ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Literature Survey on Model based Slicing

Sneh Krishna*, Alekh Dwivedi
LNCT, Bhopal, India

Abstracts

Software testing is an activity which aims at evaluating an feature or capability of system and determining that whether it meets required expectations. One way to ease this program slicing technique is to break down the large programs into smaller ones and into other is model based slicing that break down the large software architecture model into smaller models at the early stage of SDLC (Software Development Life Cycle). This is the novel methodology to extract the sub model from a big model diagrams on the basis of slicing criteria. The proposed procedure used the concept of model based slicing to slice the sequence diagram to extract the desired lump.This literature survey presents an overview of Model based slicing, including the various general approaches and techniques used to compute slices.

General Terms: UML Models, Slicing

Keywords: Dependency Graph, Model Based Slicing, Feature Based Slicing, UML/OCL Model Verification, Model Transformation Verification Through Slicing.

Introduction

Slicing of programs was introduced by [1] as a technique for the analysis of software, and has also been used for reverse-engineering and re-factoring of software. In recent year, due to increase in size and complexity of software items the importance of architectural design has been increased. The architecture of an object-oriented software system defines its high level design structures and allows an architect to reason about various properties of the system at higher level of abstraction. For this, Unified Modeling Language (UML) is best option and widely used to represent and construct the architecture of software system with the help of its various model diagram. UML diagram describe structural and behavioural aspects of architecture [2]. Structural models (e.g., class diagrams, component diagrams, object diagrams) are used to describe various relation among objects, such as aggregation, composition generalization/specialization etc. On the other hand, the behavioural models (e.g. communication and sequence diagrams, activity diagram, state diagrams) are used to describe a sequence of actions, states and their interaction, through which a use case is realized[3]. Quotes Jianjun [4], a software testing consultant, said that the first mistake that people make is thinking that the testing team is responsible for assuring quality. For better visualization of architecture, impact analysis and for test case generation the properties of system architecture with slicing can be taken into account. For this various ideas, approaches and slicing techniques are proposed by various academician' author and researcher. Second part provides a brief review of Slicing of UML models and various techniques and approaches used by the researches. Third section provides the list of tools used for model based slicing and last section provides the conclusion of this literature analysis.

UML Model Slicing

UML Model Slicing is a process of decomposition of model and extraction to identify relevant model parts across the user defined slicing criteria. In UML Model Slicing several types of model relations, and dependency such as class-class, class-operation, class-object, operation-operation, object-object, guard condition in sequence diagram , data flow, control flow, conditional predicate, etc., need to be taken into account. In this work, sequence diagram has been taken into account and various approaches present till date for UML Model Slicing have been listed.

Methodologies for UML Models Slicing

Using Dependency Relationships

Dependency Graph is an intermediate representation step by step while slicing UML Models that can describe the various types of dependencies. Zhao [4] introduced the concept of architectural slicing which operated on architectural description of software system. According to the proposed architectural description there are three types of dependencies. First is component-connector dependency where information flows from port (interface) of a component to role of a connector. Second type is connector component

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

dependency in which information flow is from role of connector to port of components. Third type of dependency is additional dependency which can be used to represent a relation between two ports or roles within a component. To compute the architectural slice a two phase algorithm has been proposed that works on SADG (software architectural dependency graph). First phase of algorithm is to compute slice 'Sg' over the SADG and second phase is to compute an architectural slice in which each element of 'Sg' is mapped to equivalent source code of description. As an extension of his previous work Zhao [4] introduced Architectural Information Flow Graph with three type of information flow arcs: Component-connector, Connector-component, internal flow arcs to apply the slicing technique on software architecture precisely.

Fangjun et.al [6] presented a method for slicing hierarchical automata. The importance of Fangjun algorithm is its ability to remove the hierarchies and concurrent states, which are irrelevant to the properties of the hierarchical automation. The given approach was based on representing the UML state chart by hierarchical automation for modelling dynamic aspects of software.The proposed method reduces the state space during model checking of UML state chart. The output slice proposed by technique is Extended Hierarchical automation instead of UML State chart models.

