



Test Assertions Guidelines

Draft 0.9.9.8-in-progress-3

7 January 2009

This Version:

Previous Version:

http://www.oasis-open.org/apps/group_public/download.php/30292/TestAssertionsGuidelines-draft-0-9-9-7.doc

Latest Version:

Latest Approved Version:

http://www.oasis-open.org/apps/group_public/download.php/29935/TestAssertionsGuidelines-cd-0-9-9-5.doc

Technical Committee:

OASIS Test Assertions Guidelines (TAG) TC

Chair(s):

Patrick Curran, Sun Microsystems

Jacques Durand, Fujitsu

Editor(s):

Stephen Green, Document Engineering Services and previously SystML

Dmitry Kostovarov, Sun Microsystems

Contributor(s):

David Marston, IBM Research

David Pawson, Royal National Institute for the Blind

Hyunbo Cho, Pohang University

Kevin Looney, Sun Microsystems

Kyoung-Rog Yi, KIEC

Lynne Rosenthal, NIST

Paul Rank, Sun Microsystems

Serm Kulvatunyou, NIST

Tim Boland, NIST

Victor Rudometov, Sun Microsystems

Youngkon Lee, Korea TAG forum

Abstract:

This document provides guidelines and best practices for writing test assertions along with mandatory and optional components of a test assertion model.

Status:

This document was last revised or approved by the Test Assertions Guidelines on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/tag/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/tag/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/tag/>.

Notices

Copyright © OASIS® 2007-2009. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Introduction

This document is a guide to test assertions. Its purpose is to help the reader understand what test assertions are, their benefits, and most importantly, how they are created. As you will discover, test assertions can be an important and useful tool in promoting the quality of specifications, test suites and implementations of specifications. You will learn that there are many ways to create test assertions.

By following the guidelines in this document, you will learn how to develop well-defined test assertions that can have useful purposes and applications such as the starting point for a conformance test suite for a specification. Experiences in developing test assertions will be shared, along with lessons learned, helpful tricks and tools, hazards to avoid, and other knowledge that may be helpful in crafting test assertions.

Organization of the document:

Section 1 describes the rationale for test assertions

Section 2 describes basic design principles sufficient for simple cases of test assertions

Section 3 explains the advanced features related to test assertions

Appendices provide a glossary of important terms, references, a listing of related reading material and a worked example.

Intended Audience

The primary audience for this document is the authors of software specifications and the writers of software test suites as these are most likely to be involved in writing test assertions. However these guidelines are also intended to be of use to anyone who needs to write or make use of test assertions or who seeks to understand them. This might include anyone involved in specifying, testing or implementing software or related technologies.

Scope

These guidelines are intended to apply to any technology or business field but some of the text may apply more to software engineering which is the primary focus. The examples describe an arbitrary mechanical device in order to ensure a general understanding of the concepts, whatever the background of the reader. This document is limited to the essentials of test assertions with an expectation that a further document will follow to cover matters in greater depth and detail.

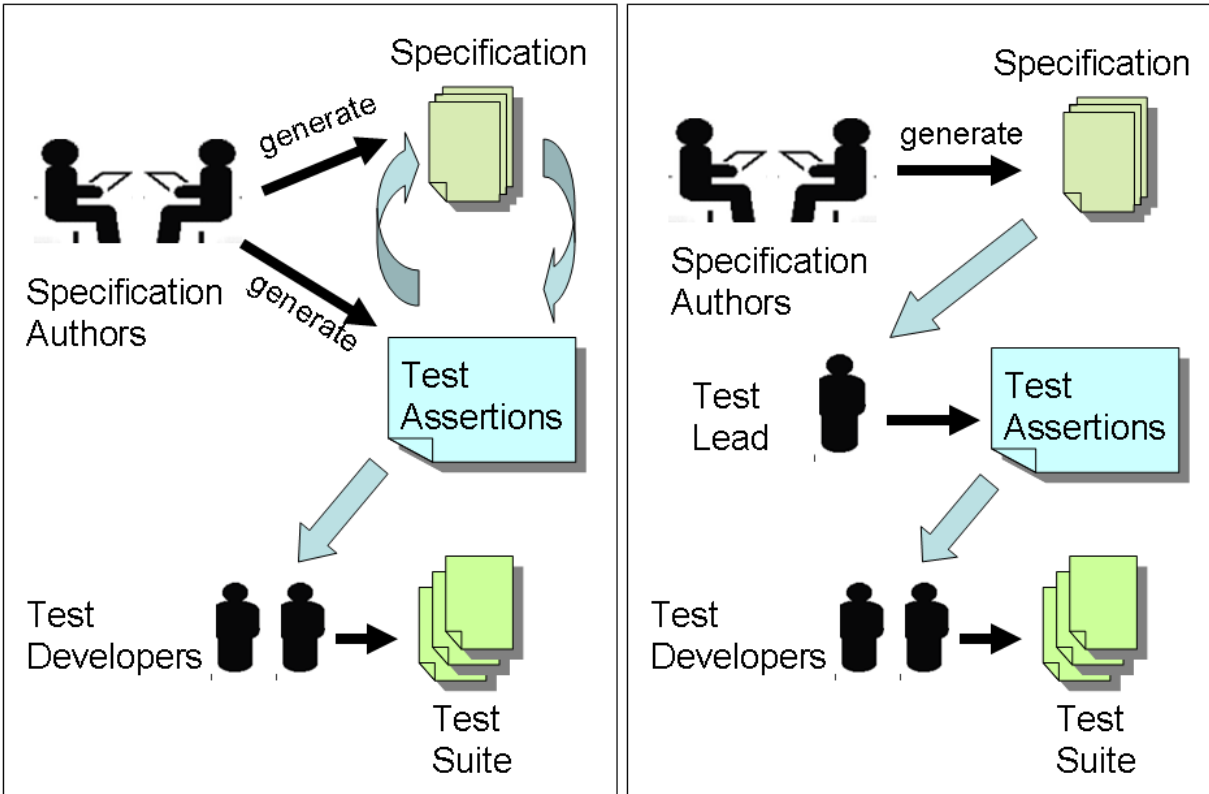
1 Rationale

1.1 Benefits of Test Assertions

Improving the Specification

Test assertions may help to provide for a tighter specification: Any ambiguities, contradictions and statements which require unacceptable resources for testing can be noted as they become apparent during test assertion creation. If there is still opportunity to correct or improve the specification, these notes can be the basis of comments to the specification authors. If not developed by the specification authors, test assertions should be reviewed and approved by them. This improves both the quality and time-to-deployment of the specification so that the best results are achieved when assertions are developed in parallel with the specification. An alternative is for test assertions to be written by the leader of a team involved in writing test suites.

