Appendix E: Developing a new module

The module interface should be constructed in the format below:

```
/**
 * Description:  PetriNet Analysis/Editor module
 * interface
 *
 * Note: The Module should have a blank constructor for
 * dynamic loading which will make an instance of the
 * class
 * the only way to get information from application is to
 * access through the runModule method.
 */

public interface Module {
// get Data from a JDOM Document, and get the analysis
result.
// the ModuleBridge class is used to return the analysis
back to the editor
public void runModule(Document doc, ModuleBridge bridge);

// return the ModuleName.
public String getModuleName();

// if the module can deal with a subnet, return "true",
otherwise return "false".
  public String isSubnetable();
}
```

The following is the code for the ModuleBridge class allowing the modules to pass information back to the application:

Bridge class allowing Modules to pass instructions back to the main program, for example to display a sequence of transitions firing.

Initialise the ModuleBridge with a reference to the active DisplayPane.
@param display a reference to the active DisplayPane which holds the Document which the module is working on.

```
public ModuleBridge(DisplayPane display) {

}
```

Signal the program to re-initialise the display from the current state of the DOM. Assumes that the type of net has not changed.

```
public void reloadDOM() {
}
```

Request the program to highlight the object, given its id.
All other objects will be un-highlighted.
@param id String holding the id of the object to be highlighted.

```
public void highlight(String id) {
}
```

Request the program to un-highlight all objects.

```
public void unHighlight() {
}
```

Request the program to animate a sequence of transitions, given a list of the transition ids.
@param trans_ids An array of transition ids, in order of firing
@param length The number of transitions in the sequence

```
public void animate(String[] trans_ids, int length) {
}
```