

Homework #4 State Chart Diagrams

1. Creating a state chart diagram in Enterprise Architect (EA)

A State chart or State Machine diagrams illustrate how an element can move between states, classifying its behavior according to transition triggers and constraining guards. In this tutorial we will create a new state chart diagram using EA as the following steps.

1. From the Enterprise architect browser window, right click on the **Restaurant package** created earlier.

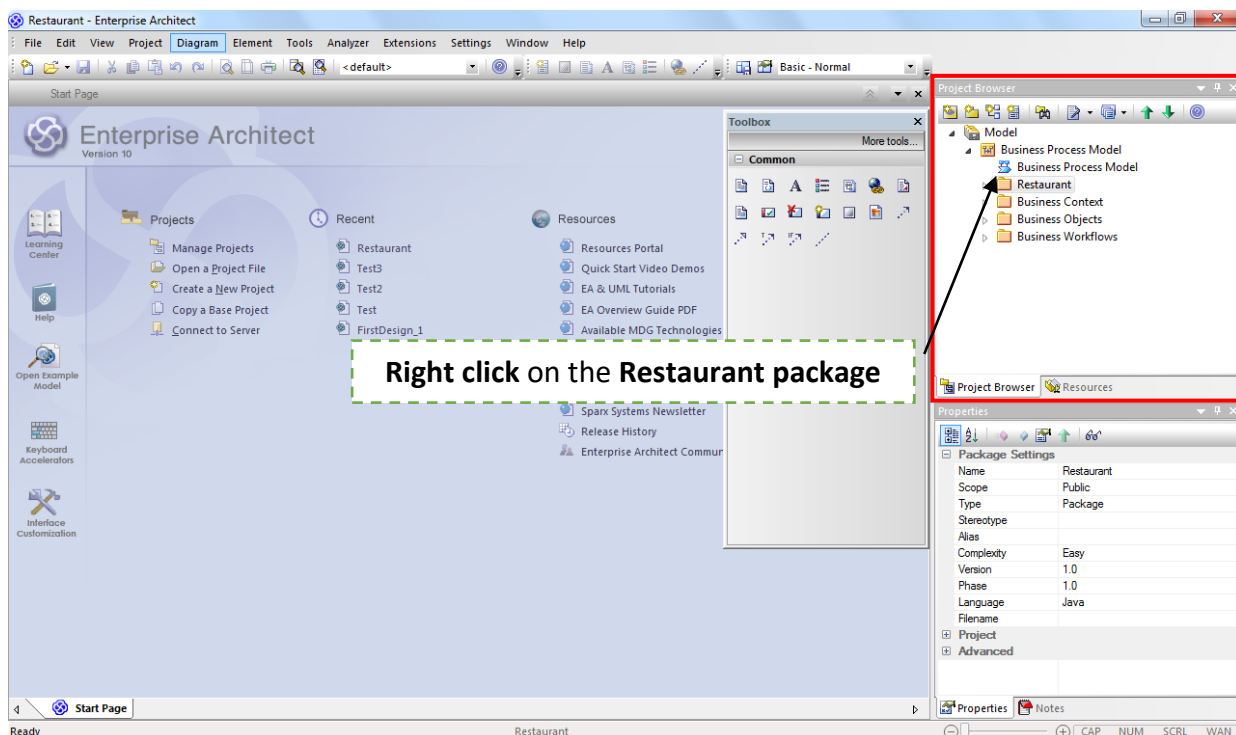



Figure 1: Enterprise architect browser window

2. Select **"Add > Add Diagrams..."**. or click at  icon on Project Browser.

