Open Geospatial Consortium Inc.

Date: 2009-10-19

Reference number of this OGC® project document: **OGC 08-131r3**

OGC Version: 1.0.0

Category: OGC® Policy Standard

Editor: Policy SWG

The Specification Model — A Standard for Modular specifications

Warning

This document is both an OGC Policies document and a Standard for writing OGC standards. It is subject to change based on membership requirements.

Contents Page

Foreword xiii

Introduction xiv

1 Scope 1

2 Conformance 1

3 Normative references 2

4 Terms and definitions 3

5 Conventions 8

5.1 Symbols (and abbreviated terms) 8

5.2 Abbreviations 8

5.3 Finding requirements and recommendations 8

6 Requirements Class: The Specification (Core) 9

6.1 Specification model 9

6.2 Using the specification model 13

6.3 The “standards” specification document 14

6.4 Conformance Test Suite 15

6.5 Requirements for Modularity 16

6.5.1 Each Conformance class tests a complete requirements class 16

6.5.2 Requirements classes contain all requirements tested by a conformance test case 18

6.5.3 Profiles are defined as sets of conformance classes 19

6.5.4 There is a Defined Core 19

6.5.5 Extensions are requirements classes 20

6.5.6 Optional requirements are organized as requirements classes 20

6.5.7 Requirements classes intersect overlap only by reference 21

7 Mapping this standard to types of models 21

7.1 Semantics 21

7.2 Data Models 21

7.2.1 Common modeling issues 21

7.2.2 Requirements class: UML model extends The Specification 22
7.2.3 Requirements class: XML schema extends The Specification 25
7.2.4 Requirements class: Schematron extends XML schema 27
7.2.5 Requirements class: XML meta-schema extends The Specification 28
Annex A (normative) Abstract Conformance Test Suite 29
A.1 Conformance Test Class: The Specification (Core) 29
A.1.1 Requirements are atomic and tests cover all the parts of each of the requirement 29
A.1.2 All components have an assigned URI 29
A.1.3 Requirements on vocabulary are appropriately placed 29
A.1.4 Requirements have a single target 30
A.1.5 Conformance test classes have a single target 30
A.1.6 Requirements are organized by clauses 30
A.1.7 Conformance test classes are consistent with requirements classes 30
A.1.8 Requirement classes and the Conformance Test classes in correspondence 31
A.1.9 No Optional Elements in Requirements classes 31
A.1.10 Certificate of conformance specifies all parameters used 31
A.1.11 Conformance class tests only one requirements class 31
A.1.12 Conformance class specifies all dependencies 32
A.1.13 Imported Conformance class tests are consistent with the specification 32
A.1.14 Naming consistency 32
A.1.15 Requirements in one and only one requirements class 32
A.1.16 Co-dependent Requirements are in the same requirements class 33
A.1.17 Modularity in implementation is possible 33
A.1.18 Requirements follow rules of inheritance 33
A.1.19 Profiles are expressed as sets of conformance classes 33
A.1.20 There is a named Core requirements class 34
A.1.21 General conditions are in the core 34

A.1.22 Every extension has a consistent target type 34
A.1.23 Specification is a core plus some number of extensions 34
A.1.24 Conformance to this standard is required for any extensions 35
A.1.25 Future extensions cannot be restricted 35
A.1.26 Optional requirements are organized as requirements classes 35
A.1.27 Requirements classes intersect overlap only by reference 35
A.2 Conformance test class: UML model extends The Specification 36
A.2.1 Dependency on Core 36
A.2.2 The UML model is normative 36
A.2.3 Dependency graph must be explicit 36
A.2.4 Leaf packages in only one requirements class 36
A.2.5 Requirements class associated to a unique package 37
A.2.6 A requirements class contains complete leaf packages 37
A.2.7 Classes common to all requirement classes are in the core 37
A.2.8 Package dependencies become requirements class extensions 37
A.2.9 Dependencies in packages are reflected in dependencies in requirements classes 38
A.2.10 Co-dependent packages are in the same requirements class 38
A.2.11 All classes are in leaf packages 38
A.3 Conformance test class: XML schema extends The Specification 38
A.3.1 Dependency on Core 38
A.3.2 Dependency on W3C XML schema 39
A.3.3 A requirements class corresponds to a single XML namespace 39
A.3.4 A reference to the URI of the test suite in AppInfo 39
A.3.5 Direct dependencies become schema imports 39
A.3.6 No requirements class modifies or redefines elements in another namespace 40
A.4 Conformance test class: Schematron 40
A.4.1 Dependency on XML Schema conformance test class 40

A.4.2 Each schematron pattern element implements one requirement....40
A.4.3 Each schematron pattern is in one schematron element....40
A.4.4 Each schematron @fpi attribute is used only once....41
A.4.5 Each schematron @see attribute is used only once....41
A.4.6 Each schematron fpi attribute is used only once....41
A.5 Conformance Class: XML meta-schema....41
A.5.1 Supports the Specification class....41
A.5.2 Each XML schema is conformant with the XML Schema class....42
Annex B (normative) Changes required in the OGC Standards; Transition plan for incorporation into the OGC procedures....43
B.1 Existing standards....43
B.2 New standards and specifications; during the first year....43
B.3 New standards and specifications; after the first year....43
Annex C (informative) Definitions....44
C.1 Semantically ordered definitions....44
C.2 UML Model....45
C.3 Specification....46
C.4 Conformance Suite....46
C.5 Conformance Class....46
C.6 Requirements class....47
C.7 Requirements module....47
C.8 Normative Statement....47
C.9 Requirement....47
C.10 Recommendation....48
C.11 Conformance test....48
C.12 StandardizationTarget....48
C.13 StandardizationTargetType....49
Annex D Bibliography....50
Bibliography for examples....50

Table of Figures

Page

Figure 6-1: Abstract superclass example 14

Figure C.1: Specification structure 45

Table of Requirements Page

Req 1 All the parts of a requirement, a requirement module or requirements class shall be tested. Failure to meet any part of any requirement shall be a failure to pass the associated conformance test class. 10

Req 2 Each component of the standard, including requirements, requirements modules, requirements classes, conformance test cases, conformance modules and conformance classes shall be assigned a URI as specified by the OGC naming authority or its equivalent. 10

Req 3 Requirements on the use and interpretation of vocabulary shall be in the requirements class where that use or interpretation is used. 12

Req 4 Each requirement in a conformant specification shall have a single standardization target type. 13

Req 5 All conformance tests in a single conformance test class in a conformant specification shall have the same standardization target. 13

Req 6 The requirements shall be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements modules and requirements classes. 14

Req 7 The requirements structure of the document shall be in a logical correspondence to the test suite structure. 14

Req 8 The requirements classes shall be in a one-to-one correspondence to the conformance test classes, and thus to the various certificate of conformance types possible for a candidate implementation. 16

Req 9 A Conformance class shall not contain any optional conformance tests. 16

Req 10 A certificate of conformance shall specify all parameter values used to pass the tests in its conformance test class. 16

Req 11 A Conformance class shall explicitly test only requirements from a single requirements class. 16