Fig 1. Role of Test Assertions in the Specification and Testing Workflows



Facilitating Testing

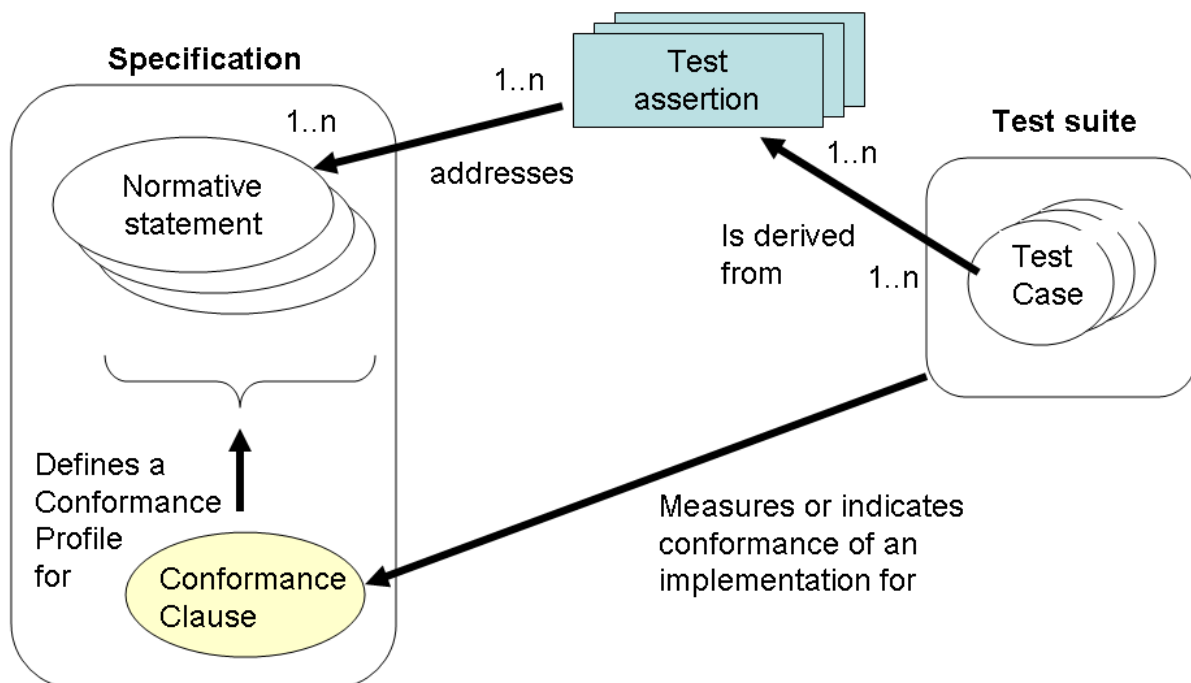
Test assertions provide a starting point for writing a conformance test suite or an interoperability test suite for a specification that can be used during implementation. They simplify the distribution of the test development effort between different organizations while maintaining consistent test quality. By tying test output to specification statements, test assertions improve confidence in the resulting test and provide a basis for coverage analysis (estimating the extent to which the specification is tested).

1.2 What is a Test Assertion?

A test assertion is a testable or measurable expression for evaluating the adherence of part of an implementation to a normative statement in a specification.

There is always a need to make explicit the relationship between a test assertion and the normative statement(s) it addresses in the specification.

Fig.1 The Role of Test Assertions



The specification will often have a clause called a conformance clause¹ [CONF1][CONF2] which defines a conformance profile [VAR] for the normative statements from which the specification is comprised . Testing such conformance involves matching test results to specification normative statements for which there is directly or indirectly a conformance requirement. Test assertions sit between the specification and any suite of tests to be conducted to determine conformance or otherwise. Such a test suite is typically comprised of a set of test cases that measure or indicate conformance of an implementation for the conformance profile defined in the conformance clause of the specification. These test cases may be derived from test assertions which address the normative statements of the specification. Reference to definitions of the following terms in the Glossary, Appendix A, may avoid confusion between related concepts: 'Conformance Clause', 'Test Assertion', 'Test Case', 'Test Metadata' and 'Tag'.

A Note on Testability

Judging whether the test assertion is testable may require some knowledge about testing capabilities and resource constraints. Sometimes there is not much knowledge of what actual testing conditions will be. In such cases the prime objective of writing test assertions is to provide a better understanding of what is expected from implementations in order to fulfill the requirements. In other cases, the test assertions are designed to reflect a more precise knowledge of testing conditions. Such test assertions can then be use as a blueprint for test suites.

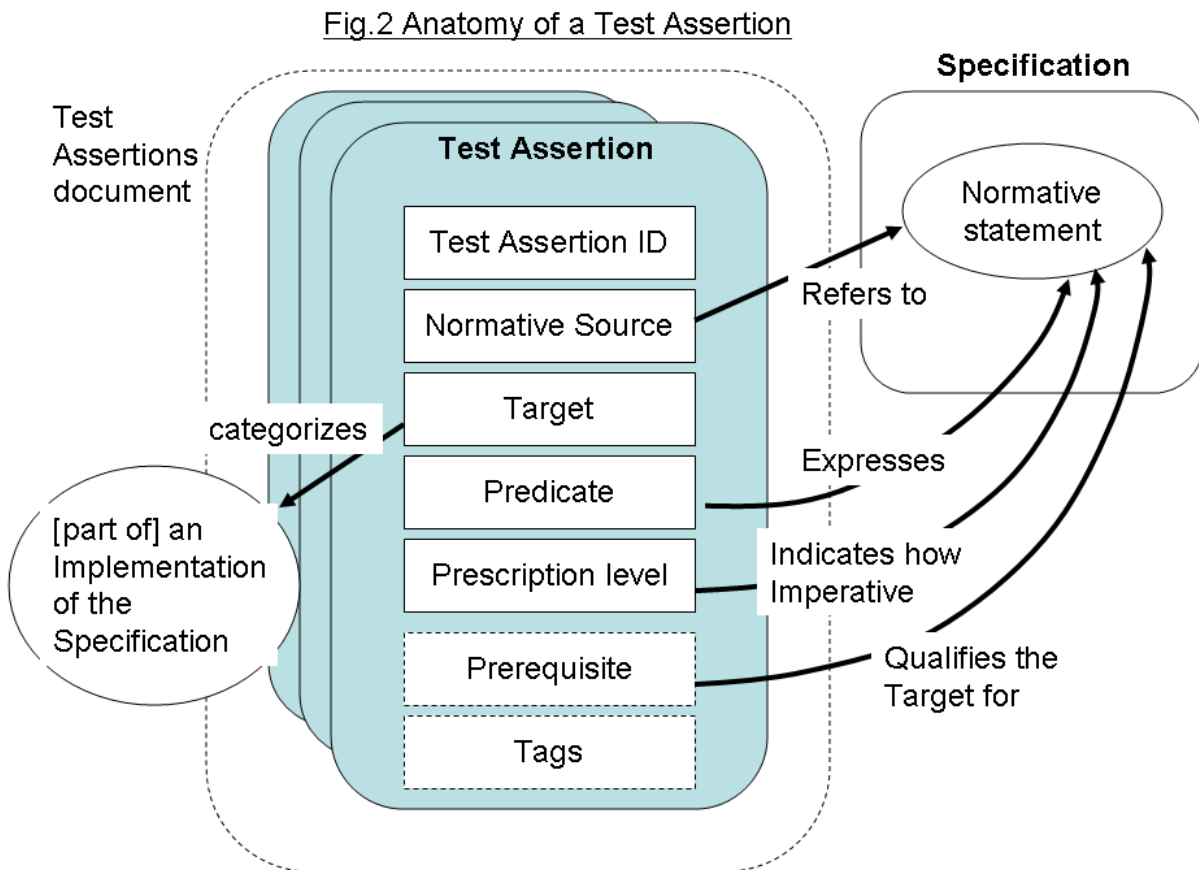
¹ See description of 'conformance clause' in Glossary, Section 4

2 Designing a Simple Test Assertion

This section aims to cover the simpler aspects of test assertions. Some of the more complex aspects are covered later in Section 3.

2.1 The Structure of a Test Assertion

Some of the elements which comprise a test assertion are considered core while others are optional.



Core Test Assertion Parts

A test assertion must include, implicitly or explicitly:

Test Assertion Identifier

This unique identifier facilitates tools development and the mapping of assertions to specification statements. It is recommended that the identifier be made universally unique.²

Normative Source(s)

These refer to the precise specification requirements or normative statements that the test assertion addresses.

Test Assertion Target

Such a target categorizes an implementation or a part of an implementation of the referred specification.

