Using GMF and M2M for Model-Driven Development

Tatiana Fesenko, Borland Software
Radomil Dvorak, Borland Software
Bernd Kolb
Markus Voelter

GMF Overview

“The Eclipse Graphical Modeling Framework (GMF) provides a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF.”

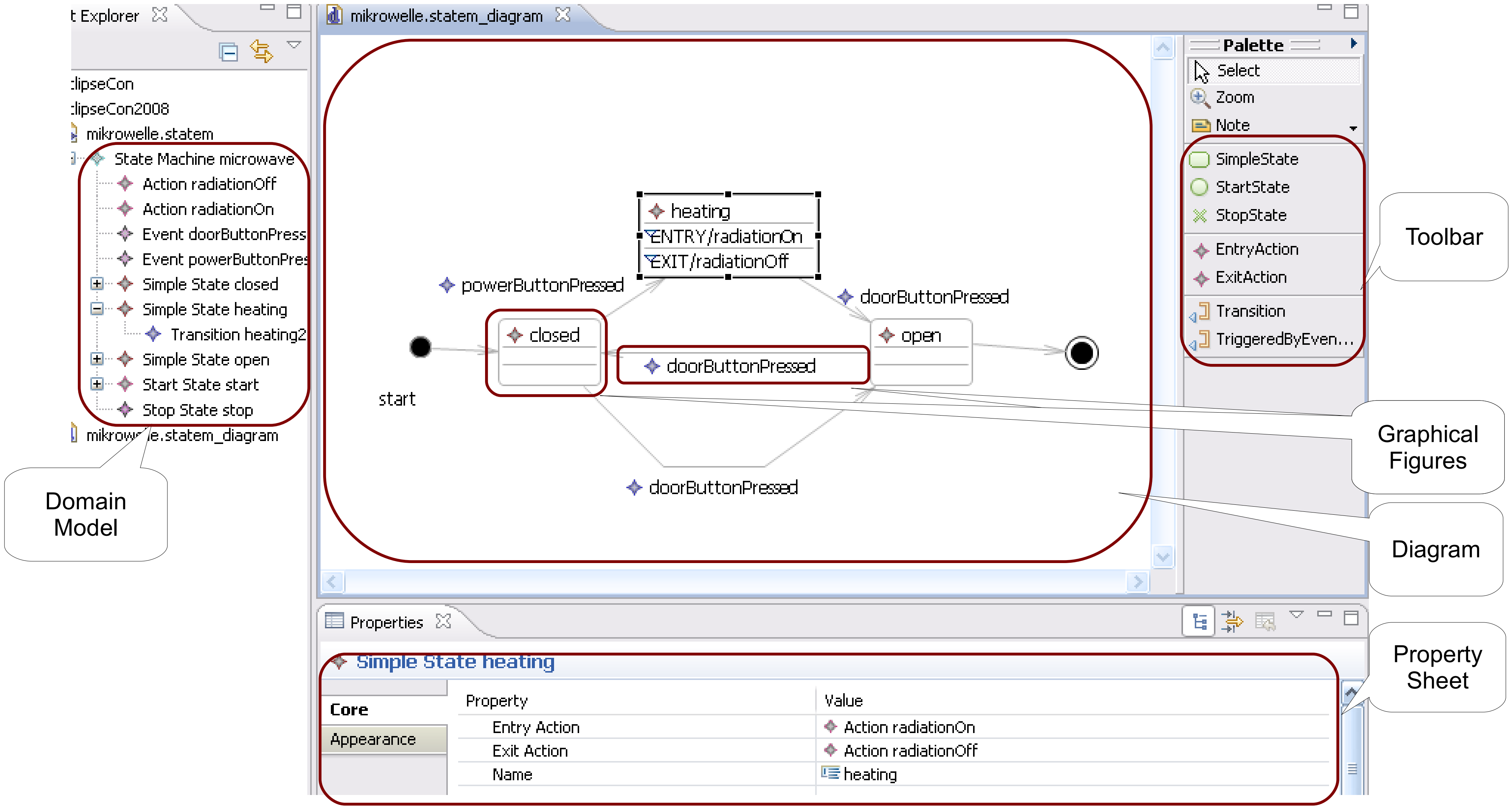
Runtime

- Binds EMF & GEF
- Notation metamodel
- Designed for extensibility

Generation (tooling)

- Models used to define graphics, tooling, mapping to domain
- Code generation targets runtime
- Promotes use of Domain-Specific Languages

