# SLAngMon - User Manual

Franco Raimondi

January 28, 2008

## Contents



## 1 SLAngMon Overview

SLAngMon is an Eclipse plugin to generate Java monitors for Service Level Agreements defined using the SLAng language. We refer to Deliverable 2.1 for a detailed description of SLAng.

SLAngMon implements the automata-based technique presented in D4.1 to monitor agreements by introducing AXIS handlers to intercept messages exchanged and to check their conformance with respect to the established SLAs.

SLAngMon is integrated into the SLA editor described in D2.2 and D2.3; we refer to these documents for a detailed description of how to create and manage a Service Level Agreement in SLAng.

## 2 Technical info

**Provider** University College London.

**Introduction** SLAngMon is a tool for the automatic generation of monitors from Service Level Agreements.

**Development status** The Eclipse plug-in is avaliable for download as part of the SLAng

**Intended audience** Developers who intend to implement monitors for Service Level Agreements.

**License** Mozilla Public Licence v.1.1 (MPL).

**Language** Java, EMOF/OCL, ant.

**Environment (set-up)** The following components are needed to compile SLAngMon:

- Eclipse 3.2 or greater.
- UCL MDA tools, available from `http://uclmda.sourceforgege.net`

**Platform** Eclipse. Operating System(s): all

**Download** The tool is available as part of the SLAng editor plug-in, available from: http://www-c.inria.fr/plastic/workpackage/wp2

**Documents** See D2.1 and D2.2 for a description of SLAng and its editor (Eclipse plug-in). Further documentation is available in D4.1 and D4.2 and from the website.

**Tasks** A binary distribution to avoid the compilation process and dependency problems is currently under development; please contact f.raimondi@cs.ucl.ac.uk for further information.

**Bugs** N/A

**Patches** N/A

**Contact** f.raimondi@cs.ucl.ac.uk

# 3 Deployment

## 3.1 Install

These are the steps required to install SLAngMon:

1. Open Eclipse.
2. Add the following CVS repositories:

   ```
   Host: uclmda.cvs.sourceforge.net
   CVS path: /cvsroot/uclmda
   User: anonymous (no password)
   ```

   In HEAD, checkout the following: UCL, GTL, EMOFOCL2, EMOFOCLPlugin
3. Compile UCL:
   - Rename project.properties.example in project.properties and edit
   - Execute build.xml
4. Compile GTL:
   - You need to obtain gnu.regexp-1.1.4.tar.gz, unzip, and place it in the lib/ directory (the archive can be obtained by looking for it in Google).
   - Execute build.xml
5. Compile EMOFOCL2
   - Rename project.properties.example in project.properties and edit
   - Execute build.xml
6. Compile EMOFOCLPlugin
   - Rename project.properties.example in project.properties and edit
   - Execute build.xml
7. Create a new general project with name SLAng
   - Unzip SLAng.zip and import everything into new project SLAng
   - Edit project.properties
   - Run ant target ”clean”
   - Run ant target ”all”
8. Create a new general project with name SLAngTA

- Unzip SLAngTA.zip and import everything into new project SLAngTA
- Edit project.properties
- Run ”all”

The plugin is at this point compiled and ready to be used.

# 4 Tutorial

SLAngMon output includes

- Java classes implementing the automata corresponding to checkers for SLAs.
- Java classes extending AXIS Basic Handlers, which should be invoked by appropriately modifying the chain of invocations.

For the purposes of this document, it is assumed that the reader is familiar with the following technologies:

- Apache, Tomcat, and AXIS.
- Eclipse and Eclipse plugins (with Ant builds)
- Meta-modelling, EMOF and OCL: please see Deliverables 6.2 and 2.1 for further details.

**Usage.** After compiling the software as described in the previous section, right-click on SLAngTA, Run As → Run... Run as Eclipse application. A new Eclipse instance should start. In this new instance, create a new General project. Then, create a new file, for instance `test.slangtaxmi` (notice: it is important to use `.slangtaxmi` extensions to activate the plugin). As an example, complete an InputThroughput clause, right click, and choose a destination directory. An example of the plugin at this point is depicted in Figure 1. This will generate a directory structure. The actual checker (the automaton) is generated in server/tautils. The handler for AXIS is generated in server/slamonitor

The generated java files can be compiled using the appropriate classpath for AXIS libraries. Copy the files and their container directories in `$AXIS_HOME/classes`

To install the handler, modify the file `$AXIS_HOME/server-config.wsdd`, for instance:

```
<handler name="slamonitor" type="java:slamonitor.SLAMonitor">
  <parameter name="filename" value="SLAMonitor.log"/>
  <parameter name="wsdlURL" value="/axis/SLAMonitorService-impl.wsdl"/>
  <parameter name="scope" value="Session"/>
  <parameter name="serviceName" value="SLAMonitorService"/>
  <parameter name="namespace"
     value="http://tempuri.org/wsdl/2001/12/SOAPMonitorService-impl.wsdl"/>
  <parameter name="portName" value="Demo"/>
</handler>
```

Finally, add the handler to you service (in `server-config.wsdd`):

```
<service name="MyService" provider="java:RPC">
    <requestFlow>
       <handler type="slamonitor"/>
    </requestFlow>
    <responseFlow>
       <handler type="slamonitor"/>
    </responseFlow>
[...]
</service>
```

The handler is now fully operational. Violations of the SLA clause are reported in the file `SLAMonitor.log`.

