1. APPROACH

We propose a rule based approach to identify and extract CTDM elements from UML activity diagrams. A CTDM parser is implemented that is able to parse the XMI files [19][20] based on the rules formulated and extract the CTDM elements. The sequence of steps in deriving the CTD model from activity diagrams is shown in Fig. 1.

Step 1: INPUT: UML Activity Diagrams Step 2: Generate XMI files

Step 3: CTDM Parser parses the XMI files as per the pre-defined rules.

Step4: OUTPUT: CTDM model

Figure 1. Steps in deriving CTDM from activity diagrams.

1. *Step 1: UML Activity Diagrams*

Activity diagrams are “*graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency”* [Wikipedia]. The activity diagram [16][17] comprises of nodes and edges. The nodes represent processes and process control such as Action state, Start state, Final state, Decision box, Synchronization point, Signal sender and receiver. The edges in activity diagram show which activities may follow one another. The various flows are control flow, Message flow and Signal flows. Swim lanes are used for bringing out the logical grouping of the nodes based on factors such as actors. ArgoUml an open source UML modeling tool is used to draw the activity diagrams [18][19]. Fig.7a shows an example of ATM withdrawal transaction requirements document [23] and Fig.7b, shows the activity diagram representation of Fig.7a. Fig.8 is the manually derived CTDM of Fig.7a in terms of parameters and values.

1. *Step 2: XMI Files*

XML Metadata Interchange XMI is an open standard file format for storing and exchanging the metadata information. The activity diagrams are exported to XMI format using the export feature of ArgoUml. In essence the exported XMI file contains the UML activity diagram information in document form. This XMI file is fed as input to the CTD parser.

1. *Step 3: CTDM Parser*

The CTDM Parser is implemented in python [22] programming language to parse the XMI file. The parser is embedded with rules necessary to identify the CTDM elements (parameters and values). The rules are formulated based on analysis of various activity diagrams.

* 1. *Rule for parameter indetification and extraction.*
     1. *Rule Name:* Rule-Parameter
     2. *Rule definition:* Action state prior to decision box is a parameter.
     3. *Rule description:* The decision box in general has single input and multiple output transitions. The Action state from which the input comes to the decision box holds the parameter, as shown in Fig.2.

In Fig. 7b, the state “choose account type” is an Action state and its outgoing transition leads to the decision box (incoming transition for the decision box is from an Action state). Hence the action state “choose account type” holds the parameter “account”. Accordingly the Action states “Enter saving amount”, Enter checking amount”, “Sufficient amount” and “appropriate Balance” hold the parameters.

* + 1. *Algorithm:* An algorithm to identify and extract the parameters is shown in Fig.3.



***Parameter***

[Guard Condition

Decision Box

[Guard Condition]

[Guard Condition]

***Values***

Action state

Figure 2. Rule description in sample activity diagram.

Step 1: Begin

Step 2: Open the XMI file in read mode

Step 3: Extract the states from all swim lanes using the tag name UML:Pseudostate

Step 4: For each of the UML:Pseudostate

1. Search for Attribute kind = junction,
2. Find its incoming transition using tag name UML:StateVertex.incoming
3. Fetch its source state using tagname UML:Transition.source
4. If source state=UML:ActionState, print the value of attribute “body” which forms

the parameter

1. Repeat for any other UML:Pseudostate and goto step 4

Step 5: End

Figure 3. Algorithm for identifying parameters.

* 1. *Rule for value identification and extraction*
     1. *Rule Name:* Rule-Value
     2. *Rule definition:* Guard conditions on the outgoing transitions from the decision box are values.
     3. *Rule description:* Cosider a decision box, if the incoming transition to it is from an Action state, then the Guard conditions on its outgoing transition are the values as shown in Fig.3. In Fig. 7b, Consider the decision box which has an incoming transition from the Action state “choose account type” and the guard conditions on the outgoing transition of this decision box are “checking account” and “savings account”. These two guard conditions are values.
     4. *Algorithm:* An algorithm to identify and extract the values is shown in Fig.5.
  2. *Association between parameter and values*
     1. *Rule Name:* Rule-Parameter Value Asscoiation
     2. *Rule definition:* The associated parameters and values are directly linked by a decision box.
     3. *Rule description:* The association between parameters and values is required to know as to which parameter the extracted values belong to. The decision box becomes an important factor while linking parameters and their values. The guard conditions on outgoing transitions identified as values belongs to the parameter which is held by the incoming action sate with respect to a decision box as shown in Fig.3. In Fig.7b, the action state “choose account type” holds the parameter and “checking account” and “savings account” are its corresponding values.They both are linked directly by the decision box.
     4. *Algorithm:* The algorithm for association of parameters and values is same as the value extraction algorithm with some minor modifications.The algorithm in Fig.5 extracts and prints the values. In addition, to print the associated parameters, we need to fetch and print the value

Figure 4. Algorithm for identifying values.

Step 1: Begin

Step 2: Open the XMI file in read mode

Step 3: Extract the states from all swim lanes using the tag name UML:Pseudostate

Step 4: For each of the UML:Pseudostate

1. Search for Attribute kind = junction,
2. Find its incoming transition using tag name UML:StateVertex.incoming
3. Fetch its source state using tagname UML:Transition.source
4. Check if source state=UML:ActionState
5. Find the outgoing transition of pseudostate using tagname UML:StateVertex.outgoing
6. Print the values of the tagname

UML: Transition.guard which forms the Values

1. Repeat for any other UML:Pseudostate and goto step 4

Step 5: End

* 1. *Rule for identifcation of constraints.*
     1. *Rule Name:*Rule-Constraint
     2. *Rule definition:* The parameter-value combination leading to an end state through an error condition and its corresponding parent parameter-value combination are involved in a constraint.
     3. *Rule description*: Fig.9 is part of the withdraw activity diagram focussing on the constraints part. As shown in Fig.9, if the parent parameter(account type) takes a particular value(savings account) then the child parameter (amount) cannot take the value (amt > 10000). If it takes, the flow leads to an end state through an error condition (amount exceeds).

Step 1: Begin

Step 2: Search for an end state that was lead by an error condition.

Step 3: Start scanning backward from the end state.

Step 4: Check if the parameter-value combination that lead to the end state through an error condition is preceded by another parent parameter-value combination.

Step 5: Extract these child and parent parameters and their values using the parameter, value and association algorithms.

Step 6: End

* + 1. *Algorithm:* An algorithm to identify the constraints is shown in Fig.5.

Figure 5. Algorithm for identifying constraints.