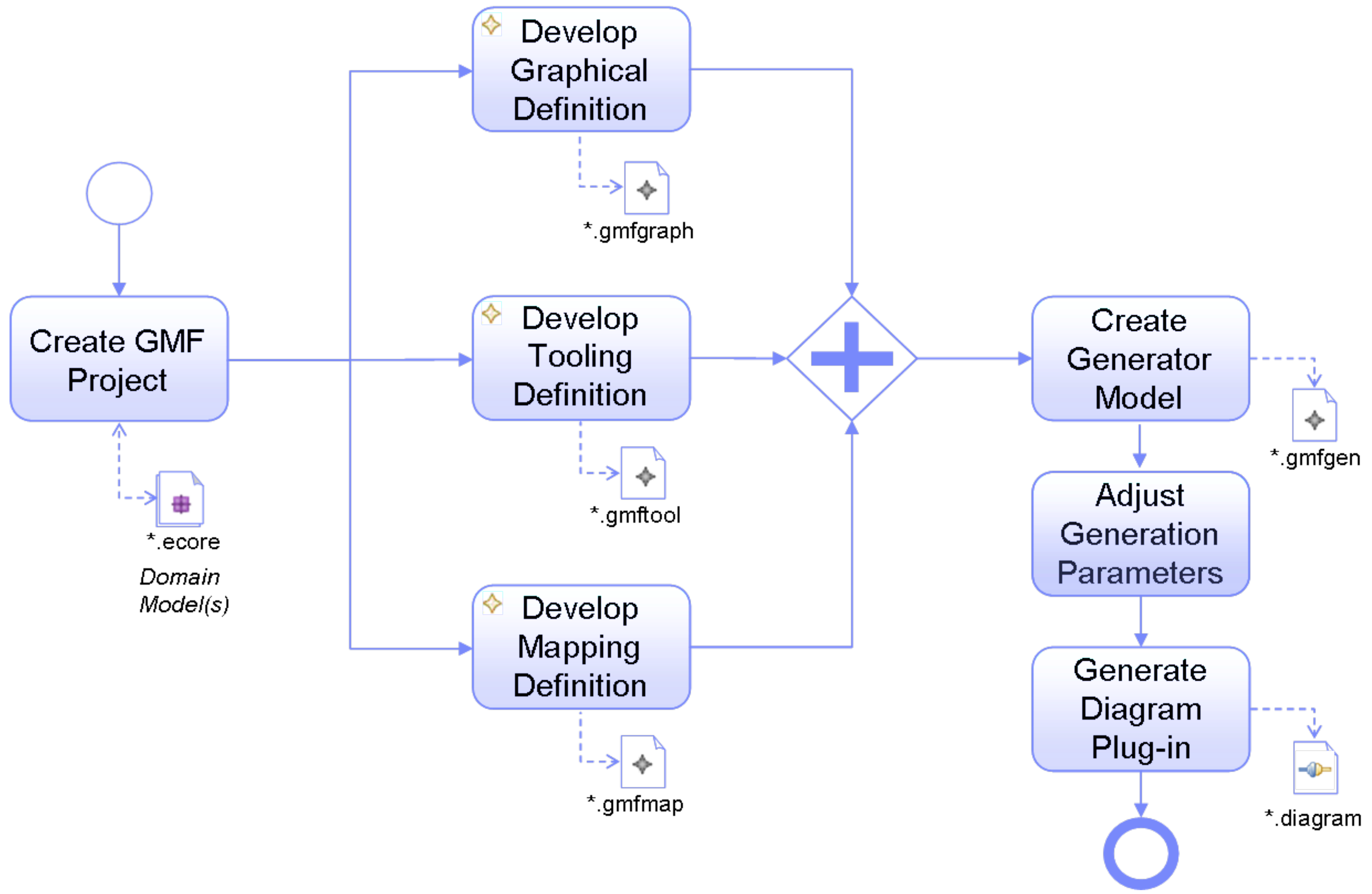
GMF-Generated Diagram Editor



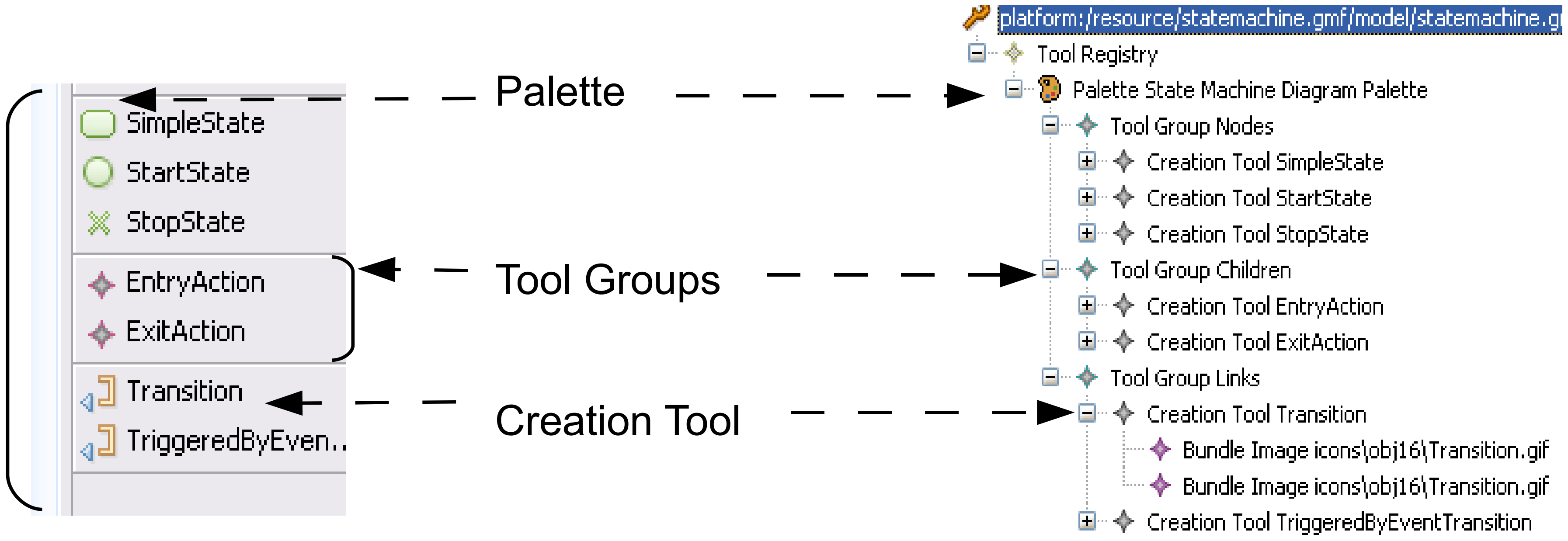
Simple State heating

Core	Property	Value
Appearance	Entry Action	Action radiationOn
	Exit Action	Action radiationOff
	Name	heating

GMF Generation overview

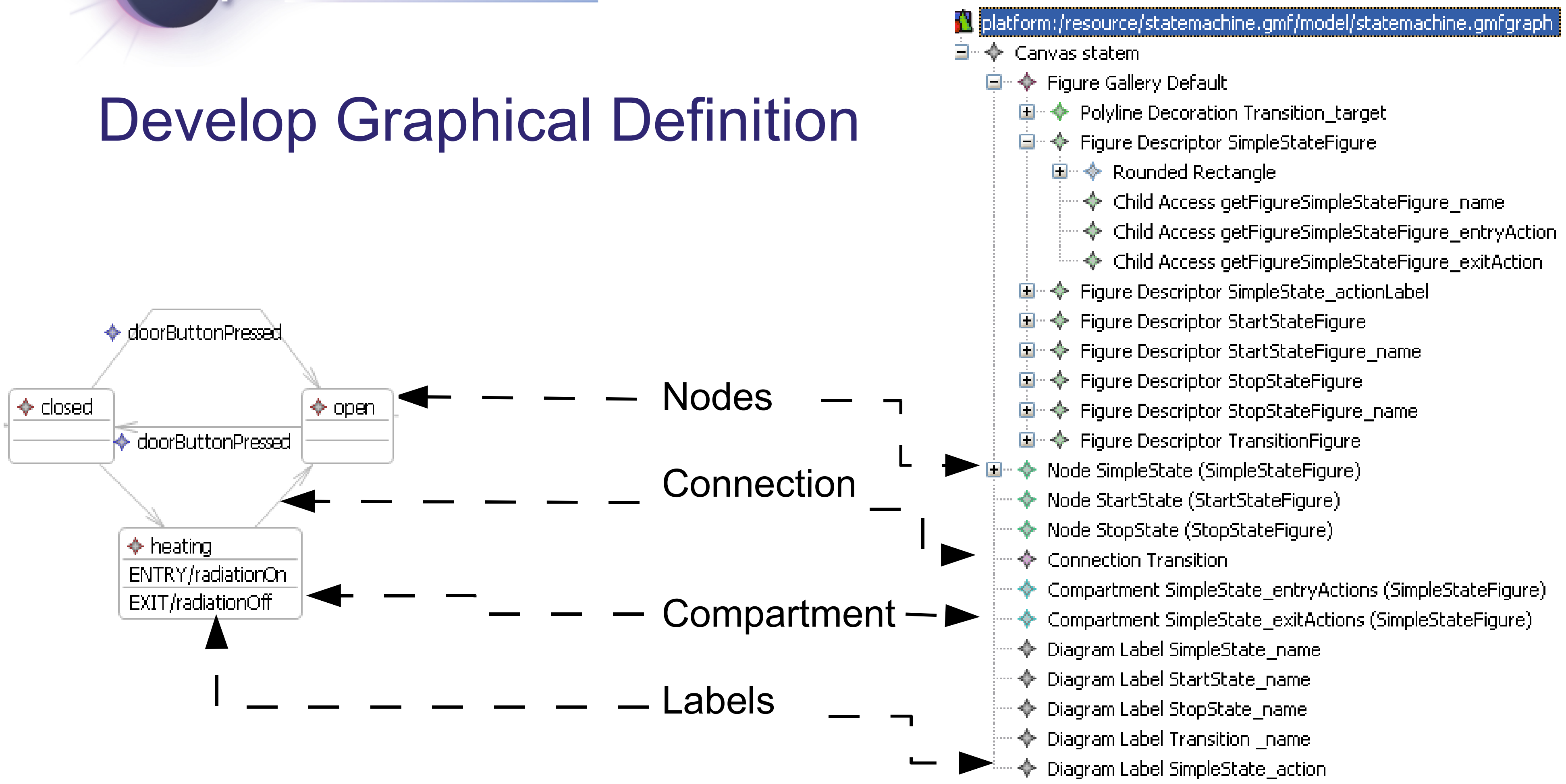


Develop Tooling Definition



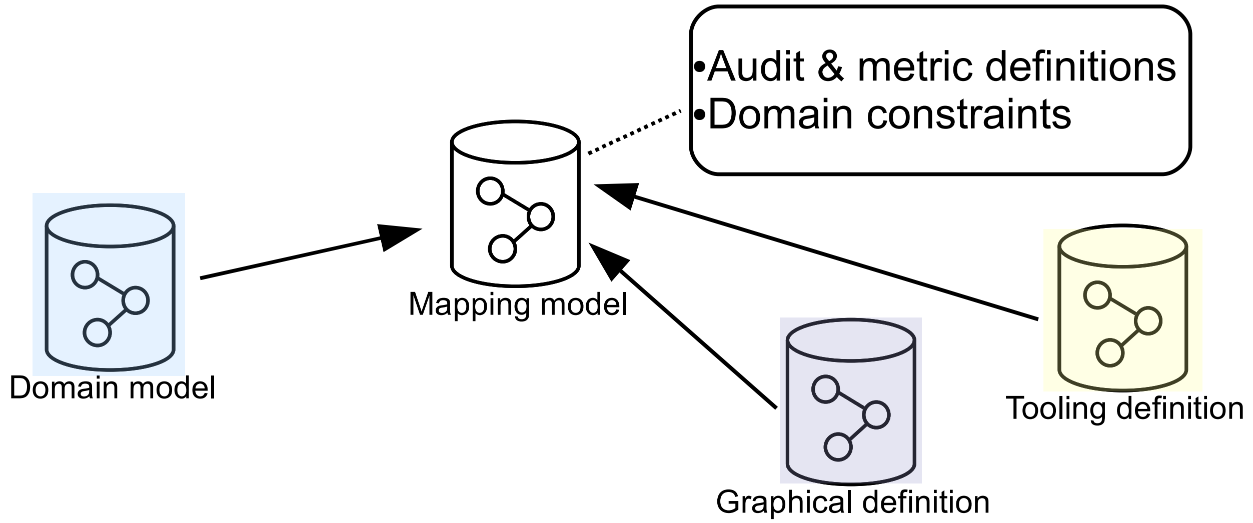
platform:/resource/statemachine.gmf/model/statemachine.g
- Tool Registry
  - Palette State Machine Diagram Palette
    - Tool Group Nodes
      - Creation Tool SimpleState
      - Creation Tool StartState
      - Creation Tool StopState
    - Tool Group Children
      - Creation Tool EntryAction
      - Creation Tool ExitAction
    - Tool Group Links
      - Creation Tool Transition
        - Bundle Image icons\obj16\Transition.gif
        - Bundle Image icons\obj16\Transition.gif
      - Creation Tool TriggeredByEventTransition

