

# SCXML JS DCharts Simulation Tools Tutorial

Jacob Beard

01/12/10

## About SCXML JS DCharts Simulation Tools Tutorial

SCXML JS is a Statechart-to-JavaScript compiler optimized for User Interface development for the World Wide Web. It is currently being developed as part of my Master's thesis. It is designed to be run against XML descriptions of statecharts, and does not have any dependencies on AToM<sup>3</sup>.

SCXML JS DCharts Simulation Tools is a collection of Python modules that integrate SCXML JS with the AToM<sup>3</sup> DCharts modelling environment.

## Features

SCXML JS DCharts Simulation Tools provides the following features:

1. Export SCXML documents from DCharts models.
2. Export JavaScript documents from DCharts models.
3. Allow interaction with the model via a generic console web client interface.
4. Start a debugging server within AToM<sup>3</sup> to allow interactions to graphically animate the original DCharts model, thus facilitating graphical simulation and debugging.

## Release Status

SCXML JS and SCXML JS DCharts Simulation Tools are *unreleased software*. At the time of this writing, SCXML JS has been under development for about 10 months, and SCXML JS DCharts Simulation Tools for about one month. You will probably find bugs. When you do, please send a bug report to [jbeard4@cs.mcgill.ca](mailto:jbeard4@cs.mcgill.ca) including the stack trace from the console and the DCharts model that caused the bug.

## System Requirements

SCXML JS DCharts Simulation Tools has only been tested on Ubuntu 10.04. It *should* work on Windows as well, but has not been tested.<sup>1</sup> AToM3 is known not to work in Mac OS X.

On Ubuntu 10.04, the following Python packages must be installed: *python-lxml* and *python-setuptools*. These packages can be installed through apt.

The web console interface has been tested with Firefox 3.6 and Chromium 6.<sup>2</sup>

---

<sup>1</sup> Don't let that discourage you, though. Anyone who wants to try this on Windows totally should, and let me know if it works.

<sup>2</sup> Again, if anyone wants to try this on Windows, in other browsers, such as Internet Explorer, the feedback would be much appreciated.

## Installation

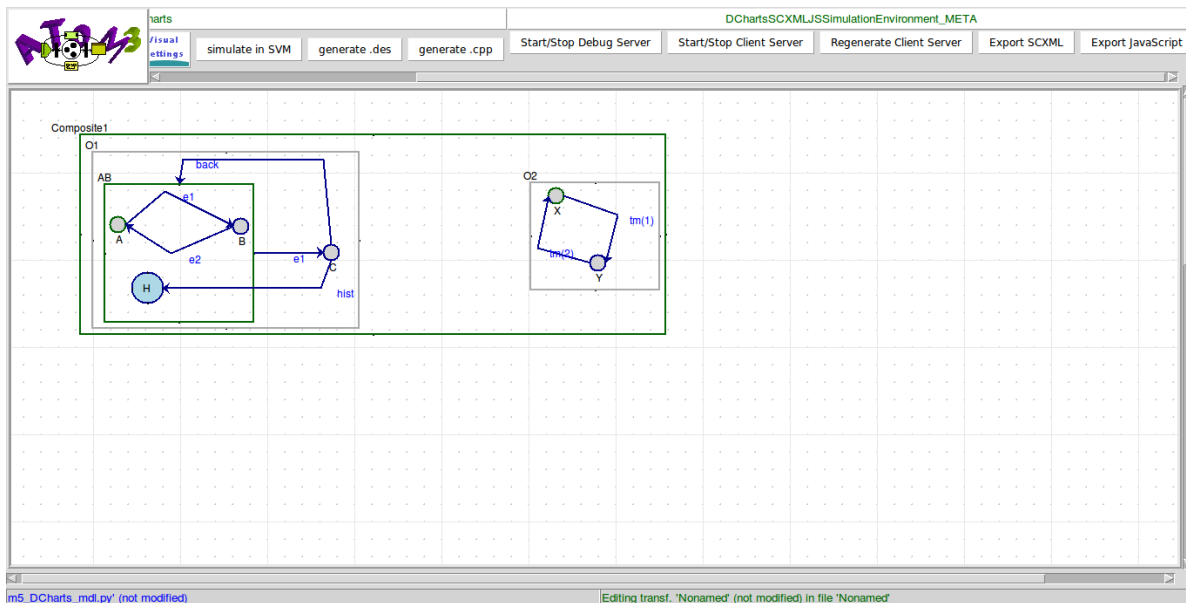
Download <http://stuff.echo-flow.com/atom3-dcharts-simulation-tools/externals.tgz> and unzip into *Externals/*

Download [http://stuff.echo-flow.com/atom3-dcharts-simulation-tools/user\\_formalisms.tgz](http://stuff.echo-flow.com/atom3-dcharts-simulation-tools/user_formalisms.tgz) and unzip into *User Formalisms/*

## Basic Workflow

This section will walk you through the basic workflow you will likely use when developing Statecharts with AToM<sup>3</sup> DCharts.

