Gamma Statechart Simulator
Contributor’s Guide

Table of Contents

Chapter 1. Environment setup

1.1. Gamma and Eclipse

This plugin uses Gamma to function, please follow their setup guide!

This includes: * Gamma * Eclipse * Yakindu (you may need a license) * Viatra * EMF * Xtext * Xtend

This is everything you need to start developing.

1.1.1. Peculiar bug

Because of a bug, the `Gamma Statechart Model` can't be imported to the same workspace as the rest of the plugin → in order to use, you have to export the `Gamma Statechart Model` as a plugin, and install it into your Eclipse.

Chapter 2. Code architecture

Currently, the code consists of two parts

- Simulator
- Model

This is because the Model plugin can't be directly inside the same workspace.

2.1. Model

This is an EMF plugin. Its purpose is to provide the simulator with runtime information, such as active states, variable values, etc.

2.1.1. Simulator Model

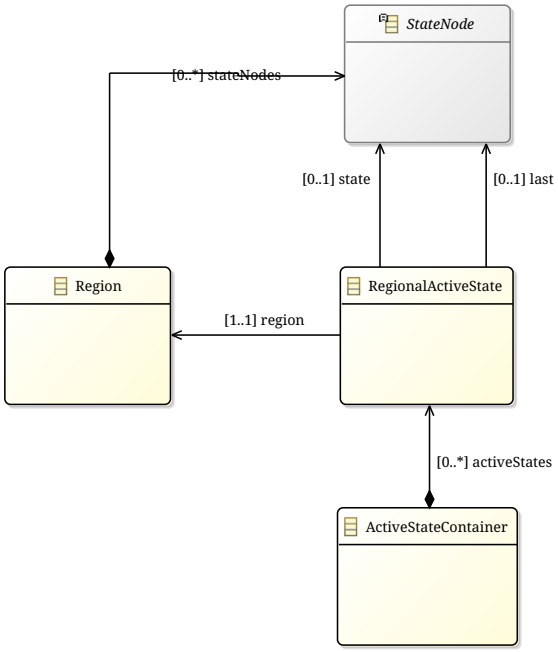

Figure 1. Simulator Model

As you can see, there are a number of custom Classes.

ActiveStateContainer

Contains all `RegionalActiveState`

RegionalActiveState

This is the heart of the simulator runtime model. All `Region` is associated with one, and contains the current active state, and the last state.

Whenever firing a transition, the source's `RegionalActiveState.state` is set to null, and the target's `RegionalActiveState.state` and `RegionalActiveState.last` is set to the target.

This way the last always contains the state to travel to whenever a history state exists.

2.2. Simulator

This is an eclipse plugin, which exposes a `View` which can be opened in a running eclipse instance.

The UI of the view is inside the `ui` package.__ See User's Guide - UI overview

2.2.1. Query

The simulator uses `Viatra` queries to fetch simulation time data. See Viatra for more info.

UI data

On demand

These are collected on demand, when the View is initialised. They are only called once, as these data do not change during runtime.

- Available ports/events
- Available variables

Event driven

These are collected and sent to the UI every time they update.

- Active states

Transition data

For ease of use, `QueryUtils` contains helper functions for querying the output of Viatra queries.

Expression and Action languages

Gamma defines Expression and Action languages. The evaluators are contained in `query.util` package. These traverse the `AST`, and process every node.

2.3. More information

For more information, see resources/slides/GammaStatechartSimulator-Demo.pdf

Chapter 3. Version control

If you want to contribute, feel free to start working on an issue, and submit a pull request!

Few requirements:

3.1. Branch names

Branches must follow the following naming scheme `issues/issue number/brief description`

For example: `issues/6/parallel_regions`

This is to easily find the related issue from github, and to prevent branch name conflicts.

3.2. PR

PRs should be assigned to the same person/people who worked on the issue. The same labels and milestones should be applied, and a small description of the changes.

Try making as small change as possible, if you can't, split up the PR into multiple ones.

3.3. PR merging

Before you merge, I have to review and accept. Before merging, merge `main` in, and only after that merge into main.

3.4. Commits

All commit must have at least a single line with the issue's number as the first work.

For example

```
#6 added regions to metamodel
```

If needed, you may provide more detail in the lines after