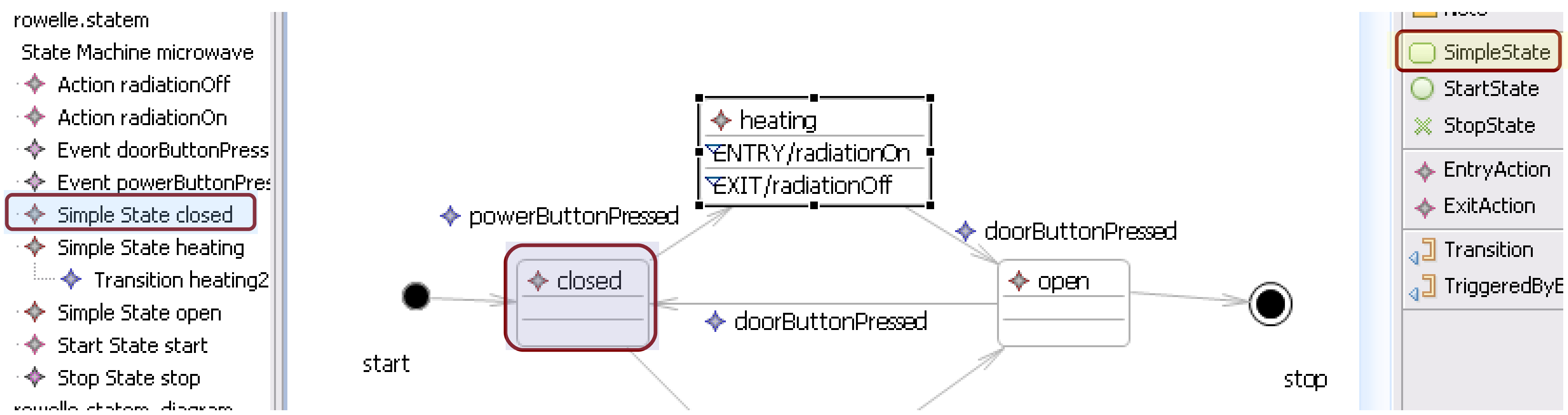
Develop Graphical Definition



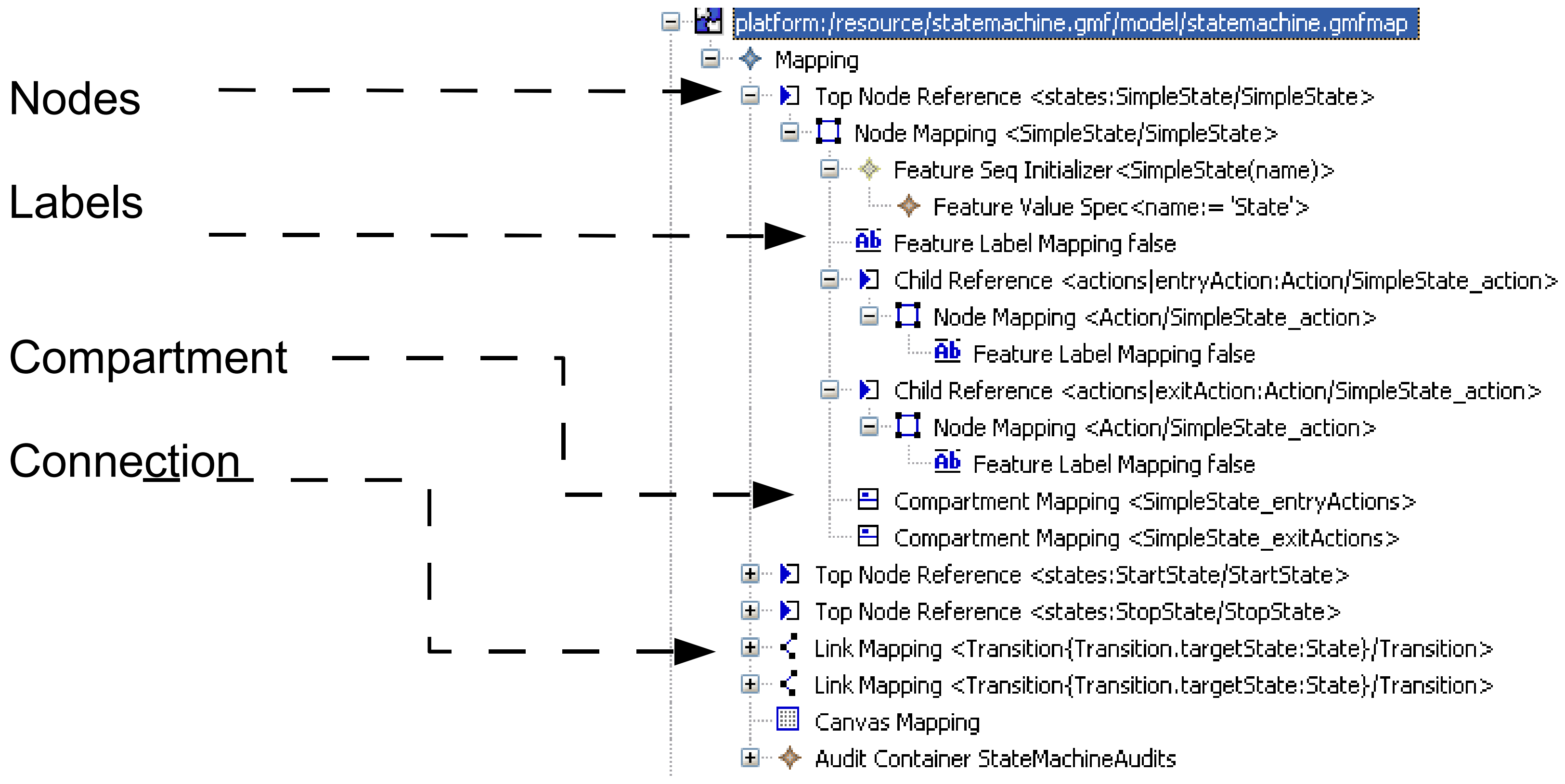
platform:/resource/statemachine.gmf/model/statemachine.gmfgraph
- Canvas statem
  - Figure Gallery Default
    - Polyline Decoration Transition_target
    - Figure Descriptor SimpleStateFigure
      - Rounded Rectangle
      - Child Access getFigureSimpleStateFigure_name
      - Child Access getFigureSimpleStateFigure_entryAction
      - Child Access getFigureSimpleStateFigure_exitAction
    - Figure Descriptor SimpleState_actionLabel
    - Figure Descriptor StartStateFigure
    - Figure Descriptor StartStateFigure_name
    - Figure Descriptor StopStateFigure
    - Figure Descriptor StopStateFigure_name
    - Figure Descriptor TransitionFigure
  - Node SimpleState (SimpleStateFigure)
  - Node StartState (StartStateFigure)
  - Node StopState (StopStateFigure)
  - Connection Transition
  - Compartment SimpleState_entryActions (SimpleStateFigure)
  - Compartment SimpleState_exitActions (SimpleStateFigure)
  - Diagram Label SimpleState_name
  - Diagram Label StartState_name
  - Diagram Label StopState_name
  - Diagram Label Transition _name
  - Diagram Label SimpleState_action