Predicate

A predicate asserts, in the form of an expression, the feature (a behavior or a property) described in the referred specification statement(s). If the predicate is an expression which evaluates to "true" over the test assertion target, this means that the target exhibits this feature. "False" means the target does not exhibit this feature.

Prescription Level

A keyword that indicates how imperative it is that the Normative Statement referred to in the Normative Source, be met. See possible keyword values in the Glossary.

Optional Test Assertion Parts

In addition, a test assertion may optionally include:

Prerequisite(s)

A test assertion Prerequisite is a logical expression (similar to a Predicate) which further qualifies the Target for the Normative Statement. It may include references to the outcome of other test assertions. Whether or not the implementation of the target fulfils the prerequisite determines whether or not the test assertion is applicable to the implementation: If it evaluates to "false" then the test assertion is to be considered 'not applicable', meaning the test assertion is not relevant to this Target instance.

Tag(s)

Test assertions may be assigned 'tags' or 'keywords', which may in turn be given values. These tags provide you with an opportunity to categorize the test assertions. They enable you to group the test assertions, for example based on the type of test they assume or based on their target properties.

² One way to do this is to designate a universally unique name for a set of test assertions and to include this name along with the identifier when referencing the test assertion from outside of this set.

2.2 Best Practices

In an actual test assertion definition, the previously mentioned properties are often explicitly represented as elements of the test assertion. For example:

Consider the following as a requirement from a specification on “widgets” (we will build on this example throughout these guidelines):

```
[requirement 100] "A widget MUST be of rectangular shape".
```

Here is a test assertion addressing this requirement:

```
TA id: widget-TA100-13
```

```
Normative Source: "widget specification", requirement 100
```

```
Target: widget
```

```
Predicate: [the widget] is of rectangular shape
```

```
Prescription Level: mandatory
```

The assertion predicate is worded as an assertion, not as a requirement (the 'MUST' keyword is absent from the predicate but reflected in the prescription level). It has a clear Boolean value: either the statement is true, or it is false for a particular target. The case of how to write a predicate for specification statements that convey optionality (for example, using keywords SHOULD, MAY, etc.) is examined later.

Note that a concrete representation of a test assertion may omit some of these elements provided they are implicit, as discussed later.

2.2.1 Granularity of Test Assertions

Consider the following statement in the widget specification:

```
[requirement 101] "A widget of medium size MUST use exactly one AA battery encased in a battery holder."
```

There are actually two requirements here that can be tested separately:

(requirement 101, part 1) A medium-size widget MUST use exactly one AA battery.

(requirement 101, part 2) A medium-size widget MUST have a battery holder encasing the battery.

³ Just as we have done with the examples, it is useful to create and follow a scheme or convention when assigning test assertion identifiers. In the examples in these guidelines, the test assertion identifier is based on a combination of broad target category and specification requirement reference number, suffixed with extra characters because it is worth remembering that there is likely to be a many-to-many relationship between specification requirements and test assertions.

Because of this it is possible to write two test assertions:

TA id: widget-TA101-1a

Normative Source: specification requirement 101, part 1

Target: medium-size widget

Predicate: [the widget] uses exactly one AA battery.

Prescription Level: mandatory

TA id: widget-TA101-1b

Normative Source: specification requirement 101, part 2

Target: medium-size widget

Predicate: [the widget] has a battery holder encasing the battery.

Prescription Level: mandatory

The granularity of a test assertion is a matter of judgement. A single test assertion instead of two could have been written here, with the predicate: "[the widget] uses exactly one AA battery AND has a battery holder encasing the battery". This choice may later have an impact on the outcome of a test suite written to verify the conformance of widgets. With a single test assertion, a test case derived from this test assertion will not be expected to distinguish between the two failure cases. Using two test assertions - one for each sub-requirement - will ensure that a test suite can assess and report independently about the fulfillment of each sub-requirement. Other considerations such as the different nature of tests implied or the reuse of a test assertion in different conformance profiles [VAR], may also lead to the adoption of "fine-grained" instead of "coarse-grained" test assertions. Usage considerations will dictate the best choice.

2.2.2 Implicit Test Assertion Parts

It was noted earlier that a concrete representation of a test assertion may omit elements provided they are implicit. A common case of implicit test assertion components is the implicit target: when several test assertions relate to the same target, the latter may be described just once as part of the context where the test assertions are defined, so that it does not need to be repeated. For example: all test assertions related to requirements about the widget power supply in a widget specification may be grouped in the section "Widget Power Supply Requirements", suggesting that they share the same target.

The predicate may be implicit: In some specifications where all requirements follow a similar pattern, it is often possible to straightforwardly derive the assertion predicate from a requirement so that the predicate does not need to be explicitly stated every time. One way to do this is to use a simple rule. Take, for example, requirement 101, part 1 "A medium-size widget MUST use exactly one AA battery". Compare this text with the predicate of its test assertion, widget-TA101-1a, "[the widget] uses exactly one AA battery". There is so much similarity between the requirement text and the test assertion predicate text that an implementation may decide there is too much overhead in writing the predicate to warrant it and the implementation may then decide to merely use a quotation of the requirement in the normative source as an implicit predicate.

2.2.3 Optional Statements: Prescription Level

The requirement 101 in the widget example in Section 2.2.1 has a mandatory character: it is an absolute requirement using the keyword MUST.

Interpreting the outcome of such test assertions is straightforward. Test cases derived from such test assertions can make a clear statement of conformance to the specification for the target under test: '(test assertion predicate = "true")' means not only that the target exhibits the specified feature of the specification, but also that the target fulfills a specification requirement, since this feature is required.

However there might be several ways to conform to a specification, also known as dimensions of variability [VAR]. While both conforming, two implementations may not exhibit the same features. This section considers one of the most obvious cases of variability: optional features.

Consider a case where the the specification statement is optional, for example, it uses RFC keywords SHOULD / RECOMMENDED or MAY / OPTIONAL as follows.

Examples:

```
[statement 102] "It is RECOMMENDED for a widget to be waterproof."
```

```
[statement 103] "A widget MAY have a metallic casing."
```

Such (normative) statements cannot be construed as formal requirements – a widget will not fail to conform to the specification if it is not waterproof, or if it has a plastic casing. However, establishing conformance is not the sole objective of test assertions. Some test suites are intended to evaluate the capabilities of an implementation, for example, which options the suite implements, regardless of conformance considerations. Even with a conformance objective in mind, a clear separation must be made between:

(a) Describing a condition under which a target can exhibit a specified feature. This is the role of the test assertion.

(b) Deciding if a target satisfies a conformance criterion. This is the role of one or more test cases that are derived from a test assertion, the outcome of which might be interpreted according to a conformance profile.

Therefore, test assertions can be written for statements 102 and 103 by simply focusing on the specified feature and its related predicate, ignoring the prescription level in the predicate and including it instead as the separate component of that name, for which suitable values are enumerated ('mandatory', 'preferred', 'permitted'):

```
TA_id: widget-TA102-1
```

```
Normative Source: specification statement 102
```

```
Target: widget
```

```
Predicate: [the widget] is waterproof.
```

```
Prescription Level: preferred
```

TA id: widget-TA103-1

Normative Source: specification statement 103

Target: widget

Predicate: [the widget] has a metallic casing.

Prescription Level: permitted

The negative keywords 'MUST NOT' and 'SHOULD NOT' are handled as follows: With 'MUST NOT' the 'MUST' determines the prescription level, in this case 'mandatory', and the 'NOT' is transferred to the predicate. For example:

[statement 203] "A widget MUST NOT have a painted casing."

TA id: widget-TA203-1

Normative Source: specification statement 203

Target: widget

Predicate: [the widget] does not have a painted casing.