Req 12 A Conformance class shall specify any other conformance class upon which it is dependent and that other conformance class shall be used to test the specified dependency. 17

Req 13 If a requirements class is imported from another standard for use within a specification conformant to this standard, and if any imported requirement is "optional," then that requirement shall be factored out as a separate requirements class in the profile of that imported standard used in the conformant specification. Each such used requirements class shall be a conformance class of the source standard or a combination of conformance classes of the source standard or standards. 17

Req 14 For the sake of consistency and readability, all requirements classes and all conformance test classes shall be explicitly named, with corresponding requirements classes and conformance test classes having similar names. 17

Req 15 Each requirement in the standard shall be contained in one and only one requirements class. Inclusion of any requirement in a requirements class by a conformance class shall imply inclusion of all requirements in its class (as a dependency). 18

Req 16 If any two requirements or two requirements modules are co-dependent (each dependent on the other) then they shall be in the same requirements class. If any two requirements classes are co-dependent, they shall be merged into a single class. 18

Req 17 There shall be a natural structure on the requirements classes so that each may be implemented on top of any implementations of its dependencies and independent of its extensions. 18

Req 18 No requirements class shall redefine the requirements of its dependencies, unless that redefinition is for an entity derived from but not contained in those dependencies. 19

Req 19 The conformance tests for a profile of a specification shall be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets. 19

Req 20 Every specification shall define and identify a core set of requirements as a separate conformance class. 19

Req 21 All general recommendations shall be in the core. 19

Req 22 Every other requirements class in a specification shall have a standardization target type which is a subtype of that of the core and shall have the core as a direct dependency. 19

Req 23 Each specification conformant to this standard shall consist of the core and some number of requirements classes defined as extensions to that core. 20

Req 24 A specification conformant to this standard shall require all conformant extensions to itself to be conformant to this standard. 20

Req 25 A specification conformant to this standard shall never restrict in any manner future, logically-valid extensions of its standardization targets. 20

Req 26 The only optional requirements acceptable in a specification conformant to this standard shall be expressible as a list of conformance classes to be passed. 20

Req 27 The common portion of any two requirements classes shall consist only of references to other requirements classes. 21

Req 28 An implementation passing the UML conformance test class shall first pass the core conformance test class. 22

Req 29 To be conformant to this UML conformance class, UML shall be used to express the object model, either as the core mechanism of the standard or as a normative adjunct to formally explain the standard in a model. 23

Req 30 A UML model shall have an explicit dependency graph for the leaf packages and external packages used by the standard consistent with the way their classifiers use those of other packages. 23

Req 31 A UML leaf package shall be associated directly to only one requirements class. 24

Req 32 Each requirements class shall be associated to a unique package in the model and include either directly or by a dependency any requirement associated to any of its subpackages. 24

Req 33 A requirements class shall be associated to some number of complete leaf packages and all classes and constraints in those packages. 24

Req 34 Classes that are common to all requirements classes shall be in a package associated to the core conformance/requirements class. 24

Req 35 In the UML model, if a "source" package is dependent on a "target" package then their requirements class shall be equal or the source package's class shall be an extension of the target package's class. 24

Req 36 If one leaf package is dependent on another leaf package, then the requirements class of the first shall be the same or an extension of the requirements class of the second. 25

Req 37 If two packages have a two-way dependency (a "co-dependency"), they shall be associated to the same requirements class. 25

Req 38 The UML model shall segregate all classes into leaf packages. 25

Req 39 An implementation passing the XML schema conformance test class shall first pass the core specification conformance test class. 25

Req 40 An implementation passing the XML schema conformance test class shall first pass the W3C Recommendation for XML schema. 25

Req 41 If a specification conformant to the XML schema conformance class defines a set of data schemas, all components (e.g. elements, attributes, types …) associated with a single conformance test class shall be scoped to a single XML namespace. 26

Req 42 The all-components schema document for an XML Schema shall indicate the URI of the associated conformance test class in the schema/annotation/appinfo element. 26

Req 43 If a specification conformant to the XML schema conformance class defines a direct dependency from one requirement class to another, then a standardization target of the corresponding conformance test class shall import a schema that has passed the associated conformance test class (dependency) or shall itself pass the associated conformance test class. 26

Req 44 No requirements class in a specification conformant to the XML schema conformance class shall modify elements, types or any other requirement from a namespace to which it is not associated. 27

Req 45 A specification passing the Schematron conformance test class shall also define or reference an XML schema that shall pass the XML schema conformance class from this standard. 27

Req 46 Each sch:pattern element shall implement constraints described in no more than one requirement. Each requirement shall be implemented by no more than one sch:pattern. 27

Req 47 Each sch:pattern element shall be contained within one sch:schema element. 27

Req 48 The value of the sch:schema/@fpi attribute shall be a URI that identifies this implementation 27

Req 49 The value of the sch:schema/@see attribute shall be the identifier for the requirements class that contains the requirement(s) implemented by the schema 27

Req 50 The value of the sch:schema/@fpi attribute shall be used on only one Schematron schema. 27

Req 51 A specification passing the XML meta-schema conformance test class shall first pass the core specification conformance test class. 28

Req 52 A specification passing the XML meta-schema conformance test class shall require that its specification targets (XML schema) pass the XML schema conformance class from this standard. 28

i. Preface

This standard contains requirements for writing standards to be used for any document whose eventual purpose is the specification of requirements for software, services or data structures.

Suggested additions, changes, and comments on this standard are welcome and encouraged. Such suggestions may be submitted through the OGC change request system (http://www.opengeospatial.org/standards/cr).

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the imperative verb form used to indicate a requirement to be strictly followed to conform to this standard.

iii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

Organization Represented
CSIRO
ESRI
SeiCorp, Inc.
Oracle USA
University of the Bundeswehr – ITS
US National Geospatial-Intelligence Agency (NGA)
interactive instruments GmbH
US Department of Homeland Security (DHS)
Bentley Systems, Inc.
BAE Systems - C3I Systems

iv. Document contributors

The following voting members of the Standards Working Group participated in editing this document:

Person	Organization Represented
Simon Cox	CSIRO
David Danko	ESRI
James Greenwood	SeiCorp, Inc.
John R. Herring	Oracle USA
Andreas Matheus	University of the Bundeswehr – ITS
Richard Pearsall	US National Geospatial-Intelligence Agency (NGA)
Clemens Portele	interactive instruments GmbH
Barry Reff	US Department of Homeland Security (DHS)
Paul Scarponcini	Bentley Systems, Inc.
Arliss Whiteside	BAE Systems - C3I Systems

v. Revision history

This is the first normative version of this document.

vi. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification and Standard will require changes to accommodate the contents of this document. At their next revision, any volume of the abstract specification or implementation standards will have to be made to conform to the requirements here.

vii. Future work

Any needed improvements in this document will be made as experience with using it increases.

Foreword

This standard began as discussions in the OGC Architecture Board (OAB) about general principals of application development taken from the OAB members' collective experience.

Since this standard is a summary of collective experience, no one should claim intellectual property rights to its contents. This standard may be used by anyone as long as its source is specified.

