Analyzing the Cohesion and Coupling of State Chart Diagrams using Program Slicing Techniques

Singh Daljeet, Kambow Lavleen
1,2Dept. of CSE, Guru Nanak Dev Engineering College, Ludhiana, Punjab, India

Abstract
The State Chart Diagrams are used for analyzing the dynamic behavior of system. The understandably of State Chart Diagrams are important for the effective and clear communication among developers. In this paper we, propose a cohesion and coupling, as Average Cohesiveness of States (ACOS) and Average number of Similar states (ASSOS), for the evaluation of the understandability of states chart diagrams using program slicing. We had given the new technique of finding the cohesion and coupling of the state chart diagram by building the dependency graph first and then program slicing of the state chart diagrams to get effective results.

Keywords
State Chart Diagrams, Metrics, Cohesion And Coupling, Uml Diagrams

I. Introduction
Now a days, maintainability has become one of the most pressing challenge. It is generally accepted that development field that the maintainability of software is highly dependent on decisions made early in the development life cycle. Conceptual modelling is an important task of this early development. Most of the approaches towards development (Rational Unified Process, OPEN, etc.), have considered conceptual modelling as a relevant task. Therefore, the maintainability of conceptual models has a significant impact on the maintainability Softwares, which is ultimately implemented.

II. A Proposal of Measures for UML State Chart Diagrams
The complexity of a statechart diagram is determined by the different elements that compose it, such as states, transitions, activities, etc. It is not advisable to define a single measure for the complexity of UML statechart diagrams, since a single measure of complexity cannot capture all possible aspects or viewpoints of complexity instead several measures are needed, each one focusing on a different statechart diagram elements. Table 1 outlines the set of metrics we propose to metric of UML state chart diagrams complexity.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Metric definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER OF ENTRY ACTIONS</td>
<td>The total number of entry actions, i.e. the actions performed each time a state is entered.</td>
</tr>
<tr>
<td>NUMBER OF EXIT ACTIONS</td>
<td>The total number of exit actions, i.e. the actions performed each time a state is left.</td>
</tr>
<tr>
<td>NUMBER OF ACTIVITIES</td>
<td>The total number of activities (do/activity) in the statechart diagram.</td>
</tr>
</tbody>
</table>

III. Different States and Scenarios of State Chart Diagrams
First transform State chart diagram into a state chart dependency graph(SDG). This approach is taken from the Monalisa Sharma et al [11] in year 2007 which describes the Ssdg is the set of all nodes representing various states of operation scenarios; each node basically represents an event. Σ sdg is the set of edges representing transitions from one state to another. SDG q0 is the initial node representing a state from which an operation begins. F sdg is the set of final nodes representing states where an operation terminates. To understand a methodology, an operation scenario as a quadruple, aOpnScn: <ScnId; StartState; MessageSet; NextState>. A unique number called ScnID identifies each operation scenario. StartState is a starting point of the ScnId, that is, where a scenario starts. MessageSet denotes the set of all events that occur in an operation scenario. NextState is the state that a system enters after the completion of a scenario. This is to be noted that an SDG has a single start state and one or more end state depending on different operation scenarios.

A. A Distance Based Definition of Complexity Measures for State Chart Diagrams

Fig. 1: The Examples of State Chart Diagram
In this section the distance-based definition of a particular measure NS (number of states) is elaborated in detail. We will follow each of the steps for measure construction proposed in the DISTANCE framework. In order to exemplify the process we will use the state chart diagrams shown in fig. 1.

**B. Dynamic Slicing of State Chart Diagrams**

Makes use of information about a particular execution of a program. A dynamic slice [21], 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. 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, the affected statements in both blocks would be included in the slice. But, in the case of dynamic slicing we consider a particular execution of the program. Sequence diagrams shows the dynamic function and if we make the slices of the sequence dependencies by taking the criteria as the starting and end state of the each traction slices were formulated as the dynamic program slicing. The next step of the thesis will calculate the software metrics as cohesion and coupling in the coming sections.

**C. Empirical Validation of the Proposed Metrics**

Makes use of information about a particular execution of a program. 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. 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, the affected statements in both blocks would be included in the slice. But, in the case of dynamic slicing we consider a particular execution of the program. Sequence diagrams shows the dynamic function and if we make the slices of the sequence dependencies by taking the criteria as the starting and end state of the each traction slices were formulated as the dynamic program slicing. The next step of the thesis will calculate the software metrics as cohesion and coupling in the coming sections.

**D. Metrics Definition**

Based on the UML meta-model, our measurement experience and the more commonly used elements when modelling an UML statechart diagram, we considered the following UML constructs as contributing to the structural complexity.