Maletic et al. [7] developed an approach that comprises different class relationship to define dependence relations corresponding to the relations among classes. Based upon these set of dependency relations they construct a dependence graph of UML class diagram. Their proposed models can be used in two important applications slicing the architecture and measurements of coupling between component. As their graph representation has been derived from class diagrams alone, usefulness is limited to understanding static aspects of a modelled system.

Sutton et al. [7] proposed the concept of model slicing to support maintenance of software through querying understanding, and analyzing large UML models. Kagdi

developed model slices from UML class models. His approach was to extracts parts of a class diagram in order to construct sub models from a given model of a system. However, class models are lacking of explicit behavioural information and represent only structural behaviour. For the purpose of model slicing they define a model 'M' as directed by multi graph for finite set of elements, their set of relationships, and a function that maps element to element via a relationship.

Van [8] presents an algorithm for reducing the number of interference dependencies in state chart by using the concept of slicing with concurrent state. The proposed approach considers data dependency from the definition and use of variables that are common to parallel executing statements. The core idea of approach is to know relation between states and transitions of orthogonal region to improving the degree of refinement in measurement of interference dependency. He achieved this by exploiting the internal broadcasting mechanism and maintaining the state chart's execution systematically.

Chae et.al [9] proposed UML metamodel slicer to manage the complexity of UML metamodels which addresses to all UML diagram by modularizing metamodels into small metamodels. The proposed approach extracts diagram-specific metamodels from the UML metamodel because the diagram-specific metamodels consist of a considerable small number of elements and relationships. The UML Metamodel Slicer generates a metamodel, 'MMdt' by a given set of key elements 'KEdt'. The elements in 'KEdt' are used for identifying the model elements as slicing criteria which are relevant to the diagram type 'dt'.

Moha et al.[10] presented an approach for meta-model pruning algorithm. The proposed pruner takes input slicing criteria, i.e. operations, classes, etc of the meta-model to slice the architecture and extract all the mandatory dependencies between them. The pruner resulted into an output slice that satisfies all the structural constraints imposed by the input metamodel.

Jaiprakash et al. [11][12] proposed a technique for constructing dynamic slices of UML model using the integrated state based information. In order to achieve this he proposed an algorithm i.e Architectural Model Slicing through MDG Traversal (AMSMT) that first represents a model dependency graph to collect all the information about dependency at different states of variables. Researchers proposed an algorithm that generate the dynamic slices corresponding to any slicing criteria by traversing the model dependency graph which hold all the dependency of variables. Such slice can be used for studying the impact of design changes, reliability predictions, understanding large architecture because it holds the object state information. By using the same algorithm (AMSMT) researchers had implemented prototype architectural slicing tool called SSUAM [13] to generate static slices for UML Architecture models. Later on, in another approach they proposed a DSUAM algorithm [14] which uses the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

MDG representation to compute dynamic slices. There slicing algorithm is based on traversing the edges in the MDG for any given slicing criteria. During MDG traversal, DSUAM identify the relevant model parts from architecture. Algorithm has two phase, in first phase an MDG is constructed by extracting model information through static analysis of the UML structural and behavioural model. And in second phase MDG is traversed according to the given slicing criteria to produce desire chunk relative to condition.

Neto et al. [13] This paper characterizes and analyzes MBT approaches from 78 technical papers. Important issues on automation of MBT approaches that are indicators of future trends are described. Current deficiencies in MBT are:
• MBT approaches are usually not integrated with the software development process. The models used for tests generation are not integrated with artifacts from the software development process, or were defined exclusively by a MBT approach. Integration tools must be used to support this problem;
• The MBT approach usually cannot represent and test NFR's, such as user behavior, software usability, security and reliability. Solutions for these issues must be provided.
• Most MBT approaches are not evaluated empirically and not transferred to the industrial environment. Empirical knowledge provides precise information about costs, effort, quality and context to apply a MBT approach. Expanding on the qualitative analysis and providing recommendations to practitioners is proposed for the future.