This standard, while it specifies the structures of other standards, does not supply them with specific content, since its level of abstraction is one level higher than any standard that would normally claim conformance. Where possible, this standard is conformant with itself (with respect to the core conformance test class, Clause 6 and Annex A.1).

Since this standard specifies requirements for standards to be acceptable by the OGC TC, it is logically an annex of the TC Policy and Procedures (TC-PnP). It is however a radically different document, since the TC-PnP is a procedural set of rules, and this document is a set of testable constraints against a finished document. The mechanism for enforcement of this standard is the purview of the TC, its subgroups, and in particular the OAB.

Recent OAB discussions have identified the need for a "check list" for each candidate standard to assure its conformance to OGC policy and rules. Conformance to this document should be added to that list.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Introduction

"A really Useful Engine"

Thomas the Tank Engine,
by Rev. W. V. Awdry, 1946

Our friend Thomas, in a Zen-like moment unusual for a child's story and definitely unusual for a steam locomotive, decides that his purpose in life is to be a "Really Useful Engine[1]." A standard's usefulness and hence worth can be measured by:

- The ease with which it can be implemented.
- The number of times it has been implemented.
- The number of times those implementations are used.
- The ease with which those implementations interact.
- The number of times it has been extended through inclusion in other useful standards.

Some of these are affected by the choice of topic, but all are affected by the internal logical structure of the standard. This standard specifies generic rules for this internal structure conducive to being a really useful engine.

A standard is a partial solution to a design problem which limits "conformant" solutions to enhance interoperability and harmony between disparate implementations. Design issues and requirements are transformed into statements about the solution design, and are then presented in the standard as requirements of the solution, usually as requirement statements targeting the solution.

Thus, a standard presents requirements targeting implementations of solutions of the original problem, which must be satisfied by passing the tests of the conformance suite. These tests are organized into conformance classes, each of which represents a mechanism for partial satisfaction of the standard. These give the standard a modular structure, each requirements class represented by a conformance class. In a well written standard, the normative clauses and any model or schema are organized in a manner that parallels the conformance clause.

NOTE This standard has been referred to as the "core and extension model" due to its insistence on a modular structure throughout all parts of a specification and its implementation.

[1] The capitalization is the Rev. Audrey's. See http://en.wikipedia.org/wiki/Thomas_the_Tank_Engine

OGC 08-131r3

1 Scope

This standard specifies some desirable characteristics of a standards specification that will encourage implementations by minimizing difficulty determining requirements, mimicking implementation structure and maximizing usability and interoperability.

Clause 6.1 contains the UML model of a specification upon which this standard is described. Annex C contains informal and non-normative definitions ordered for ease of understanding. These two sections can be read first to aid in the understanding of the rest of the document.

2 Conformance

Since this standard adds requirements to the procedure for the adoption of OGC standard, for proper use it should be adopted by the OGC Technical Committee in a motion worded something like the following:

> **This standard shall be considered as a normative annex to the "current" OGC Policy and Procedures[2].**

The effect on these procedures and a transition plan of existing and "soon to be adopted" standards is detailed in Annex B. These procedures should also be adopted by the OGC TC.

Conformance to this standard by technical implementation standards and specifications can be tested by inspection. The test suite is in Annex A.

There are 5 conformance classes for this standard:

1. Specification documents in general (the core) – see Clause 6 and Annex A.1
2. Specifications using UML to state requirements, extending the core – see Clause 7.2.2 and Annex A.2
3. Specifications using XML schema to state requirements, extending the core – see Clause 7.2.3 and Annex A.3

[2] The reference to the OGC TC PnP is floating, and does not apply to a particular document, but to the policy and procedures of the Open Geospatial Consortium whether defined by an OGC document, by *Robert's Rules of Order*, or by common consent and practice of the membership. The status of this standard in the OGC Policy and Procedures must be found in that "virtual document". This requirement is for that document and is not a requirement of this one.

4. Specifications using Schematron to state requirements, extending XML schema – see Clause 7.2.4 and Annex A.4

5. Specifications defining requirement for a new category of XML schemas, extending the core, whose target XML schemas must be conformant with the XML schema conformance class above – see Clause 7.2.5 and Annex A.5.

This standard contains normative language and thus places requirements on conformance, or mechanism for adoption, of candidate standards to which this standard applies. In particular:

- Clause 6 specifies the core requirements which shall be met by all conformant standards.

- Clause 7 gives information on how this standard is to be applied to requirements, conformance clauses, UML models, or XML Schemas.

- Annex B gives a transition plan to deal with "work in progress."

The various subclauses of Clause 7 list requirements partially derived from the core, but more specific to the conditions where a data model expressed in one of the specified languages (UML or XML) is involved. These requirements classes are extensions of the core presented in Clause 6.

3 Normative references

While this document references UML, SQL and XML, and their technical specifications, it does not derive any of its requirements from these documents. While this standard may be applied to extensions of those standards, conformance to them is the purview of those standards, not this one.

The following are normative references for this standard in the sense that they supplied definitions used here

[1] ISO/IEC 10000-1: Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework

[2] ISO Directives Part 2; available at ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards

[3] ISO 19105: Geographic Information — Conformance and testing

[4] ISO/IEC 19501, Information technology -- Open Distributed Processing -- Unified Modeling Language (UML)

[5] OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2, OMG Document Number: formal/2007-11-04, Standard document URL: http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF

[6] OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, OMG Document Number: formal/2007-11-02; Standard document URL: http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF

[7] ISO/IEC 19757-3:2006 Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

[8] W3C XML Schema Part 1: Structures Second Edition. W3C Recommendation (28 October 2004)

[9] W3C XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (28 October 2004)

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply. Terms not defined here take on their meaning from computer science or from their Standard English (common US and UK) meanings. The form of the definitions is defined by ISO Directives.

Many of these definitions depend upon the model given in Clause 6.1.

4.1 all-components schema document

XML schema document which includes, either directly or through the inclusion of other schema documents, all schema components associated to its namespace

4.2 certificate of conformance

evidence of conformance to all or part of a standard, awarded for passing one or more of the **conformance test classes** specified in that standard

NOTE "Certificates" do not have to be instantiated documents; having proof of passing the conformance test class is sufficient. For example, the OGC currently keeps an online list of conformant applications at http://www.opengeospatial.org/resource/products.

Each certificate of conformance is awarded to a **standardization target**.

4.3 conformance test case

test for a particular requirement or a set of related requirements

NOTE When no ambiguity, the word "case" may be omitted. i.e. **conformance test** is the same as **conformance test case**.

4.4 conformance test module

set of related tests, all within a single **conformance test class**

[ISO 19105]

NOTE When no ambiguity, the word "test" may be omitted. i.e. **conformance test module** is the same as **conformance module**. Conformance modules may be nested in a hierarchical way.

This term and those associated to it are included here for consistency with ISO 19105.

4.5 conformance test class; conformance test level

set of **conformance test modules** that must be applied to receive a single **certificate of conformance**

NOTE When no ambiguity is possible, the word "test" may be left out, so **conformance test class** maybe called a **conformance class**.