Mapping Definition



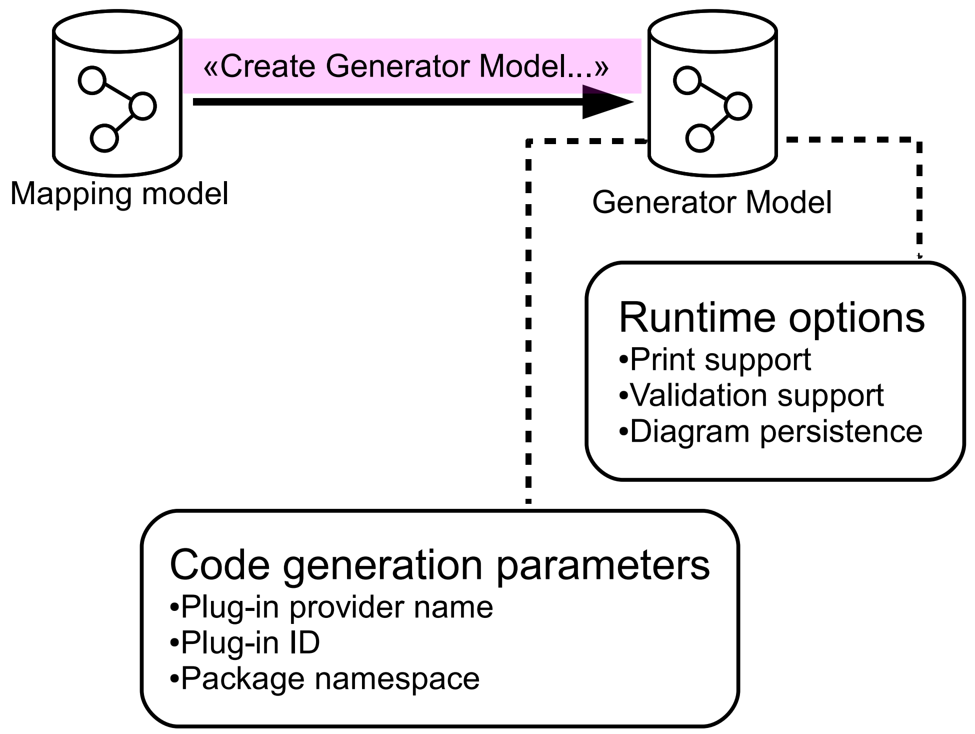


Develop Mapping Definition



```
platform:/resource/statemachine.gmf/model/statemachine.gmfmap
  Mapping
    Top Node Reference <states:SimpleState/SimpleState>
      Node Mapping <SimpleState/SimpleState>
        Feature Seq Initializer <SimpleState(name)>
          Feature Value Spec<name:= 'State'>
        Feature Label Mapping false
        Child Reference <actions|entryAction:Action/SimpleState_action>
          Node Mapping <Action/SimpleState_action>
            Feature Label Mapping false
        Child Reference <actions|exitAction:Action/SimpleState_action>
          Node Mapping <Action/SimpleState_action>
            Feature Label Mapping false
        Compartment Mapping <SimpleState_entryActions>
        Compartment Mapping <SimpleState_exitActions>
    Top Node Reference <states:StartState/StartState>
    Top Node Reference <states:StopState/StopState>
    Link Mapping <Transition{Transition.targetState:State}/Transition>
    Link Mapping <Transition{Transition.targetState:State}/Transition>
    Canvas Mapping
    Audit Container StateMachineAudits
```

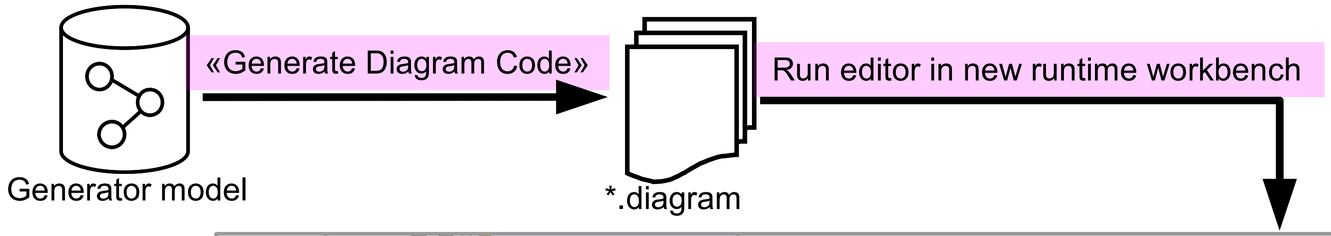
Create generator model

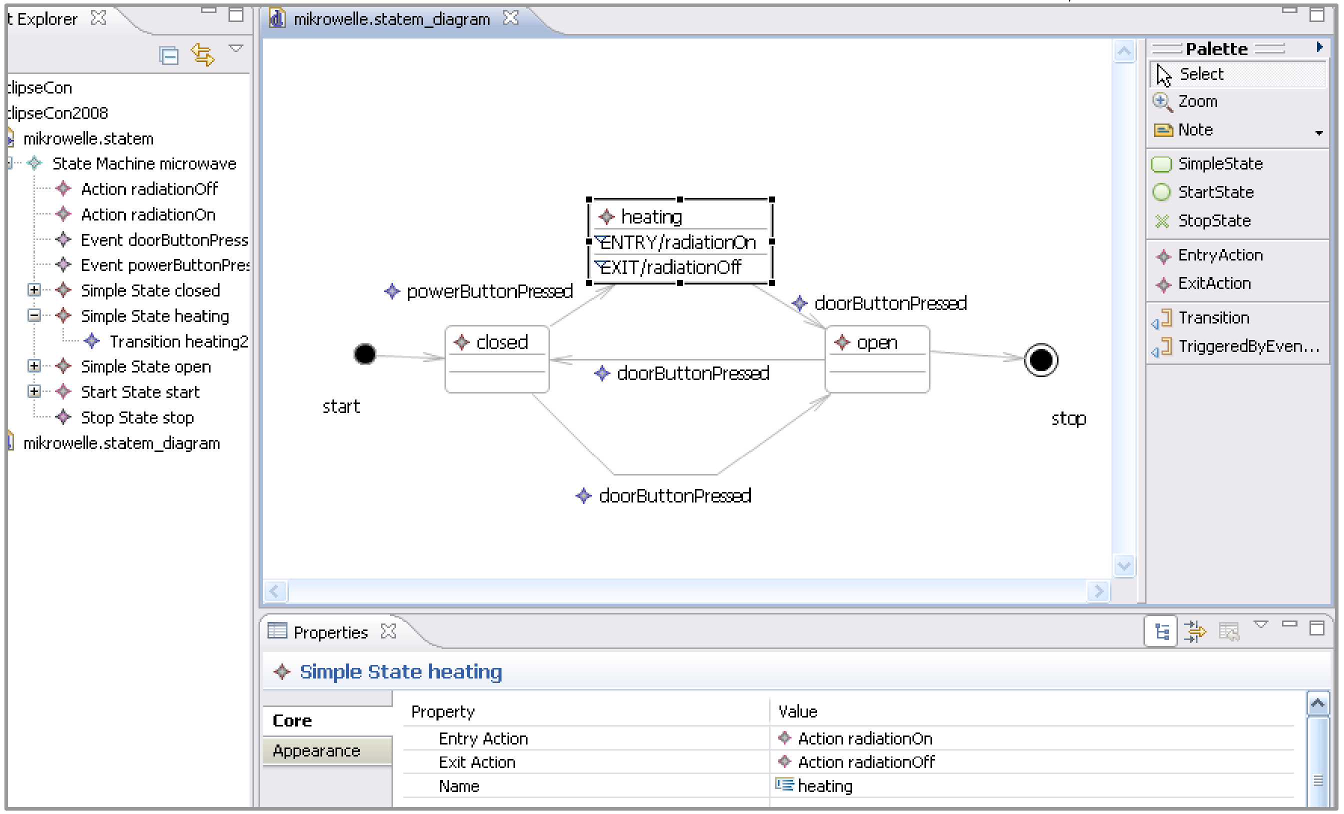


Runtime options
- Print support
- Validation support
- Diagram persistence

Code generation parameters
- Plug-in provider name
- Plug-in ID
- Package namespace

