A practical approach for model based slicing

Rupinder Singh\(^1\), Vinay Arora\(^2\)
\(^1\)(CSED, Thapar University, Patiala, India)
\(^2\)(CSED, Thapar University, Patiala, India)

Abstract: Software testing is an activity that will aim at evaluating an attribute or capability of system and determine that whether it meets required expectations. Test cases can be designed at model level of software development. But to visualize the software model or architecture is difficult due to its large and complex structure. We have presented a novel methodology to extract the sub-model from model diagrams correspond to point of interest to ease the software visualization. The proposed methodology use the concept of model based slicing to slice the sequence diagram to extract the desired chunk.

Keywords – Software Testing, Sequence diagram, Model based slicing.

Submitted date 17 June 2013

I. INTRODUCTION

Software testing is an evaluation process to determine the presence of errors in code snippet. Software testing cannot completely test software because exhaustive testing is not possible due to time and resource constraints. According to ANSI/IEEE 1059 standard [1, 2], Testing can be defined as “A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/bugs) and to evaluate the features of the software item”. The prime objective of testing is to discover faults that are preventing the software in meeting customer requirements. Moreover, testing requires planning and designing of test cases. The testing process progresses from component level to system level in an incremental way. The most intellectually challenging part of testing is the design of test cases. Test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle. Now days, UML has been widely used for object oriented modeling and design. This is due to the fact that a UML metamodel extends support to describe structural and behavioral aspects of an architecture. For instance, the structural models (e.g. class diagrams) are used to describe various relations among classes, such as aggregation, association, composition, and generalization/specialization. On the other hand, the behavioral models (e.g., communication and sequence diagrams) are used to depict a sequence of actions in an interaction that describe how the objects are interacting to perform their respective action [3]. But due to large and complex architecture of software products it’s hard to visualize and test the software. To overcome this problem of visualization large models, ‘Model based slicing’ technique came into existence. Model Based slicing is a decomposition technique to extract and identify relevant model parts (or fragments) or related elements across diverse model views. Slicing can be referring as process or strategy to break body of information into smaller parts to examine it from different viewpoints that can yield more precise information so that one can understand it better [35]. The term is also used to mean the presentation of information in a variety of different and useful ways to ease the visualization. But in term of automation of software testing, the direct use of UML diagrams is not feasible. Conversion of UML into XML is one of the integral part of model based slicing for automation of whole technique. The core principle of model based slicing technique is to decompose the structure into submodels without affecting their core structure and functionality. It helps the developer to gain the perfect view of software according to their requirement.

II. RELATED WORK

In Model based slicing several types of model relations and dependency such as class-class, class-operation, operation-operation, class-object, object-object, guard condition in sequence diagram, conditional predicate, control flow, data flow etc., need to be taken into account as discussed in previous paper [34]. In this work, sequence diagram has been taken into account and various approaches present till date for slicing UML diagram have been listed. Zhao [4] introduced the concept of architectural slicing which operates on architectural description of software system. As an extension of his previous work Zhao [5] introduced Architectural Information Flow Graph with three types of information flow arcs: Component-connector, Connector-component, internal flow arcs to apply the slicing technique on software architecture precisely. Wang et.al [6] presented a method for slicing hierarchical automata. The importance of Wang’s algorithm is its ability...
to remove the hierarchies and concurrent states, which are irrelevant to the properties of the hierarchical automata. Kagdi et al. [7] proposed the concept of model slicing to support maintenance of software through the understanding, querying and analyzing large UML models. Langehove [8] presents an algorithm for reducing the number of interference dependencies in state charts by using the concept of slicing with concurrent states. The proposed approach considers data dependencies from the definition and use of variables that are common to parallel executing statements. Lallchandani et al. [9] propose a technique for constructing dynamic slices of UML models using the integrated state-based information. In order to achieve this they proposed an algorithm for Architectural Model Slicing through MDG Traversal (AMSMT). By using the same algorithm (AMSMT) researchers had implemented a prototype architectural slicing tool called SSUAM [10] to generate static slices for UML Architecture models. Later on, in another approach [11] they proposed a DSUAM algorithm which uses the MDG representation to compute dy-namic slices.

Samuel and Mall [12] presented a scheme to generate slice and test cases with the help of edge marking dynamic slicing algorithm for activity diagrams. They used the flow dependency graph (FDG) which shows the dependencies among activities that arise during run time. Noda et.al [13] proposed a sequence diagram slicing method to visualize the object oriented program's behavior. In order to achieve this, a tool has been proposed that named 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.