In this standard, the set of **requirements** tested by the conformance tests within a **conformance class** is a **requirements class** and its dependencies. Each optional **requirement** will be in a separate **requirements class** with other **requirements** that are part of the same option. Each **requirements class** corresponds to a separate **conformance class**

In other words, the only options in a specification conformant to this standard will be if a particular **requirements class** is tested. Each requirements class will be in a 1 to 1 correspondence to a similarly named **conformance class** that tests all of the **requirements class's requirements**.

All **requirements** in a **conformance class** will have the same **standardization target**.

The term "level" is a synonym for "class" and is part of the ISO nomenclature. Here the two terms are treated as equivalent but ISO usually has differences in the way they are named and how they related to one another level or class.

4.6 conformance suite

set of **conformance classes** that define tests for all **requirements** of a standard

NOTE The **conformance suite** is the union of all **conformance classes**. It is by definition the **conformance class** of the entire specification.

In this standard, each **requirement** is mandatory within its **conformance class** and each **requirement** is tested in at least one **conformance test**.

4.7 conformance test

test, abstract or real, of one or more **requirements** contained within a standard, or set of standards

4.8 core requirements class

unique **requirements class** that must be satisfied by any conformant **standardization targets** associated to the specification

NOTE The **core requirements class** is unique because if it was possible to have more than one, then each **core** would have to be implemented to pass any **conformance test class**, and thus would have to be contained in any other **core**. The **core** may be empty, or all or part of another standard or related set of standards.

The "**core**" can refer to this **requirements class**, its associated **conformance test class** or the software module that implements that requirements class.

4.9 direct dependency (of a requirements class)

another **requirements class** (the dependency) whose **requirements** are defined to also be **requirements** of this **requirements class**

NOTE A **direct dependency** of the current **requirements class** will have the same **standardization target** as the current **requirements class**. This is another ways of saying that the current **requirements class** extends, or uses all the aspects of the **direct dependency**. Any tests associated to this **dependency** can be applied to this **requirements class**.

When testing a **direct dependency**, the **standardization target** is directly subject to the test in the specified **conformance test class** of the **direct dependency**.

4.10 indirect dependency (of a requirements class)

requirements class with a different **standardization target** which is used, produced or associated to by the implementation of this **requirements class**

NOTE In this instance, as opposed to the **direct dependency**, the test against the consumable or product used or produced by the **requirements class** does not directly test the **requirements class**, but tests only its side effects. Hence, a particular type of feature service could be required to produce valid XML documents, but the test of validity for the XML document is not directly testing the service, but only indirectly testing

the validity of its output. **Direct dependencies** test the same **standardization target**, but **indirect dependencies** test related but different **standardization targets**.

The **standardization target** of the **indirect dependency** is different from the target of "this requirements class" but it may be of the same or related **standardization target type**. For example, if one service is related to another second service, then a service **requirement** may be placed against the second associated service to assure that the first service has access to its functionality. For example, if a DRM-enabled service is required to have an association to a licensing service, then the requirements of a licensing service are indirect requirements for the DRM-enabled service. Such a requirement may be stated as the associated licensing service has a **certificate of conformance** of a particular kind.

4.11 extension (of a requirements class)

requirements class which has a **direct dependency** on another **requirements class**

NOTE Here **extension** is defined on **requirements class** so that their implementation may be software extensions in a manner analogous to the extension relation between the **requirements classes**.

4.12 general recommendation

recommendation applying to all entities in a specification model

4.13 home (of a requirement or recommendation)

official statement of a **requirement** or **recommendation** that is the precedent for any other version repeated or rephrased elsewhere

NOTE Explanatory text associated to normative language often repeats or rephrases the requirement to aid in the discussion and understanding of the official version of the normative language. Since such restatements are often less formal than the original source and potentially subject to alternate interpretation, it is important to know the location of the **home** official version of the language.

These alternative statements use non-normative language and are **statements** using the definitions of the ISO Directives Part 2.

4.14 leaf package

UML model package that does not contain any subpackages, but contains classifiers

[UML]

4.15 model
abstract model
conceptual model

theoretical construct that represents something, with a set of variables and a set of logical and quantitative relationships between them.

NOTE Derived from *Wikipedia*

The "theoretical construct" is essentially a **conceptual metaphor** with the **target** of the **metaphor** as the thing being modeled, and the **source** of the **metaphor** as the **model**. The terms are almost interchangeable, with **model** being preferred when the **source** is a constructed entity, and **metaphor** being preferred when the **source** already exists, and the emphasis is the mapping between it and the **target**.

The definition in ISO 19101, Clause 4.4 is
conceptual model - model that defines concepts of a universe of discourse.

While adequate in the context of a "universe of discourse" as the something addressed by a standard, a model need not have any "universality" property at all. Most often models are representative of only a relatively small portion of a larger universe, and part of the process of modeling is to factor out the

properties and things to which no interest is directed within the present standard It also fails to define "model" which is in fact the central issue within this discussion.

The **abstract** or **conceptual** is actually redundant and will often be dropped in the text. **Models** are by their vary nature not the same as what they are describing, and thus must contain a **conceptual metaphor** to describe their relationship to the **target** (the thing being described) of the model. This inherently makes them abstractions.

4.16 profile

specification or standard consisting of a set of references to one or more base standards and/or other profiles, and the identification of any chosen **conformance test classes**, conforming subsets, options and parameters of those base standards, or profiles necessary to accomplish a particular function.

[ISO/IEC TR 10000-1]

NOTE This definition has been adopted from ISO 10000: Part 1. The wording has been changed to accommodate the shared vocabulary of OGC and ISO TC 211 and for editorial reasons. The original text is "A set of one or more base standards and/or ISPs, and, where applicable, the identification of chosen classes, conforming subsets, options and parameters of those base standards, or ISPs necessary to accomplish a particular function."

In the usage of this standard, a profile will be a set of requirements classes or conformance classes (either preexisting or locally defined) of the base standards.

This means that a **standardization target** being conformant to a profile implies that the same **target** is conformant to the standards referenced in the **profile**.

4.17 recommendation

expression in the content of a document conveying that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited

[ISO Directives Part 2]

NOTE Although using normative language, a **recommendation** is not a **requirement**. The usual form replaces the "shall" (imperative or command) of a **requirement** with a "should" (suggestive or conditional).

4.18 requirement

expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted

[ISO Directives Part 2]

NOTE Each **requirement** is a normative criterion for a single **type of standardization target.** In this standard, **requirements** will be associated to **conformance tests** that can be used to prove compliance to the underlying criteria by the **standardization target**.

The implementation of a **requirement** is dependent on the type of specification being written. A data specification requires data structures, but a procedural specification requires software implementations. The view of a standard in terms of a set of testable **requirements** allows us to use set descriptions of both the standard and its implementations.

The specification of a **requirement** is usually expressed in terms of a model of the **standardization target**, such as a UML model, or an XML or SQL schema. Anything without a defined test is *a priori* not testable and thus would be better expressed as a **recommendation.**

Requirements use normative language and in particular are commands and use the imperative "shall" or similar imperative constructs. Statements in standards which are not requirements and need to be either conditional or future tense normally use "will" and should not be confused with requirements that use "shall" imperatively.

