ISSN 2278 – 0211 (Online)

Process for Extracting Subpart from UML Sequence Diagram

Sneh Krishna
M.Tech. Research Scholar, LNCT, Bhopal, Madhya Pradesh, India
Alekh Dwivedi
Assistant Professor, LNCT, Bhopal, Madhya Pradesh, India
Vineet Richhariya
Head of the Department, CSE Department, LNCT, Bhopal, Madhya Pradesh, India

Abstract:
Software visualization is lone promising technique meant at helping developers and testers to recognize the performance of object-oriented systems successfully. However, it is still complex to understand this performance, because the size of involuntarily generated model diagrams tends to be ahead of the study capacity. In this paper, a novel methodology to simplify the problem of software visualization has been planned, that uses the concept of slicing UML diagrams and model based slicing technique.

Keywords: *Slicing, sequence diagram, software visualization, UML, model based slicing*

1. Introduction

In current years, due to the enhance in size and difficulty of software products the significance of architectural design has been improved. The design of an object-oriented software system define its high level design structure and allows an architect to motive about various property of the system at higher level of concept. For this, Unified Modeling Language (UML) is the best choice and widely used to symbolize and construct the design of software system with the help of a variety of model diagrams. UML diagrams explain structural and behavioral aspect of architecture [1, 2]. Structural models (e.g. class diagrams, object diagrams, component diagrams) are used to explain various relationships among objects, such as aggregation, connection, work and simplification/specialization etc. On the other hand, the behavioral models (e.g. communication and sequence diagram, activity diagram, state diagram) are applied to depict a sequence of events, states and their dealings, through which a use case is realized. The task of analyzing UML Models is bit tricky since the information about system can be spread across several model views. To surmount this problem the thought of model based slicing came to reality. Model Based slicing is a breakdown technique to extract and recognize relevant model parts (or fragments) or associated elements across varied model views. It takes the user defined slicing criteria as input and slices the architecture, as a view of importance [3]. Slicing is useful in software protection, reengineering, testing, breakdown and integration, recompilation, program comprehension, and debugging. The aim of software testing is to ensure excellence. Software testing is essential to produce highly dependable systems, since static verification techniques bear from several hurdles to detect all software faults [16]. The most intellectually challenging part of testing is the plan of test cases. Test cases are typically generated based on program source code. Another approach is to generate test cases from specifications developed using formalisms such as UML models. Slicing is best technique which can decay the structure into sub models without disturbing their core structure and functionality. It helps the developer to increase the perfect view of software according to their condition.

2. Literature Analysis on Model Based Slicing

In early stage of growth, Model based slicing has been useful to state machines [4] where similar benefits as those scheduled above has been claimed. State machine slicing is an example of applying slicing to a model of a scheme rather than to the system accomplishment. However, system models such as those represented in terms of the UML-family of languages are much more difficult than state machines (and contain state machine sub-languages). Several approach and attempt have been made for slicing UML diagrams. The loom in [5] describe context-free slicing of UML class models where the issue of situation has been defined to be object position, which is a dynamic property of the situation therefore context free slicing is a static slice of a structural model. As noted in [5] the criterion used for slicing a model is more composite than that used in program or state machine slicing since there are more types of elements and relationships. This has been noted that OCL (object constraint language) should be use to state the slicing criteria. A similar approach has been used to modularize the UML meta-model into collection of works that are relevant to the different UML diagram types in [6] though the predicate used to resolve the slicing criteria has been set in terms of traverse the metamodel elements opening with a group of supplied classes. Class models has sliced in combination with OCL invariants in [7] thereby dropping the state-space explosion that would otherwise occur when using a model-checker (in this case Alloy) to verify a class-model. UML state-charts can be sliced as described in [8] [9] [10] though these approaches do not simplify the results to include other parts of the UML language family. Both static and dynamic aspects of UML can be joint and sliced as described in [11] [12] where class and sequence diagrams are combined into a single symbol (a model dependency graph MDG) that can be consequently sliced to show partial dynamic and structural information resulting from criteria containing both structural and dynamic constraint. Slicing UML sequence diagrams in order to produce test cases has been described in [13] [15]. UML sequence diagrams (or scenarios) are basically an essential part of execution of a program. It depicts the objects and classes involved in the condition and the sequence of messages exchange between the objects. Sequence diagrams are normally associated with use case realization in the Logical View of the system under progress. It has been meticulously analyzed that for the process of slicing sequence diagram no combine technique have been developed to remove the point of interest from architecture of software to ease the software visualization that uses provisional predicate for finding out a relative slice. Figure 1 shows the general view of functional behavior of software models in the form of sequence diagram. Rectangular box specify the objects within model diagram that are interacting with each other. Doted lines signify the life line of the objects on which instances of the objects has been produced. Arrow represents the particular action (in the form of messages) of objects and their way, where round brackets specify the guard condition that specifies the truth and false value to interact. Sign of cross specify the end of life line where all the instance of the objects will devastate after the completion of their relevant action.

