Developing Technical Specifications
Guidance for Working Groups

Prepared by

The Open Group
July 18th, 2005
Revision 0.3

© 2005 The Open Group
ALL RIGHTS RESERVED
Table of Contents

1. Introduction ............................................................................................................. 3
2. General Guidelines .......................................................... ................................. 3
3. Specific Guidelines ................................................................................................ 3
4. Writing Style ........................................................................................................... 5
5. Review Guidelines ................................................................................................ 5
6. Conformance Guidelines ......................................................................................... 6
7. Test Development Guidelines ................................................................................. 6
   7.1 Criteria for Test Development ................................................................. 8
       7.1.1 The Test Specification ........................................................................ 8
       7.1.2 The Test Suite .................................................................................. 8
1. Introduction

This document is intended to provide general guidance to Working Groups engaged in developing technical specifications. It has a particular emphasis on testability, and focuses on issues which will help to make the resulting specification clear and unambiguous, with the goal that implementations will be capable of being tested, either by automated test means or by clear and verifiable assertions of conformance by the implementor.

This document consists of a series of guidelines, the observance of which will help to achieve the goals of clarity and lack of ambiguity. It is important to be aware that this set of guidelines is not exhaustive, and should be backed up with a healthy application of common sense.

2. General Guidelines

Most technical specifications contain two sorts of information

- Normative information is information that defines the technical specification for the purposes of determining conformance. It may include both mandatory and optional parts.
- Informative information is provided as explanation and guidance. It does not form part of the specification for the purposes of conformance. In the event that there should be contradictions between normative and informative information, the normative information is always definitive.

It is very important to separate clearly the normative and informative parts of the specification. This can either be done physically (by creating separate chapters or appendices), or typographically (by means of distinct fonts or the use of differing colors).

An important area to take into consideration is that of inter-specification and intra-specification dependencies. Many specifications have external dependencies on other specifications. Care must be taken to ensure that these are clearly spelled out. In addition, there is a need to ensure that the external specification does not change in such a way that it fails to meet the requirements. This can be achieved by making references to specific versions of external specifications, rather than to generic specification titles.

Internal dependencies can be created as a result of optional parts of the specification. Care must be taken to ensure that the optional parts are clearly marked as such, and that there are no dependencies to optional functionality embedded within required parts of the specification.

3. Specific Guidelines

Technical specifications need to be written in very specific, well defined terminology that makes clear what behavior is mandatory and what is optional. The exact terminology used is not important, provided that it is properly defined. The following terms used by The Open Group are offered as an example:
can
Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms the specification. An application can rely on the existence of the feature or behavior.

implementation-defined
Describes a value or behavior that is not defined by the specification but is selected by an implementor. The value or behavior may vary among implementations that conform to the specification. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations.

The implementor shall document such a value or behavior so that it can be used correctly by an application.

legacy
Describes a feature or behavior that is being retained for compatibility with older applications, but which has limitations which make it inappropriate for developing portable applications. New applications should use alternative means of obtaining equivalent functionality.

may
Describes a feature or behavior that is optional for an implementation that conforms to the specification. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

To avoid ambiguity, the opposite of may is expressed as need not, instead of may not.

shall
For an implementation that conforms to the specification, describes a feature or behavior that is mandatory. An application can rely on the existence of the feature or behavior.

For an application or user, describes a behavior that is mandatory.

should
For an implementation that conforms to the specification, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

undefined
Describes the nature of a value or behavior not defined by the specification which results from use of an invalid program construct or invalid data input.
4. Writing Style

These general guidelines are designed to take account of internationalization and translation considerations:

- Sensitivity to your readers' needs is important. It is necessary to start with a good understanding of who your readers are likely to be, and their technical expertise. This should be clearly identified in the Intended Audience section of the Preface.
- Anticipate the readers' questions.
- Provide information clearly and concisely, so that readers can find information quickly.
- Keep sentences short — preferably less than twenty five words (approximately one and one half lines) — and clear (but complete).
- Use consistent language, terminology, and typographical conventions. Try to avoid the use of synonyms.
- Make sure statements are unambiguous — do not use contractions.
- When choices exist, explain the advantages and disadvantages of the alternatives.
- Avoid humor, jargon, irony, idioms, adages, slang, sexist language, and political and religious references.
- If necessary, when using computer terms that can be interpreted as jargon, make sure that you and the reader understand the meaning of the word as used in your document. Ideally such terms should be included in the Glossary.
- Define key terms at the first mention, if the terms may be new to the reader. (And add to the glossary.)
- Avoid general modifiers, such as “nice.”
- Avoid long strings of modifiers. For example, “The previously sent destination protocol address ...” would be better worded as “The destination protocol address that was previously sent ...”
- Do not use the same word in different grammatical categories.
- Avoid using symbols to represent words, such as an ampersand (&) to represent the word “and.”
- Avoid referring to the authors. If such a reference is essential, use “The Open Group ...”
- Use x to refer to a generic letter, and n to refer to a generic number.
- Do not use double spaces between sentences
- Do not refer to holidays.
- Do not use analogies and terms based on local culture.
- Do not use non-English abbreviations and terms (see Appendix A on page 55).
- Do not use terms that attribute human characteristics to software, including gender.
- Do not use words that do not appear in the dictionary (see Section 2.33 on page 19).