4.19 requirements class

aggregate of all **requirement modules** that must all be satisfied to satisfy a **conformance test class**

NOTE There is some confusion possible here, since the testing of indirect dependencies seems to violate this definition. But the existence of an indirect dependency implies that the test is actually a test of the existence of the relationship from the original target to something that has a property (satisfies a condition or requirement from another requirements class).

4.20 requirements module

aggregate of **requirements** and **recommendations** of a specification against a single **standardization target type**

NOTE This term is included to be consistent with the use of modules in ISO 19105. The third type of normative language, the "permission" which uses "may," is not considered here mainly because it is usually used to prevent a requirement from being "over interpreted" and as such is considered to be more of a "statement of fact" than a "normative" condition.

4.21 specification

document containing **recommendations**, **requirements** and **conformance tests** for those **requirements**

NOTE This definition is included for completeness. See Clause 5.3.

This does not restrict what else a standard may contain, as long as it does contain the three types of element cited.

4.22 standard

specification that has been approved by a legitimate Standards Body

NOTE This definition is included for completeness. **Standard** and **specification** can apply to the same document. While **specification** is always valid, **standard** only applies after the adoption of the document by a legitimate standards organization.

The legitimate Standards Bodies for OGC consist of OGC, ISO and any of the other standards bodies accepted and used as a source of normative references by OGC or ISO in their standards. In the normal meaning of the word "standard", there are other conditions that may be required, but this standard has chosen to ignore them in the process of abstraction.

4.23 standardization target

entity to which some **requirements** of a **standard** apply

NOTE The **standardization target** is the entity which may receive a **certificate of conformance** for a **requirements class.**

4.24 standardization target type

type of entity or set of entities to which the **requirements** of a **standard** apply

NOTE The **standardization target types** give the **standardization targets** a typing system similar to the UML classifiers. In general, the types inherit from one another in the same way that UML classes do. The same class/metaclass semantics apply, and two targets can be considered to have the "same type" (in a particular situation) if their instantiation types share the appropriate supertype, as is the case in UML.

In OGC for example, all service types that must pass the OWS (Open Web Services) Common specification are some extension of the "Open Web Service" **standardization target type**. This makes OWS Common a default "global core" for all OGC Services.

In some cases, the **standardization target type** may be another specification. A GML application schema is a **standardization target** for the GML standard, but is in turn a specification of instances of that application schema.

4.25 statement

expression in a document conveying information

[ISO Directives Part 2]

NOTE Includes all statements in a document not part of the normative **requirements**, **recommendations** or **conformance tests**. Included for completeness.

5 Conventions

5.1 Symbols (and abbreviated terms)

All symbols used in this document are either:

1. common mathematical symbols
2. UML 2 (Unified Modeling Language) as defined by OMG and accepted as a publicly available standard (PAS) by ISO in its earlier 1.3 version.

5.2 Abbreviations

In this document the following abbreviations and acronyms are used or introduced:

ERA	Entity, Relation, Attribute (pre-object modeling technique)
ISO	International Organization for Standardization (from Greek for "same")
OGC	Open Geospatial Consortium (http://www.opengeospatial.org/)
OOP	Object Oriented Programming
OOPL	OOP Language (such as C++ or Java)
SQL	ISO/IEC 9075 query language for relational databases, not originally an acronym, but now often cited as "Structured Query Language"
TC	Technical Committee (usually either in ISO or OGC)
UML	Unified Modeling Language (an object modeling language)
XML	eXtensible Markup Language
OMG	Object Management Group (http://www.omg.org/)
OCL	Object Constraint Language (part of UML)

5.3 Finding requirements and recommendations

For clarity, each normative statement in this standard is in one and only one place and is set in a **bold font.** If the statement of the requirement or recommendation is repeated for clarification, the "**bold font" home** of the statement is considered the official statement of the normative

requirement or recommendation. In this sense, all requirements in this standard are listed in the Table of Requirements at the beginning of this standard (page vii).

In this standard, all requirements are associated to tests in the abstract test suite in Annex A. The reference to the requirement in the test case is done by a requirements label (in the form "**Req #**", where "#" is a number) associated to the "**bold font" home** of the statement described above. Recommendations are not tested and are not labeled, although they still use a **bold font** for their unique home statement.

Requirements classes are separated into their own clauses and named, and specified according to inheritance (direct dependencies). The Conformance test classes in the test suite are similarly named to establish an explicit and mnemonic link between requirements classes and conformance test classes.

In this standard, other documents which may be standards but are being tested for conformance to this standard are referred to as "specifications." The purpose of this linguistic artifact is to prevent confusion between this standard and the standardization targets of its requirements.

6 Requirements Class: The Specification (Core)

6.1 Specification model

This standard lays requirements against other specifications by using a set-theoretic description of those specifications based on their structure as organized sets of criteria, those that are to be tested ("requirements") and those that are not tested ("recommendations"). All specifications, formally or informally, create an abstraction of the things to which they apply (called an "abstract model" or "metaphor" if more informally constructed) in order to provide a context for stating requirements. This standard is considered one of its own standardization targets and thus a subject of its own requirements.

This standard assumes that the specifications addressed are in a commonly used logical form. This form can be specified by the following descriptions:

1. A specification contains Clauses (corresponding to numbered sections as they might appear in a table of contents) which describe its standardization target and its requirements.

2. A specification contains Annexes or is associated to other documents (both a logical type of Clause), one of which is the Conformance Test Suite (which may be an abstract description of the test suites to be implemented separately).

3. All requirements, recommendations and models are introduced and defined first in the numbered Clauses.

4. All requirements are identifiable as requirements. In OGC and ISO, this means use of "normative" language, meaning the proper use of SHALL, SHOULD, CAN and MAY or similar wording in the passive voice. (MUST in ISO is reserved for "external statutory obligations" which are not usually carried as requirements)

5. All tests for conformance to those requirements are defined in the Conformance Test Suite.

6. Tests may be grouped for convenience into conformance test modules (ISO 19105)
7. The tests, if conducted, will determine to some degree of certainty whether an implementation meets the requirements which the tests reference.
8. The tests are organized into some number of conformance "classes". If a standard does not do this, it is has by default only one "conformance class".
9. Certificates of conformance (see 4.1) are awarded by a testing entity based on these conformance classes.
10. There is a clear distinction between normative and. informative parts of the text.
11. Examples and notes are informative, and do not use "normative" language.[3]

A UML representation of important properties of this model is given in Annex C.2.

This standard defines a "requirement" of a specification as an atomic testable criterion. See the formal definition of requirement in 4.18

NOTE "Atomic" means non-decomposable from the Greek word for "indivisible." Thus, each requirement is not to be further divided (even if it can logically be done) into smaller requirement statements.

Req 1 All the parts of a requirement, a requirement module or requirements class shall be tested. Failure to meet any part of any requirement shall be a failure to pass the associated conformance test class.

NOTE This means that any failure to pass the test specified for a part of requirement is a failure to pass the requirement.

Req 2 Each component of the standard, including requirements, requirements modules, requirements classes, conformance test cases, conformance modules and conformance classes shall be assigned a URI as specified by the OGC naming authority or its equivalent.

