
Testing POSIX® Realtime Systems

Andrew Josey

ajosey@opengroup.org

THE *Open* GROUP

This talk covers

Status update of POSIX RT Test Project

Next Phases

Development process

POSIX Realtime amendments

.1d, POSIX P1003.1d-1999 , Additional Realtime Extensions

.1j, POSIX P1003.1j-2000, Advanced Realtime Extensions

.1q, POSIX 1003.1q-2000, Tracing

The POSIX RT Test Project

Initially builds upon existing test tools used for POSIX testing.

Phased development - first two phases now complete

Scoping phase

test suite specification

analysis of reqts for testing .13/.13a

profiles to propose additional project phases

Phase 1

Scoped the project, produced a detailed project plan and test specification.

This phase also included a study of the POSIX .13/.13a profiles and proposed further phases beyond those specified to address conclusions of that study.

These are documented in the white paper

Phase 2

Test assertions deliverable for 1003.1d
and 1003.1j

Test development for 1003.1d and
1003.1j

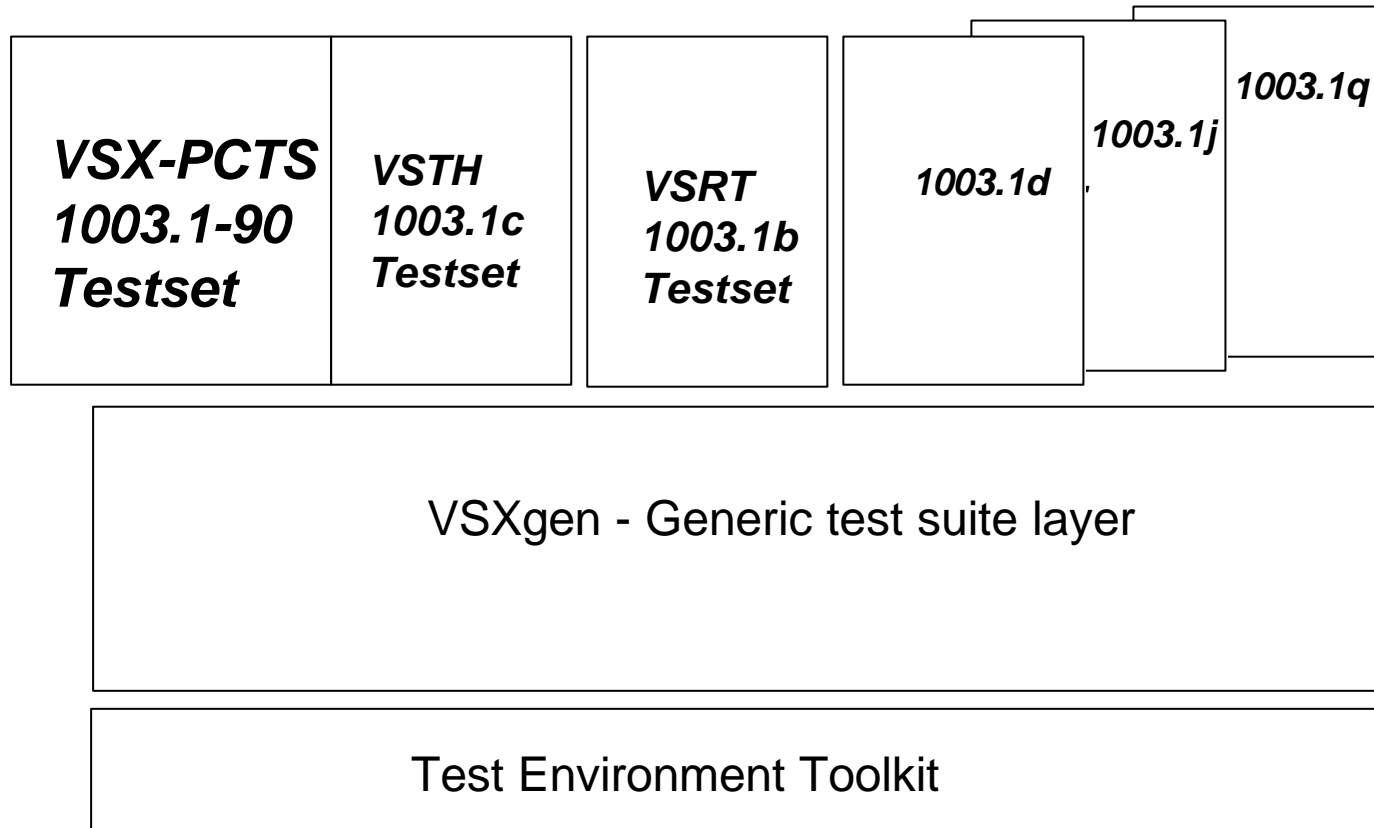
Completed May 2000

Phase 3

Test assertion deliverable for 1003.1q

Test development for 1003.1q, in progress will complete in early November 2000

Test Suite Architecture

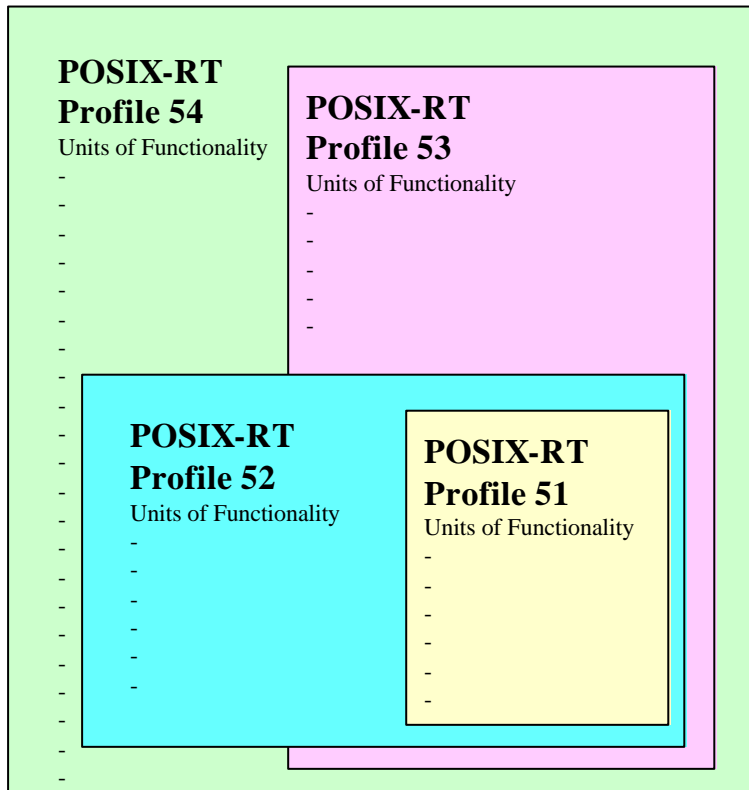


Future Phases

Address .13 Profiles as per 1003.13

Approach outlined in the white paper

POSIX RT Profiles (1003.13)



Portable Operating System Interface (POSIX)
IEEE Industry Standard
ISO 9001 Industry Standard

Profile 54: All POSIX.1,
Multi-process, Threads, File System

Profile 53:
Multi-process, Threads; No File System

Profile 52:
Single Process, Threads, File System

Profile 51:
Single Process, Threads; No File System

POSIX RT Profiles

Test suites only exist for Profile 54 and larger systems

Are POSIX profiles sufficient for real-world solutions?

Do we need to add further to them?

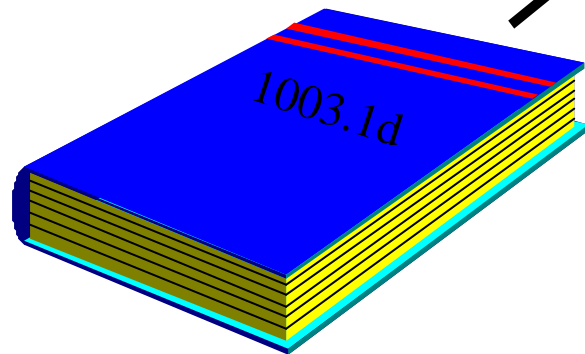
Test Development Process

Objectivity and traceability of test code to a written specification are the key

The written specification is the definitive reference

In cases of doubt assertions and tests are secondary and defer to the specification.

Traceability



Assertion 3.1.6.1-1(A)
If the value specified by
attrp is invalid
posix_spawn() returns
EINVAL

```
test1()
{
  vsx_assert 3.1.6.1-1(A)
  vsx_tfunc 2>&1
}
```

Assertion 4.8.1.2-1(C)
If the Spawn option is
supported then
sysconf(_SC_SPAWN) returns 1

```
test2()
{
  vsx_assert 4.8.1.2(C)
  #ifdef _POSIX_SPAWN
  ..
  #else
  .
  #endif
}
```

Test Process overview

1. Development of test specification
2. A set of formal test assertions
3. Test suite developed and taken through a controlled development cycle before approval
4. Test suite moved to support & maintenance mode.

Test Suite Specification

A high level description of the test suite

test methodology

functional coverage

test architecture

assumptions

development schedule

Formal Test Assertions

Bridges the gap between language specification and test suite code.

The goal of test code is to produce an executable representation of the requirements of the specification under test.