Prescription Level: mandatory

Similarly with 'SHOULD NOT':

[statement 204] "A widget SHOULD NOT have a plastic battery holder."

TA id: widget-TA204-1

Normative Source: specification statement 203

Target: widget

Predicate: [the widget] does not have a plastic battery holder.

Prescription Level: preferred

3 Advanced Features

We have considered the five essential elements of the test assertion: identifier, reference, target, predicate and prescription level. In practice there may be a need for further features to better cater for corner cases such as

- specifications where the normative statements are embedded wholly or partly in tables and diagrams
- specifications which normatively reference other specifications
- specification and test assertions versions
- inheritance and dependencies between specifications
- redundancy of excessively repeated assertions elements
- test assertion targets which are categorized in conformance clauses.

3.1 Complex Predicates

Recall the previous example requirement:

```
[requirement 101] "A widget of medium size MUST use exactly one AA battery and be encased in a battery holder."
```

The target can be defined as "a medium-size widget" (as in section 2.2.1) or as just "a widget". The latter is a natural decision if the specification requirement uses the wording: "[requirement 101] If a widget is medium size, then it MUST use exactly one AA battery and be encased in a battery holder." For the simplicity of this example, if the two test assertion predicates for widget-TA101-1a and widget-TA101-1b are combined into one example, one possible outcome is:

TA id: widget-TA101-2a

Normative Source: requirement 101

Target: widget

Predicate: if [the widget] is medium-size, then [the widget] uses exactly one AA battery AND the battery is encased in a battery holder.

Prescription Level: mandatory

The target category is broad, but the predicate part is really of interest only for a subset of this category (the medium-size widgets). Usage considerations should again drive the decision here: a test suite that is designed to verify all widgets, and does not assume a prior categorization of these into small / medium / large sizes, would be improved with test assertions that only use “widget” as the target, such as widget-TA101-2a.

Note: As an important part of the test assertion, even when the target is implicit, not explicit, the target for each assertion must be clearly identifiable.

A test assertion predicate may, then, be a Boolean expression - a composition of atomic predicates using logical operators AND, OR, NOT. A test assertion predicate may also be of the kind: “if (condition) then (expression)”.

The predicate is worded in an abstract way, still close to the wording of the specification. No indication of what kind of test procedure will be used, such as how to determine the number and type of batteries, is given. Detailed criteria for the condition evaluation, such as what kind of battery holder is acceptable, is also not provided. These details are normally left to the test cases that can be derived from the test assertions. These test cases will determine the precise criteria for conforming to the specification. However, if a precise criterion for interpreting the battery holder requirement is provided in an external specification, either referred directly by the widget specification or by a related conformance clause, then a test assertion must use this criterion in its predicate. Such a test assertion must then refer not only to the specification requirement in its reference property, but also to the external specification or to the conformance clause that refers to this specification.

3.2 Prerequisites

An issue with the previous test assertion (widget-TA101-2a) is that it will apply to all widgets, while the specification requirement is obviously of interest only for targets that are medium-sized. With widget-TA101-2a , the target predicate will always evaluate to “true” even when the widget is NOT medium-sized. Indeed, from a logical viewpoint, a predicate of the form “if <condition> then <property>” will always be true when <condition> is “false”. (The only way such a predicate evaluates to “false”, is when <condition> is “true” and <property> is “false”).

On the other hand if all widgets are categorized according to their claimed size prior to testing, then a test case implementing widget-TA101-2a will uselessly repeat the “size” test. In this situation the test assertions written in section 2.2.1 (widget-TA101-1a and widget-TA101-1b) are a better choice than widget-TA101-2a.

Assuming that the size of widgets is not a given but is subject to testing, how can we indicate that a preliminary test on the widget size must be done, and that the test assertion predicate must only apply if the widget is medium-size, meaning that otherwise the test assertion is considered “Not Applicable”? This is done by introducing a prerequisite element in the test assertion:

TA id: widget-TA101-2b

Normative Source: requirement 101

Target: widget

Prerequisite: [the widget] is medium-size

Predicate: [the widget] uses exactly one AA battery AND has a battery holder encasing the battery.

Prescription Level: mandatory

The Prerequisite element is a logical expression of the same nature as the Predicate, which concerns the same target instance.

Then the possible outcomes of a test assertion with a prerequisite are:

Not Applicable: If the prerequisite evaluates to “false”, then the test assertion does not even apply for this target (or its outcome can be stated as “Not Applicable”). A test case derived from this test assertion should not even be executed on this target: the result of doing so would be meaningless.

True: If the prerequisite evaluates to “true”, and the test assertion predicate evaluates to “true”, then the target is exhibiting the feature described in the addressed specification requirement.

False: If the prerequisite evaluates to “true”, and the test assertion predicate evaluates to “false”, then the target is NOT exhibiting the feature described in the addressed specification requirement.

3.3 Test Assertions for Properties

Requirements addressed by test assertions may be related to specific properties of a target. Assume there are specification requirements that define under which conditions a widget qualifies as “medium-size”. In other words, widgets do not come with a sticker that makes this categorization obvious by announcing small / medium / large. Instead, the size label is a property that is itself defined in the widget specification and that is subject to verification, like any other normative statement. In such a case, when writing test assertions, it is not a good idea to consider this property as part of the definition of the target category as in the case widget-TA101-1a and widget-TA101-1b, because the category of a widget could not be identified prior to doing any test on this widget.

Assume that the following requirement defines the “medium-size” property:

```
[requirement 104] "A widget that weighs between 100g and 300g and is from 5 to 15 centimeters long in its longer dimension, is a medium-size widget."
```

There is a major distinction between requirement 104 and requirement 101:

requirement 101 uses “medium-size” as a prerequisite: its predicates only concern widgets that are already established as medium-size.

requirement 104 defines how to qualify a test assertion as medium-sized.

The test assertions for requirement 104 can be written as:

TA id: widget-TA104-1

Normative Source: specification requirement 104

Target: widget

Predicate: [the widget] weighs between 100g and 300g.

Prescription Level: medium-sized:mandatory

Note that the "mandatory" prescription level is now relative to the claimed property ("medium-size"): the specification does not mandate all widgets to be "medium-size", but for a widget to claim the "medium-size" property, the predicate must be "true". The prescription **intent** must clearly be indicated in the Prescription Level element by prefixing the level value (here "mandatory") with the property name ("medium-size").

TA id: widget-TA104-2

Normative Source: specification requirement 104

Target: widget

Predicate: [the widget] is from 5 to 15 centimeters long in its longer dimension.

Prescription Level: medium-sized:mandatory

The test assertions widget-TA104-1 and widget-TA104-2 will be used to derive test cases that verify if the property "medium-size" applies to some widget. A "false" outcome for their predicates is an indicator that the medium-size property does not apply. It is not indicative of a violation of the specification itself. Such test assertions are called in this document "Property test assertions" to distinguish them from test assertions that are used as indicators of conformance to a specification. However, both types of test assertions are designed in the same way, with a predicate that indicates whether or not a target satisfies some feature or property.

There is no mention of the "medium-size" property in the predicates of test assertions 'widget-TA104-1' and 'widget-TA104-2'. This is because this property is precisely what needs to be established by a test suite containing test cases that are derived from these test assertions. Only when a target (here a widget) evaluates to "true" for these two test assertions, will it be considered medium-size. These test assertions are only concerned with the nature of these tests, not with how to interpret their outcome.

