Testing Interoperability of SML & SML-IF Implementations

Change Log:

Date	Person	Remarks
4/4/2008	Kumar Pandit	Created initial draft.

1. Overview

In order to progress from Candidate Recommendation status to Proposed Recommendation, SML and SML-IF must satisfy the exit criteria for Candidate Recommendation agreed upon by the Director and the chairs of the SML Working Group. At a minimum, the W3C process document requires that each specification have at least one implementation of each feature, and preferably two inter-operable implementations; additional exit criteria may be (and often are) agreed by the Director and chairs. As a consequence, the basic requirement for our test plan is that the SML WG must construct a test suite suitable for documenting (a) which features of the spec have been implemented and (b) for each feature implemented more than once, whether the implementations are consistent and interoperable. This document defines the approach adopted by the SML working group to meet that goal. ~~The SML working group is required to have at least one implementation of each SML & SML-IF required feature and preferably two inter-operable implementations in order to meet the Candidate Recommendation (CR) exit criteria. This document defines the approach adopted by the SML working group to meet that goal.~~

This is defined in the following sections:

1. Test packaging
2. Test storage and organization
3. Test execution
4. Analyzing test results
5. Test Cases

2. Test Packaging

Each test involves processing a set of schema/rule documents and instance documents. There are two basic approaches to package these documents.

1. Keep each document in its own file and have a test-description file per test that points to files involved in that test.

   Although this method saves disk space by eliminating duplication of files, it has the following disadvantages:

   a. Change to a single file potentially affects multiple tests.

b. The test-description file must store information about file category (definition/instance/rule), file URLs (aliases), schema-completeness, rule bindings, etc. In other words, this file must define and use syntax that is already defined for the SML-IF files.
c. Sending a complete test case to someone is complicated because one needs to carefully copy all required files and the associated test-description file.
d. A test harness must have additional code must be written that understands how to use the information in the test-description file.

2. Package all files required for a test in a single SML-IF file.

   Although this method requires more disk-space than the previous method, the actual amount of disk space additional disk-space is insignificant compared to current disk sizes. This method has the following advantages:
   a. Changes to each test are localized to that test.
   b. No special syntax needs to be invented because SML-IF already defines it.
   c. Sending a complete test case to someone is easy because one needs to send only 1 file.
   d. An SML-IF consumer already knows how to process an SML-IF file therefore that part does not have to be implemented by a test harness.

In view of the above, each SML test case is packaged in a single SML-IF file. There is only one exception. Testing non-embedded documents pointed to by a locator element requires multiple files per test. However, since support for the locator element is optional, that test is not included in interoperability testing. Note that a test for the locator element is still required to prove that an implementation, that supports locator elements, processes them correctly.

3. Test Storage and Organization

Test cases and associated files are stored on W3C servers so that they can be accessed by people who need them. There are less than 200 test cases that cover SML & SML-IF. This allows the test cases to be stored in a relatively flat directory structure. They are stored under the sml directory in the cvs repository as shown below.

- SML
  - test
    - testsForRequiredFeatures
    - testsForOptionalFeatures
    - documentation

The testsForRequiredFeatures directory has the SML-IF files that contain tests for the required features. The testsForOptionalFeatures directory has the SML-IF files that contain tests for the optional features. The documentation directory has test documentation, including this document.

4. Test Execution

How the test case files are supplied to a test harness and how the test harness is invoked depends on each implementation. This document does not define requirements for the following:

1. Hardware configuration
2. Operating System or its version
3. Test execution behavior of any test harness.

5. Analyzing Test Results

As mentioned earlier the SML working group is required to have at least one implementation of each SML & SML-IF feature and at least two inter-operable implementations. The process of determining whether all features are covered by the test cases cannot be automated. The working group must determine this based on a review of the test cases. However, the inter-operability can be determined in an automated fashion.

For the purpose of this test plan, two implementations are said to be interoperable if they produce identical model validation result for each test case that tests a required feature of SML and SML-IF. Two implementations are allowed to produce a different model validation result for a test case that tests an optional feature.

Comparing test results is trivial where the results indicate a valid model. Comparing test results where a model is invalid is not so easy. This is because a model could be invalid because of a number of reasons and the SML & SML-IF specifications do not define an inter-operable way to encode or compare the reasons. This is because of the following:

1. The SML specification does not define the order in which model validity assessment steps must be carried out. One implementation may evaluate id constraints before Schematron constraints and some other implementation may do it in the reverse order.
2. The SML specification does not define whether model validation must stop at the first error and whether all possible errors are returned to the invoker. This combined with the previous item can result in two implementations producing two different results for the same invalid model.
3. The SML/SML-IF specifications do not define specific error codes or error messages that represent an error condition. This means that even though two implementations find the same error in a model, they may produce different error codes/messages. There is no inter-operable way to compare them.

As a result of these difficulties, this test document does not require automated granular comparison of test results between two different implementations. The result comparison is limited to comparing model validity as a Boolean value.

Each participating implementation must pass each test that tests a required feature. A test is said to pass if the actual model validity result, expressed as a Boolean value, is identical to the expected result.

If an implementation produces descriptive error messages when model validity is assessed, such messages are compared manually with the results from other implementations for the same model. The comparison of such error messages cannot be automated for reasons mentioned earlier.

The name of each test file includes the test name followed by the expected result. This obviates the need to maintain a separate test metadata file. For example,

id-constraint-KeyMissing-invalid.xml
ref-dangling-valid.xml

6. *Test Cases*

As defined earlier, model validation results are compared as a Boolean value. Test cases are written such that they focus on a single issue at a time and consequently they result in a single model validation error. This avoids ambiguity in cases where a model may be invalid due to multiple reasons.

The following sections lists all test cases used for interoperability testing. They are divided in 2 parts,

1. Tests for required features:
   ~~Each participating implementation must pass all tests in this section.~~ These features are "required" in the sense that conforming processors must support them. If any partial implementations participate in the testing effort, those partial implementations may not pass all of tests in this group.

2. Tests for optional features:
   Not all implementations support optional features, therefore the behavior of an implementation cannot be reliably compared with that of another when an optional feature is involved. An implementation may silently ignore the presence of an optional feature or it may produce a fatal error or it may produce a warning if required by SML or SML-IF specification.

Tests for Required Features

tbd

Tests for Optional Features

Tbd

Open Issues

1. We need to specify behavior when an optional feature is not supported.
2. Define directory structure to hold files related to interop testing.

Formatted: Numbered+ Level:1 + NumberingStyle:1, 2, 3, … + Startat: 1 + Alignment:Left + Alignedat: 0.25" + Indent at: 0.5"

3. Need to decide test result format.
4. Decide whether two implementations are allowed to produce a different model validation result for a test case that tests an optional feature.
5. Need test cases for validation of SML-IF format.