Samuel et al. [14] presented scheme to generate slice and test case with the help of edge marking dynamic slicing algorithm for activity diagram. They used the flow dependency graph (FDG) which showed the dependency among activities that arise during run time. The proposed technique used the edge marking concept to mark the stable and unstable edges in FDG so that they can slice the graph according to slicing criteria from FDG correspond to conditional predicate present at each activity edges and can generate test case according to them.

Mall et al. [15] presented a methodology to generate dynamic slices and test case with the help of UML sequence diagram. In this Message dependency graph (MDG) gets constructed which represent every message as node. To identify the conditional predicate associated with message in a sequence diagram, slicer can create dynamic slice according to the criteria. As an extension of previous work to generate automatic test case according to the functionality of the system at a designing part of the SDLC, they proposed an approach [16] to use slicing technique on the UML sequence diagram. Sequence Diagram can capture time dependent sequence of interaction between different object and component. By analyzing these relation a proper functionality of the system can be visualize which can capture to generate test cases for better verification. This was the way to generate test data in their proposed approach to select conditional predicate from sequence diagram to make a slicing criteria in the slicer while keeping all other variable constant while traversing the every node of sequence diagram until the solution is found.

Kobayashi et.al [17] proposed a sequence diagram slicing method to visualize the object oriented program behaviour. In order to achieve this, a tool has been proposed that name as 'Reticella' which is implemented as eclipse plug-in. The proposed tool take java program as input and after analyzing, fetch the static information and draw B-model tree. The slicer extract a slice according to user define slicer criteria from graph and Drawer converts the data sequence slice into sequence diagram with the help of Quick sequence diagram editor.

Panthi et.al [18] proposed an approach to generate test case from UML interaction diagram by using the condition slicing. In their approach they identified the message guard condition from interaction diagram and use the condition slicing to generate test cases. In the proposed approach, they first build a message dependency graph from UML interaction diagram and then applied the conditional slicing on a predicate node of the graph by considering guard conditions of message flow as slicing criterion to compute slices and to generate test cases.

Hyeon et.al [19] proposed an approach to address the hierarchy and orthogonality problems while tracing the data dependency in slicing of UML state machine diagram. They first, constructed a control flow graph (CFG) to track every transaction and parallel flow, and then they created a hierarchy graph that represent a parent-child relationship among region, state and behaviour of state. By utilizing CFG and hierarchy graph he have generated dependency graphs that represent the related functionality.

Nisansala et al. [20] focused on Model Checking as fully automated technique to reduce the size of model with the help of slicing. They used Behaviour Tree Dependency Graph (BTDG) to capture all functional requirements and dependency between components and attributes.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

After creating the BTDG they useed the slicing criterion which consisted of all state-realization node that updated the state of one of the component or attributes mentioned in the temporal logic property to reduce the parallelism and size of model in order to improve verification time in model checking techniques.

Uses of Control and Data Flow System Modelling

Control and Data Flow are the most important aspect of system modelling or UML models that describes the nature of every component, their behaviour, and working with other component and sequential pattern of interaction. Many researchers dedicated their work to slice the models and the architecture of the system into desirable small chunks. Korel et.al [21] dedicated their work on slicing the state based models, such as EFSMs (Extended Finite State Machines). As a result of two types of slicing came into existence deterministic and non deterministic slicing. Their approach also includes a slice reduction technique to reduce the size of a computed EFSM slice by isolating the parts of the model that may contribute to faulty behaviour. Their research of slicing techniques for state machines is thoroughly based upon control and data-flow analysis. To automate the slice computation they proposed tool that constitute of graphical editor, an EFSM executor and EFSM slicer.

Crest [22] defined that slicing can be carried out for UML state machines, using data and control flow analysis to remove element of the machine that do not contribute to the value of a set of features in a selected state of the machine. The proposed technique of slicing by refactoring enables the models to be simplified and factored on the basis of features. He also represented the pre and post condition relationship of the state during the path predicate coverage.