Generate diagram plug-in and run diagram



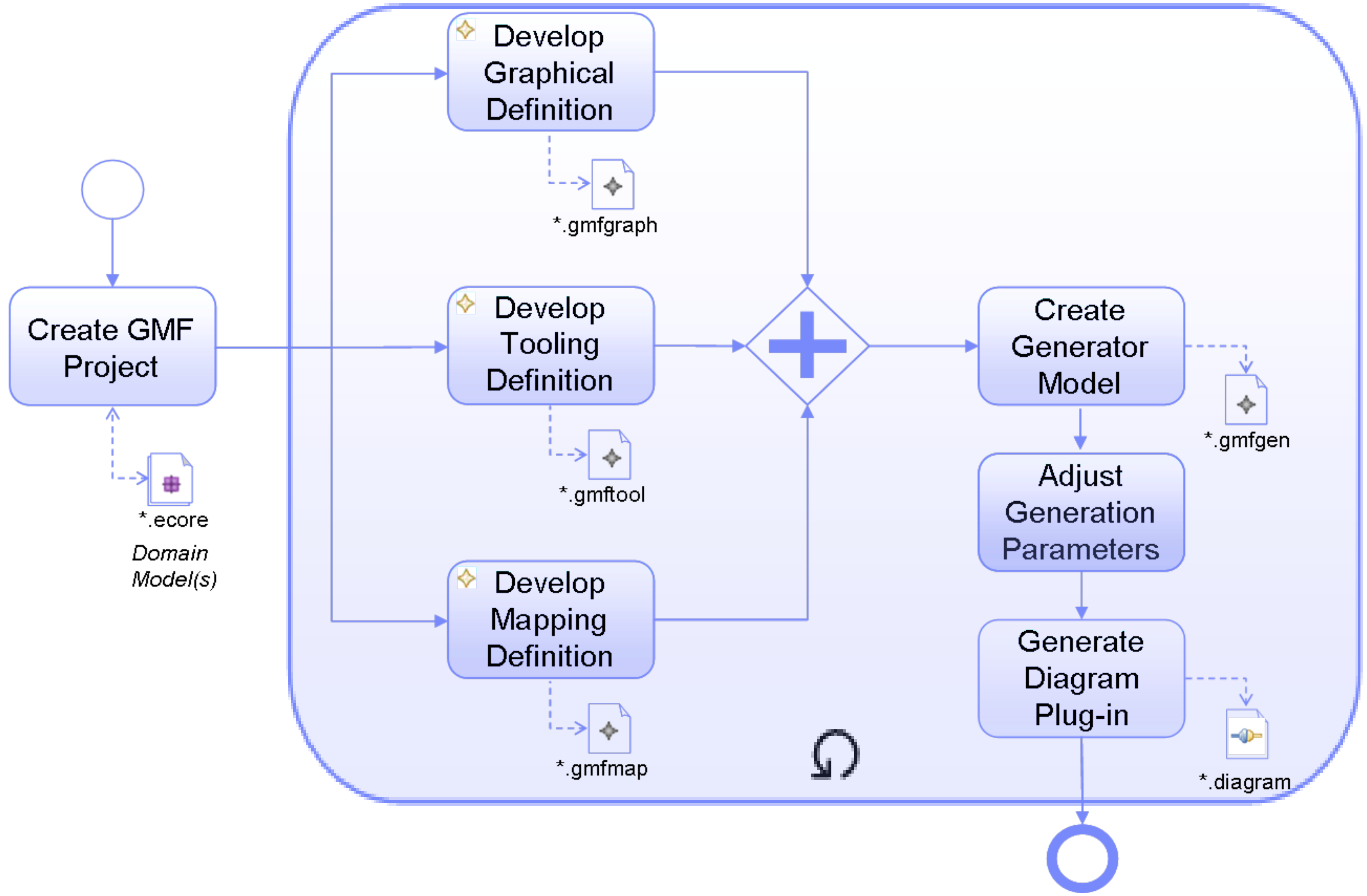


clipseCon
clipseCon2008
mikrowelle.statem
State Machine microwave
Action radiationOff
Action radiationOn
Event doorButtonPress
Event powerButtonPres
Simple State closed
Simple State heating
Transition heating2
Simple State open
Start State start
Stop State stop
mikrowelle.statem_diagram

Simple State heating

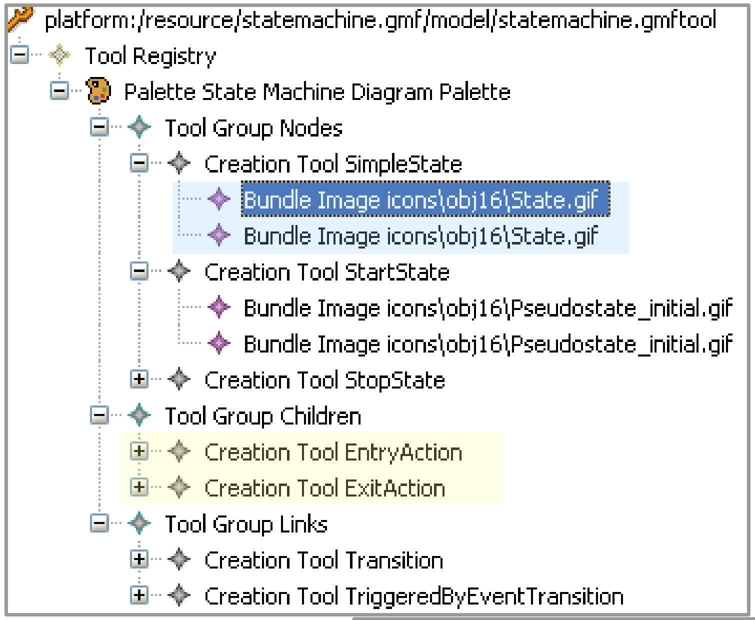
Property	Value
Entry Action	Action radiationOn
Exit Action	Action radiationOff
Name	heating

GMF Generation is iterative

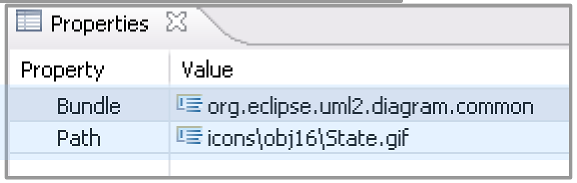


Tooling Definition: Beautiful icons

- Set Beautiful icons
  - Delete default image
  - Create icon image
- Add 'Children' tool group
  - Create EntryAction
  - Create ExitAction



```
platform:/resource/statemachine.gmf/model/statemachine.gmftool
  Tool Registry
    Palette State Machine Diagram Palette
      Tool Group Nodes
        Creation Tool SimpleState
          Bundle Image icons\obj16\State.gif
          Bundle Image icons\obj16\State.gif
        Creation Tool StartState
          Bundle Image icons\obj16\Pseudostate_initial.gif
          Bundle Image icons\obj16\Pseudostate_initial.gif
        Creation Tool StopState
      Tool Group Children
        Creation Tool EntryAction
        Creation Tool ExitAction
      Tool Group Links
        Creation Tool Transition
        Creation Tool TriggeredByEventTransition
```