These URI identities should be used in any external documentation that reference these component elements in a normative manner, including but not limited to other standards, implementations of the conformance test suite, or certificates of conformance for implementations conformant to the standard in question. The precise enforcement of this requirement and its associated recommendation is the purview of the OGC URI/URN Naming Authority or its equivalence.

A requirement may have a variety of parts spread throughout the standard, but somewhere in the normative clauses, the requirement will be defined in terms of the local modeling paradigm or in terms of the conceptual model of the target of the standard. This place of its definition shall be its "**home**" (see clause 5.3) and will be the only place where full normative language is used. This is because two separate statements of the same requirement could eventually diverge in meaning or interpretation.

[3] In this standard, in informative sections, the word "will" implies that something is an implication of a requirement. The "will" statements are not requirements, but explain the consequence of requirements.

In the conformance test suite there will be a test defined to verify the validity of the claim that an implementation of the standard (standardization target) satisfies each requirement. Since the normative language of the body of the standard and the conformance test classes both define what conformance to the standard means, they will be equivalent in a well-written specification. This standard requires specifications to be well-written, at least in stating requirements and conformance tests.

Conformance tests are aggregated into conformance classes that specify how certain "certificates of conformance" are achieved. The natural inclination is to aggregate the requirements. The issue that blocks this approach is that some requirements are optional while others are mandatory. To achieve a cleaner separation of requirements, this standard separates them into sets (called "requirements classes"), each of which has no optional components. Since the statement of each requirement has only one "**home**", it will be in a clause associated to its requirements class.

So, this standard defines a "requirements class" as a set of requirements that must all be passed to achieve a particular conformance class (see 4.5). Because ISO 19105 includes a "middle" structure called a conformance test module, this standard also includes requirements modules to parallel the conformance test modules. A specification written to this standard may use this "module" structure in any manner consistent with the rest of this standard and consistent with ISO 19105

Those requirements that are sometimes referred to as "optional" requirements will be segregated into separate requirements classes by themselves. This allows the options in the testing procedure to be grouped into non-varying optional conformance classes To conform to this standard, requirements classes may be optional, but each requirement within a requirements class is mandatory when that requirements class is implemented. When needed, a particular requirements class may contain only a single requirement.

Care must be taken, since the requirements classes are not always in a one-to-one correspondence to conformance classes in other standards which may be the source of requirements for a specification conformant to this standard. If other standards are used, their options shall be specified to be useable within a standard conformant to this standard, see 6.5.1.

Conformance classes contain dependencies on one another. These are represented by tests in one conformance class that state that another conformance class must be passed to qualify to pass this conformance class. In terms of requirements, that says that the dependent conformance class contains tests (by reference) for all requirements of the "included" conformance class.

Translating this into this standard's view of requirements classes, one requirements class is dependent on another if the other is included through such a reference. In this manner, requirements classes can be treated as sets of requirements (each in a single requirements class but included in others by reference to its "home" requirements class).

NOTE The set containment lattice (partial ordering) defined by:

$$[A < B] \Leftrightarrow [\{\text{requirements in } A\} \subset \{\text{requirements in } B\}]$$

This says that to pass "B", an implementation must pass "A." If "A<B", we say that "A" is a profile of "B" (see 4.11) and that "B" extends "A" and that "B" depends on "A" (see 4.9).

A "core" requirements class is one which is a dependency of all others (see 4.8). So, if we consider a standard "**S**" as a set of normative requirements classes:

$$[A \text{ is core in standard } \mathbf{S}] \Leftrightarrow [\forall\ B \in \mathbf{S}:\ A < B\}]$$

So, a core is a "universal lower bound," and a base is a "universal upper bound."

In this standard, each conformance requirement is separated in its own labeled paragraph, such as Req 1 above.

The distribution of the information in a specification is not restricted. The only requirement will be that the specification's requirements be grouped in a manner consistent with the conformance test classes, see Req 6 and Req 7. **The informational and structural universals of the specification may be included in the core text and its associated models without violations of this standard.** This is true as long as the requirements of the extension are not implicit in what is included in the core.

In this manner, the core requirements class and its associated contents can be thought of not only as the requirements of the core conformance class, but as a form of reference model for establishing core vocabularies and schemas for the entire specification.

The core may contain the definition and schema of commonly used terms and data structures for use in other structures throughout the specification.

This may include the list of the names of all operations and operation parameters to be used in any request-response pairs defined in any conformance class of the specification. If a service receives a request that is not supported in its conformance claim, then the service may return an error message text stating that the requested operation is part of a non-supported extension.

Req 3 Requirements on the use and interpretation of vocabulary shall be in the requirements class where that use or interpretation is used.

Importation of external vocabularies and schemas may be in the core.

Example In the specification of a metadata service, the Dublin Core concept of a "Title" and the XML schema structure used for its specification can be in the core of the service specification. How a particular request-response pair uses the data structure to mean the title of a particular document or dataset will be specified in the requirements class in which the request-response pair is defined and set against requirements.

6.2 Using the specification model

The primary difficulty in speaking of specifications (or candidate standards)[4] as a group is their diverse nature. Some specifications use UML to define behavior, others use XML to define data structures, and others use no specific modeling language at all. However, they all must model the standardization target to which they apply since they need to use unambiguous language to specify requirements. Thus, the only thing they have in common is that they define testable requirements and recommendations against some model of an implementation of the specification (the standardization target). For completeness, they should also specify the conformance tests for these requirements that are to be run for validation of the implementations against those requirements.[5]

For simplicity in the text here, we assume that each specification has a single (root) standardization target type from which all extensions inherit. If this is not true, then the specification can be logically factored into parts each corresponding to a "root" standardization target type, and that the specification addresses each such part separately (see the definition of requirements class in 4.19). In this sense, the next requirement divides specification into parts more than restricting their content.

Req 4 Each requirement in a conformant specification shall have a single standardization target type.

In practice, the standardization target of the core requirements class is the root of an inheritance tree where extensions all have the core's target as an ancestor, and thus can be considered as belonging to the same "class" or type as the core's target.

Req 5 All conformance tests in a single conformance test class in a conformant specification shall have the same standardization target.

This means that all requirements are considered as targeting the same entity being tested for a particular certificate of conformance. The test may specify other types as intermediaries or indirect dependencies (see 4.10).

If needed, a requirement may be repeated word for word in another requirement up to but not including the identification of the standardization target type. This second statement will be in a separate requirements class, since it will have a separate standardization target and thus belong to the requirements to be tested by a separate conformance class. For example, in a service interface, a specification may be written that requires both the client and server to use a particular language for data transmission. Since the client and server are different standardization targets types (except in some special circumstances), they will have different conformance test classes.

One solution is to state the requirement twice, once for each target. The most common alternative is to introduce a new "superclass". **The specification may introduce an abstract superclass of**

---

[4] This is purposely written as "as yet not adopted" standards, since it is during the authoring process that this standard must be considered, not *ex post facto*. It is also handy since the word "standard" in this section means this document; while the term "specification" here means another document to which this standard is being applied..