Mall et.al [23] proposed a scheme known as 'Ctest' that automatically generate test cases from UML communication diagram. According to Mall the first step of the approach is to construct communication tree from communication diagram on the basis of data flow and control flow. After selecting the predicate from tree, tool named as UTG (UML behavioral Test case Generator) transform the predicate according to 'CTest' schema to find the test data. Communication diagram has been taken as input by tool in xml format. In this approach Document Parser class parses the XML file for the message name, arguments, sequence numbers and constructs the communication tree, while Test Data Finder uses the parsed information and finds the test data in the form of a string

Julliand et.al [24] proposed an approach based on domain abstraction for generating test cases on the basis of synthetic abstraction and variable elimination with the help of model slicing. In the proposed approach source model is taken as input with set of abstract variable then reduced by syntactic abstraction followed by semantically abstraction to generate abstract model from which symbolic tests are extracted according to selection criteria. They proposed three methods for identifying the relevant variable and generating abstract model. The first one is to consider data flow dependency only. Second one uses both data-flow and control-flow dependency. Third method is to use data flow and partial control flow dependencies to find as much as possible strong relevant variables. Once the set of abstract variable 'Xa' is defined the next step is to define the Slice function that abstract the predicate P according to 'Xa'. The core idea of the work is to use the combination of syntactic and semantic abstraction or slicing to refine the result more precisely while generating the test cases.

Using UML/OCL Constraints

In the MDD and MDA approaches , models become the primary artifacts of the development process. Therefore , assessment of the correctness of such models is a key issue to ensure the quality of the final applications. In that sense, this paper presents an automatic method that uses the constraint programming paradigm to verify UML class diagrams extended with OCL constraints. In our approach , both class diagrams and OCL constraints are translated into a constaint satisfaction problem.

Memon et.al [25][26] proposed a verification technique to check the correctness of model with the help of slicing. The proposed technique increases the scalability of verification by partitioning the original model into submodel. To define the binary association and inheritance relation, dependency graph and flow graph has been used in the proposed approach. By which slicing can be done easily to decompose the model into sub models with the help of dependency graph to extract instances of models and correct component relations.

In another approach [27][28] author proposed a tool (UOST) to enable the efficient verification of UML/OCL Class diagram with the help of model slicing technique. The tool can verify the properties of the diagram with disjoint and non-disjoint sets of slicing. Tool take the class diagram as input in XMI format with text specified OCL constraints and break the file into several slices with the help of model slicing technique after parsing the file. Researcher used the eclipse solver to translate the slices into CSP and check the existence of solution to generate the respective object diagram for satisfiabile and unsatisfaible sub models with their

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

specific invariants.

Acher et al. [29] proposed an algorithm for automatic generation of test cases from sequence diagrams. They first transform UML sequence diagram into graphical representation named as SDG (Sequence diagram graph). They follow a graph based methodology with a depth first search algorithm to traverse the SDG and to generate test cases according to all message sequence path coverage criteria. To retrieve the information for a specification of input/output, pre and post conditions for test cases generation they use the use case template, class diagram and data dictionary and expressed in OCL.

Using Feature Based Criteria

Archer et al. [30] proposed a novel slicing technique on the feature model by taking cross-tree constraints into account with respect to set of features which are acting as slicing criteria. The core idea of proposed algorithm is to compute proposition formula representing the set of configuration and rules and to apply propositional logic reasoning techniques to construct an FM (representing its hierarchy, variability information, feature groups and cross-tree constraints).By extended the previous author [31] also proposed the concept that how set of complementary set of operators like aggregate, merge and slice can provide practically and efficient support for separation of concerns from feature modeling. They defined that slicing process is both semantic and syntactic so they analyze the cross-cutting constraints to define the features that must be or cannot be sliced. In their proposed technique, the feature model and its cross-cutting constraints are first analyzed by transformation into predicates and then these predicates are transformed in a sliced feature model.

Hubaux et al. [32] proposed a slice feature diagram to design three different views of an input diagram to provide more flexibility to the configuration environment. The sliced diagram does not keep the same structure as the input diagram. The proposed approach does not consider cross-cutting constraints and is thus syntactic. The important property of the approach is that it should always lead to valid configurations but the problem can arises in the approach when features belong to more than one view or, more generally, when the selection of a feature in one view affect the selection of another feature in a concurrent view.

