A software that transfer UPPAAL mode to Yakindu mode

Shuo Yan

***Abstract*—This paper describe a software that convert UPPAAL mode to Yakindu mode. This software is used after construction the model by UPPAAL and then convert it to Yakindu mode. Uppaal is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automate, extended with data types (bounded integers, arrays, etc.). Meanwhile, Yakindu is another software that used mostly for modeling and integrated tool environment for Editing,Validation,Simulation and Code Generation. Therefore, as students, we thought it would be beneficial study how to transfer from Uppaal to Yakindu as we need using UPPAAL to verify the model and using Yakindu to generate the JAVA code. We were able to do below attributes. First, we create a UI interface to load the Yakindu/Uppaal xml file and then show it on the window. Then, transfer the mode to Yadindu mode. Lastly, we show the transfer result on the UI interface.**

***Index Terms*—Uppaal, Yakindu, Conversion.**

I. Introduction

YAKINDU Statechart Tool is a open source toolkit provides an integrated modeling environment for the specification and development of reactive, event-driven systems based on the concept of state charts.What makes Yakindu state of popular even today is its a open source project and its kinds of UI interface.In addition, it is based on the open source development platform EclipseBut Yakindu do not integrated a model checker function like UPPAAL or an importer for state flow.So we need to use UPPAAL to realize the model checker function. After checking model, we need to transfer the xml file from UPPAAL to Yakindu to generate JAVA code. In addition, the tool can realize this feature.

II. File format

UPPAAL supports three file formats for models: XML, XTA and TA. XML and XTA files can be loaded and stored via the Open Project, Open System, Save System, and Save System As menus. Yakindu file saved as a sct file but that is also a xml file. It is important to note that in general, each application would generate some xml content which can be used by themselves. For Yakindu, the content block ”NOTATION:Diagram” is specification for Yakindu. One of the challenges of working with this particular dataset is the data type requirement between the two software. Since there are different organization architecture and the software specification data. To cope with this, our implementations modify or add some contents for unmatched data when transfering. In the end, we reconstruct a *.sct file for Yakindu application.

III. Background

In this section, we provide a brief review of the software realize process. Fig.1 shows the features and their relation to each other for Yakindu application. We can see that Yakindu include the feature - Code Generation and that is special in Yakindu. So that is the reason that we need the application to convert Uppaal model to Yakindu Model.In the framework, a practitioner first load XML file into DOM and then parse it. Then, the parsed data are fed into a waterfall type data structure to save those data. Finally, the data structure will reorganized with the required type of Yakindu. Finally, add the Yakindu ordered data type and data to construct an State chart Tools (SCT) file. However, due to the time constraint, we are unable to construct the ”‘notation:Diagram”’ area since that is special ordered by Yakindu.

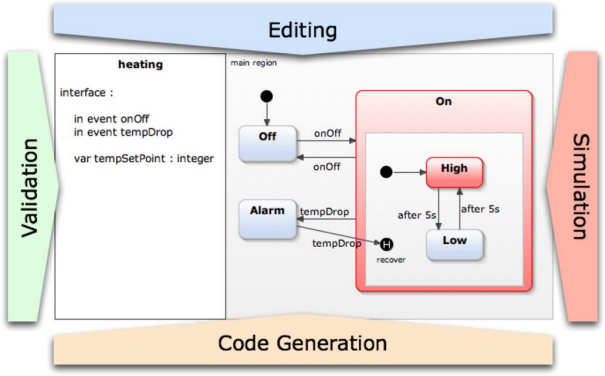

Fig. 1. features and their relation to each other

The YAKINDU SCT meta model is designed based on the UML state chart model and below is the component for it.

1. Region YAKINDU state are organized in regions, Due to this it is possible to organize multiple state machines in different regions and to run them concurrently. In the SCT xml file, it is represented in the region of ¡region¿ with the region id and region name.

2. State That is the central elements of a state machine, seems like a node in the graphic chart. It is used to describe one statement that appears in the region. The name should be unique inside the region. In the SCT file, it is represented in the region of ¡vertices¿ with a state type and state id.

3. Transition A transition is the transfer of one state to another. In the graphic chart, it is shows as arrows and can carry events and actions. In the SCT file, it is represented int the region of ¡vertices¿ with the ’incomingTransition’ and

'outgoingTransitions'. Meanwhile, the event that transition is carried is represented by the 'specification' area.