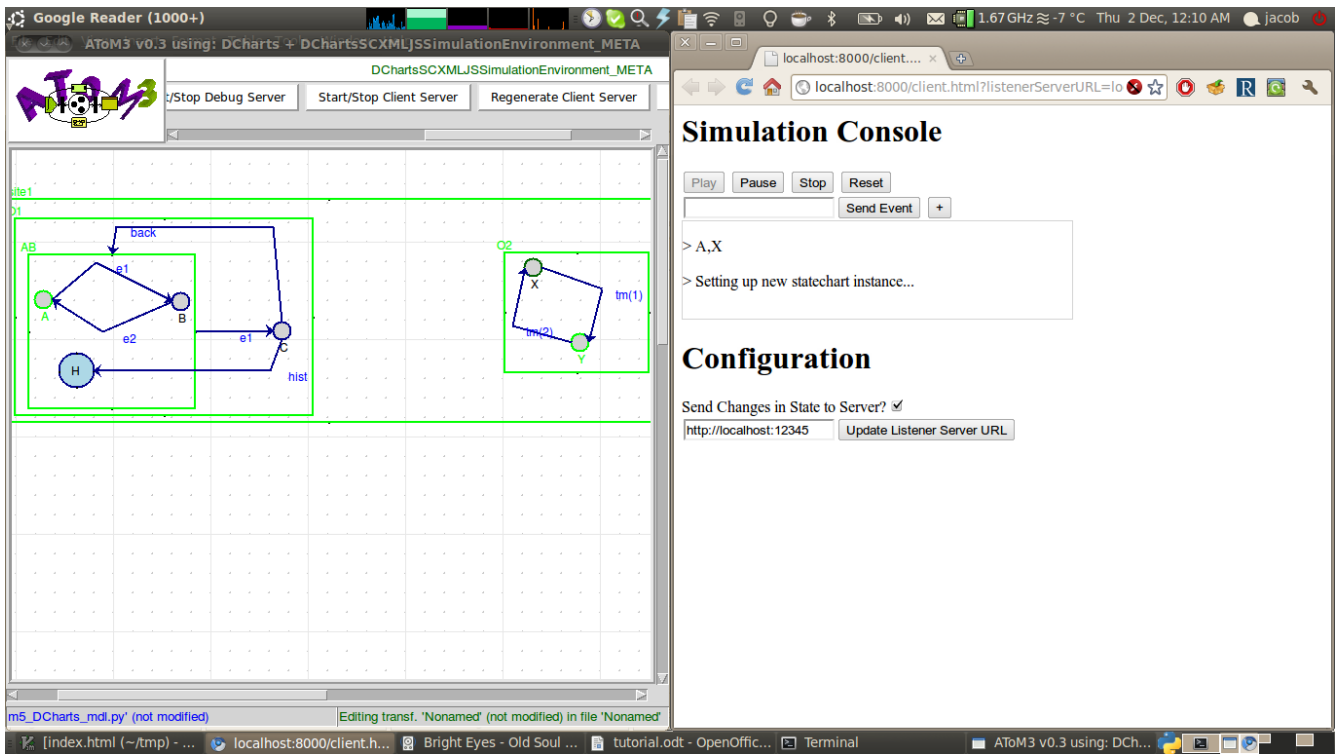
1. Start AToM<sup>3</sup>, preferably in a terminal so that you can see output to stdout and stderr.
2. Open a DCharts model or open the DCharts formalism. Throughout this tutorial, I'll be using the model *Models/DCharts/m5\_DCharts\_mdl.py*.
3. Open the SCXML JS DCharts Simulation Tools formalism at *User Formalisms/atom3-dcharts-scxml-js-simulation-environment-formalism*. AToM3 should look something like the following, with both formalisms loaded:



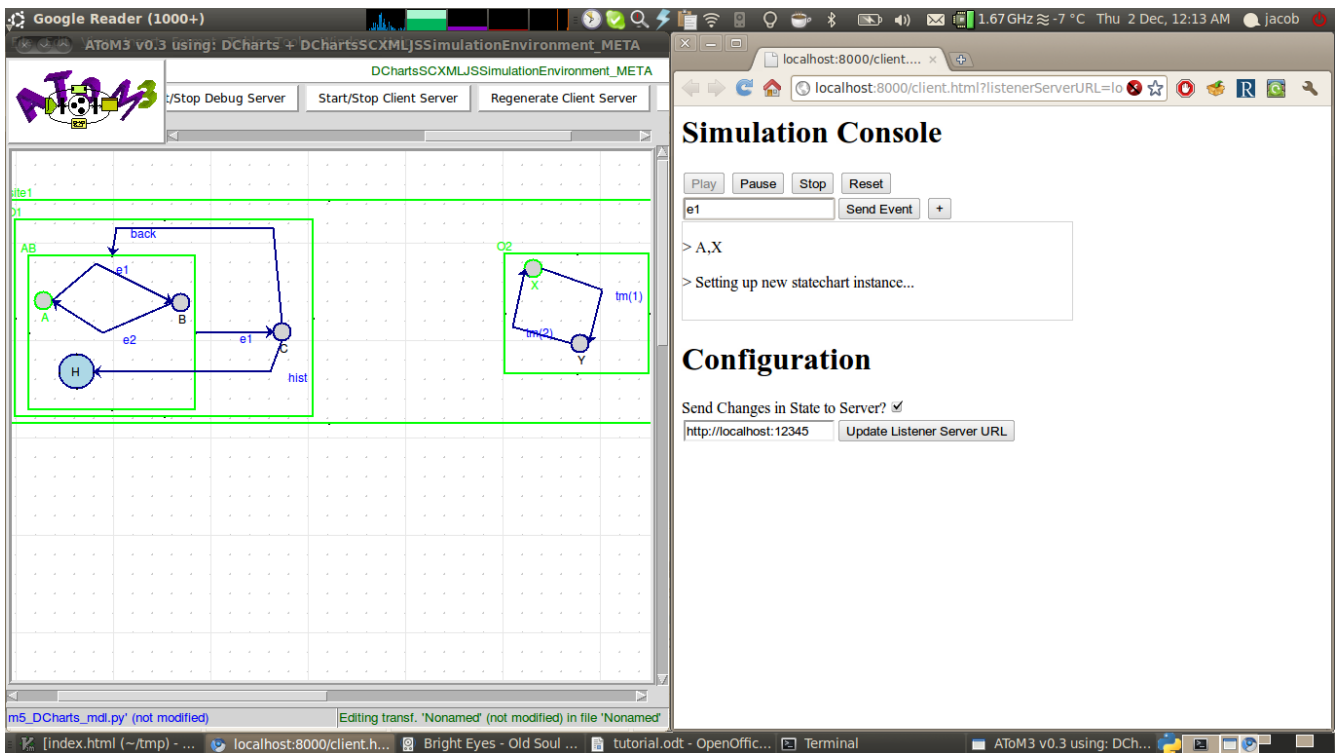
4. Work on your model.
5. When you are ready to simulate the model from the web console interface, click on *Start Client Server* and *Start Debugging Server*. You should see in the console that the servers have started. Your default web browser should open so that the web console interface is displayed:



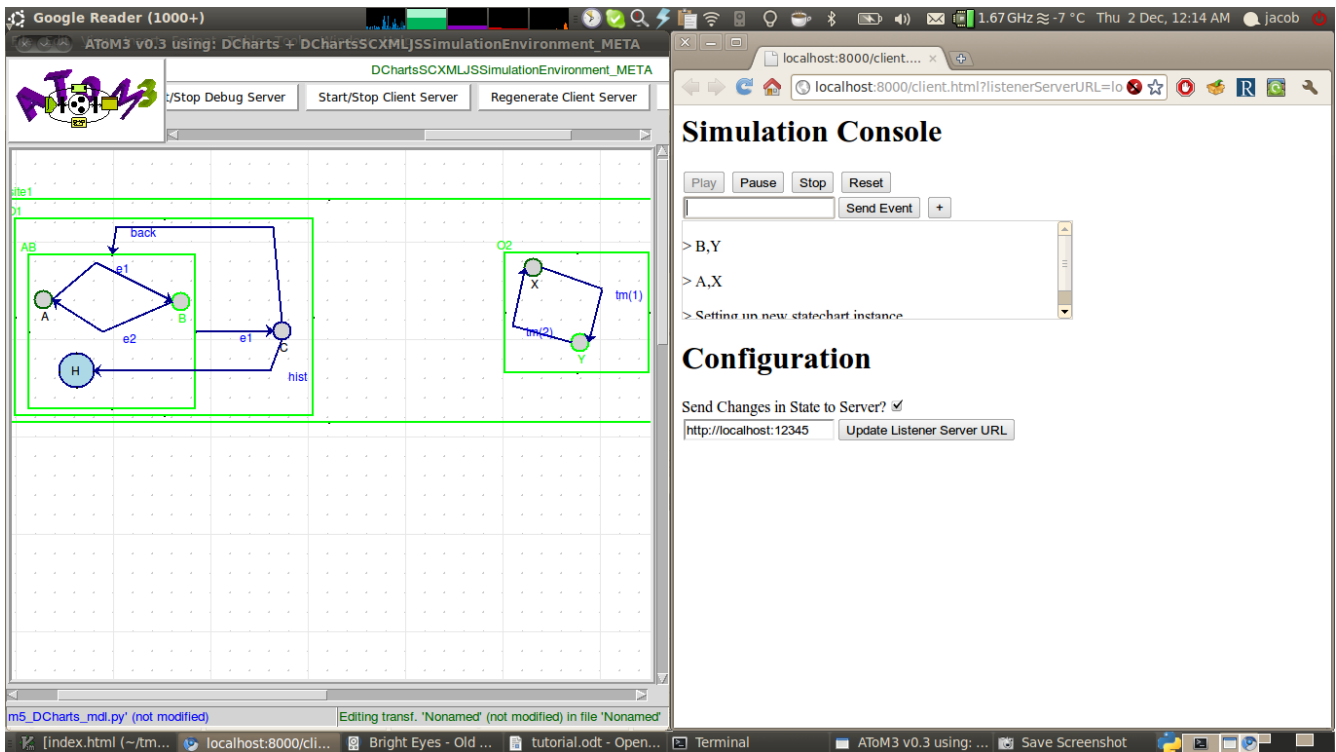
6. The console allows you to *Start*, *Stop*, *Pause*, and *Reset* the execution of your compiled DCharts model. Pressing the Start button should start simulating the model, so you should see something like this:



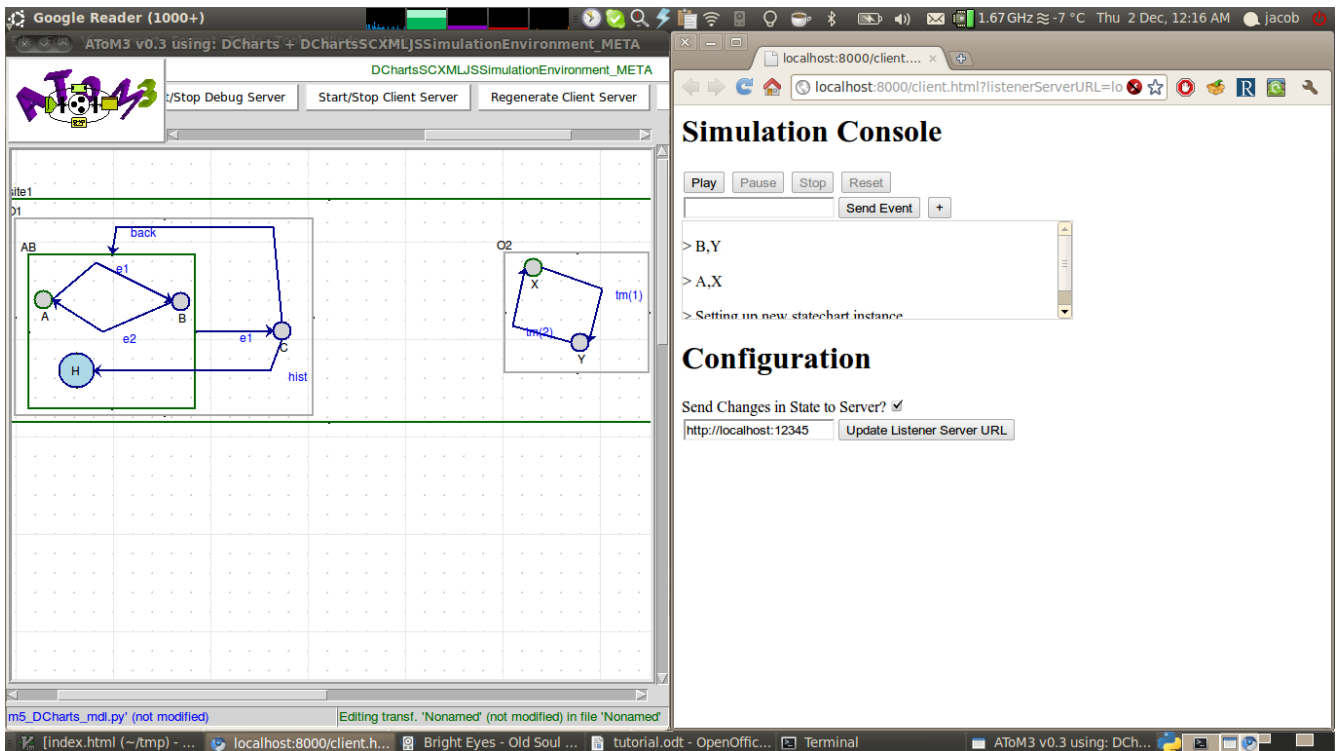
7. You can type into the box next to the button called *Send Event*, then either press <Enter> or click the *Send Event* button to send events into the Statechart. For example, here is how it looks sending in the event e1:



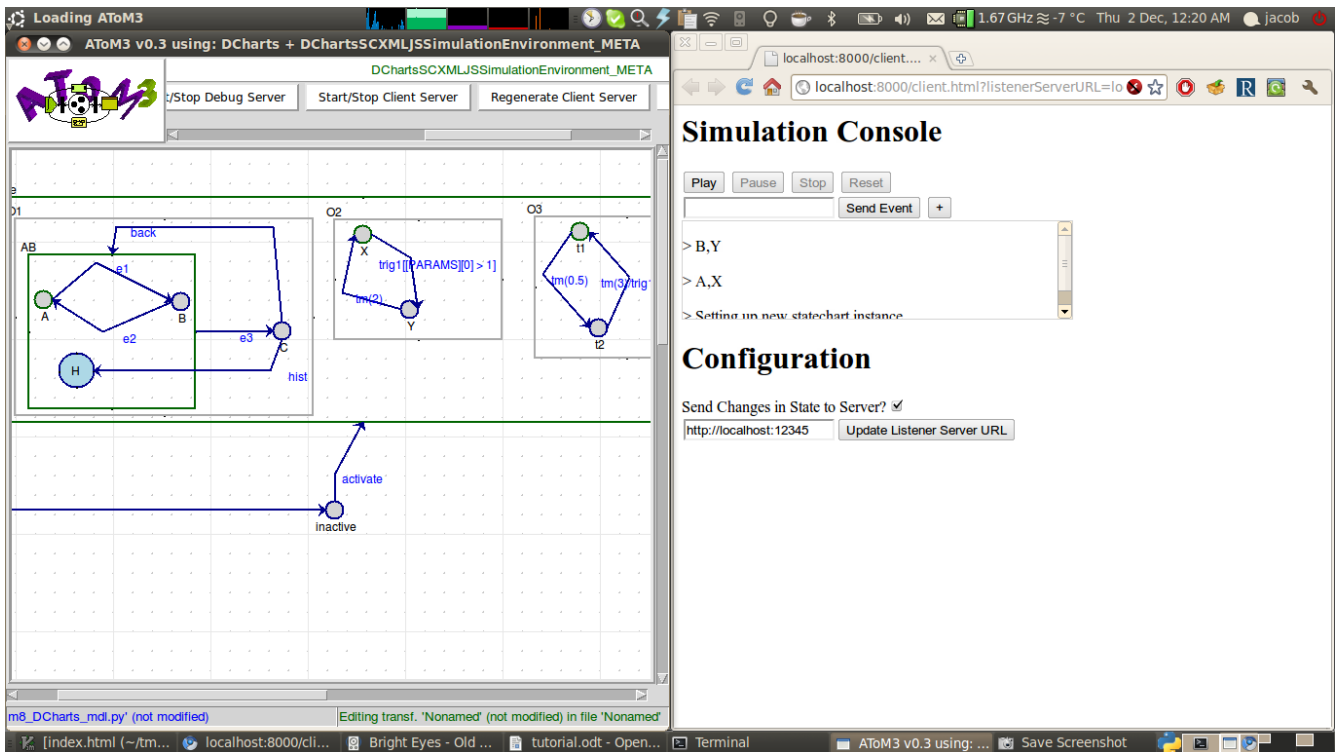
8. And here is how it looks after sending in the event:



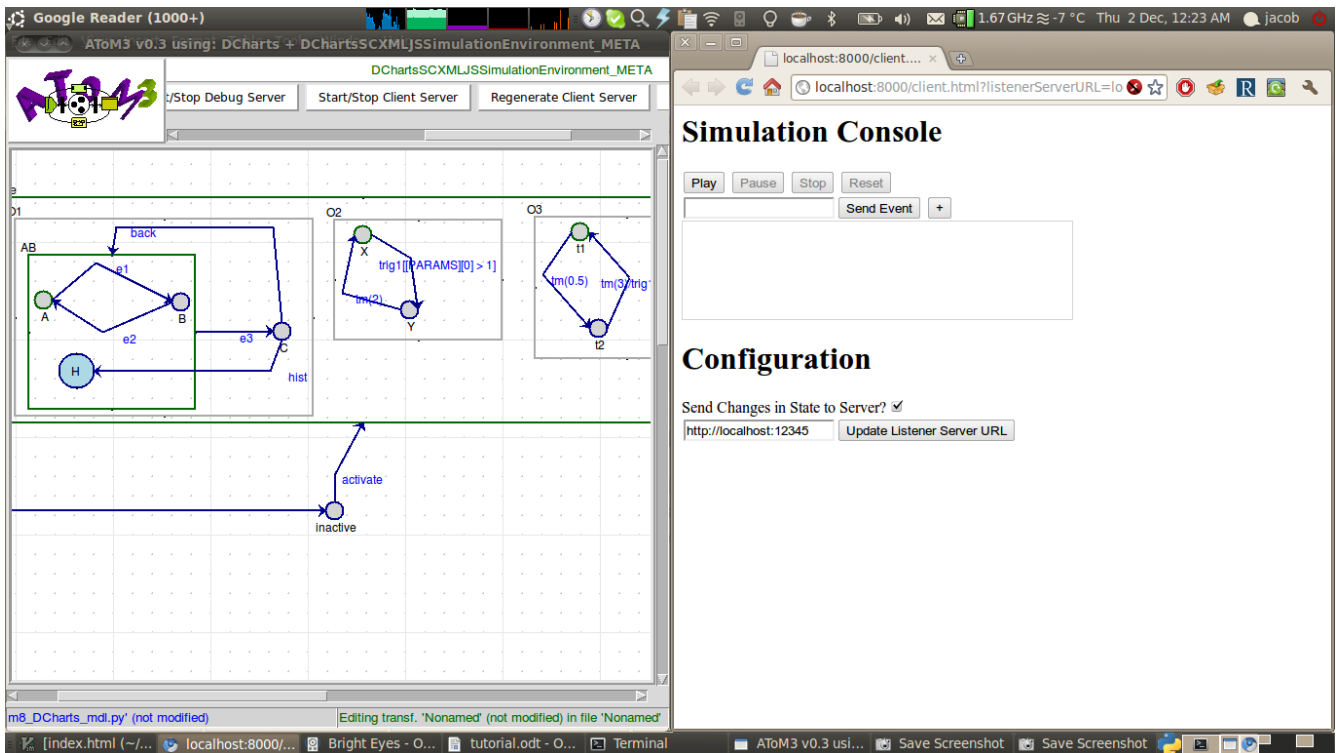
- To stop the model press the *Stop* button. This will stop the model from executing in the browser, and clear the visual DCharts model of highlighting:



- After stopping the model's execution, it is safe to continue editing your model, or to open a new model (when opening a new model, make sure that you close the old model first by pressing Ctrl+Delete). Here, I will open *Models/DCharts/m8\_DCharts\_mdl.py*:

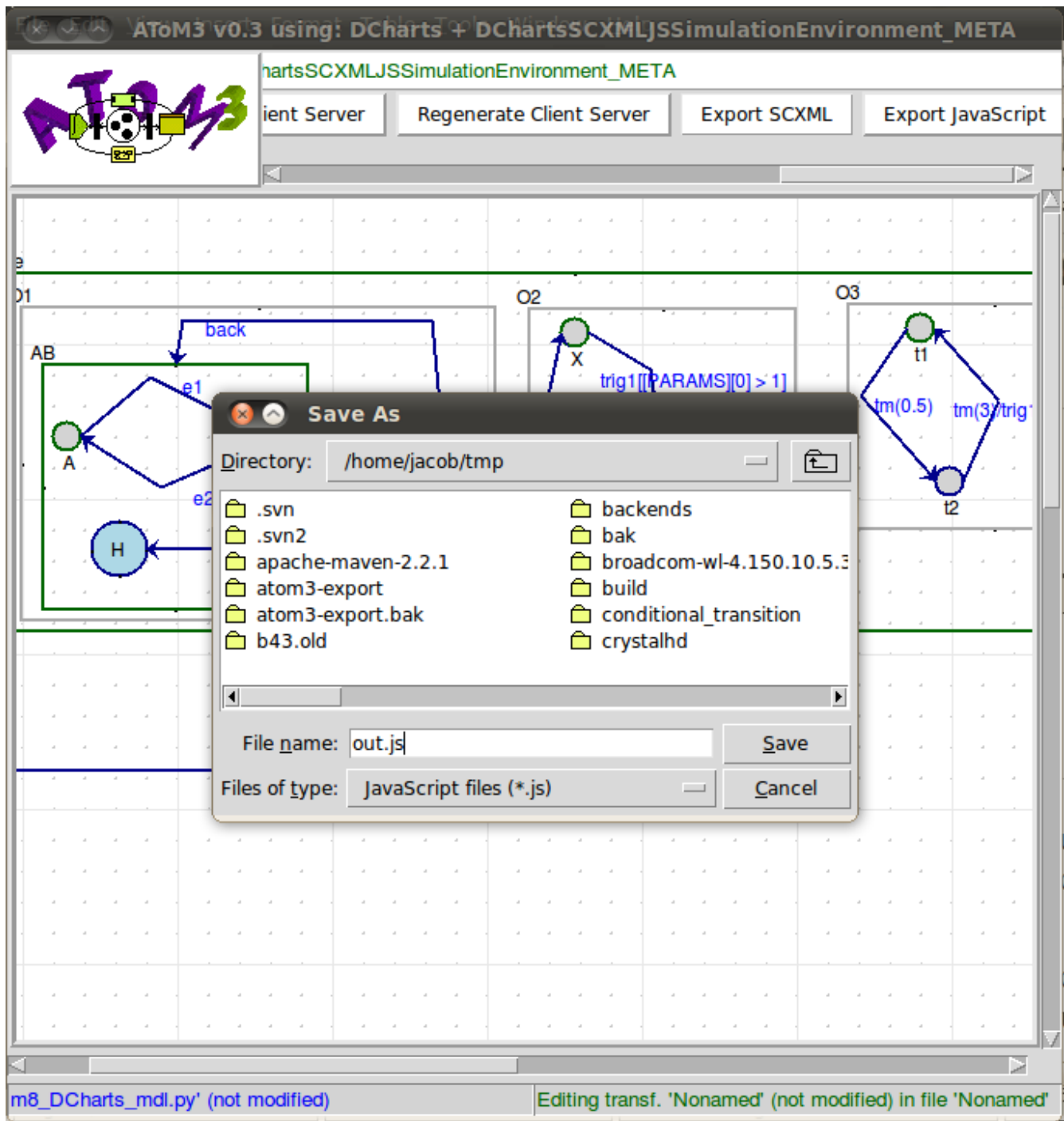


- To simulate this new model, you do **not** need to restart either of the servers. All that is needed is to regenerate the compiled, executable JavaScript from the DCharts model, and reload the web page so that it loads this new JavaScript code in place of the old code. To accomplish this, click on *Regenerate Client Server* in ATOM<sup>3</sup> and refresh the browser web page:



- You can now repeat follow steps 6 – 11 to iteratively develop and simulate/debug your DCharts model.

13. When you are satisfied that the your DCharts model is complete and correct, and wish to Export JavaScript code that can be run standalone, click on the button *Export JavaScript* in AToM<sup>3</sup>. It will prompt for a location to save the XML representation of the model, and the exported JavaScript code. By default, these files are named out.scxml and out.js, respectively.



14. You could then import this script into a web page by using a regular `<script>` tag (although this is already done for you in the Digital Watch assignment).