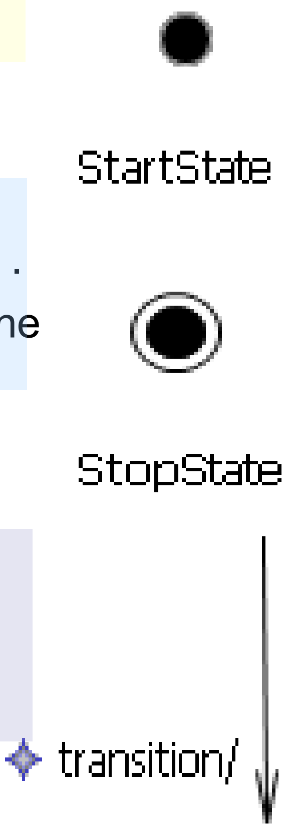


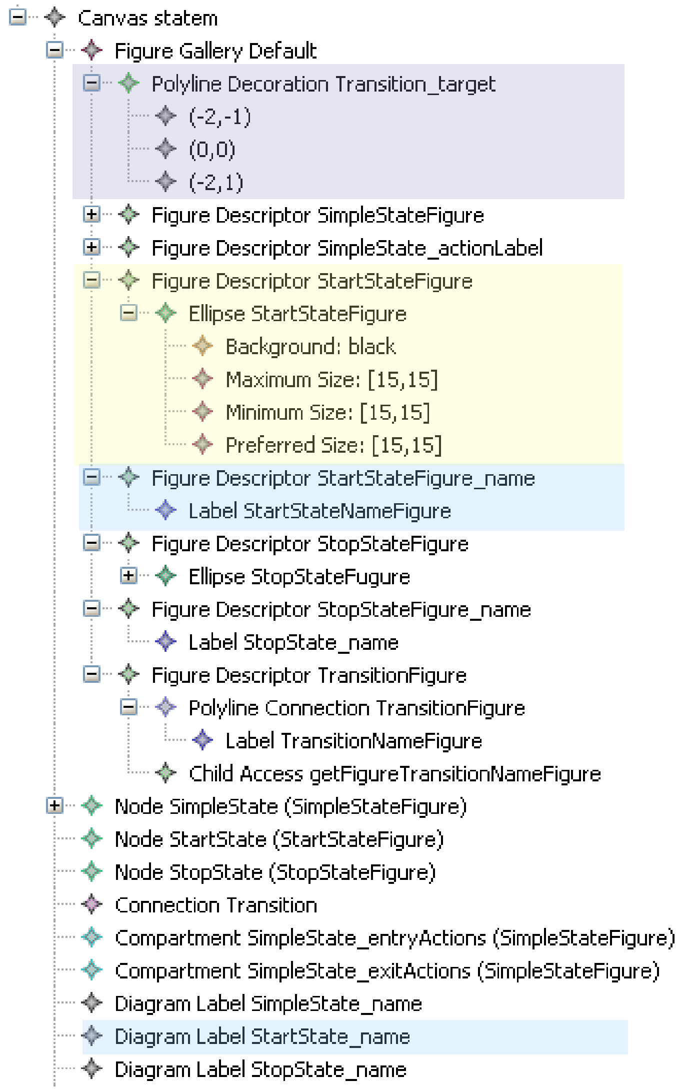
Properties

Property	Value
Bundle	org.eclipse.uml2.diagram.common
Path	icons\obj16\State.gif

Graphical Definition: Intelligent figures

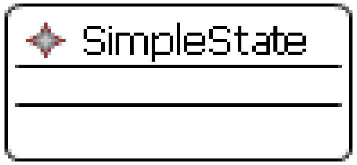
- Turn StartStateFigure into Ellipse, its background is black
- Turn StopState figure into the Ellipse containing inner Ellipse.
- Create Figure Descriptor for StartStateName, create Label inside it .
- Create DiagramLabel referencing it. The label became external.
- Repeat with StopStateName.
- Set Arrow decoration for Transition
    - Create PolylineDecoration 'ConnectorTarget' in Figure Galle Set TemplatePoints (-2,-1), (0,0), (-2,1).
    - Select Polyline Connection TransitionFigure inside Figure Descripor TransitionFigure. Set choose 'ConnectorTarget' for TargetDecoration property.



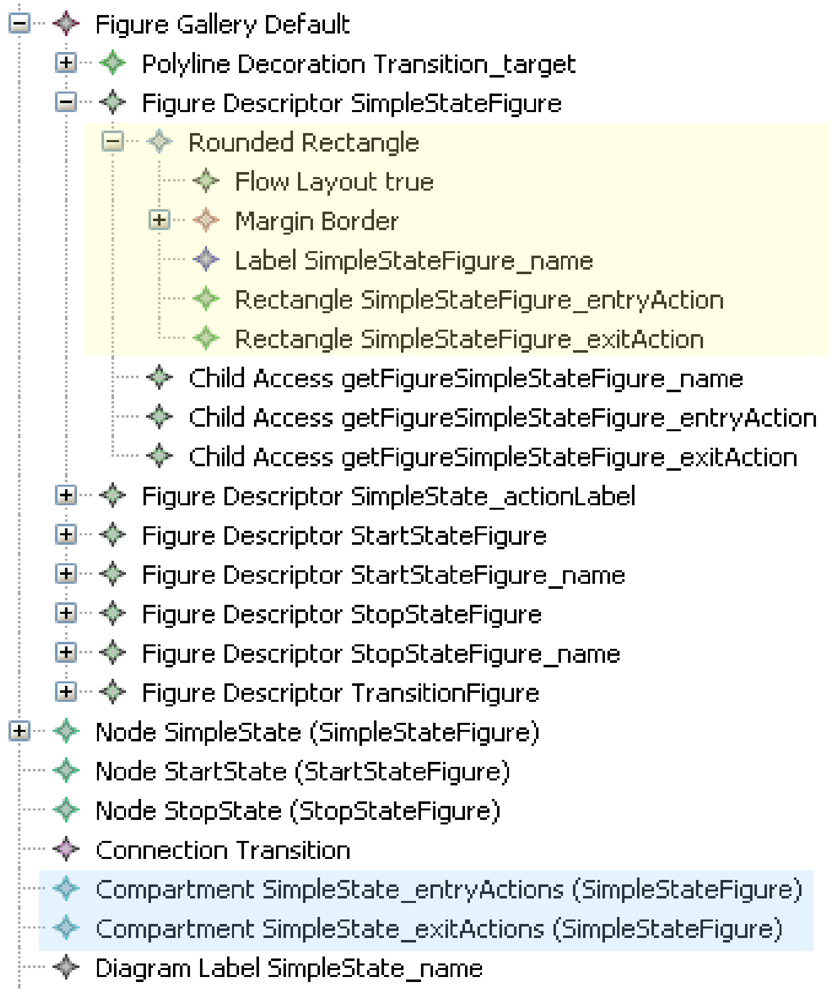


```
Canvas statem
  Figure Gallery Default
    Polyline Decoration Transition_target
      (-2,-1)
      (0,0)
      (-2,1)
    Figure Descriptor SimpleStateFigure
    Figure Descriptor SimpleState_actionLabel
    Figure Descriptor StartStateFigure
      Ellipse StartStateFigure
        Background: black
        Maximum Size: [15,15]
        Minimum Size: [15,15]
        Preferred Size: [15,15]
    Figure Descriptor StartStateFigure_name
      Label StartStateNameFigure
    Figure Descriptor StopStateFigure
      Ellipse StopStateFugure
    Figure Descriptor StopStateFigure_name
      Label StopState_name
    Figure Descriptor TransitionFigure
      Polyline Connection TransitionFigure
        Label TransitionNameFigure
      Child Access getFigureTransitionNameFigure
  Node SimpleState (SimpleStateFigure)
  Node StartState (StartStateFigure)
  Node StopState (StopStateFigure)
  Connection Transition
  Compartment SimpleState_entryActions (SimpleStateFigure)
  Compartment SimpleState_exitActions (SimpleStateFigure)
  Diagram Label SimpleState_name
  Diagram Label StartState_name
  Diagram Label StopState_name
```