Swain et.al [14] proposed an approach to generate test cases from UML interaction diagram by using the condition slicing. In their approach they identify the message guard condition from interaction diagram and use the condition slicing to generate test cases. J. Kim et.al [15] proposed an approach to address the hierarchy and orthogonality problems while tracing the data dependency in slicing of UML State machine diagram. Yatapana et al. [16] focused their work on Model Checking as fully automated technique to reduce the size of model with the help of slicing. They used Behavior Tree dependency graph (BTDG) to capture all functional requirements and dependency between components and attributes.

Korel et.al [17] dedicated their work on slicing the state based models, such as EFSMs (Extended Finite State Machines). As a result two types of slicing came to existence—deterministic and nondeterministic slicing. Lano [18] defined that slicing can be carried out for UML state machines, using data and control flow analysis by factoring the model on the basis of features. Archer et al. [19] 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. Julliand et.al [20] proposed an approach based on domain abstraction for generating test cases on the basis of syntactic abstraction and variable elimination with the help of model slicing.

Shaikh et.al [21] 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 sub model. Kim [22][23] introduced the slicing technique called dynamic software architecture slicing (DSAS) using ADL (Architecture description language). 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. Lano et al. [24] 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. Zoltán et.al [25][26] proposed dynamic backward slicing of model transformations technique with respect to program slicing. 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. Blouin et.al [27] [28] proposed a DSML (Domain Specific Model Language) 'Kompren' to model the model slicers for particular domain. Kompren refers to the selection of the set of classes and relations from the input metamodel expressed using an object-oriented meta-language. Table 1 depicts the list of model based slicing tools extracted from literature analysis. Table 2 shows the comparison of different approaches of model based slicing.

<table>
<thead>
<tr>
<th>Year</th>
<th>Tool Name</th>
<th>Technique Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>EFSM Slicing Tool</td>
<td>Control and Data flow analysis.</td>
</tr>
<tr>
<td>2007</td>
<td>UTG</td>
<td>Data Flow and Control Flow dependency, Communication Tree.</td>
</tr>
<tr>
<td>2008</td>
<td>SSUAM</td>
<td>Model Dependency Graph.</td>
</tr>
<tr>
<td>2008</td>
<td>UML Slicer</td>
<td>MetaModel Diagram, Key Elements.</td>
</tr>
<tr>
<td>2009</td>
<td>Reticella</td>
<td>B-Model dependency Graph.</td>
</tr>
<tr>
<td>2011</td>
<td>Archlice</td>
<td>Model Dependency Graph.</td>
</tr>
<tr>
<td>2012</td>
<td>UOST</td>
<td>UML + OCL Constraints.</td>
</tr>
</tbody>
</table>
Table 2: Approaches/Techniques/Tools for model based slicing

<table>
<thead>
<tr>
<th>Approach/Technique</th>
<th>Slicing Process</th>
<th>Model</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature model slicing [19, 29]</td>
<td>✓</td>
<td>Feature model</td>
<td>Separation of concerns</td>
</tr>
<tr>
<td>UML slicer [30]</td>
<td>✓</td>
<td>UML metamodel</td>
<td>Modularization</td>
</tr>
<tr>
<td>UML statechart [18, 31]</td>
<td>✓</td>
<td>State charts</td>
<td>Reactive system, Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>checking</td>
</tr>
<tr>
<td>Safe slicer [32]</td>
<td>✓</td>
<td>System Models</td>
<td>Safety</td>
</tr>
<tr>
<td>Domain specific model language [27, 28]</td>
<td>✓</td>
<td>UML metamodel</td>
<td>Dynamic model slicing</td>
</tr>
<tr>
<td>Context free UML slicing [18]</td>
<td>✓</td>
<td>UML class diagram</td>
<td>Sub model extraction</td>
</tr>
<tr>
<td>Dynamic software architecture slicing [22, 23]</td>
<td>✓</td>
<td>Architecture description language</td>
<td>Architectural slicing</td>
</tr>
<tr>
<td>EFSM slicing [17]</td>
<td>✓</td>
<td>State based models</td>
<td>Size reduction</td>
</tr>
<tr>
<td>UML activity diagram [12]</td>
<td>✓</td>
<td>Activity diagram</td>
<td>Test case generation</td>
</tr>
<tr>
<td>UML/OCL slicing [21, 33]</td>
<td>✓</td>
<td>UML/OCL</td>
<td>Verification</td>
</tr>
<tr>
<td>Model transformation [24, 25, 26]</td>
<td>✓</td>
<td>Graph and UML diagram</td>
<td>Program slicing</td>
</tr>
<tr>
<td>Behavior tree slicing [16]</td>
<td>✓</td>
<td>Behavior tree</td>
<td>Model checking</td>
</tr>
</tbody>
</table>

III. Preliminary Work

We had proposed a method for extracting subpart from UML sequence diagram, based on conditional predicate in our previous work titled “Technique for extracting subpart from UML Sequence diagram”. This paper provides the detailed view on the practical implementation of the already proposed technique, for finding the chunk from a given sequence diagram.

