Sam Dembinski
Zach Taylor
Kaoutar Maghfour
Nathan Spaeth

Use Cases

Use Case: CreateDiagram
Participating Actors: User
Priority Level: High
Flow of Events:

1. The user runs the application.
2. A movable start and end point are automatically generated in the upper left and lower right corners of the screen, respectively.
3. The user optionally edits and saves the diagram as they wish (See EditDiagram and SaveDiagram use cases).

Entry Conditions:

- The program has been executed at some point

Exit Conditions:

- A diagram exists, with a start and end point.

Use Case: EditDiagram
Participating Actors: User
Priority Level: Medium
Flow of Events:

1. The user selects an edit option.
2. The edit option is performed on the diagram.
3. The edit options are as follow: delete, creating state figure, create nested state figure, draw transition.

Entry Condition:

- The diagram exists.

Exit Condition:

- The new changes are shown on the diagram.

Use Case: CheckForErrors
Participating Actors: User
Priority Level: Low
Flow of Events:

1. The user selects the check for errors option.
2. A pop-up window tells the user if their UML Diagram has errors or not.

Entry Condition:

- None.

Exit Condition:

- A pop-up report has been displayed

Use Case: SaveDiagram
Priority Level: High
Participating Actors: User
Flow of Events:

1. User selects the "Save" option.
2. If there is no path associated with the diagram, the user is prompted for a path.
3. The file is saved to the specified path.
4. The user is returned to the diagram view.

Entry Conditions:

- There is a diagram open.

Exit Conditions:

- The diagram has been saved to a file.
- The diagram has not been altered.

Use Case: OpenDiagram
Participating Actors: User
Priority Level: High
Flow of Events:

1. The user selects "open" option.
2. The user navigates to a predetermined file location and selects it.
3. The diagram is displayed to the user.

Entry Condition:

- The file exists.

Exit Condition:

- The user sees the diagram

Use Case: GenerateXML
Priority Level: Low
Participating Actors: User
Flow of Events:

1. User selects the "Export XML" option.
2. If there is no XML path associated with the current diagram, the user is prompted to specify a path different from the file path of the diagram.
3. The file is saved to the newly specified path.
4. The user is returned to the diagram view.

Entry Conditions:

- There is a diagram currently open.

Exit Conditions:

- The diagram has been saved in XML format to the specified path.
- The diagram has not been altered.

Use Case: DrawnDiagramSimulation
Participating Actors: User
Priority Level: Low
Flow of Events:

1. The user selects the "Simulate" option.
2. A text file is generated based off the currently open diagram listed each state, each state transition, and the effects after each transition.

Entry Conditions:

- A diagram is currently open in the program

Exit Conditions:

- A text file describing the states, transitions, and effects of each transition exists.
- The diagram remains open.

Use Case: XMLSimulation
Participating Actors: User
Priority Level: Low
Flow of Events:

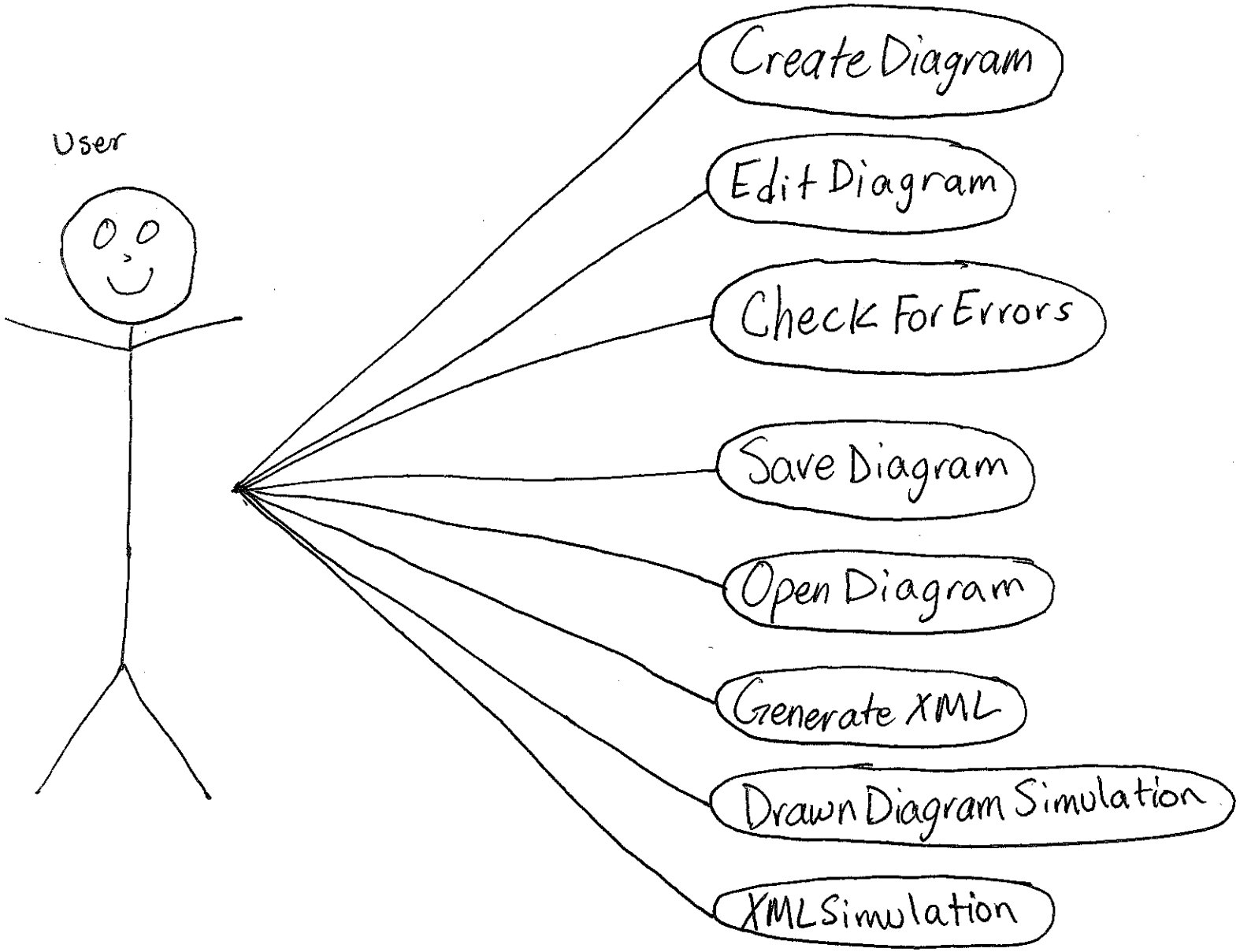
1. The user selects the "SimulateXML" option.
2. A window opens allowing the user to select a file location.
3. The user navigates to an already existing XML representation of a diagram
4. The user is prompted for an input file to be executed
5. The user navigates to an already existing text file
6. A text file is generated based off the XML representation listing the effects of each transition.

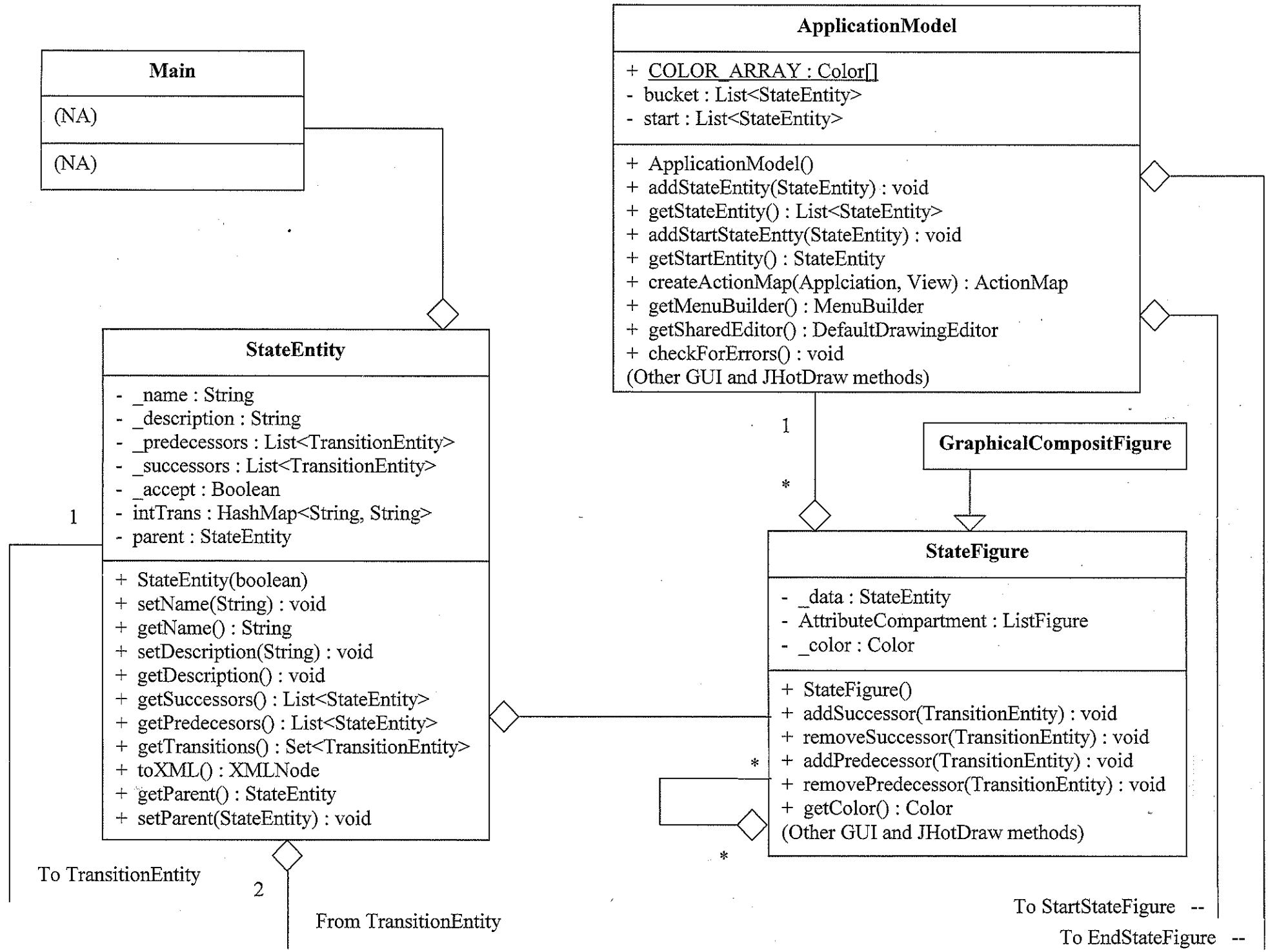
Entry Conditions:

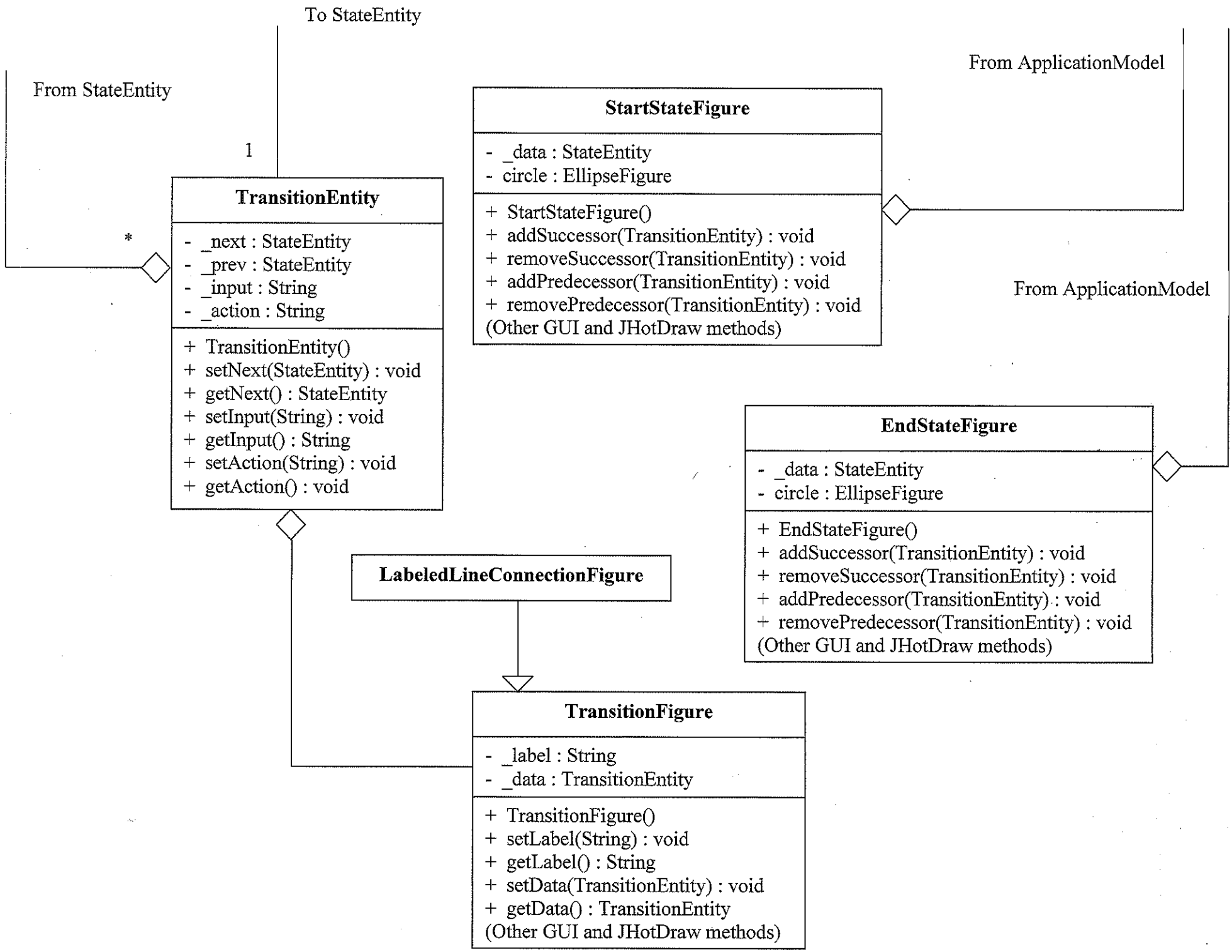
- An XML representation of a UML state diagram exists.
- The user must generate an XML representation of a particular diagram if one does not already exist for it.