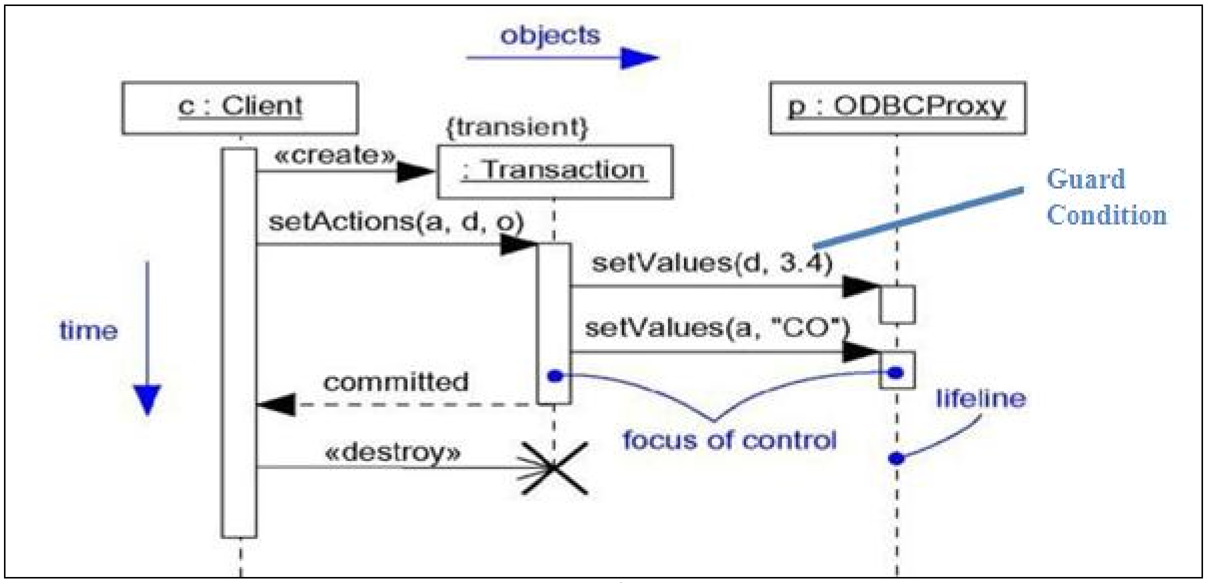

Figure 1: Generic view of Sequence diagram [1]

3. Proposed Methodology

The proposed work addresses the slicing of sequence diagram to ease the software visualization through using conditional predicate for finding relevant slices. In the planned methodology, following steps has been followed:

- Step1. Generation of UML (Sequence) diagrams of/from a demanding requirement specification.
  1.1. Visual paradigm for UML, Magic-draw, and Rational rose; etc can be used to create the UML diagrams.
- Step 2. Create XML from the particular UML diagram (Sequence diagrams).
  2.1. Visual paradigm for UML 10.0 version gives the in-built functionality to export the diagram into XML format.
- Step 3. Document Object Model (DOM) parsers for parsing XML code and produce an output file (with .txt extensions) having Object name, message name, Identifier, message to & fro information.
  3.1. Java API DOM is applied to parse the XML code file produced in step 2.
  3.2. DOM parser utilizes the function Document Builder Factory ( ) to constructs the instance of the class to parse the files.
  3.3. DOM parser will create a txt file having information regarding objects name and its identifier. This file also holds the information related to the entire messages and the objects with which the message is floating.
  3.4. The entire information supplied by parser will be store up in separate .txt file.
- Step 4. Passing file achieved from step 3 and slicing criteria to a .java programs (which acts as slicer) for getting the relative/required large piece of information in a separate .txt files.
  4.1. Slicer will take .txt file produced in step 3 as input.
  4.2. Slicer will request user to name the slicing criteria at run time to produce the chunk/slice as per particular requirement.
  4.3. Computed slices will be store in different .txt file which contains the information of messages, their guard conditions and objects id's between which messages are being passed.