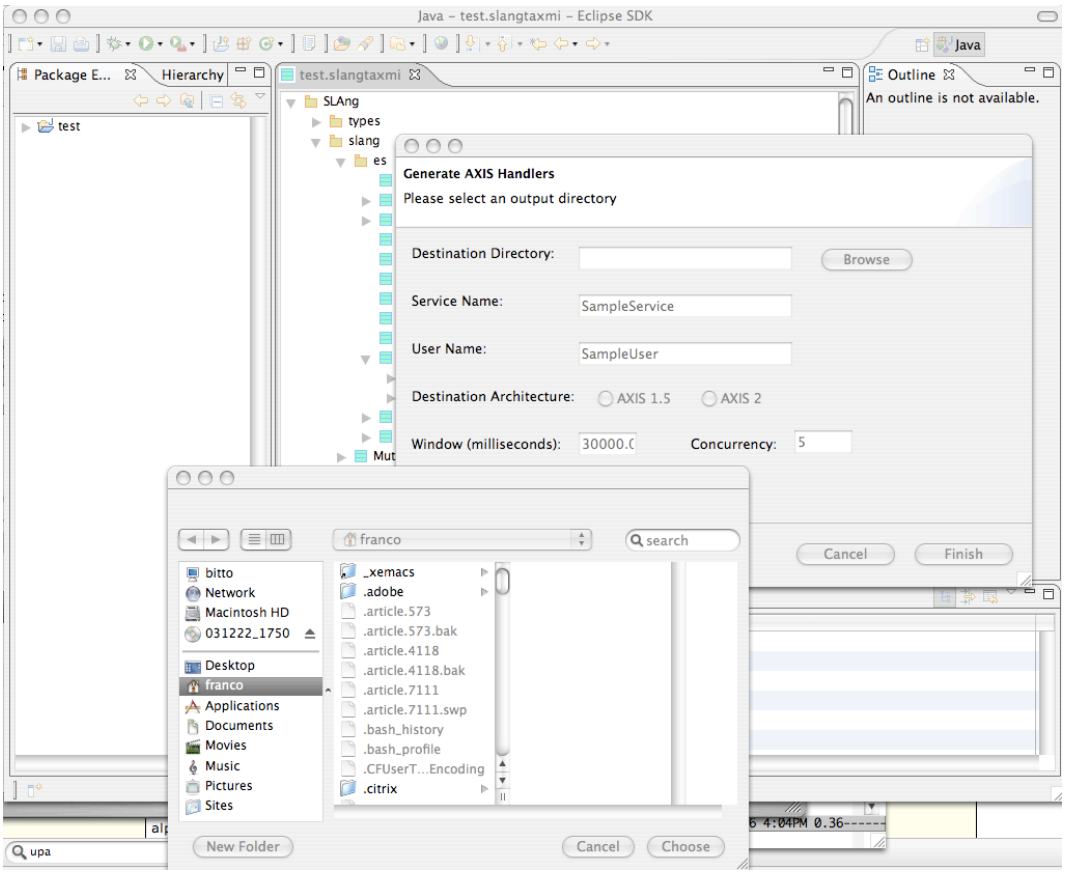


**Figure 1: Plugin screenshot for SLAngMon: generation of checkers.**

# 5 Appendix

This is a brief overview of the structure of the project:

- SLAng is defined in the file `src/uk/ac/ucl/cs/slangta/specification/slang.emof`. By modifying this file it is possible to define / modify SLAng clauses to define custom SLAs
- The SLA editor with the SLAngMon plugin is defined in src/uk/ac/ucl/cs/slangta/editor/SLAngTAEditor.java.
- The actual pluging to generate the checkers is defined by the files in src/uk/ac/ucl/cs/slangta/editor/action and src/uk/ac/ucl/cs/slangta/editor/tautils (this is the automata stuff)

By modifying any of the previous entities, SLAngMon can be easily customized to suit many applications scenarios.

## 5.1 FAQ

- **How do I create an SLA in SLAng using the editor?** Please see D2.1 and D2.2 for the description of the SLAng language and its associated editor.