Exit Conditions:

- A text file describing the states, transitions, and effects of each transition exists.
- The program remains open.


User
Create Diagram
Edit Diagram
Check For Errors
Save Diagram
Open Diagram
Generate XML
Drawn Diagram Simulation
XML Simulation


Main
(NA)
(NA)
ApplicationModel
+ COLOR_ARRAY : Color[]
- bucket : List<StateEntity>
- start : List<StateEntity>
+ ApplicationModel()
+ addStateEntity(StateEntity) : void
+ getStateEntity() : List<StateEntity>
+ addStartStateEntty(StateEntity) : void
+ getStartEntity() : StateEntity
+ createActionMap(Applciation, View) : ActionMap
+ getMenuBuilder() : MenuBuilder
+ getSharedEditor() : DefaultDrawingEditor
+ checkForErrors() : void
(Other GUI and JHotDraw methods)
StateEntity
- _name : String
- _description : String
- _predecessors : List<TransitionEntity>
- _successors : List<TransitionEntity>
- _accept : Boolean
- intTrans : HashMap<String, String>
- parent : StateEntity
+ StateEntity(boolean)
+ setName(String) : void
+ getName() : String
+ setDescription(String) : void
+ getDescription() : void
+ getSuccessors() : List<StateEntity>
+ getPredecesors() : List<StateEntity>
+ getTransitions() : Set<TransitionEntity>
+ toXML() : XMLNode
+ getParent() : StateEntity
+ setParent(StateEntity) : void
GraphicalCompositFigure
StateFigure
- _data : StateEntity
- AttributeCompartment : ListFigure
- _color : Color
+ StateFigure()
+ addSuccessor(TransitionEntity) : void
+ removeSuccessor(TransitionEntity) : void
+ addPredecessor(TransitionEntity) : void
+ removePredecessor(TransitionEntity) : void
+ getColor() : Color
(Other GUI and JHotDraw methods)
1
*
1
*
*
2
To TransitionEntity
From TransitionEntity
To StartStateFigure --
To EndStateFigure --


To StateEntity
From StateEntity
From ApplicationModel
1
*
TransitionEntity
- _next : StateEntity
- _prev : StateEntity
- _input : String
- _action : String
+ TransitionEntity()
+ setNext(StateEntity) : void
+ getNext() : StateEntity
+ setInput(String) : void
+ getInput() : String
+ setAction(String) : void
+ getAction() : void
StartStateFigure
- _data : StateEntity
- circle : EllipseFigure
+ StartStateFigure()
+ addSuccessor(TransitionEntity) : void
+ removeSuccessor(TransitionEntity) : void
+ addPredecessor(TransitionEntity) : void
+ removePredecessor(TransitionEntity) : void
(Other GUI and JHotDraw methods)
From ApplicationModel
EndStateFigure
- _data : StateEntity
- circle : EllipseFigure
+ EndStateFigure()
+ addSuccessor(TransitionEntity) : void
+ removeSuccessor(TransitionEntity) : void
+ addPredecessor(TransitionEntity) : void
+ removePredecessor(TransitionEntity) : void
(Other GUI and JHotDraw methods)
LabeledLineConnectionFigure
TransitionFigure
- _label : String
- _data : TransitionEntity
+ TransitionFigure()
+ setLabel(String) : void
+ getLabel() : String
+ setData(TransitionEntity) : void
+ getData() : TransitionEntity
(Other GUI and JHotDraw methods)