3.4 Prerequisites Referring to Other Test Assertions

Now that there is a means to establish the "medium-size" property, we can use a more precise prerequisite element in the test assertion for the requirement 101. Because Test Assertions have been written for testing the property "medium-size", we can refer to these in the Prerequisite:

TA id: widget-TA101-2c

Normative Source: specification requirement 101

Target: widget

Prerequisite: widget-TA104-1 AND widget-TA104-2

Predicate: [the widget] uses exactly one AA battery AND has a battery holder encasing the battery.

Prescription Level: mandatory

When a prerequisite element is quoting other test assertions, as seen above, with the prerequisite: (widget-TA104-1 AND widget-TA104-2), such references must be understood as short for: (widget-TA104-1 outcome = 'true' AND widget-TA104-2 outcome = 'true').

The prerequisite could have been stated as an explicit predicate in widget-TA101-2c. In other words, widget-TA101-2c could have been rewritten as:

TA_id: widget-TA101-2d

Normative Source: specification requirement 101

Target: widget

Prerequisite: [the widget] weighs between 100g and 300g AND [the widget] is from 5 to 15 centimeters long in its longer dimension.

Predicate: [the widget] uses exactly one AA battery AND has a battery holder encasing the battery.

Prescription Level: mandatory

Here widget-TA101-2c is semantically equivalent to widget-TA101-2d. However, because the notion of “medium-size” is itself specified as a property that is subject to verification and enforcement, it is useful to write test assertions for this property. It is then preferable to reuse such test assertions as prerequisites whenever this property is assumed. If the notion of medium-size evolves in future releases of the widget specification, the test assertion does not need to be altered: only its prerequisite test assertion needs to be, while all test assertions that explicitly state the prerequisite predicate would need to be updated.

3.5 Various Normative Sources

The normative content addressed by a test assertion is not always a single, well-identified specification requirement. The normative source may include:

- Multiple (non contiguous) specification statements.
- Non-textual content: tables and diagrams.
- Normative statements that present some testability challenges.

In the previously mentioned cases, it is often useful or necessary to “derive” a new textual statement that will be the actual normative source for the test assertion, and for which the predicate outcome will unequivocally indicate fulfillment or violation. This derived statement may in turn be worded so that the Predicate is implicit (see 2.2.2). In the case of “multiple statements”, although using several references is possible in the Normative Source element, it is recommended to derive a new consolidated statement.

Identifying the normative source subject to a test assertion may be a delicate exercise, as the source material is often dependent on its context for its meaning. A derived statement may be then necessary.

Even when the normative statement is a well identified portion of text in the specification, the following cases may occur:

- The normative source is implicit, as explained in 2.2.2. This means that there is no explicit element in the test assertion pointing at the specification part that is addressed. This may be the case when the test assertion normative source can be inferred from the location of the test assertion within the specification document itself.
- The specification document is itself expressed as a set of test assertions. This is possible by inserting in the "normative source" part of the test assertion, the normative statement itself instead of a reference to it.

3.6 Test Assertion Grouping

When writing test assertions while the specification is being analyzed, it is typical to group certain test assertions together. You can group assertions that have a special status, such as all accredited test assertions for a given specification, or those that share a particular characteristic, such as a common category of test assertion target.

A special kind of grouping is the container of all test assertions which belong to a particular specification or profile. The container may be the specification document itself if it includes the test assertions to be associated with it.

There are two special ways to group test assertions: - explicitly listing test assertions by their identifiers (section 3.6.1) and a more implicit grouping by a common but not unique property such as the tag names or tag values assigned to the test assertions (section 3.6.2).

3.6.1 Lists (Dimensions of Variability)

To explicitly identify a group of test assertions they can be listed by their unique test assertion identifiers. Listing in this way clarifies which assertions belong to a particular group and which do not. In addition to such a list, the logical reason which determines whether a test assertion is a member of that list or not must be stated in order to help with list maintenance.

For example:

TA List id: A001

List Description: all assertions describing 'Size' requirements

List Members: TA001, TA002, ..., TA008

Note that although, in the previous example, we have avoided enumerating each and every test assertion identifier by using an ellipsis ('...'), such methods introduce a possible weakness. During test assertion or test assertion list maintenance, it might be overlooked if a later test assertion is given an ID of TA002a and therefore is implicitly rather than explicitly made a member of this particular list. This may be a mistake since the new test assertion TA002a might not relate to the list description of 'Size' requirements. A list is completely explicit about every test assertion member when every member test assertion is listed explicitly by its test assertion identifier.

For example:

TA List id: A001

List Description: all assertions describing 'Size' requirements

List Members: TA001, TA002, TA003, TA004a, TA004b, TA005, TA006, TA007, TA008

Such a list may be regarded as 'fixed', 'frozen' or 'closed'. A test assertion added later with identifier TA005a, if it is to be included in this list, would require a change to the list (with possible version or change control implications) or the creation of a new list (with a new list identifier, if a list identifier is included with the list).

Test Assertion Document

A list of test assertions related to either conformance or interoperability testing will need special care with respect to version control and change management. Therefore, the criteria used to determine which test assertions are members of the list and which are not must be clear. The special case of a container for all test assertions related to a given specification or profile is a special example of an explicit list, although here the method used to define such a list may involve the use of inclusion of the test assertion itself rather than just its identifier within a special document or package. One way to create such a list is to include all such related test assertions within a document, called a 'Test Assertion Document'. Other synonymous terms might be 'Test Assertion List', 'Specification Analysis' or 'Test Assertion Set'. Note that the container of this complete set of test assertions might instead be the document of the specification or conformance profile [VAR] itself, when test assertions are included within the text of the actual specification or profile.

3.6.2 Tags (Test Assertion Metadata)

Another way to define a group of test assertions is to use a non-unique property of such assertions rather than just using their unique identifiers in a list or containing the test assertions in a document. To this end, test assertions may be assigned metadata in the form of non-unique 'tags' or 'labels'.

For example, test assertion 'widget-TA104-2' might be tagged as 'Size-Property-Description':

TA id: widget-TA104-2

Normative Source: specification requirement 104

Target: widget

Predicate: [the widget] is from 5 to 15 centimeters long in its longer dimension.

Prescription Level: medium-sized:mandatory

Tag: Size-Property-Description

Then it might be included in a list of test assertions related to 'Medium Size' requirements, along with other assertions tagged 'Size-Related' but NOT with test assertions 'Small-Size-Related'.

TA List id: A002

List Description: all assertions describing 'Medium Size Widget' requirements

List Members: All test assertions with Tag 'Size-Property-Description' AND Tag 'Size-Related' AND NOT Tag 'Small-Size-Related'

This we have called a 'List' but it is in fact defined rather more implicitly than if every member were listed by its identifier, as described in section 3.6.1. In fact a more explicit and well-defined list might combine both tags and identifiers to group the assertions:

For example:

TA List id: A002

List Description: all assertions describing 'Medium Size Widget' requirements

List Description: All test assertions with Tag 'Size-Property-Description' AND Tag 'Size-Related' AND NOT Tag 'Small-Size-Related'

List Members: TA001, TA002, TA003, TA004a, TA004b, TA005, TA006, TA007, TA008

So a tag is a further, optional test assertion element useful in grouping test assertions. It may sometimes be useful to create tags as name-value pairs.

For example, tagging a test assertion:

```
Tag: Widget-Size=Medium
```

This tag would allow all test assertions related to requirements for medium-sized widgets to be grouped to facilitate testing just medium-sized widgets, or a conformance profile relating only to medium-sized widgets.

Note that several such filters can be applied to the same set of assertions and any given assertion can appear in more than one grouping.

Special consideration when using tags for grouping is to be given to the stages in the workflow of test assertion authoring and maintenance and subsequent use at which changes might be made to tags and their values. New tags may be added, perhaps by adding metadata which is separate from the documented test assertion. If metadata for test assertions is defined and maintained separately from the test assertions it may be subject to an entirely different set of version and change control rules and methodologies. In this case, a distinction might need to be made between tags which were part of the original test assertion and those whose list membership might be different to that which was known or expected when the list was defined.