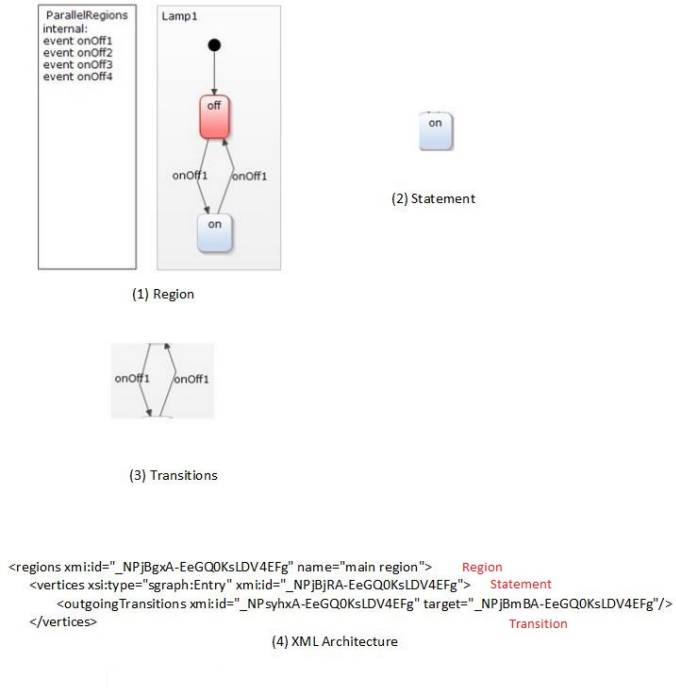

Fig. 2. Yakindu Statechart elements

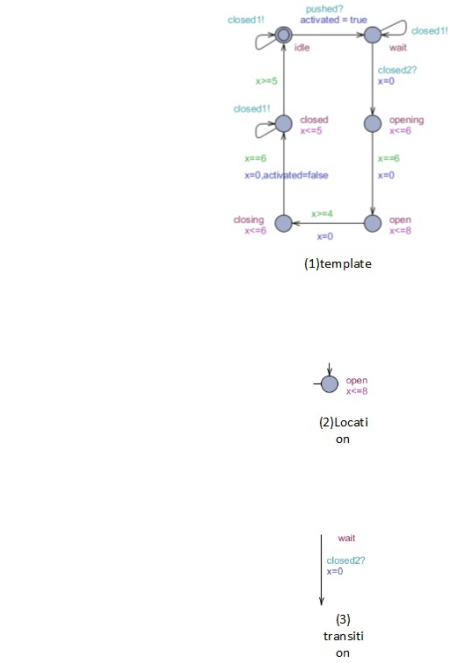

Fig. 3. uppaal Statechart elements

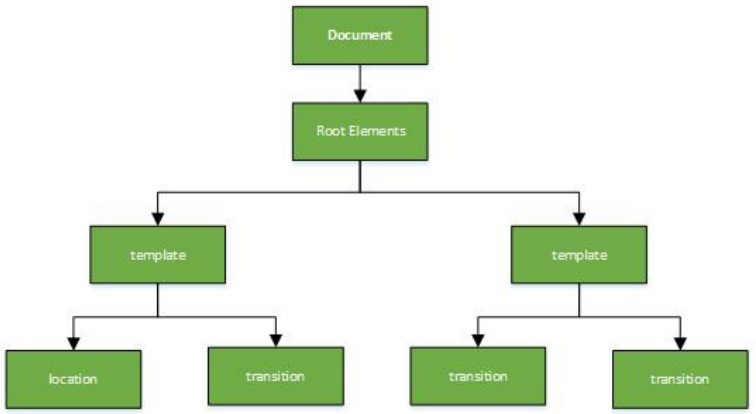

Fig. 4. Parsed DOM Tree

Fig3 is the corresponding component for UPPAAL

1.Template This region are defined with a set of parameters that can be of any type. Meanwhile, it also notice the region name. The corresponding name in YAKINDU is 'region'

2.Location That is the central elements of a state machine and describe the status that appears in the region. The corresponding name in YAKINDU is 'status'

3.Transition That is same as the region in YAKINDU.

With Fig 2 and Fig3, components in different model are similar and we can transfer the corresponding component to another model. While, some component cannot be transferred from UPPAAL to Yakindu as UPPAAL do not have that component. such as 'notation:Diagram' and 'interface'

IV. Implementation Details

A. loading XML file into DOM and parse it

After selecting the source file, all data will be accessed and updated by DOM. Document Object Model(DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update and style of a document. DOM defines a standard for accessing documents like XML and HTML. After parse it, we will generate the tree like below.

After loading the document by DOM, we can use below command to load each parameter which we want. We need to using below command to parse wanted parameter: document.getElementsByTagName("h1")

Parameter that need to parse:

1) declaration - That is used to parse the parameter area and data will be saved into the list of declarationList.

2) template - That means the region area. This will parse how many regions this file have. and then each the region will saved into each 'Automata' data structure and

3) location - The status id, name will be parsed and saved into the list of stateList

4) transition - The status transition will be parsed and saved into the list of transitionList

In order to store those parsed data, we create a data structure. the structure automata include three lists, State List, Transition List, Declaration List. Meanwhile, if there are two regions in this chart, we will have two automata structure and each structure have same architecture.