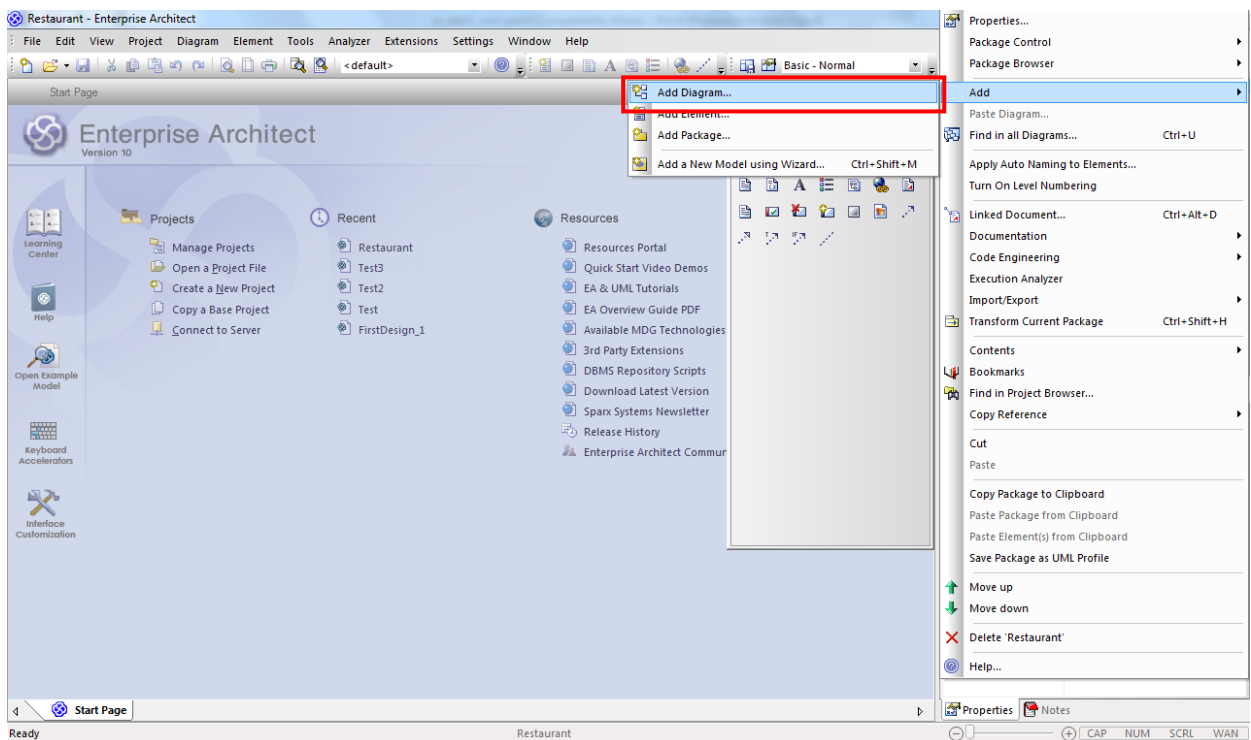


Figure 2: Add New State chart diagram from **Add Diagram...** menu

3. Choose **"UML Behavioral -> State Machine"** on **"New Diagram"** window.

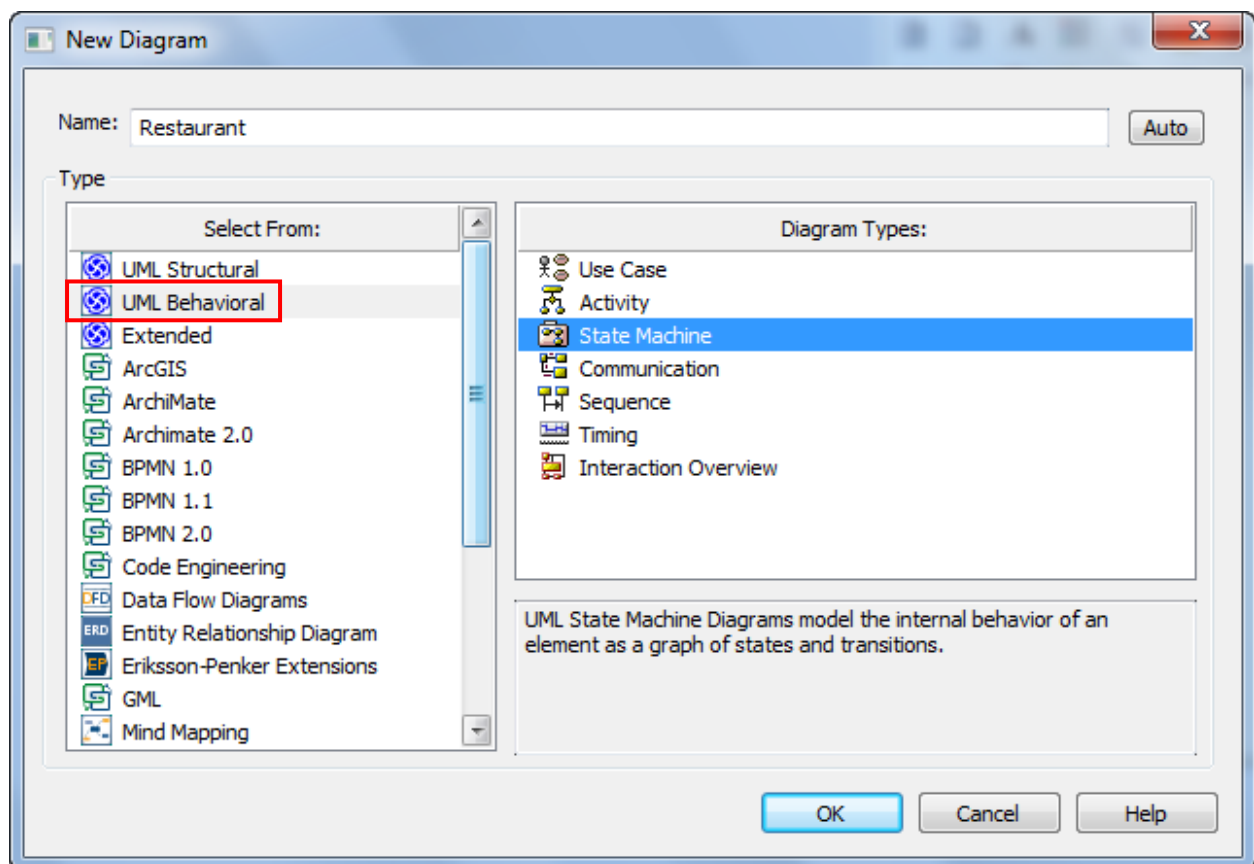


Figure 3: Select State Machine on Diagram Types.