- Step 5. Shifting object id with relative object name between which message is passing so that information can be get back easily (this step will only deal through sliced part).
  5.1. To ease the regained of information objects id's will exchanged by their corresponding objects name (in the file retrieved from step 4.3).
  5.2. The entire information will store in different .txt file which contains the information of messages and the object name (between which they are communicating relative to user define slicing criteria).
- Step 6. Passing txt file as achieved from step 5 to a .java programs so that it can be exchanged into input file format for Quick Sequence Diagram Editor.
- Step 7. Tool will produce the absolute and relatively small sequence diagram.
  7.1. Tool will capture the input format described at step 6 as input to transform into its correspondent diagram.
  7.2. Filtered slice (small sequence diagram) will be created as final output according to slicing criteria by means of requirement to ease the software visualization.

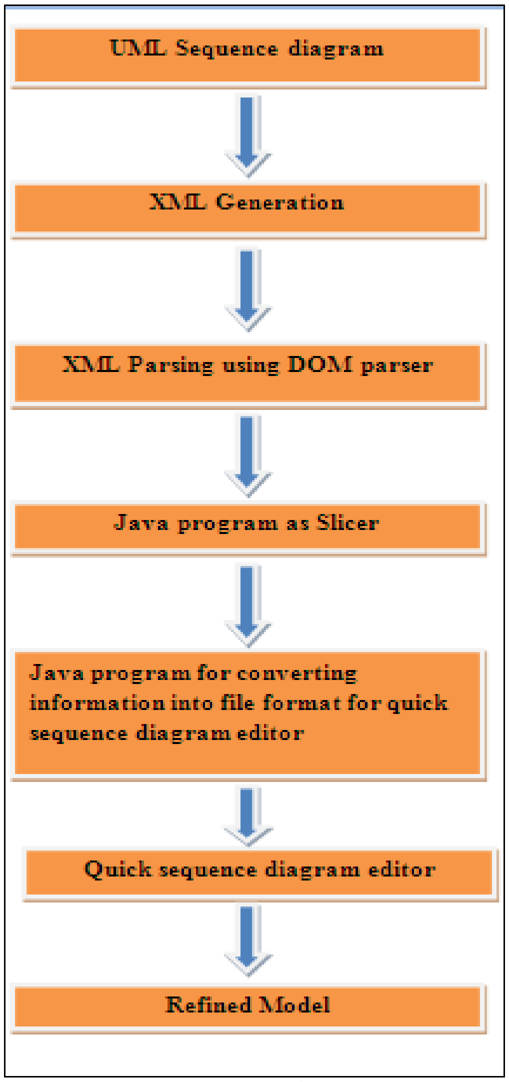
UML Sequence diagram
XML Generation
XML Parsing using DOM parser
Java program as Slicer
Java program for converting information into file format for quick sequence diagram editor
Quick sequence diagram editor
Refined Model

Figure 2: Overview of Methodology

Consider an example UML sequence diagram as shown in figure: 3. the point of selecting this example is to display the concept of proposed methodology. In the example there are four objects which are interacting with each other thorough message passing (using guard order as true to interact with each other). We illustrate our methodology by explaining the generation of chunk or refined model diagram as shown in figure 4 with respect to slicing criteria. Here in this example let the slicing criterion is variable „z". Given criteria is a variable used in conditional predicate of message guard condition. True and false value of these guard conditions are used by objects to interact with each other. According to user defined slicing criteria, proposed methodology will slice the model diagram shown in figure 3 and generate the resultant small chunk or refined sequence diagram as shown in figure 4.