[5] This "test suite" specification is a requirement for ISO and for OGC, but is often ignored in less formal standardization efforts. In such cases, if there exists a "validation authority" for conformance, they must interpret the requirements to be tested, *ex parte*, possibly separated from the authors of the standard, leading to issues of separate interpretations of the same specification.

all affected standardization target types and use this for the requirements common to all of the affected target types. This is diagrammed in Figure 6-1.

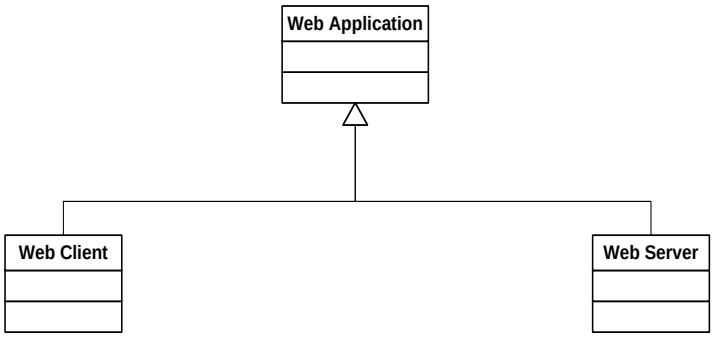

Figure 6-1: Abstract superclass example

Example 6-1 Abstract Superclass

In Figure 6-1, "`Web Application`" has been created as an abstract superclass of "`Web Server`" and "`Web Client.`" In this manner, any common requirement for "`Web Server`" and "`Web Client`" can be laid against the abstract superclass "`Web Application`" and does not have to be repeated. Because the subclassing created a direct dependency from the packages containing "`Web Server`" or "`Web Client`" and that containing "`Web Application`" a corresponding direct dependency will exist at the conformance class level so that any requirement or compliance test defined for "`Web Application`" will also apply to directly to both "`Web Server`" and "`Web Client.`" The conformance class of "`Web Application`" is tested whenever either "`Web Service`" or "`Web Client`" is tested see Req 12.

Using this technique may pose some problems at the concrete level because the semantics of the requirement may shift subtly between the "`Web Server`" and the "`Web Client.`" If this happens, the original requirement statement at "`Web Application`" might be ambiguously stated (it has multiple semantically distinct interpretations depending on the subclass of "`Web Application`" to which it is applied). It would be appropriate in this case to move the requirement down from the abstract superclass, replicating it into each of the subclasses, where it can be stated more precisely.

The conformance class of "`Web Application`" is tested whenever either "`Web Service`" or "`Web Client`" is tested, see Req 12.

6.3 The "standards" specification document

Each specification is a set of requirements and their associated conformance tests.

Req 6 **The requirements shall be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements modules and requirements classes.**

Req 7 **The requirements structure of the document shall be in a logical correspondence to the test suite structure.**

This structure is described in the following clause. Thus, if two requirements are in the same clause of the body of the document, they should be tested in the same class in the conformance suite. **Each requirement may be separately identifiable either by a label as is done in this document, by its subclause number, or by its subclause number and full text.**

The level of the clauses at which the requirements classes corresponding to the test suite classes are organized is the purview of the editing body of the document, but should be obvious from the language of the clause titles. For example, if UML packages or XML schemas are used to express the factoring of the document into test suite defined conformance classes, the document should use package or schema names in the "table of contents" outline of the standard to indicate the correspondence to test suite classes.

Since these clauses, conformance classes, packages and schemas represent the most important divisions of the test suite and thus of the requirements classes, their name should be included in the table of contents (TOC) of the standard document, and should all be at the same level of heading. For example, in a UML organized standard, where packages are used for test classes in the test suite, the following may be a reasonable clause subdivision for a version of "ISO 19107: Spatial Schema."

- Topic at heading level 1, such as Geometry, Topology.
- Subdivision of related requirements classes at level 2, such as coordinate systems, points, curves, surfaces and solids at level 2.
- Test Suite subdivisions at level 3, such as linear interpolations, conics, Bézier splines, NURBS, etc.
- Classes at level 4, and so forth.

In this case, at least level 3 headings should be in the table of contents. Level 4 and beyond are optional in the TOC.

In summary, the structure of the requirements and requirements classes of the model should be reflected in the organization of the conformance tests and classes, and also in the structure of the normative clauses in the specification document.

NOTE This makes it more difficult to write a standard, but is expected to make the standard easier to read and understand. This trade-off is usually worth it, since the readers of a specification should be orders of magnitude more numerous than its contributing authors.

Making it easier to implement a specification is an important goal of this standard.

6.4 Conformance Test Suite

The requirements here will be applied directly to the test suite, and in particular to the conformance classes. By definition, a "test suite" is a collection of identifiable conformance classes. A conformance class is a well-defined set of conformance tests. Each conformance test is a concrete or abstract (depending on the type of suite) description of a test to be performed on each candidate conformant implementation, to determine if it meets a well-defined set of requirements as stated in the normative clauses of the standards document.

NOTE The Test Suite is normative in the sense that it describes the tests to be performed to pass conformance, but it specifies no requirements in any other sense. The requirements should be specified in the body of the standard. The test suite only describes in detail how those requirements should be tested.

In each of the profiles defined in the Clauses to follow, some set of entities, types, elements or objects are defined and segregated into implementation requirements classes.

Req 8 The requirements classes shall be in a one-to-one correspondence to the conformance test classes, and thus to the various certificate of conformance types possible for a candidate implementation.

Strict parallelism of implementation and governance is the essence of this standard.

6.5 Requirements for Modularity

6.5.1 Each Conformance class tests a complete requirements class

Req 9 A Conformance class shall not contain any optional conformance tests.

It is reasonable to assume that two standardization targets that have the same certificates of conformance could be considered to be functionally equivalent. This is not always the case since many standards have "optional" requirements that are tested or not as selected by the implementors of the target. This requirement stops conformance classes from containing optional requirements and tests, and, at least as far as the specification is concerned, makes all certificates of conformance mean that exactly the same tests have been conducted. Specification may use recommendations for such options, but the conformance test classes do not test recommendations.

A Conformance class may be parameterized. This means that the class's tests depend on some parameter that must be defined before the tests can be executed. For example, if a XYZ conformance class needs to specify a data format such as GML or KML to be tested, then XYZ(GML) is XYZ using GML, and XYZ(KML) is XYZ using KML. **Because the parameters choose which requirements will be tested, two conformance classes with distinct parameters should be considered as distinct conformance classes.**

The most common parameters are the identities of indirect dependencies. For example, if a service uses or produces feature data, the format of that data may be a parameter, such as GML, KML or GeoJSON. When reading a certificate of conformance, the values of such parameters are very important.

Req 10 A certificate of conformance shall specify all parameter values used to pass the tests in its conformance test class.

Conformance to a particular conformance class means exactly the same thing everywhere.

Req 11 A Conformance class shall explicitly test only requirements from a single requirements class.

This means that there is a strict correspondence between the requirements classes and the conformance test classes in the test suite. Recall that a conformance test class may specify

dependencies causing other conformance test classes to be used, but this is a result of an explicit requirement in the "home" requirements class.

