# Python Statechart Code Generator

(Last Updated: September 20, 2012, Accurev Snapshot: INTERNAL_JPL_Release_5August2011)

Team & Technolgies

This is an internal JPL web page providing the latest user's guide and distribution release of the JPL Statechart Autocoder and Python Quantum Framework. This release is considered a beta test release since few people have used it. Please use it and explore possible applications.

NOTE: Currently it is best to use this release with Python 2.5 because of bugs introduced in the xmlrpclib library of later versions of Python. XMLRPC is used as middleware to connect active state machine implementations to Python or C++ state machines. Also currently C++ state machines must be started manually after both C++ and Python are generated. This is done by first executing sim_state_start.py and then executing a valid linux/active built from C++. For Python 2.7 we have introduced a work around for errors associated with xmlrpclib. A new script called kill_sim_state.py was added to ease killing the Python scripts that in later versions of Python do not exit gracefully.

## Downloads:

- Python Statechart Autocoding User's Guide
- 5 August 2011 Python Statechart Autocoder Download Release

Actually the ***autocoder.tar*** code generator tool included in the above tar file is capable of generating Promela (design verification language) and C/C++ state machines based on the Quantum Framework public domain infrastructure. We currently do not have a user's manuals for ether the Promela or C/C++ code generation but it follows roughly the same process as for Python. The follow link will provide you with downloadable tar files that include extra support for these languages:

- 5 August 2011 Statechart Autocoder Release Downloads

## Release Notes:

1. Currently the Python Quantum Framework works on Mac OS X, Red Hat Linux and Windows XP. To validate that the autocoder.jar file is working and the Python state-machine Quantum Framework functions are correct we recommend executing the unit tests. This can be done by execution of the command

   ```
   <root>/QF_Py/bin/pythonsuite.py
   ```

   If desired you can also execute this command with the –g option as

   ```
   <root>/QF_Py/bin/pythonsuite.py –g
   ```

   and the tests will run while making various Statechart trace GUI widgets appear. New this release is the incorporation of XMP-RPC middleware between the target state machines executing within individual threads and the Statechart Python trace GUI widgets. The XML-RPC provides a language neutral layer that allows the generated Python trace GUI widgets to be used with virtually any language (e.g. Python, C/C++ or Java, etc.). Use of the XML-RPC also allows us to easily configure the software for execution of generated state machine code on remote machines as well.

2. Note that on some versions of cygwin you will need to do an "export PYTHONPATH" before executing sim_state_start.py. Also note that on cygwin you will need to execute sim_state_start.py by typing

   ```
   python –i sim_state_start.py
   ```

   with the python command explicitly given. We have not figured out why the cygwin bash shell cannot read the execution tag within the .py files.

3. There is still trouble building the autocoder.jar on Windows and Cygwin. For this distribution it should not be a problem since we have pre-build autocoder.jar and only include the Java binaries.

4. The following have been added or fixed in this version:

    - Added support for FinalState to Python backend, including cleaning up GUI state when resetting a StateMachine.
    - Added support for "else" or "no" as else branches of junction and choice pseudostates.
    - New XML-RPC GUI connection added. This fixes the non-support for subprocess.PIPE in Windows-native Python so that now generated Python all executes on Windows.
    - A new gui_server.py file replaces the gui.py legacy trace GUI widget handler. This is executed within the qf package. The qf.framework_gui.py module is smart and attempt retry connections to gui_server 10 times before throwing exception.
    - New capability in sim_state_start.py to generate Application.py with query method to check whether all windows are displayed. Previous this was done with a fixed time delay.
    - All regular python unit-tests successfully run on both 32 and 64 bit Linux, with or without GUI enabled (with GUI enabled).
    - On Windows-native Python, the GUI connector will now work – thanks to the addition of XML-RPC!
      Caveat: pexpect isn't supported in native Windows, so no unit-test run possible.
    - Within the Statechart the user can now use the "after" notation in addition to the "at" notation for timer event transitions. It is recommended that "after" is used in Statecharts to specify relative time delay per the specification.

5. The following UML State Machine Features are supported within this release:

- Simple States (or Leaf States)
- Composite States
- Orthogonal Regions in states
- Initial Pseudo-State
- Shallow History
- Junction and Choice pseudo-states
- Transition
- Transition to self
- Guards (using implementation functions that return true/false rather than expressions)
- Actions (Entry, Exit, Transition and signalEvent using implementation functions rather than code snippets)
- Sub-machines
- Threaded do activity actions
- Final Pseudo-state

3. The JPL Statechart Autocoder does not yet support:

- Terminate
- Entry point
- Exit point
- Connection Point Reference
- Deep History
- Fork/Join
- Guards using OCL logic expressions
- Actions using code snippets

Support:

The software was developed for the Ares1 Model Validation Task and SMAP Flight Software development. The Python generation has been successfully used on Ares1 is used on SMAP to support Fault Protection System engineers doing function requirements level simulation. At this time the Python code generation is no longer being actively developed, however, if a bug is discovered we are very interested to fix it and have the software used by others. We invite you to evaluate the tool for your project.  We are interested to hear your feedback and suggestions for other tools we might develop in the future.

If you have questions about the tools, suggestions for feature additions or wish to report a problem send email to the following people:

Garth Watney
watney@jpl.nasa.gov

Leonard Reder
reder@jpl.nasa.gov

Shang-Wen (Owen) Cheng
Shang-wen.cheng@jpl.nasa.gov