Graphical Definition: SimpleState



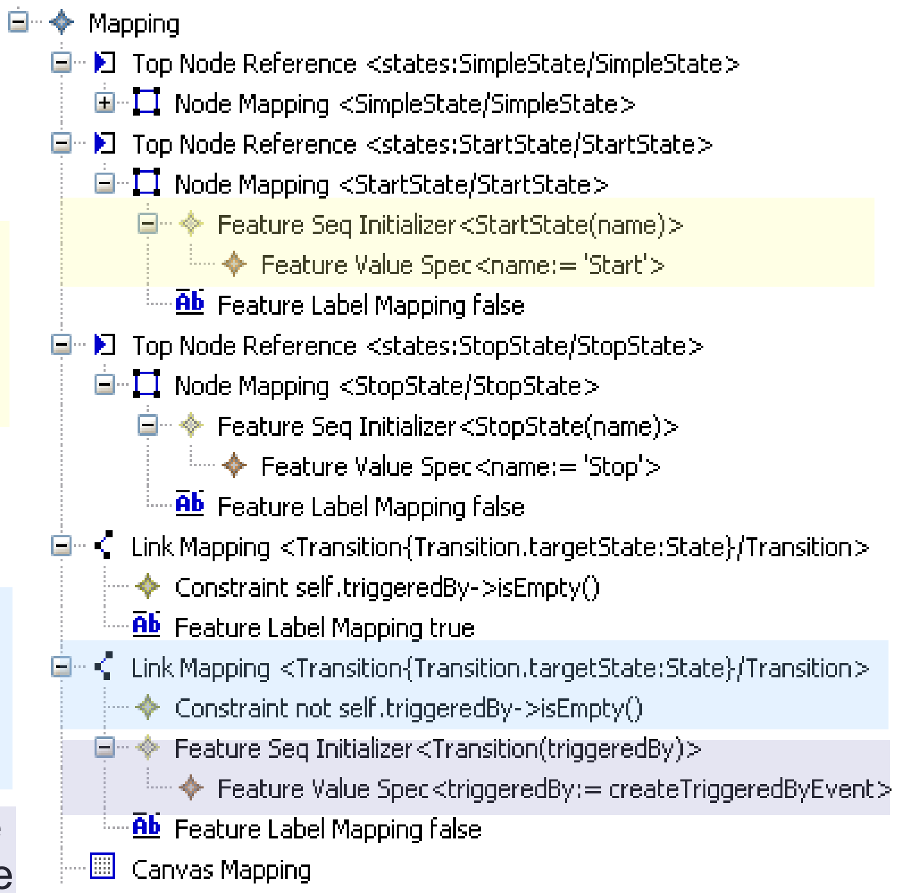
- Turn SimpleStateFigure into rounded rectangle
- Add rectangles for Entry and Exit Action compartments inside SimpleAction.
- Create compartments for Entry and Exit Actions. Choose SimpleStateFigure in 'Figure' property,reference them to just created rectangles in 'Accessor' property.



```
Figure Gallery Default
  Polyline Decoration Transition_target
  Figure Descriptor SimpleStateFigure
    Rounded Rectangle
      Flow Layout true
      Margin Border
      Label SimpleStateFigure_name
      Rectangle SimpleStateFigure_entryAction
      Rectangle SimpleStateFigure_exitAction
    Child Access getFigureSimpleStateFigure_name
    Child Access getFigureSimpleStateFigure_entryAction
    Child Access getFigureSimpleStateFigure_exitAction
  Figure Descriptor SimpleState_actionLabel
  Figure Descriptor StartStateFigure
  Figure Descriptor StartStateFigure_name
  Figure Descriptor StopStateFigure
  Figure Descriptor StopStateFigure_name
  Figure Descriptor TransitionFigure
Node SimpleState (SimpleStateFigure)
Node StartState (StartStateFigure)
Node StopState (StopStateFigure)
Connection Transition
Compartment SimpleState_entryActions (SimpleStateFigure)
Compartment SimpleState_exitActions (SimpleStateFigure)
Diagram Label SimpleState_name
```

Mapping Definition: Feature Initializers and OCL Constraints

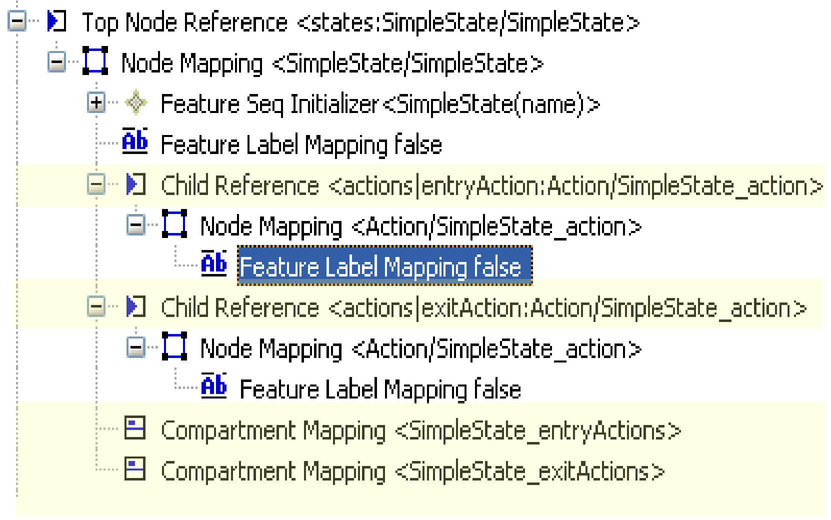
- Automatically set default name for elements
  - Create feature Seq Initializer for SimpleState Node Mapping. Create Feature Value Spec. Choose 'name' feature, set 'State' to value.
  - Repeat with Start/Stop States
- Event-triggered transition
  - Create additional Link Mapping for Transition.
  - Set Constraints in order to distinguish links.
  - Create java FeatureValueInitializer. We will implement it to create and reference Event automatically.



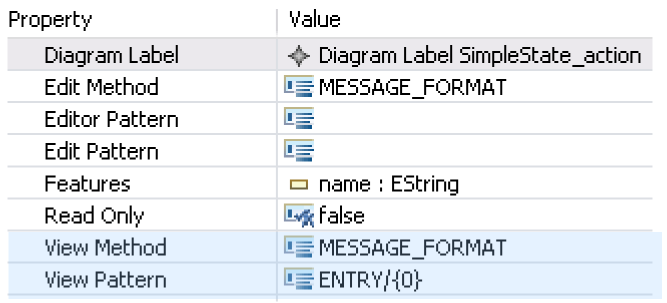
```
Mapping
  Top Node Reference <states:SimpleState/SimpleState>
    Node Mapping <SimpleState/SimpleState>
  Top Node Reference <states:StartState/StartState>
    Node Mapping <StartState/StartState>
      Feature Seq Initializer <StartState(name)>
        Feature Value Spec <name:= 'Start'>
      Feature Label Mapping false
  Top Node Reference <states:StopState/StopState>
    Node Mapping <StopState/StopState>
      Feature Seq Initializer <StopState(name)>
        Feature Value Spec <name:= 'Stop'>
      Feature Label Mapping false
  Link Mapping <Transition{Transition.targetState:State}/Transition>
    Constraint self.triggeredBy->isEmpty()
    Feature Label Mapping true
  Link Mapping <Transition{Transition.targetState:State}/Transition>
    Constraint not self.triggeredBy->isEmpty()
    Feature Seq Initializer <Transition(triggeredBy)>
      Feature Value Spec <triggeredBy:= createTriggeredByEvent>
    Feature Label Mapping false
  Canvas Mapping
```

Mapping Definition: Simple State

- Entry/Exit Actions compartment Mapping
- Distinguish Entry and Exit actions
  - Set View Pattern 'ENTRY/{0}' for the MessageFormat parser of EntryAction
  - Set View Pattern 'EXIT/{0}' for the MessageFormat parser of ExitAction