These guidelines are taken from The Open Group Technical Publications: Writing Style. This document may be found at http://www.opengroup.org/bookstore/catalog/1801a.htm.

5. Review Guidelines

When undertaking a review of a specification (which could include test assertions) it is important to have clear review instructions to be followed that will lead to clear comments.

An example follows:

Comments should take the following format, which allows us to collate comments by section.

Subject: BUG in Specification Name, Version Identifier

Include Assertion reference number
Problem:

*Explain why here. Be sure to add sufficient explanation to enable someone not familiar with the problem to be able to make a decision.*

Action:

*Be specific; that is give precise editorial instructions for change.*

For example

Specification Ref: 3.1-1

Problem:

*Tool-4* may not be provided if the *Class A1* extension is not supported. The specification says clearly that the Class A1 extension is optional.

Action:

Change Section 3.1-1, lines 4-6

From:

3.1-1 The implementation shall provide *Tool-4* on the default screen menu.

To:

3.1-1 If the implementation supports the Class A1 extension then *Tool-4* shall be provided on the default screen menu.

6. Conformance Guidelines

Careful consideration needs to be given to testability. When specifying behavior, a key question to ask is “Can this behavior be measured/tested?” If the answer is “No”, there is probably little benefit in specifying it. Similarly, any behavior which occurs internally to a system conforming to the specification, and which does not affect any externally observable behavior, has little benefit. For example, if the specification defines methods for storing and retrieving some item of data, there is no benefit in requiring any specific method of internal storage, provided that it is capable of returning stored values without modification.

This implies that any behavior must be accompanied by an externally observable metric.

English, or any other natural language, is an imprecise tool for specification. It is always worth considering the use of a formal notation for testable entities. Possibilities include the use of VDM, Z, and ADL. But, assuming that the specification is intended to be read by people rather than by machines, a natural-language translation of the formal-language text should always be provided.

A basic rule for conformance is to recognize that the authors of the specification are the authoritative body for interpretations of the specification and not test authors.

7. Test Development Guidelines
The process for developing a set of tests (termed a test suite) involves a multi-step process where objectivity, and relationship to the written specification for which the tests are being written is the key.

The first step is development of a test specification that should undergo formal review. Secondly, a test suite should be developed based on that test specification. At regular points there should be feedback from the customer audience including the underlying specification owners to verify any assumptions.

Typically a test specification should follow the IEEE POSIX 1003.3 guidelines and consist of a set of assertions, that is a set of statements, which evaluate as true or false based on the specification under test. This can be thought of the translation between the natural language specification and the testable specification.

For example, there might be a statement in the specification:

**Section 3.1 Required tools for Class A Conformance**

*The following tools shall be provided in the default screen menu for products meeting Class A conformance.*

*{A list of 34 tools}*

This is written as 34 separate test assertions, for example

Reference 3.1-1 (A)

The implementation provides tool tool-1 on the default screen menu.

Result: PASS/FAIL

Reference 3.1-4 (A)

The implementation provides tool tool-2 on the default screen menu

Result: PASS/FAIL

etc.

The above example is a simple case; more complex examples are for conditional tests. The key is to produce logical, testable statements that can be reasoned about, these are called test assertions. The test assertions should be reviewed and, once they are approved, tests can be produced. Such tests might be automated or they might be manual procedures based on the assertions.

In order for development to proceed in an objective fashion, review guidelines and a pro-forma response are used for getting feedback from the community that should include the underlying specification authors if possible. Review comments are then collated and then a set of proposed resolutions produced.

The review comments together with the proposed resolutions are then circulated to the review group for confirmation.

This allows test development to proceed based on the specifications. At this point an iterative process occurs as test assertions are validated or shown to be false. This may be due to an incorrect assumption by the author of the test assertion, or in some cases what is now considered an error in the underlying specification. The authors of the specifications should only decide the latter, and not the test suite author, and these issues should be fed back to the specification owners so that corrective action can be taken. The test suite should note any issues in its output,
but the final correct strategy cannot be undertaken until the specification owners have issued corrections or possibly a version of the specification.

7.1 Criteria for Test Development

The following are a set of sample criteria that a test specification and test suite should meet.

7.1.1 The Test Specification

The premise for all the tests is that they have to be based on a written, publicly available test specification.

The test specification upon which the test suite is based shall conform to the POSIX.3 methodology (which means use of test assertions).

7.1.2 The Test Suite

The test suite is aligned with the current draft/issue of the relevant specification. (Mandatory)

The test suite shall be able to establish conformance with all legitimate options and variables defined by the specification. (Mandatory)

Each individually executable test shall provide its own setup and clean up functionality. (Mandatory)

No individually executable test shall introduce side effects that will affect the results of any other tests. (Mandatory)

The unit of test execution should cover no more than a single interface function. (Desirable)

The test suite shall have a capability to check complete and correct configuration and installation. This may include a set of defined confidence checks. (Mandatory)

Configuration and installation checking facilities should be automated. (Desirable)

The test suite should include clear and comprehensive user and programmer documentation