For example, consider a list defined using tags but without explicitly listing test assertion identifiers:

```
TA List id: A002
```

```
List Description: all assertions describing 'Medium Size Widget'  
requirements
```

```
List Description: All test assertions with Tag 'Size-Property-Description'  
AND Tag 'Size-Related' AND NOT Tag 'Small-Size-Related'
```

If TA004a is originally tagged 'Size-Related' but the workflow allows it also to be subsequently tagged 'Small-Size-Related', then there will need to be rules which determine whether the test assertion is still a member of List 'A002'.

3.7 The Case of Multiple Specifications

Modularity and succinct description within specifications can be achieved by leveraging existing specifications that are referenced by other specifications. Specification writers often create "umbrella specifications". These are widely scoped specifications that delegate certain normative descriptions to other "referenced specifications".

Specification modularity may also come from a "prototypical specification" - a "base specification" - from which specialized derivative specifications may define specific information for a given context.

For proper specification analysis, inclusion of test assertions from both the umbrella or base specification and all the referenced specifications should be considered.

Two aspects of analysis of multiplicity of the normative sources are considered:

- Specification Visibility (Section 3.7.1) deals with how assertions from a referenced specification are considered or described in the umbrella specification.
- Composition of Assertions (Section 3.7.2) describes how predicates and prerequisites can express various relationships between umbrella and referenced specifications.

3.7.1 Specification Visibility

Consider a case where a specification is the normative source but itself refers normatively to a second specification. Consider first the case where there is no visibility of any test assertions covering this second specification. Here there may be a prerequisite that all test assertions for the second specification be treated as a single Boolean requirement.

In a second case the specification refers to a second specification for which the test assertions are given visibility and here the test assertions of the first specification may refer to individual or groups of test assertions of the second specification as explicit prerequisites. Of course, any number of normative sources may be involved, some with test assertions visible, others not.

3.7.2 Composition of Assertions

There are three dimensions that describe how assertions from a referenced specification may be included within an umbrella specification:

- 'scope of inclusions',
- 'conditionality of inclusions',
- 'modification of inclusions'.

These relationships between specifications can be expressed by including a set of references in the test assertions document for the umbrella spec. The references would be to any test assertions from the referenced specification which are required according to the conformance clause of the umbrella specification. An alternative, if no such test assertions for the referenced specification exist, would be to

summarize conformance requirements to the referenced specification in the form of a test assertion using its predicate:

TA_id: widget-TA108-1

Normative Source: [interpretation of conformance clause to WidgetSpec 1.0] "All widgets conformant to the WidgetSpec 1.0 specification must also be conformant to the 'smaller box' assertions of the WidgetMobile Small Box Specification 1.2"

Target: widget

Predicate: [the widget] conforms to Mini-Widget Small Box Specification 1.2

Prescription Level: mandatory

Scope of Inclusions

An umbrella specification usually relates to a referenced specification by assuming or requiring conformance of its implementation to this specification. Variations exist as follows :

- conformance to an (entire) umbrella specification
- conformance to a profile of the umbrella specification
- conformance to a specific normative statement from the umbrella specification

Similarly with a conformance requirement for varying scopes of the external specification as follows:

- conformance to an (entire) referenced specification
- conformance to a profile of an referenced specification
- conformance to a specific test assertion from an referenced specification

The predicate or the list of external test assertion references would reflect such variations.

Conditionality of Inclusions

It might be that the conformance or otherwise to a referenced specification is a condition which is included in another test assertion as a prerequisite. The prerequisite of the assertion may:

- a. require that optional portions of the referenced specification be implemented in the umbrella,

- b. conditionally require optional portions of the referenced specification be implemented in the umbrella (for example, based on the presence of hardware or some other such support), or
- c. make remaining (required) portions of the referenced specification optional.

Modification of Inclusions

This dimension of inclusion describes where an umbrella specification is conformant to a referenced specification, where some subset of assertions must be modified. This means of inclusion assumes some partitioning of the unchanged assertions and modified assertions. You can use "lists of assertions" to describe in the prerequisite the subset of assertions that the umbrella specification is conformant to "unchanged". The remaining test assertions (the changed set) can be individually specified as test assertions of the umbrella specification.

Typically, assertions are modified in a referenced specification that can be strengthened in a few ways:

- strengthening the prescription level of an assertion (eg. x MAY do y => x MUST do y), or
- strengthening the meaning of an assertion with additional requirements (eg. IF x THEN z => IF (x AND y) THEN z).

3.8 Specification Versions

Where a specification is the basis for test assertions there needs to be consideration of how to support further versions and maybe any previous versions of that specification. One solution is to create a set of test assertions for each specification version. The references to specifications may include the identification of the precise specification version, but this may restrict that test assertion to just one version of a specification. This simple strategy is less than ideal as it results in a need to re-author the test assertions each time there is a new specification version.

Another method of dealing with multiple specification versions is to create a repository of test assertions for a specification and properly tag each test assertion for the versions of the specification where it is valid. This can be accomplished by introducing two tags, `VersionAdd` and `VersionDrop`.

tag: `VersionAdd`: the lowest numerical version to which the test assertion applies.

tag: `VersionDrop`: the lowest numerical version number to which the test assertion does NOT apply.

Both `VersionAdd` and `VersionDrop` are optional tags. The absence of both tags would mean that the test assertion is valid in all specification versions. If only a `VersionAdd` tag exists and its value is X, the test assertion will be valid in version X of the specification and all subsequent versions. If only a `VersionDrop` tag exists and its value is Y, the test assertion will be valid in all versions of the specification prior to version Y. If both `VersionAdd` and `VersionDrop` tags exist, the test assertion will be valid in version X and all subsequent versions up to but not including version Y. Based on these rules, you can easily generate the set of test assertions that apply to a specific version of the specification.

Care must be taken when going from one version of a specification to another. The test assertion author must identify all test assertions that are the same between versions, dropped from one version, added to

one version, and modified between versions. Test assertions that are the same, are dropped, or are added can easily be handled with the `VersionAdd` and `VersionDrop` tags.

Test assertions that are modified are trickier to handle. One could treat the modified test assertions as two separate test assertions and tag them with the appropriate `VersionAdd` and `VersionDrop` tags. Unfortunately, this approach does not provide any indication that the updated assertion is related to the test assertion in the prior specification.

Note that in order to track the evolution of a test assertion, it is important to preserve the test assertion identifiers across revisions of the specification. This is important because it is also important to maintain the relationship between a test assertion and the tests associated with that test assertion.

3.9 Variables

Variables have a similar role in test assertions as in other technologies. They provide a means for consistently sharing values across multiple assertions and with other processes. The writer of a set of test assertions can use a variable to assert that all occurrences of that variable must have the same value, even if that value cannot be known at the time the test assertions are written. For example, consider a set of assertions that share a reference to the line (mains) voltage supplied by the local electric utility, where the specific location varies. By declaring a variable for this value and distinguishing the name by, for example, representing it in upper case letters (UTILITY-VOLTAGE), each test assertion can use it within elements and expressions.

Note: Some people prefer the term "parameter" and would say that the test assertions in question are "parameterized."