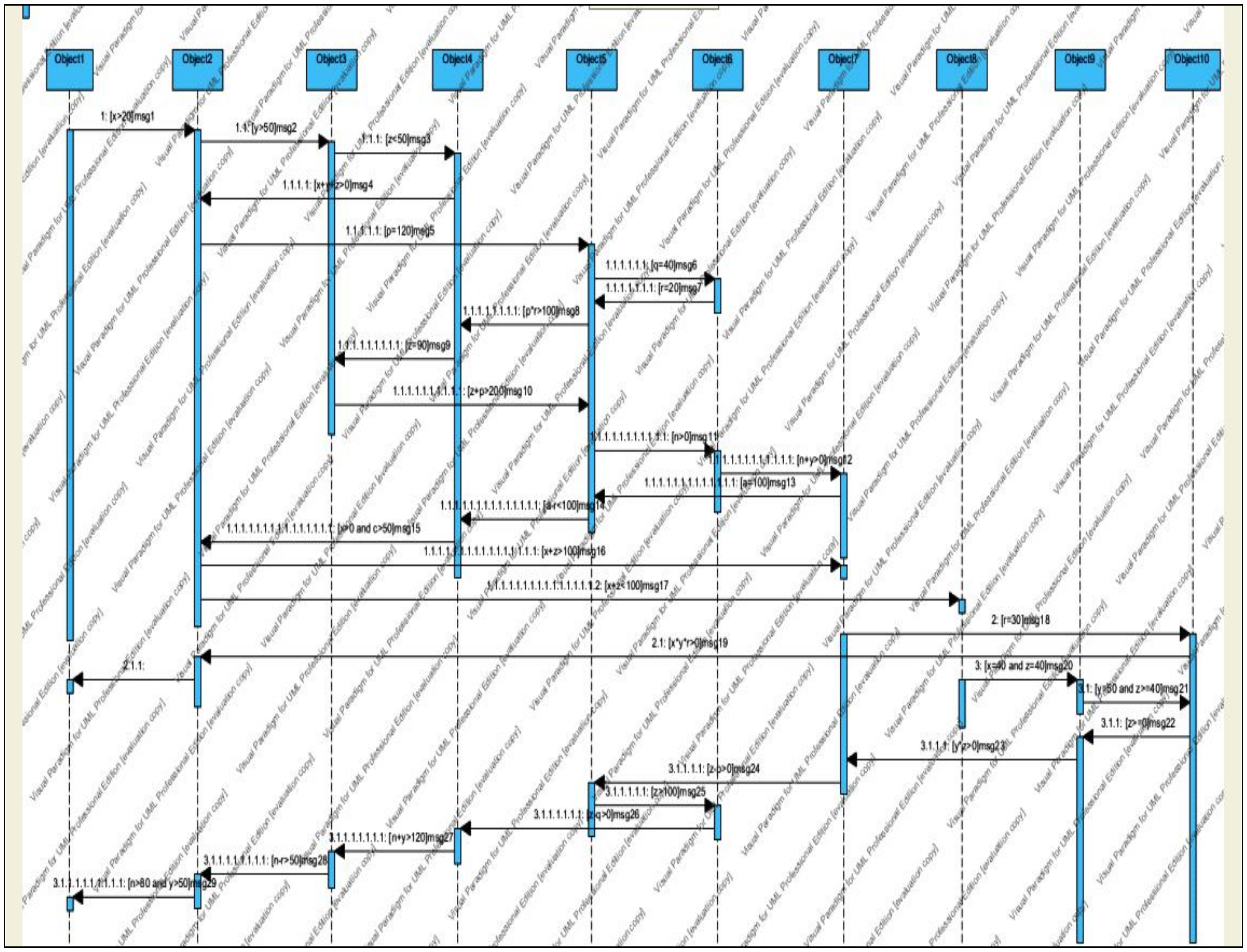
Figure 3: Example Sequence diagram

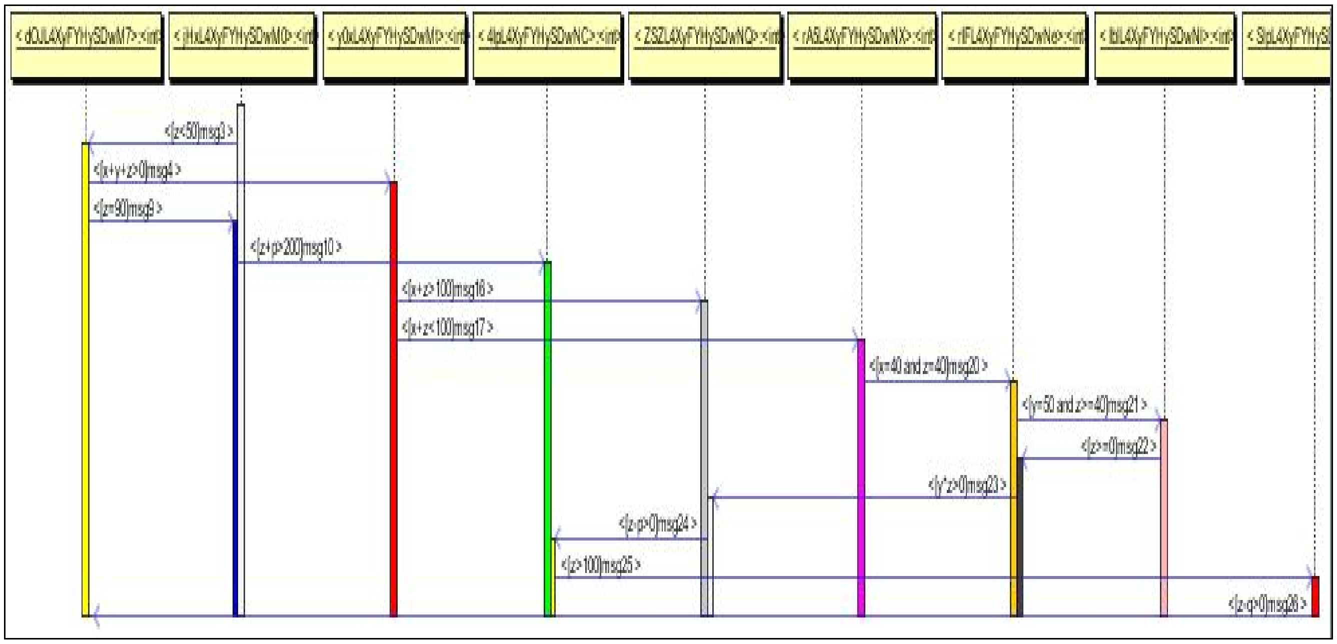
Figure 4: Refined Sequence diagram

Related Work	Type of Model	Sliced Entity	Slice Type	IR*	Elements of Slicing Criterion	Elements of the Slice
Zhao[2]	Wright ADL	Architectural Specification	Static	AIFG*	Set of ports of a component, or a set of rules of a connector	Set of components, and connectors
Zhao[17]	Acme ADL	Architectural Specification	Static	SADG*	Set of ports of a component, or a set of rules of a connector	Set of components, connectors, and attachments
Kim[18]	Rapide ADL	Architectural description and its implementation	Dynamic	DNU-IR*	Input values for ADL executable, event name and no. of occurrences of an event	Set of components, and ports
Korel *et al*.[19]	EFSM	State-based EFSM model	Static	EDG+++	Pair of variable and transition in EFSM	Transition in EFSM that affect an variable
Kagdi *et al*.[5]	UML	Class Models	Static	Digraph	Set of initial elements, set of selected elements, and set of relationship	Set of class elements and its relations
Ojala[9]	UML	State Machines	Static	CFG	Set of transitions in state machines	CFG nodes
Wang *et al*.[20]	UML	State Charts	Static	EHA	Set of states and transitions based on LTL** property	States and transitions in eha
Langenhove [10]	UML	State Charts	Static	FSM	States in state chart and some data values	Program statements
Bae *et al*[6]	UML	UML metamodel	Static	Digraph	UML diagram type	Elements of a UML diagram
Lallchandani [11]	UML	UML model(structural and behavioural)	Dynamic And Static	MDG	class, scenario, and data values	UML model elements
Present Work	UML	UML model(sequence diagram)	Static And Dynamic	Guard Condition	Model Based Slicing	UML sequence diagram

Table 1: Summary of Comparison with Related Work

*IR is acronym for "intermediate representation" #DNU-IR means "Do not use any intermediate representation" +AIFG is an acronym for "Architecture information flow graph" ++SADG is an acronym for "software architectural dependence graph" +++EDG is an acronym for "Extended finite state machine dependence graph" **LTL is an acronym for "Linear temporal logic".