Using Model Languages

Kim [33][34] introduced the slicing technique called dynamic software architecture slicing (DSAS). In the situations where a large number of ports are present and their invocation can change the values of some variables, or the occurrence of certain events, Kim's work is very efficient there because it's able to generate a smaller number of components and connectors in each slice according to slicing criteria. In this approach software architecture is first designed by using ADL (Architecture description language) and later on mapped onto program statement as executable architecture. Dynamic slicer takes slicing criterion as input, and reads the ADL source code of the architecture to identify the information of component and connector along with the event names used in the ADL and parameter names combined with those events. The proposed algorithm filters out the events that are not relevant and passes only those which are relevant to slicing criterion and generate resulting software architecture slice as shown in Fig 2.1.

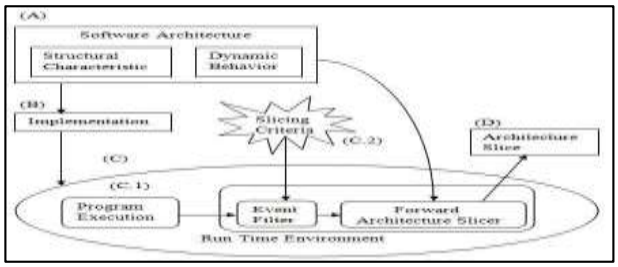

Fig 2.1 Dynamic Software Architecture Slicing Methodology Proposed by Kim [32]

Falessi et al. [34] used the concept and technique of model slicing to automate the safety inspection of system. In order to achieve this, a tool named "Safe Slicer" has been proposed that use model based slicing to enables automatic extraction of the safety-related slices (fragments) of design models. They proposed and elaborate a design methodology which ensures the traceability of links required for automated slicing. The methodology and the slicing algorithm proposed by these researchers are the basis for the Safe Slice tool. Evaluation conducted by this tool indicates that the use of design slices substantially reduces the amount of information that needs to be inspected.

Lano et al. [35] defined the technique for slicing of UML model using Model Transformation, particular for restriction of model to those parts which specifies the properties of subset within. The proposed technique use class diagrams, individual state machines and communicating sets of state machines to perform slicing of UML Model. Researcher used the client-supplier relation between different classes to form a tree structure. According to proposed technique slicing will be carried out upon class invariants and operation pre and post conditions by considering the predicates P. To slice the behaviour and communicating state machines they define criteria's for a slice 'S' of a state machine

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

'M' are S <syn M, if S has fewer elements than M. S =sem M. This means that any analysis which concerns the value of the slice features V in the selected state s, over all paths to this state, can be performed on the slice S, and the result will also apply to state model M. their techniques focused on structure-preserving and amorphous slicing of class diagrams and state machines by using Model transformations to perform slicing, each transformation will satisfy the <syn and =sem relations, resulting in a slice which also satisfies these relations compared to the original model.

Zoltán et.al [36][37] proposed dynamic backward slicing of model transformations technique with respect to program slicing. To slice the models they used model transformation language as a core of technique with the help of Dynamic Backward slicing by considering the Execution traces of program to generate final slice. The proposed technique take three inputs, the model transformation program, the model on which the MT program operates and the slicing criterion and generate the output as transformation slices and model slices. Transformation of models into MT Language is done by three consecutive processes, Graph pattern, Graph transformation rules and control language on VIATRA2 platform. The algorithm keep the record of execution traces during graph pattern calls and GT rules, to provide traceability information between source and target models and uses these traces to slice MT programs and models simultaneously according to slicing criteria.

Model based slicing tools [41]

Table 1: List of Tools

Year	Name of Tools	Techniques Used
2003	EFSM Slicing Tool	Control and Data flow analysis.
2007	UTG	Data Flow and Control Flow dependency, Communication Tree
2008	SSUAM	Model Dependency Graph.
2008	UML Slicer	MetaModel Diagram.
2009	Reticella	B-Model dependency Graph.
2011	Archlice	Model Dependency Graph.
2011	Safe Slicer	System Model Language, Traceability Links and Rules.
2012	UOST	UML + OCL Constraints.