Assertion Based technique

Follows the IEEE Standard for Test Methods (1003.3).

An assertion is developed for each definitive statement in the specification under test.

Each assertion is designed to test whether the statement is true or false for the implementation under test.

Test Assertion Development

The key is to produce logical, testable statements that can be reasoned about, these are called test assertions.

By producing test assertions, we can then produce tests (which maybe produced manually or if using Assertion Definition Language automatically).

Typical test suites can contain thousands of test assertions.

Test coverage

Assertions also need to consider the level of testing. POSIX.3 defines three levels:

- Exhaustive

- Thorough

- Identification

Reviews

Used to validate the test suite

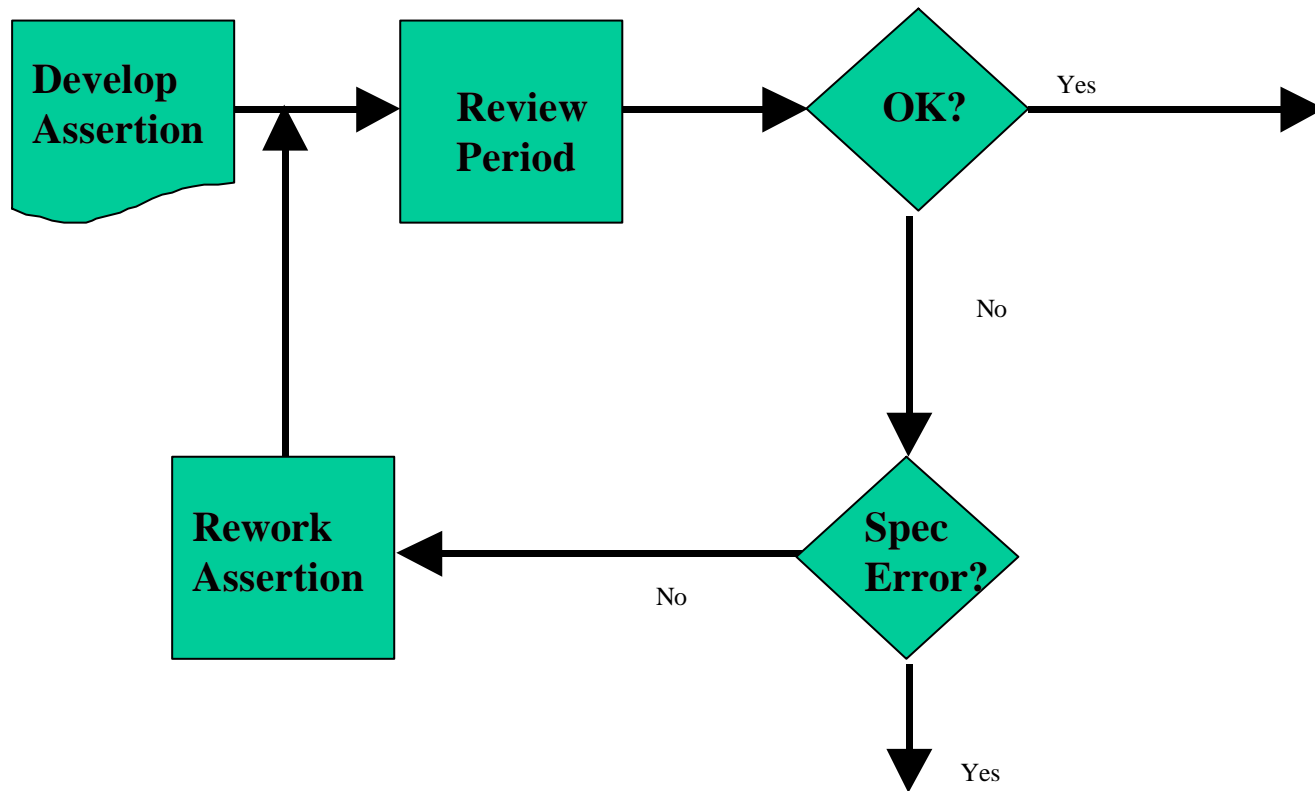
Important to manage the review of feedback

Use review guidelines and pro-forma responses

Comments should be specific

Comments should include a specified action

Review Model



Problems with Assertions.

Test assertions can be invalid for two main reasons

- Invalid assumptions

- Specification errors

Specification Errors

Highlighted by test suite failures

Should be determined by the owners of the specifications, and not the test suite author!

Should be fed back to the specification owners so that corrective action can be taken.

Test Development

Once the test assertions have been reviewed, test development proceeds

Reviews needed to validate test development

- Alpha test

- Beta test

Experience

Writing test suites can make you unpopular!

Testing ultimately leads to better products and to better specifications