B. Reconstruction

1) matched component between two model: Once the xml file have been parsed, it i time to handle these data. Function "write" in YakinduWriter.java take the parsed parameter

parsed from xml file, and follow the format of Yakindu to reconstruction those collected data to match the required format of Yakindu. The reason to reconstruct is the format is different.Such as the transaction region, Fig 6 shows this.

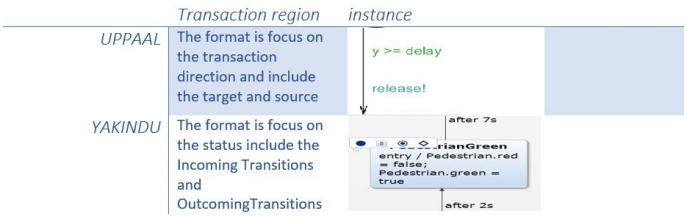

Fig. 5. format different

The algorithm we used is that

1.Fetch each transaction data from the list transition and then using the target id to find out the outgoingTransitions of status node.

2.Using the source id to find out the ingoingTransitions of the source status node.

Fig.6 shows the detail process

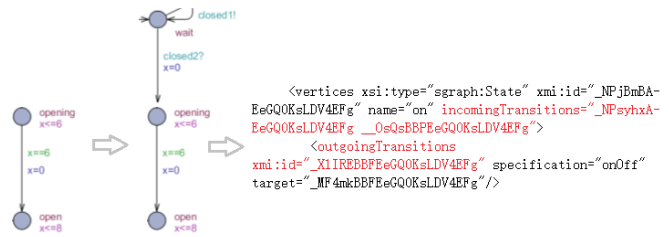

Fig. 6. Reconstruction

2) unmatched architecture between two model: It is not enough to convert the matched component from UPPAAL to Yakindu as it required some special component such as ¡notation:Diagram¿...¡/notation:Diagram¿ which uppal do not have. So we need to analysis how to generate it. But that is different as we need to analysis the source code of Yakindu to find out the algorithm. That is what we will do next.

V. Results

A. UI Interface

For the purpose of loading the UPPAAL and Yakindu xml file, we design a UI Interface to operate each action instead by the button.

As show in Fig. 7, there are 3 main components, the left text window shows the untransformed file and the right window shows the converted file content. There are 5 buttons include as Open File, Save target file, Load Model Application and UPPAAL to Yakindu/Yakindu to UPPAAL conversion.

B. Experimentation

we parsed the xml and reconstruct it to the format that Yakindu required. Unfortunately, one of the segment cannot

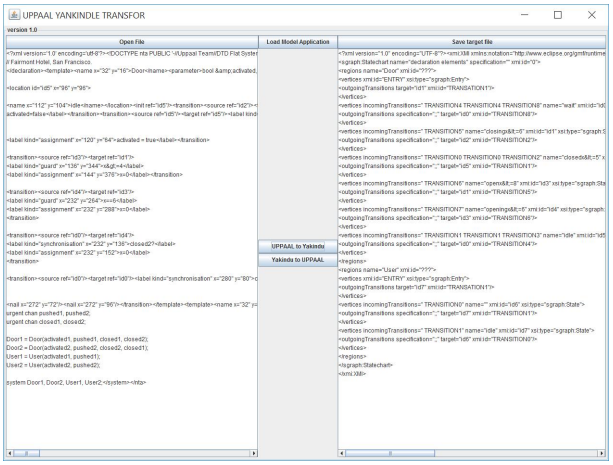

Fig. 7. UI interface.

transfer from Uppaal as Uppaal do not have that apartment. While, we will find another solution such as analysis Yakindu source code to find out the solution.

VI. Conclusion

I have learned about very much from this project. Swing is a GUI widget toolkit for Java. It is an API for providing a graphical user interface (GUI) for Java programs. We use it to design our UI interface. Firstly, we design to load Yakindu or UPPAAL UI interface within our application and we can edit the chart and save it directly. But as time is limited, we have not realized this function. We also learned the format of xml file which is used for Uppaal and Yakindu model. The archicture of xml likes a waterfall model. There include a root element, and the child of root element is the region area. As we know, YAKINDU/Uppaal state are organized in regions. So the region indicate the group of the status that have same attribute. While, transaction are organized in status. So that seems like the waterfall as each component is include its child. We use DOM to parse the xml file. It can parse the parameter by the Tag name. The result include the subtag also include in it. Yakindu have its special order context and Uppaal do not have. So that introduce the difficult to convert Uppaal model to Yakindu module we are under checking that issue now.

Appendix A
References

@article title=http://www.uppaal.org/ http://statecharts.org/,