5. Conclusion

A new technique for model based slicing has been planned that will extract the sub-model from architecture of the software to ease the software visualization. The key contribution of the method is to generate the refined model slices related to user defined slicing criteria using conditional predicate in sequence diagram. Sequence diagrams capture time dependent (temporal) sequences of relations between objects. The basis of the proposed technique is „UML‟ and „Slicing‟. With this, the problem of visualization of large and complex software can be sorted out. This technique can further be improved and applied in the field of concurrent programming.

6. References

1. Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide," 2nd Edition, May 2005, Publisher. Addison Wesley.
2. Jianjun Zhao, "Slicing Software Architecture," Technical Report 97-SE-117, pp.85-92, Information Processing Society of Japan, Nov 1997.
3. Sneh Krishna, Alekh Dwivedi , "Literature Survey on Model based Slicing" International Journal of Engineering Sciences &

Research Technology , pp. 576-583, [Krishna, 3(10): October, 2014].
4. K. Androutsopoulos, D. Clark, M. Harman, Z. Li, and L. Tratt. Control dependence for extended finite state machines. Fundamental Approaches to Software Engineering, pp. 216–230, 2009.
5. H. Kagdi, J.I. Maletic, and A. Sutton, "Context-Free Slicing of UML Class Models," Proc. 21st IEEE Int"l Conf. Software Maintenance, pp. 635-638, 2005.
6. J.H. Bae, K.M. Lee, and H.S. Chae. Modularization of the UML metamodel using model slicing. In Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on, pages 1253–1254. IEEE, 2008.
7. A. Shaikh, R. Clarisó, U.K. Wiil, and N. Memon, "Verification-driven slicing of UML/OCL models," In Proceedings of the IEEE/ACM international conference on Automated software engineering, pages 185–194. ACM, 2010.
8. Kevin Lano Crest, "Slicing of UML State Machines," Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '09), 2009.
9. V. Ojala, "A slicer for UML state machines," Helsinki University of Technology, 2007.
10. S. Van Langenhove, "Towards the Correctness of Software Behavior in UML: A Model Checking Approach Based on Slicing," Dissertation, Department of Mathematics, Ghent University, 2006.
11. J.T. Lallchandani and R. Mall, "Slicing UML architectural models," ACM SIGSOFT Software Engineering Notes, vol.33, no.3, pp. 1–9, 2008.
12. J.T. Lallchandani and R. Mall, "Integrated state-based dynamic slicing technique for UML models," Software, IET, vol. 4, no. 1, pp. 55–78, 2010.
13. P. Samuel and R. Mall. A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams. e-Informatica Software Engineering Journal Selected full texts, vol. 2, no. 1, pp. 61–77, 2008.
14. J. Qian and B. Xu, "Program slicing under UML scenario models," ACM SIGPLAN NOTICES, vol. 43, no. 2, 2008.
15. P. Samuel, R. Mall, and S. Sahoo, "UML Sequence Diagram Based Testing Using Slicing," IEEE Indicon 2005 Conference, pages 176–178, IEEE, 2005.
16. R. V. Binder, "Testing object-oriented software: a survey," Software Testing Verification and Reliability, vol. 6(3/4), pp: 125 – 252, 1996.
17. Jianjun Zhao, "Applying slicing technique to software architectures," In Fourth IEEE International Conference on Engineering of Complex Computer Systems, pp.87 –98, 1998.
18. T. Kim, Y.-T. Song, L. Chung, and D.T. Huynh, "Dynamic Software Architecture Slicing", In Proceeding 23rd International conference on Computer Software and Applications , pp. 61-66, 1999.
19. B. Korel, I. Singh, L. Tahat, and B. Vaysburg, "Slicing of State Based Models", In Proceeding of International Conference of Software Maintenance, pp.34-43, 2003.
20. J. Wang, Wei Dong, and Zhichang Qi, "Slicing Hierarchical Automata for Model Checking UML Statechart," In Proceeding of 4th International Conference of Formal Engineering Methods and Software Engineering, pp. 435-446, Oct. 2002.