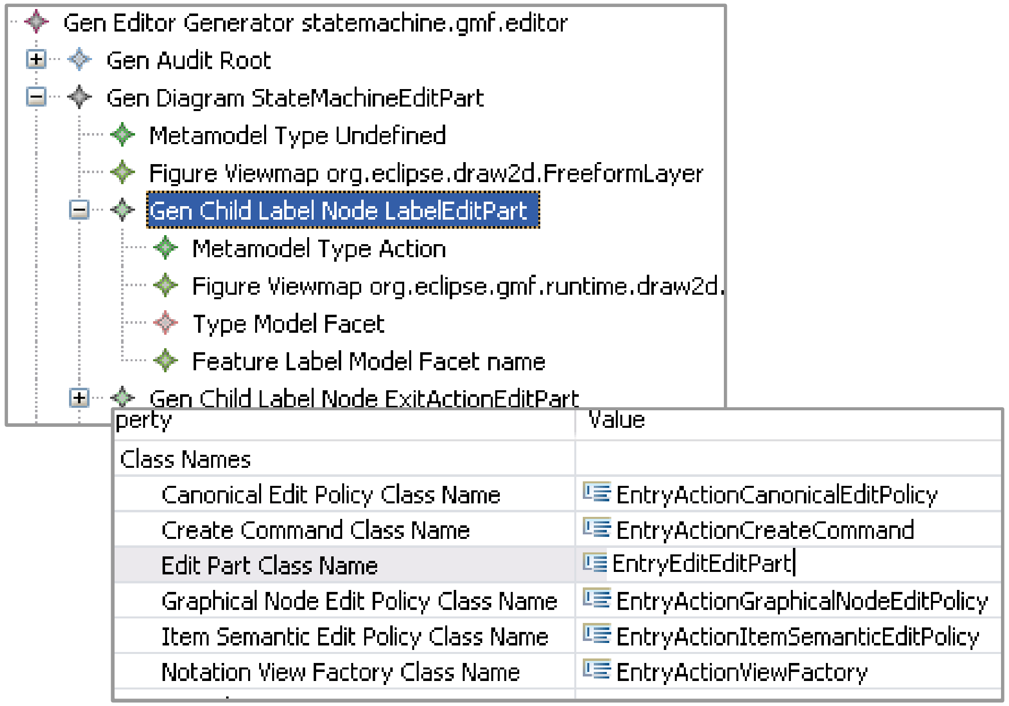
```
Top Node Reference <states:SimpleState/SimpleState>
  Node Mapping <SimpleState/SimpleState>
    Feature Seq Initializer <SimpleState(name)>
    Feature Label Mapping false
    Child Reference <actions|entryAction:Action/SimpleState_action>
      Node Mapping <Action/SimpleState_action>
        Feature Label Mapping false
    Child Reference <actions|exitAction:Action/SimpleState_action>
      Node Mapping <Action/SimpleState_action>
        Feature Label Mapping false
    Compartment Mapping <SimpleState_entryActions>
    Compartment Mapping <SimpleState_exitActions>
```



Property	Value
Diagram Label	Diagram Label SimpleState_action
Edit Method	MESSAGE_FORMAT
Editor Pattern	
Edit Pattern	
Features	name : EString
Read Only	false
View Method	MESSAGE_FORMAT
View Pattern	ENTRY/{0}

Generator model: Code generation parameters

- Make Entry EditPart class names intelligible
  - Rename LabelEditPart to EntryActionEditPart
- Repeat with Labels2EditPart (EntryActionEditPart)



```
Gen Editor Generator statemachine.gmf.editor
  Gen Audit Root
  Gen Diagram StateMachineEditPart
    Metamodel Type Undefined
    Figure Viewmap org.eclipse.draw2d.FreeformLayer
    Gen Child Label Node LabelEditPart
      Metamodel Type Action
      Figure Viewmap org.eclipse.gmf.runtime.draw2d.
      Type Model Facet
      Feature Label Model Facet name
    Gen Child Label Node ExitActionEditPart
```

perty	Value
Class Names	
Canonical Edit Policy Class Name	EntryActionCanonicalEditPolicy
Create Command Class Name	EntryActionCreateCommand
Edit Part Class Name	EntryEditEditPart
Graphical Node Edit Policy Class Name	EntryActionGraphicalNodeEditPolicy
Item Semantic Edit Policy Class Name	EntryActionItemSemanticEditPolicy
Notation View Factory Class Name	EntryActionViewFactory

Generated plugin: Code modification

- Generated code can be changed to implement domain-specific requirements
- Changed code is marked with 'generated NOT' tag.
  - Modify *EntryActionCreateCommand* and *ExitActionCreateCommand* in order to create them inside StateMachine and be referenced by SimpleState
  - Implement java FeatureValueInitializer for EventTriggeredTransition in *ElementInitializers* class.

```
/**
 * @generated NOT
 */
protected EObject doDefaultElementCreation() {
    Action newElement = StatemFactory.eINSTANCE.createAction();
    SimpleState simpleState = (SimpleState) getElementToEdit();
    simpleState.getStateMachine().getActions().add(newElement);
    simpleState.setEntryAction(newElement);
    return newElement;
}
```

```
    static class Java {

        /**
         * @generated NOT
         */
        private static Event createTriggeredByEvent(Transition self) {
            State targetState = self.getTargetState();
            if (targetState == null) {
                return null;
            }
            Event event = StatemFactory.eINSTANCE.createEvent();
            targetState.getStateMachine().getEvents().add(event);
            return event;
        }
    }
```

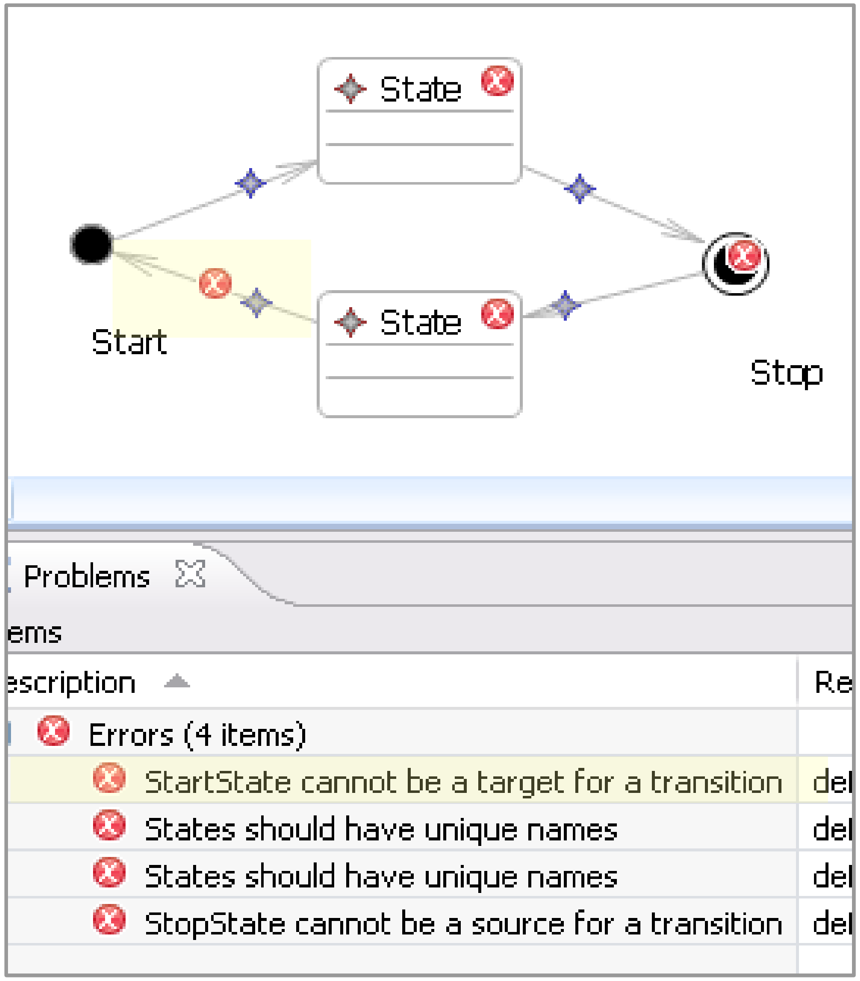
StateMachine Diagram: Discover GMF Runtime Features

- Tool palette and overview
- Layout and selection tools
- Diagram image export (svg, bmp, jpeg, gif)
- Tabbed properties view
- Font and color options for selected element
- Link routing and style options
- Animated zoom and layout
- Diagram printing

Validation

Validation: Define rules

- Validation Rules are written in OCL
- They defined in GMF Map model.
  - State has 'States should have unique names' validation rule
  - Rules for source and target of a Transition
    - 'StopState cannot be a source for a transition' for StopState
    - 'StartState cannot be a target for a transition' for Transition



ems

escription	Re
Errors (4 items)	
StartState cannot be a target for a transition	del
States should have unique names	del
States should have unique names	del
StopState cannot be a source for a transition	del

Enable and Run Validation

- Make validation enabled in GMFGen model:
  - 'Validation Enabled' of GenDiagram is 'true'
  - 'Validation Decorators' is 'true'
  - 'Validation Provider Priority' is 'Medium'
- Validation runs on diagram action:
  - Call 'Validate' action from the 'Diagram' tool menu.

Summary

- We created GMF-generated diagram editor for the StateMachine model.
  - It was quick and easy
- Using GMF is an iterative process.
  - We can modify selected tooling models and enjoy improvements in regenerated diagram
- Code GMF produces can be customized.
  - We modified the generated code

Using GMF and M2M for Model-driven development

Thank you!

Questions?

http://www.eclipse.org/gmf