**E. UML Statechart Diagrams**

1. **Action**

An action is a specification of an executable statement that forms an abstraction of a computational procedure that results in a change in the state of the model, and can be realized by sending a message to an object or modifying a link or a value of an attribute. In a state, we can find several types of actions: entry actions, exit actions and do/Activity actions, i.e., sequences of actions that are executed consecutively while staying in the state.

2. **State**

A state is an abstract meta-class that models a situation during which some invariant condition holds. This invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions such as the process of performing some activity; that is, the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed.

3. **Composite State**

A composite state is a state that contains other states vertices (states, pseudo-states, etc.). The association between the composite and the contained vertices is a composition association. Hence, a state vertex can be a part of at most one composite state.

4. **Simple State**

A simple state is a state that does not have sub-states.

5. **Event**

An event is the specification of a type of observable occurrence. The occurrence that generates an event instance is assumed to take place at an instant in time with no duration. Strictly speaking, the term ‘event’ is used to refer to the type and not to an instance of the type. However, on occasion, where the meaning is clear from the context, the term is also used to refer to an event instance. An event can have the association parameter, which specifies the list of parameters defined for the event.

6. **Guard**

A guard is a boolean expression that is attached to a transition as a finegrained control over its firing. The guard is evaluated when an event instance is dispatched by the state machine. If the guard is true at that time, the transition is enabled, otherwise, it is disabled. Guards should be pure expressions without side effects and have the attribute expression, which is the boolean expression that specifies the guard.

7. **Transition**

A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to a particular event instance. Based on these constructs, we defined a set of metrics for measuring structural complexity. A working hypothesis underlying the metric definition is that the more a particular construct is used when developing a statechart diagram, the more that construct adds to the structural complexity of the diagram. Hence, each metric captures the extent to which a particular construct is used in a diagram.
F. Definition of Average Cohesiveness of States (ACOS)

Average Cohesiveness Of States (ACOS) is a cohesion metric for a state diagram by averaging cohesiveness values of all states. ACOS of a state diagram SD is defined as follows:

\[ \text{ACOS}(SD) = \frac{\sum_{s \in S} \text{COS}(s)}{|S|} \]

where \( S \) is SSD ACOS is defined to have between 0 and 1. ACOS is 1 if cohesion of each state in SD are 1. It means that all states are only one semantic. On the other hand, low ACOS suggests that each state has low cohesion.

G. Definition of Average Number of Similar States of States (ASSOS)

Average number of Similar States Of States (ASSOS), a coupling metric for a state diagram, is defined as the average number of similar states of each state in the state diagram. ASSOS of a state diagram SD can be formulated as follows:

\[ \text{ASSOS}(SD) = \frac{\sum_{s \in S} |SS(s)|}{|S|} \]

where \( S \) is SSD ASSOS can be more than or equal or 0.

H. Calculation of Slice Based Coupling Of State chart

If a state \( s \) has exactly one semantic, the two states split from \( s \) have the same incoming and outgoing transitions. In other words, the two states represent the same semantic if they are linked to transitions with the same event and start/end states. shows the two states \( s_2-1 \) and \( s_2-2 \) split from \( s_1 \). Understandability of state diagrams can worsen if states in the state diagrams have many similar states. Because similar states represent the same semantic as a result of unnecessary partitioning, it can not only distract user’s attention but also increase size and complexity of the state diagram such as SD3. If one semantic is represented by multiple states, users have trouble in understanding the separated states and are likely to make mistakes of analyzing State chart diagrams.

Different slices ASSOS # of similar states of states.

<table>
<thead>
<tr>
<th>Table 2: Cohesion and Coupling Values of State Chart Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State chart diagrams</strong></td>
</tr>
<tr>
<td>1. Number of diagram</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>2. Number of diagram</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

IV. Conclusion

The state chart diagrams of the UML has been taken for calculating the software metrics as the cohesion and coupling various steps are involved in this research method as dependency graph, program slicing and obtaining the results of the cohesion and coupling. By using the program slicing techniques the complexity of the diagram is reduced and becomes easy to trace the problem.

References


Singh Daljeet received his B-Tech degree in Computer Science and Engineering from Punjab Technical University, Jalandhar College, B.C.E.T, Ludhiana, Punjab, India, in 2008, the M-Tech, degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, College, Guru Nanak Dev Engineering College, Ludhian, Punjab, India in year 2012. He is a Assistant Professor at present, with Department of Computer Science and Engineering, in Guru Nanak Dev engineering College. His research interests include Software Engineering, Software Metrics, UML, Object Oriented Paradigm, Object Oriented Metrics. At present, He is engaged in Research of UML diagrams simplifications.