Concluding remarks and future work

From the given literature this has been listed out that for model based slicing techniques there is use of dependency relation, control and data flow, UML/OCL constraints, model language are present in literature with great emphasis on dependency relation. Hence there is a need for such technique that can reduce the effort of generation of dependency graph as intermediate state. Slicing UML architectural models is a difficult problem since the model information is distributed across several diagrams with implicit dependencies among them. We had to first construct an intermediate representation called MDG by synthesizing information present in various architectural model elements. Such slices can be used for studying the impact of design changes, reliability prediction, understanding large architectures, etc. We are now trying to enhance our intermediate model by integrating the state and activity models into MDG to compute more accurate slices.

References

1. Blouin, B. Combemale, B. Baudry, O. Beaudoux, "Kompren Modeling and Generating Model Slicers," Journal of Software and System Modeling, Springer, 2012.
2. Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide," 2nd Edition, May 2005, Publisher. Addison Wesley.
3. M. Weiser, Program slicing, IEEE Transactions on Soft. Eng., 10, July 1984, pp.352–357.
4. Jianjun Zhao, "Slicing Software Architecture," Technical Report 97-SE-117, pp.85-92, Information Processing Society of Japan, Nov 1997.
5. Jianjun Zhao, "Applying slicing technique to software architectures," In Fourth IEEE International Conference on Engineering of Complex Computer Systems, ICECCS'98, pp 87 –98, 1998.
6. W. Fangjun and Y. Tong, "Dependence Analysis for UML Class Diagrams," J. Electronics (China), vol. 21, no. 3, pp. 249-254, May 2004, doi 10.1007/BF02687879.
7. H. Kagdi, J.I. Maletic, and A. Sutton, "Context-Free Slicing of UML Class Models," Proc. 21st IEEE Int'l Conf. Software Maintenance, pp.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

