Rhapsody Tutorial

This tutorial gives a step-by-step demonstration on how to build the microwave oven example as given in the *demo.zip* using "Rhapsody in J" 6.2. It only covers limited features of the Rhapsody tool that might be helpful for your assignment 1. To learn/use more features of the tool, please go for the Rhapsody tutorials at "Help / List of Books".

1 Modeling Procedures

The process of constructing a system includes four steps - analyze, design, build and animate(simulate).

1. Analyze
   Before modeling the system, we first need to analyze the system specification to identify the components of the systems and their behaviors(functionalities).

   For example, our microwave oven contains three buttons for three different heating options (heat for 1000 time unit, heat 500 time unit, or stop current heating), as well as one engine which does the actually heating.

2. Design
   To design this system using Rhapsody, we need following steps.

   - Create a new project by selecting File / New
   - Create class diagram: expand "Object Model Diagrams" on the left panel, select the default diagram "Model1" (you may choose to rename it). To create a class, click the "class" icon on the middle toolbar, then click on the drawing area at the right. By double click a class you have created, the class property window appears, from which you can change the class name, add attributes and operations. Similarly, you can create and configure associations

between classes. Note that when an object of class A needs to communicate with another object of class B, class A and class B must have association in the class diagram.

In our example, we create two classes, “Button” and “Engine”. We have an attribute named “period” in the “Button” class, to capture the heating period of the three buttons (0 for stop). An constructor “Button(int p)” is created to initialize the value of “period”. Moreover, a directed association from “Button” to “Engine” (named “itsEngine”) is created so that we can send messages from “Button” objects to the engine.

- Create object diagram: We first create a new object model diagram by right click “Object Model Diagrams” and choose “Add New Object Model Diagram”. In the object diagram, we create three buttons objects (longButton, shortButton, and stopButton), one engine, and links (instances of associations) from each button to the engine. We can initialize the value of “period” for each button object in the object diagram. To do so, double click a button object, click “initialization” under “general” tab, set values of the parameters of “Button” constructor.
- Create statecharts: create a statechart for a class by right clicking the class in the class diagram, then choose Add new statechart.
  - States: Simple, Or, and And states. States may have action on entry/exit which are executed when the states are entered/exited, respectively.
  - Transitions: Upon receiving an event, an object changes its state by executing a transition. A transition includes the trigger event, a guard (boolean expression on local variables) and action. Action may involve any Java statements or event sending to another objects. For example, when the “stopButton” is pressed (receives an external event “press”), it will send an “stop” event to the engine by calling “getItsEngine().gen(new stop())”.
  - Reception: Events which are used as the communication mechanisms among objects. When you type a event name in the “trigger” field of a transition, Rhapsody will automatically create such an event for you. In the browser / package / Events, you can add or modify the events used in the package. Events can carry data with them. To make an event have data associated with it, right click the event, choose Features,

and add parameters and their types corresponding to the data you want to associate with the event.

3. Build
   Once you have created all the necessary diagrams for the model, you need to generate and build java code for the model. This is the preparation step before doing animation to debug a model at design level. These are the steps:

   - Create a component
     A component contains the configuration used to generate, build and run the model. You can use the default component (in the Components category in the browser) and rename it if you want. Set the component features as follows:
     - Type: Executable
     - Scope: All Elements
   - Create a configuration for the component
     Each component contains a default configuration named "Default-Config". Expand the component in the browser to see it. Double click the "DefaultConfig" under it, set the configuration as follows.
     - Initialization tab, initial instances: explicit
     - Settings tab, Instrumentation mode: animation. Time model: Real. Statechart implementation: flat.
   - Generate code
     Set the active configuration for which you generate code by right-clicking the configuration, select Set as Active Configuration. From the menu, select Code / Generate / <active configuration name>.
   - Build code
     Select Build from menu Code or Make tool in the Code toolbar. Debug the compilation error if any.

4. Animate
   When doing animation, you can open animated diagrams to observe the model running. First create a new sequence diagram by click "Tools / Sequence Diagram". In the diagram, draw the "instance line" for each object you want to monitor during animation. Draw a "system border" to indicate where the external events come from. Then start to run your model by click "Code / Run". Please refer to "Rhapsody in J" tutorial, page 65-71 for details of how to run the model and generate external events (inputs) to the model.