The following example also shows the use of the variable within the predicate. Here, specification requirement 130 depends on the utility voltage, expressed as variable 'UTILITY-VOLTAGE'.

```
Variable: 'UTILITY-VOLTAGE' the AC voltage the voltage commonly provided
for hand-held electrical appliances and laptop computers.
```

```
...
```

```
TA id: widget-TA130-1
```

```
Normative Source: specification requirement 130
```

```
Target: electrical widget
```

```
Predicate: [the electrical widget] contains an embedded AC adapter for
the UTILITY-VOLTAGE.
```

```
Prescription Level: mandatory
```

Variables can be used across elements in a single test assertion, for example, in the prerequisite (see section 3.2) and other parts of a test assertion. For example, requirement 130 may be restricted only to widgets that have a compliance requirement for this voltage:

```
Prerequisite: [the electrical widget] is compliant with UTILITY-VOLTAGE
```

In some cases, the variable has a value known or assigned during test assertion authoring, and is simply used to allow agile resetting of its value. In other cases, the variable can be declared in name only, leaving

the value to be assigned at a subsequent stage, such as in an implementation or in a conformance clause for a profile, level or module. The value might be measured during testing and could be associated with a property (see section 3.3). For example, if the medium-size property is true for a widget, then the SIZE variable is set to the value "medium".

A particular consideration in using variables is variable scope. For example, a grouping construct might be a place to declare variables whose scope only applies to those test assertions associated with that grouping. This allows the same variable and its value to be used across several test assertions while avoiding problems with name clashes in test assertions outside of the variable's scope.

3.10 Target Categories

As mentioned in section 2.1, the Target element of a test assertion generally defines a category of objects or parts of an implementation under test. For example, the test assertion target "widget" represents any object that qualifies as a widget.

It is often the case that different targets or categories are related, for example one target is a subcategory of another target . As a consequence:

- Two test assertions that apparently have different targets, will in fact apply to the same implementation or part of that implementation.
- A test assertion may be used as a prerequisite of another test assertion, in spite of having a different target definition.

For these reasons target dependencies should be explicitly stated. Such dependencies may be defined outside test assertions, such as in an object-oriented model. But you can also note these dependencies in the test assertion itself.

Example: consider a Target named "electrical-gizmo" that is a subcategory of the "widget" Target. This means all test assertions for widget also apply to electrical-gizmo. One way to express this dependency is to use tags. In the following example, a tag named "target-isa" will remind the reader of the test assertion that the electrical-gizmo target is also a widget target:

TA id: gizmo-TA300

Normative Source: specification requirement 300

Target: electrical-gizmo

Prerequisite: [The gizmo] has a low-battery indicator.

Prescription Level: mandatory

Predicate: The low-battery indicator of [the gizmo] is a red LED that is flashing below 25% charge.

Tag: target-isa = widget

Another way to indicate that this target is also a widget, is to use the prefix notation 'widget:electrical-gizmo' instead of just 'electrical-gizmo' for the target element in a test assertion:

```
TA_id: gizmo-TA300
Target: widget:electrical-gizmo
```

Both modes of annotation (tag or target prefix) make it clear that an electrical-gizmo is also a widget. This helps grouping test assertions based on the target to which they apply. It also helps writing prerequisites that refer to other test assertions. For example, knowing that the electrical-gizmo is also a widget, and assuming there is already a test assertion (widget-TA123) written:

```
[statement 123] A widget MAY have a low-battery indicator.
```

then it is possible to reuse the test assertion for Requirement 123 as a prerequisite:

```
TA_id: gizmo-TA300
...
Prerequisite: widget-TA123
```

Another kind of target dependency is the Composition relationship: Target T2 is a component of Target T1 if an instance of T1 contains an instance of T2. For example, assuming widgets always have at least one switch to control their operation, Target "switch" is a component of Target "widget". Knowledge of this target composition relationship brings the same benefits as for subcategories:

- Grouping all test assertions that apply to widgets often should include the test assertions that apply to widget components.
- A test assertion on the switch of a widget may be used as a prerequisite to a test assertion of the widget itself. For example, addressing the requirement "the gizmo must stop when the switch is off (Req# 400)" may use as prerequisite a test assertion (called here "TA-switchoff") that addresses the requirement "A switch must cut its electrical circuit when in OFF position". This requirement would guarantee that the switch is tested first in related test programs.

```
TA_id: gizmo-TA400
Normative Source: specification requirement 400
Target: electrical-gizmo
Prerequisite: TA-switchoff(target: the switch of the gizmo)
Predicate: [the gizmo] stops when the switch is off.
Prescription Level: mandatory
```

Appendix A. Glossary

Conformance

The fulfillment of specified requirements by a product, document, process, or service.

Conformance Clause

A statement in the Conformance section of a specification that provides a high-level description of what is required for an artifact to conform. The conformance clause may, in turn, refer to other parts of the specification for details. A conformance clause must reference one or more normative statements, directly or indirectly, and may refer to another conformance clause.

Implementation

A product, document, process, or service that is the realization of a specification or part of a specification.

Normative Source

The part of the test assertion that identifies the precise specification requirements or normative statements that the test assertion addresses. (See also Section 2.1.)

Normative Statement, or Normative Requirement

A statement made in the body of a specification that defines prescriptive requirements on a conformance target.

Predicate

An expression that asserts the feature (a behavior or a property) described in the referred specification statements. If the predicate is an expression which evaluates to “true” over the test assertion target, this means that the target exhibits that feature. “False” means the target does not exhibit that feature.

Prerequisite

A logical expression which further qualifies the applicability of the test assertion to the test assertion target. A prerequisite may include references to other test assertions. It evaluates to true or false and if false then the assertion is to be considered 'not applicable', meaning that the test assertion is not relevant to its target.

Prescription Level

The test assertion defines a normative statement which may be **mandatory** (MUST/REQUIRED), **permitted** (MAY/OPTIONAL) or **preferred** (SHOULD/RECOMMENDED). This property can be termed the test assertion's prescription level. NOTE: in the case of the normative source including keywords 'MUST NOT' the prescription level '**mandatory**' is used and the 'NOT' included in the predicate.

Tag

Metadata that allows the grouping of a test assertion and a means to categorize the test assertion targets.

Test Assertion

A testable expression for evaluating the adherence of part of an implementation to a normative requirement statement in a specification. A test assertion describes the expected output or behavior for the test assertion target within specific operation conditions, in a way that can be measured or tested.

Test Assertion Document

A container for a complete set of test assertions, often those related to all or part of a specification or conformance profile. In some cases the container is the specification itself with test assertions included within it. Test assertions can be added to the document, removed or changed using a change and version management procedure.

Test Assertion Target

Implementation or part of an implementation that can be the object of a test assertion or test case. (See also Section 2.1.)

Test Case

A set of a test tools, software or files (data, programs, scripts, or instructions for manual operations) that verifies the adherence of a test assertion target to one or more normative statements in the specification. Typically a test case is derived from one or more test assertions. Each test case includes: (1) a description of the test purpose (what is being tested - the conditions / requirements / capabilities which are to be addressed by a particular test), (2) the pass/fail criteria, (3) traceability information to the verified normative statements, either as a reference to a test assertion, or as a direct reference to the normative statement.

Test Metadata

Metadata that is included in test cases to facilitate automation and other processing.

Variable

A parameter or attribute employed by the writer of a test assertion to refer to a value that is not known at the time the test assertion is written, but will be determined at some later stage, possibly as late as the middle of running a set of tests. A variable is also employed to enable several assertions to share a value (set once, used by many), like a variable in other technologies.

