SED Sherpa-SPECView SAMP Application

Project Test Plan

Project ID:	1.3
Revision:	1.0
Project Lead:	Janet Evans
QA&T Lead:	Brian Thomas
Document Status:	Ready for Review
Software Version:	1.0 - Year 1 Release
Reviewed by:	*{LEAVE BLANK (to be filled in later)}*
Review Date:	*{LEAVE BLANK (to be filled in later)}*

1 Goal

This document describes the details of how the Spectral Energy Distribution (SED) Sherpa-SAMP software will be tested through its life cycle for requirements to be implemented in Year 1 of the project.

2 References

- Sherpa-SAMP derived requirements and design. This document details the MTypes which are in use.
- SED PDD
- Main SED testing page. Contains links to other SED project test plans.
- IVOA SAMP specification
- Python SAMPY software

3 Project Description

The SED Sherpa-SAMP software is a component of the SED project. It is a sub-project which will provide the architecture and connectivity, using an extension to the SAMP protocol, between the SPECView and Sherpa software components. These sub-components (SPECView, Sherpa, Sherpa-SAMP) comprise the Iris application.

4 Test Environment, Tools and Execution Considerations

This component utilizes SAMP, a IVOA messaging protocol standard, for communication between the Sherpa and SPECView SED software components. It is built on top of SAMPY 3rd party python software (see reference).

The scope of testing for this service shall encompass static analysis and unit testing of this application using mock Sherpa and SPECView applications to ensure that all expected communications are being handled by the application. Acceptance testing shall comprise manual inspection of all associated documentation.

5 Additional Information

To faciliate reporting, unit test cases are to be implemented in files which are clearly labelled as that test case, ex. "TestCase3.py" and aggregate all of the unit tests for that test case within them. Alternatively, in the test cases listed below, the individual files which execute the (automated) tests for each test case may be indicated.

Document(s) which lays out the basic Sherpa, SPECView interfaces to SAMP and the MTYPES which will be utilized/handled by the app (e.g. the design doc) should be indicated in a document link in this test plan.

6 Test Cases

6.1 Static Analysis Tests

TestCase0 : Static Analysis of Code

Procedure: Incorporate pylint and nosetests software into build.

The build process shall incorporate Static Code Analysis software in order to catch defects. Both pylint and nosetests software shall be incorporated, and no defects of "Fatal/5" or greater shall be allowed.

6.2 Acceptance Tests

TestCase1 : Artifact Completeness

Requirement Identifier: **???**
Procedure: Manual inspection of project artifacts.

The system shall be checked for deliverable artifact completeness as specified by Requirements **???**. Documentation will be checked for completeness, and comprehensibility.

6.3 Unit Tests

TestCase2 : VAO Platform Build Compliance

Requirement Identifier: **???**
Procedure: Automated test? using Hudson build **???**

The application will built on one or more supported VAO platforms (Fedora Core 14).

TestCase3 : Compatibility with Sherpa and SPECView SAMP interfaces

Requirement Identifier: SED.an.1, SED.an.4.3, SED.an.4.5
Procedure: Automated test using Jenkins CI/Build environment

This testcase will check that a mock Sherpa instance can read/load in data via SAMP application which has been passed to it from a mock SPECView application. This test shall also test that mock Sherpa instance can pass back fitted model information (fitting statistics, confidence levels, to mock SPECView instance via the application.

These data should be based on an aggregate SED. This test case shall exercise Sherpa-SAMP interface to test that all of the Sherpa models can be passed to/from (Iris/SPECView) to Sherpa via Sherpa-SAMP framework. Models must be specified both singly, and for selected models, should be specified as part of an algebraic formula which Sherpaunderstands to fit multiple model components. Finally, the interface must be tested for specifying a range of spectral coordinates to be considered which can be either the whole range or multiple disjoint intervals of spectral coordinates. **Note: aspects of the above para *could* be broken up into sub-test cases for ease of reporting, and designing tests. The expectation is that each of the**

MTypes which have been declared in the design document are going to be unit tested for correct behavior. This includes error conditions which are specified in the design, e.g. pg 17 of the design document.

So, in summary, we are looking for a grid of unit tests which is approximately (an upper limit) the number of MTypes X Number of Models X (# error conditions + 1). **This is a lot of tests, but automating the test harness should allow for it to be done relatively easily.**

7 Schedule

Activity	**Task**	**Dependencies**	**Responsible party**	**Start Date**	**End Date**	**Comments**
Static and Unit Tests	execution	Model List and SED data specified for testing	SED SAMP/Sherpa team	April 2011	Jul 2011	Verify unit tests during beta test 1,2 and at delivery
Acceptance Test	execution	Passed prior testing	QA&T team	April 2011	Jul 2011	Verify project deliverables.