IV. Proposed Methodology and Implementation

After reviewing the literature of software testing techniques, slicing techniques, software visualization and unified modeling language, it has been analyzed that slicing UML diagrams is one of the major areas in which work can be extended for various constructs like sequence diagram, state transition diagram, activity diagram, class diagram, etc. It has been thoroughly analyzed that for the process of slicing sequence diagram no consolidate technique have been developed to extract the point of interest from architecture of software to ease the software visualization that uses conditional predicate for finding out a relative slice.

Consider an example UML sequence diagram as shown in figure: 1. The purpose of selecting this example is to demonstrate the concept of proposed methodology. In the example there are four objects which are interacting with each other thorough message passing (using guard condition as true to interact with each other). We illustrate our methodology by explaining the generation of chunk or refined model diagram with respect to slicing criteria. Here in this example let the slicing criterion is variable ‘c’. 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 1 and generate the resultant small chunk or refined sequence diagram.
A practical approach for model based slicing

To extract subpart from UML Sequence diagram following technique has been proposed:

1. Generation of UML (Sequence) diagram of/from a particular requirement specification.
   1.1. Visual paradigm for UML, can be used to generate the UML diagram for specific requirement specifications. Figure 2 shows the designing of UML sequence diagram using visual paradigm tool.

Fig 1: Example Sequence diagram

Fig 2: Designing sequence diagram using visual paradigm

II. Create XML from the specified UML Sequence diagram.

2.1. Visual paradigm for UML 10.0 version provides the in-built functionality to export the diagrams into XML format.
III. Document Object Model (DOM) parser parse the XML code and generate an output file (with .txt extension) having Object name, identifier, message name, message to & fro information. Figure 4 shows the DOM parser and Figure 5 depicts the output file generated by DOM parser after parsing the XML file.

A practical approach for model based slicing
IV. Passing file obtained from step 3 and slicing criteria to a .java program (which act as slicer) for getting the relative/required chunk of information in a separate .txt file.

4.1. Slicer will take .txt file generated in step 3 as input.
4.2. Slicer will ask user to define the slicing criteria at run time to generate the chunk/slice as per specified requirements as shown in figure 6. In this example user define the ‘c’ variable as slicing criteria.
4.3. Computed slices will be store in separate .txt file which holds the information of messages, their guard condition and objects id’s among which messages are being passed as shown in figure 7.

![Java program](image1)

Fig 6: Java program for finding out the specified chunk

<table>
<thead>
<tr>
<th>Build Output</th>
<th>Debug Output</th>
</tr>
</thead>
</table>

![Output file](image2)

Fig 7: output file generated after applying slicing

V. Changing object id with relative object name among which message is passing so that information can be retrieved easily (this step will only deal with sliced part).

5.1. To ease the retrieval of information, objects id’s will be replaced with their corresponding object name (in the file retrieved from step 4.3) (as shown in figure 8.)
5.2. All the information will store in separate .txt file which holds the information of messages and the objects name (among which they are communicating relative to user defined slicing criteria).
VI. Passing txt file as obtained from step 5, to b .java program so that it can be converted into input file format

for ‘Quick Sequence Diagram Editor’ as shown in figure 9.

VII. Tool will generate the final and relatively small sequence diagram.

7.1. Tool will take the input format defined at step 6 as input to convert into its equivalent diagram as shown in figure 10.

7.2. Refined slice (small sequence diagram) will be generated as final output according to slicing criteria as per requirement to ease the software visualization.
VIII. Conclusion

Practical implementation of technique that will extract the sub-model from architecture of software to ease the software visualization has been discussed. The key contribution of the technique is to generate the refined model slices related to slicing criteria using conditional predicate in sequence diagram. The foundation of the proposed technique is ‘UML’ and ‘Slicing’. With this, the problem of visualization of large and complex software can be handled efficiently. The proposed technique has focused on the generation of chunk using model based slicing but still there are the few points that can be explored further like model reduction in concurrent and distributed programming

REFERENCES