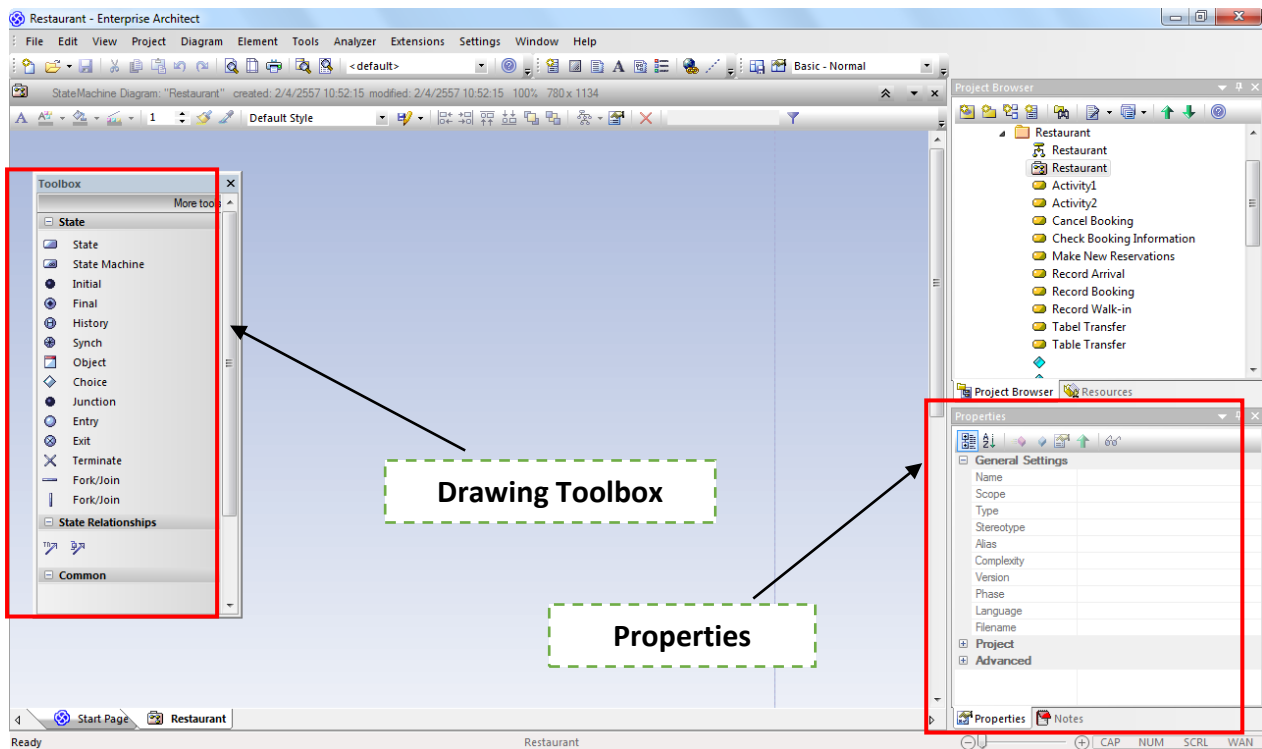





Figure 4: Drawing Toolbox component of Sate Machine

4. After creating a state chart diagram, you can right click on it and select **“Properties...”** to change the diagram features such as name, description, and properties.
5. To start adding components into the state chart diagram, if the drawing toolbar is not already shown, open the drawing toolbar by press **"Alt+5"**. You can select a component, from the diagram tools in the drawing toolbar, to add into the state chart diagram.
6. Right-click on a component in the diagram and then select **“Appearance...”** from the menu if you want to customize the component appearance and select **“Properties...”** to add to it any further detail description. You can select a component, from the diagram tools in the drawing toolbar, to add into the statechart.
7. Use the Initial tool  **Initial** to create an Initial Pseudo state. Click OK in the State dialog.
8. Use the State tool  **State** to create States named “Created”, “Identifying” and “Moving” respectively.
9. Use the Transition tool  to add Transitions from the Initial state to the Created state, from “Created” to “Identifying”, from “Identifying” to “Moving”, and from “Moving” to itself (click and release within Moving). Arrange your diagram to look like this:

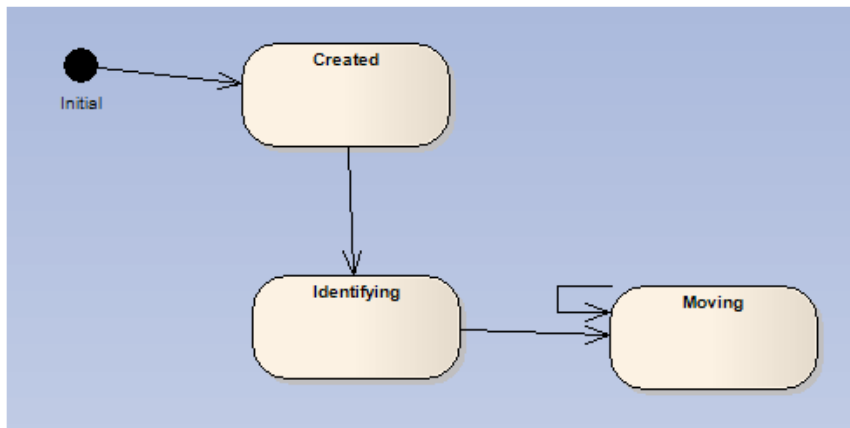


Figure 5

10. Select the self Transition (transition to itself) of the Moving state, right click and select Line Style --> Bezier.

Note : For more details about this , please follow the link below.

<http://www.sparxsystems.com/resources/index.html>

- Draw the following state chart for the “BookingSystem” class:

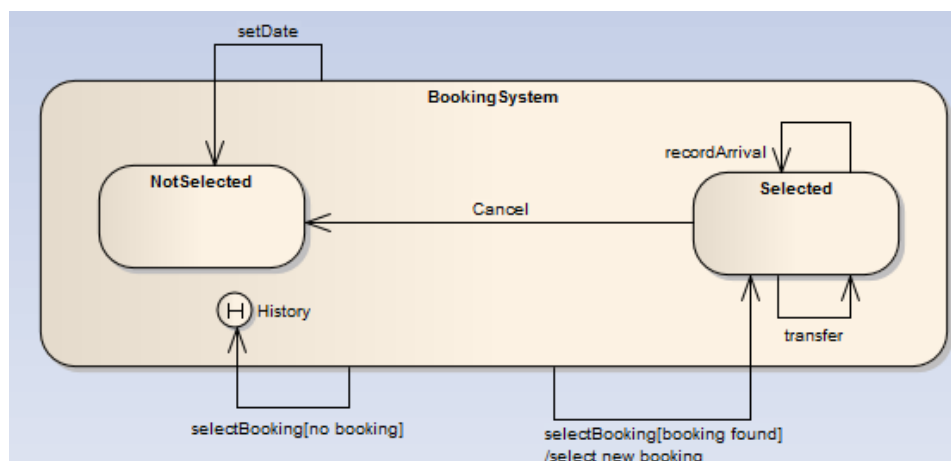


Figure 6

- Draw the following state chart for the “Reservation” class:

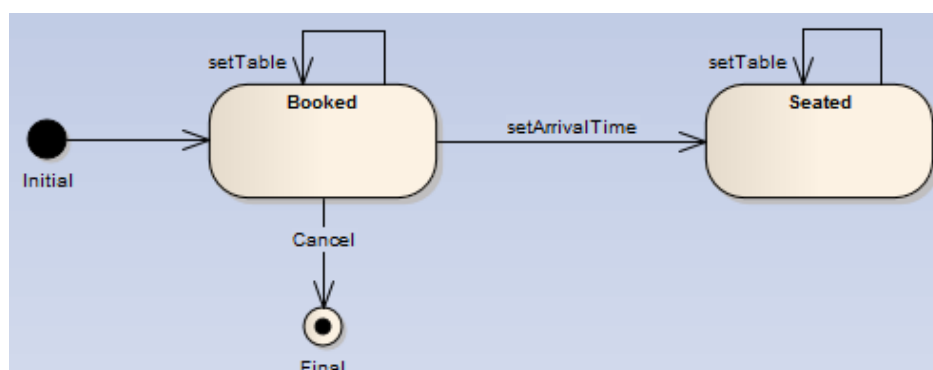


Figure 7

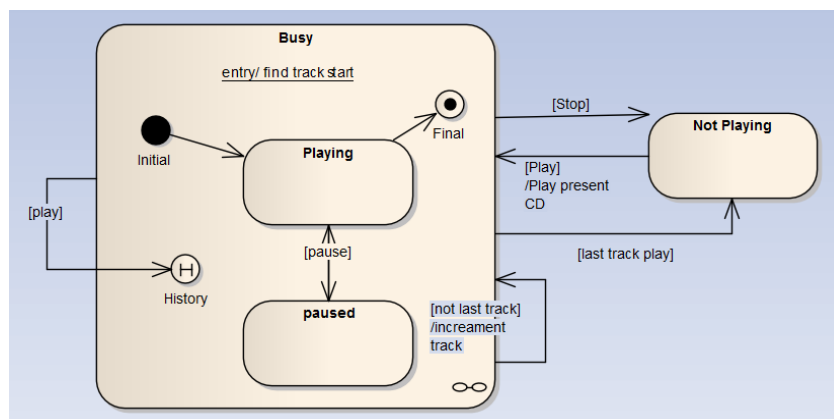
When Not to Draw a State chart

It is not necessary to draw a state chart for every class in a system. Normally, state charts are only drawn for classes with ‘interesting’ behavior: typically, these will be classes that expect to receive messages in a certain fixed order or which exhibit state-dependent behavior, responding to the same message in different ways at different times.

For example, it might seem that a state chart could be drawn for the customer class, to show, for example, a distinction between a customer who has a reservation made for some time in the future and one who hasn’t. The difference like this between customers can certainly be identified, but from the perspective of the system it is entirely irrelevant. The operations in the interface of the customer class can be called at any time, and have the same effect at all times. A state chart for the customer class specifying this would therefore not be a useful addition to the system design documentation.

Exercises

1. Create the following statechart for showing the CD player behavior using Enterprise Architect:



2. Use Enterprise Architect to create a statechart for the “BookingSystem” class that is equivalent to the statechart shown in Figure 6, but does not use composite states.

3. To refer to your Java group project, for each member create a **different** statechart for an important behavior of some classes in your system design.