Installing the Roads Source Code (Eclipse IDE)

The ROADS framework uses PRISM to model check Discrete-Time Markov Chains modelling human agents completing tasks during disaster recovery.

Download and install the Prism Model Checker from: www.prismmodelchecker.org
Note Prism's lib directory e.g. PRISMLIB = /Users/kjohnson/prism-4.5-osx64/lib

Download the Eclipse IDE from www.eclipse.org/downloads

Download the JGraphT Java Graph Library from: jgrapht.org
Note the install directory e.g. GRAPHLIB = /Users/kjohnson/jgrapht-1.3.0/lib

Roads Source Code Download contains source code, scenario output files and scripts to obtain model checking results for each scenario

1. In Eclipse, create a new workspace. Open the Roads project via the menu:
   1.1. Select **File** -> **Import...** Under the **General** folder
   1.2. Select **Existing Projects into Workspace** and click **Next**.
   1.3. Click on **Select archive file** and choose roads-tool.zip source code archive.
   1.4. Click **Finish**. A new java project called Roads is created.

2. In the Eclipse Project Explorer Right click the Roads project and select **Properties** from the menu.
   2.1. Select **Java Build Path** from the list and select the **Libraries** tab.
   2.2. Select **Classpath** and click the **button Add External JARs…**
   2.3. Navigate to PRISMLIB and select the file prism.jar. Click **Open**.
   2.4. Navigate to GRAPHLIB and select the file jgrapht-core-1.3.0.jar. Click **Open**.
   2.5. Click **Apply and Close**.

Analysing Tasks on a Socio-Cyber-Physical System:

3. In the Eclipse Project Explorer Right click the Roads project and select **Run As** -> **Run Configuration…** from the menu.
   3.1. From the list, right Click on **Java Application** and select **New Configuration**
   3.2. In the **Main Class** field type: clients.RoadsTest
   3.3. Click the Environment tab and click the New… button
   3.4. Add the Environment Variable with name DYLD_LIBRARY_PATH and value PRISMLIB (the directory of your Prism library installation)
   3.5. Click **Apply**
   3.6. Click **Run**

The console outputs the transcript below.

```
Number of input graph vertices: 6
Number of input graph edges: 18
Deliver (All Paths) Scenario
a -> b -> c -> f  0.70162  8.6714  20  28
```

a -> b -> c -> d -> f 0.74106 10.279 21 30
a -> b -> c -> e -> f 0.74870 12.682 21 30
a -> c -> f 0.81310 11.672 19 26
a -> c -> d -> f 0.85882 13.535 20 28
a -> c -> e -> f 0.86767 16.320 20 28
a -> d -> f 0.88538 11.821 19 26
a -> d -> c -> f 0.80506 21.387 20 28
a -> d -> c -> e -> f 0.85909 25.989 21 30
554ms

Emergency All Paths Scenario
e -> c -> d 0.93178 9.7812 12 16
e -> c -> f -> d 0.83004 13.371 13 18
e -> c -> b -> a -> d 0.77990 14.043 14 20
e -> c -> a -> d 0.90383 17.281 13 18
e -> f -> d 0.92227 10.831 12 16
e -> f -> c -> d 0.83860 19.950 13 18
e -> f -> c -> b -> a -> d 0.70191 23.786 15 22
e -> f -> c -> a -> d 0.81344 26.700 14 20
167ms

Deliver Parametric (Shortest Path) Scenario

5ms