635-638, 2005.
8. S. Van Langehove, "Internal Broadcasting to Slice UML State Charts: As Rich as Needed," Proc. Abstracts of the FNRS Contact Day: The Theory and Practice of Software Verification, Oct.2005.
9. J.H. Bae, K. Lee, and H.S. Chae, "Modularization of the UML Metamodel Using Model Slicing," Proc. Fifth Int'l Conf. Information Technology: New Generations, pp. 1253-1254, 2008.
10. Sagar Sen, Naouel Moha, Benoit Baudry, and Jean Marc Jézéquel, "Meta-model Pruning," In 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09), 2009.
11. Jung Ho Bae and Heung Seok Chae, "UMLSlicer: A tool for modularizing the UML metamodel using slicing," In 8th IEEE International Conference on Computer and Information Technology (CIT), pp.772–777, 2008
12. Jaiprakash T. Lallchandani, R. Mall, "Static Slicing of UML Architectural Models," Journal of Object Technology, vol. 8, no. 1, pp. 159-188, January-February 2009
13. Arilo C. Dias Neto, Rajesh Subramanyan, Marlon Vieira, Guilherme H. Travasso "A Survey on Model-based Testing Approaches: A Systematic Review".
14. J. Lallchandani and R. Mall, "Slicing UML Architectural Models," ACM SIGSOFT, vol.33, no.3, May 2008.
15. J. Lallchandani and R. Mall, "A Dynamic Slicing Technique for UML Architectural Models," IEEE Transaction on Software Engineering, Vol. 37, No. 6, NOV/DEC 2011.
16. Philip Samuel, Rajib Mall, "Slicing-Based Test Case Generation from UML Activity Diagrams," ACM SIGSOFT Software Engineering Notes, Vol. 34 No. 6, November 2009.
17. Philip Samuel, Rajib Mall, "A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams," e-Informatics Software Engineering Journal, Vol. 2, Issue 1, 2008.
18. Ranjita Kumari Swain , Vikas Panthi, Prafulla Kumar Behera, "Test Case Design Using Slicing of UML Interaction Diagram," 2nd International Conference on communication, computing and security, vol.6 , pp.136-144, ELSEVIR, 2012.
19. Hyeon-Jeong Kim , Doo-Hwan Bae, Vidroha Debroy, W. Eric Wong, "Deriving Data Dependence from UML State Machine Diagrams," Fifth International Conference on Secure Software Integration and Reliability Improvement (IEEE), 2011.
20. Nisansala Yatapanage, KirstenWinter, and Saad Zafar, "Slicing behavior tree models for verification," In IFIP Advances in Information and Communication Technology, Vol. 323, pp. 125–139, 2010.
21. B. Korel, I. Singh, L. Tahat, and B. Vaysburg, "Slicing of State Based Models," Proc. Int'l Conf. Software Maintenance, pp. 34-43, 2003.
22. Kevin Lano Crest, "Slicing of UML State Machines," Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '09), 2009.
23. Philip Samuel , Rajib Mall, Pratyush Kanth, "Automatic test case generation from UML communication diagrams," Information and Software Technology (ELSEVIER), 2007.
24. J. Julliand, N. Stouls, P-C. Bue, P-A. Masson, "B model slicing and predicate abstraction to generate tests," Software Quality Journal, vol. 21, pp.127-158, 2013.
25. Asadullah Shaikh, Robert Clarisó, Uffe Kock Wiil, and Nasrullah Memon, "Verification-driven slicing of UML/OCL models," In Proceedings of the IEEE/ACM international conference on Automated software engineering, pp. 185–194, ACM, 2010.
26. Asadullah Shaikh, Uffe Kock Wiil, and Nasrullah Memon, "UOST: UML/OCL aggressive slicing technique for efficient verification of models," In System Analysis and Modeling: About Models, 6th International Workshop SAM'10, pp. 173–192, 2010.
27. Asadullah Shaikh, Uffe Kock Wiil, and Nasrullah Memon, "Evaluation of tools and slicing techniques for efficient verification of UML/OCL class diagrams," Advances in Software Engineering, vol.18, pp 173-192, 2011.
28. Monalisa Sarma, Debasish Kundu, Rajib Mall, "Automatic Test Case Generation from UML Sequence Diagrams," 15th IEEE International Conference on Advanced Computing and Communications, 2007.
29. Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France, "Slicing feature models," In 26th
30. Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France, "Separation of

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Concerns in Feature Modeling: Support and Applications," In Aspect-Oriented Software Development(AOSD'12), ACM

31. Press, 2012. T. Kim, Y.-T. Song, L. Chung, and D.T. Huynh, "Dynamic Software Architecture Slicing," Proc. 23rd Int'l Computer Software and Applications Conf., pp. 61-66, 1999.
32. Arnaud Hubaux, Patrick Heymans, Pierre-Yves Schobbens, Ebrahim Khalil Abbasi, and Dirk Deridder, "Supporting multiple perspectives in feature-based configuration," Software and Systems Modeling, 2012.
33. T. Kim, Y.-T. Song, L. Chung, and D.T. Huynh,"Software Architecture Analysis: A Dynamic Slicing Approach," J. Computer and Information Science, vol. 1, no. 2, pp. 91-103, 2000. .
34. Davide Falessi, Shiva Nejati, Mehrdad Sabetzadeh, Lionel Briand, and Antonio Messina, "SafeSlice: a model slicing and design safety inspection tool for SysML," In SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13rd European Software Engineering Conference (ESEC-13), ACM, 2011.
35. Kevin Lano and Shekoufeh K. Rahimi, "Slicing of UML Models Using Model Transformations," Model Driven Engineering Languages and Systems, 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Lecture Notes of Computer Science, Vol. 6395, pp. 228-242, Springer, 2010.
36. Zoltán Ujhelyi, Ákos Horváth, and Dániel Varró, "Towards dynamic backward slicing of model transformations," In 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp.404–407, IEEE Computer Society, 2011.
37. Zoltán Ujhelyi, Ákos Horváth, and Dániel Varró, "Dynamic Backward Slicing of Model Transformations," IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012.
38. R. Singh, Vinay Arora,"Literature analysis on model based slicing" *International Journal of Computer Applications (0975 – 8887)*