Appendix B. References

- [CONF1]** Conformance requirements for Specifications (OASIS, March 2002)
see http://www.oasis-open.org/committees/download.php/305/conformance_requirements-v1.pdf
- [CONF2]** Conformance testing and Certification Framework (OASIS, Conformance TC, June 2001) see http://www.oasis-open.org/committees/download.php/309/testing_and_certification_framework.pdf
- [TD]** Test Development FAQ, WG note (W3C, 2005)
see <http://www.w3.org/QA/WG/2005/01/test-faq>
- [VAR]** Variability in Specifications, WG note (W3C, 2005)
see <http://www.w3.org/TR/2005/NOTE-spec-variability-20050831/>
- [TMD]** Test Metadata, QA Interest Group note, (W3C, September 2005)
see <http://www.w3.org/TR/2005/NOTE-test-metadata-20050914/>

Appendix C. Related Material

Voting Systems Standard (EAC), (2007)

<http://www.eac.gov/files/vvsg/Final-TGDC-VVSG-08312007.pdf>

The use of targets and their categorization using classes and sub-classes was noted. Object oriented classification is an interesting technique for categorizing test assertion targets, though it was considered that the treatment of target categorization in the present guidelines should take a broader scope.

Unisoft has published their own Glossary of Assertion Based Testing Terms, again for POSIX

<http://www.unisoft.com/glossary.html>

[See also DejaGnu Testing Framework below.]

The DejaGnu Testing Framework - A POSIX conforming test framework – is based on the use of test assertions

http://www.delorie.com/gnu/docs/dejagnu/dejagnu_6.html

This framework builds on the POSIX assertions definitions. Of particular note is the analysis of outcome interpretations. The present guidelines do not give extensive coverage to this topic because it fits better with the domain of test suites where outcomes can be related to the knowledge of testing methods to be used. However, test assertions may be used as a means of generating or authoring test suites and in these cases a special note may be needed to list the various possibilities inherent in interpreting the outcomes of tests based on a given test assertion.

Test assertions developed with a particular testing framework in mind include those developed by the WS-I for their web services profiles. For example for the Basic Profile version 1.1

http://www.ws-i.org/Testing/Tools/2005/01/BP11_TAD_1-1.htm

Similarly W3C publishes test suites based on test assertions for HTML and SOAP specifications:

HTML 4.01 Test Suite - Assertions (2002)

http://www.w3.org/MarkUp/Test/HTML401/current/assertions/assertions_toc.html

SOAP Version 1.2 Specification Assertions and Test Collection (2003)

<http://www.w3.org/TR/soap12-testcollection/>

The following documents are milestones in the development of the current thinking on test assertions as presented at a basic level in the present guidelines:

A Test Assertion Guideline draft was initiated within the OASIS Technical Advisory Board, (2004-2005)

http://www.oasis-open.org/committees/document.php?document_id=20661&wg_abbrev=ebxml-iic

Work in W3C lead to the QA Framework: Specification Guidelines (W3C, November 2004)

<http://www.w3.org/TR/qaframe-spec/>

A Test Assertion Guide was later drafted in W3C (W3C Editors' Draft, 2006)

<http://www.w3.org/2006/03/test-assertion-guide>

Other methodologies for the creation of test assertions or similar artefacts:

Constraint Languages have special relevance when defining test assertions for specifications involving specification normative statements which make use of standards such as XML Schemas and UML diagrams.

Object Constraint Language (OCL) and Unified Modeling Language (UML), Object Management Group, Inc.

http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL

OCL is a formal constraint language used to define constraints for UML. It can have a special role in testing based on UML methodologies. OCL includes methods for defining preconditions and prerequisites as well as postconditions: There may be some overlap with test assertion prerequisites. Where a specification includes UML diagrams it may be useful to use OCL expressions within or in place of test assertion predicates. Similarly, XPath expressions may play a key role in predicates involving XML. In both of these special cases there may be good reason to specialize the structure of the test assertion to facilitate use of such expression languages.

Appendix D. Worked Example

Dummy Widget Specification 1.1

Worked Example 'Specification'

April 2008

...

Section 100

A widget MUST be of rectangular shape, as shown below.

<< diagram of widget showing basic rectangular shape >>

Fig 67

Section 101

A widget of medium size MUST use exactly one AA battery and have a red button on top (see below).

<< diagram of widget with battery and red button on top >>

Fig 68

The mechanisms by which the widget delivers its functionality is not subject to this specification.

Section 102

It is RECOMMENDED for a widget to be waterproof. If it is not waterproof then it MUST have a warning label stating that it is not waterproof.

Section 103

A widget MAY have a metallic casing. If it does have a metallic casing it MUST have a waterproof coating.

Section 104: Localizations of Widget Size

For implementations of widgets for use in the European Union a widget that weighs between 100g and 300g and is from 5 to 15 centimeters long in its longer dimension, is a medium-size widget. However, in USA the widget is medium-sized if it weighs between 4oz and 12oz and is from 2 inches to 6 inches long.

<< Table of widget sizes >>

Table 21

Test Assertions for Dummy Widget Specification 1.1

Example Test Assertions

June 2008

...

Test Assertions for Specification Sections 100 to 104

TA id: widget-TA100-1

Target: widget

Normative Source: specification requirement 100

Predicate: [the widget] is of rectangular shape

Prescription Level: mandatory

TA id: widget-TA101-1a

Target: medium-size widget

Normative Source: specification requirement 101, part 1

Predicate: [the widget] uses exactly one AA battery.

Prescription Level: mandatory

TA id: widget-TA101-1b

Target: medium-size widget

Normative Source: specification requirement 101, part 2

Predicate: [the widget] has a red button on top.

Prescription Level: mandatory

TA id: widget-TA102-1

Target: widget

Normative Source: specification statement 102, part 1

Predicate: [the widget] is waterproof.

Prescription Level: preferred

TA id: widget-TA102-2

Target: widget

Normative Source: specification statement 102, part 2

Predicate: [the widget] has a label warning that it is not waterproof.

Prescription Level: mandatory

Prerequisite: (widget-TA102-1 = false)

TA id: widget-TA103-1

Target: widget

Normative Source: specification statement 103, part 1

Predicate: [the widget] has a metallic casing.

Prescription Level: permitted

TA id: widget-TA103-2

Target: widget

Normative Source: specification statement 103, part 2

Predicate: [the widget] has a waterproof coating over its metallic casing.

Prescription Level: mandatory

Prerequisite: widget-TA103-1

TA id: widget-TA104-1

Target: widget

Normative Source: specification requirement 104

Predicate: [the widget] weighs between WEIGHT-A and WEIGHT-B.

Prescription Level: medium-sized:mandatory

TA id: widget-TA104-2

Target: widget

Normative Source: specification requirement 104

Predicate: [the widget] is from LENGTH-A to LENGTH-B long in its longer dimension.

Prescription Level: medium-sized:mandatory

Variable Scope for Localizations of Widget Sizes

The following variables apply to:

Test Assertion References:

widget-TA104-1

widget-TA104-2

Variable: 'GEOPOLITICAL-LOCATION' the geopolitical location of use of the widget, allowed values being strings enumerated in country code list ...

Variable: 'WEIGHT-A' a weight and its units. If GEOPOLITICAL-LOCATION is 'US' then WEIGHT-A is 4oz. If GEOPOLITICAL-LOCATION is 'EU' then WEIGHT-A is 100g.

Variable: 'WEIGHT-B' a weight and its units. If GEOPOLITICAL-LOCATION is 'US' then WEIGHT-B is 12oz. If GEOPOLITICAL-LOCATION is 'EU' then WEIGHT-B is 300g.

Variable: 'LENGTH-A' a length and its units. If GEOPOLITICAL-LOCATION is 'US' then LENGTH-A is 2 inches. If GEOPOLITICAL-LOCATION is 'EU' then LENGTH-B is 5cm.

Variable: 'LENGTH-B' a length and its units. If GEOPOLITICAL-LOCATION is 'US' then LENGTH-B is 6 inches. If GEOPOLITICAL-LOCATION is 'EU' then LENGTH-B is 15cm.