Req 12 A Conformance class shall specify any other conformance class upon which it is dependent and that other conformance class shall be used to test the specified dependency.

Such referenced conformance classes may be in the same standard or may be a conformance class of another standard.

Example 6-2 Indirect dependency on schema

If a service specifies that a particular output is required to be conformant to a conformance test class in a specific standard (say a normatively referenced XML schema), then the conformance class of that normative reference will be used to test that output. For example, if a WFS specifies that its feature collection output is compliant to a particular profile of GML, then that profile of GML will be used to validate that output. This means that the service is indirectly tested using the GML standard. In other words, GML is an indirect dependency of the original service.

Requirements classes may be optional as a whole, but not piecemeal. This means that every implementation that passed a particular conformance class satisfies exactly the same requirements and passes exactly the same conformance tests. Differences between implementations will be determined by which conformance test classes are passed, not by listing of which options within a class were tested. If a specification's authors wish to make a particular requirement optional, Req 9 forces them to include it in a separate requirements class (and therefore in a separate conformance test class) which can be left untested.

Other standards do not follow a strict parallelism between requirement specification and testing, so for use within a specification compliant to this standard, special care must be taken in importing conformance test classes from other standards.

Req 13 If a requirements class is imported from another standard for use within a specification conformant to this standard, and if any imported requirement is "optional," then that requirement shall be factored out as a separate requirements class in the profile of that imported standard used in the conformant specification. Each such used requirements class shall be a conformance class of the source standard or a combination of conformance classes of the source standard or standards.

The tracking of the parallelism between requirements and test should be easy if the specification is non-ambiguous. To insure this, the following requirement places a default mapping between the two, by utilizing the names of the two types of classes.

Req 14 For the sake of consistency and readability, all requirements classes and all conformance test classes shall be explicitly named, with corresponding requirements classes and conformance test classes having similar names.

NOTE Logically, a requirements class (set of requirements) and a conformance class (set of tests) are not comparable. This can be remedied by noting that both have a consistent relation to a set of requirements. A requirements class is a set of requirements. A conformance class tests a set of requirements. Therefore we could say that a requirements class corresponds precisely to a conformance class if they both are related (as described) to the same set of requirements.

A particular standard will be simpler the fewer requirements classes it contains. The Einstein criteria of "as simple as possible but no simpler" applies. Another way to say this is that a requirements class should require a reasonable amount of work to achieve. If the requirements class (given all its dependencies have been passed) requires little, then it probably should be merged with another related requirements class, most likely one of its dependencies or a "sibling" requirements class with similar dependencies, purpose and semantics.

6.5.2 Requirements classes contain all requirements tested by a conformance test case

Req 15 Each requirement in the standard shall be contained in one and only one requirements class. Inclusion of any requirement in a requirements class by a conformance class shall imply inclusion of all requirements in its class (as a dependency).

Unless a requirement is referenced in a conformance test and thus in a conformance class, it cannot be considered a requirement since no test has been defined for it. **If possible, the structure of the normative clauses of the standard should parallel the structure of the conformance classes in the conformance clause.**

NOTE This in conjunction with Req 9 means that all requirements in a conformant specification will be tested in some conformance class. In the best example, a requirement should be contained explicitly in one and only one requirements class and tested in one and only one conformance class. This is not really a requirement here, since a single requirement can be stated twice in different requirements classes.

Req 16 If any two requirements or two requirements modules are co-dependent (each dependent on the other) then they shall be in the same requirements class. If any two requirements classes are co-dependent, they shall be merged into a single class.

Normally, circular dependencies between implementation components are signs of a poor design, but they often cannot be avoided because of other considerations (code ownership for example). **Circular dependencies of all types should be avoided whenever possible.**

Req 17 There shall be a natural structure on the requirements classes so that each may be implemented on top of any implementations of its dependencies and independent of its extensions.

NOTE The only certain manner to test this requirement maybe to create a reference implementation.

This requirement is more important and may be more difficult than it seems. It states simply that conformance classes and their associated requirements classes can be put in a one-to-one correspondence to a fully modular implementation of the complete standard (at least all of the specification against a single standardization target). Implementors who wish to sacrifice modularity for some other benefit can still do what they want; the requirement here only states that if the software requirements classes are properly separated, they can be implemented in a "plug'n'play" fashion.

Req 18 No requirements class shall redefine the requirements of its dependencies, unless that redefinition is for an entity derived from but not contained in those dependencies.

This means, for example, that a UML classifier cannot be redefined in a new extension. If a new version of the classifier is needed it has to be a valid subtype of the original.

In terms of generalization; subclassing, extension and restriction (into a new class or type) are all acceptable, redefinition (of an old class or type) is not.

NOTE Clause 6.3 makes some pointed suggestion as to how to organize the conformance classes and normative clauses in parallel to make this requirement easier to verify.

Most standards include examples, which are useful for illustrative or pedagogical purposes. However, it is not possible to write a specification "by example" that leads to conformance tests. Examples are therefore non-normative, by definition.

6.5.3 Profiles are defined as sets of conformance classes

All the conformance classes created in a specification form a base (an upper bound of all conformance classes) for defining profiles as defined in ISO/IEC 10000 (see [2]). The base for creating a profile can be defined as the union of all requirements classes.

Req 19 The conformance tests for a profile of a specification shall be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets.

NOTE This means that a standard conformant to this standard predefines all of its possible profiles through the dependencies between the conformance classes. In essence, a profile of a conformant standard is precisely a transitive closure of a subset of requirements classes under the dependency relations. If a standard has "n" requirements classes and they are all independent, it can have no more than "$2^n - 1$" profiles. If there is a core and "n" independent extensions, then there are no more than "2^n" profiles. Dependencies will usually decrease these numbers radically.

6.5.4 There is a Defined Core

Req 20 Every specification shall define and identify a core set of requirements as a separate conformance class.

Req 21 All general recommendations shall be in the core.

Req 22 Every other requirements class in a specification shall have a standardization target type which is a subtype of that of the core and shall have the core as a direct dependency.

This core may be partially or totally abstract. The core should be as simple as possible. The core requirements class may be a conformance class in another standard, in which case the current specification should identify any optional tests in that conformance class that are required by the current standard's core requirements class. See Req 13.

Since the core requirements class is contained (as a direct dependency) in each other requirements class with a similar standardization target type, the general recommendations are thus universal to

all requirements classes. **Since the basic concept of some standards is mechanism not implementation, the core may contain only recommendations, and include no requirements.**

NOTE In most cases, if someone feels the need to define a "simple" version of the standard, it is probably a good approximation of the best core. For example, the core of a refactored GML might be the equivalent of the "GML for Simple Features" profile. The core for any SQL version of feature geometry is probably "Simple Features."

6.5.5 Extensions are requirements classes

A common mechanism to extend the functionality of a specification is to define extensions, which may be either local or encompass other standards. **Specifications should use extensions where feasible, but should never hinder them.**

Req 23 **Each specification conformant to this standard shall consist of the core and some number of requirements classes defined as extensions to that core.**

Req 24 **A specification conformant to this standard shall require all conformant extensions to itself to be conformant to this standard.**

Since software is evolutionary at its best, it would not be wise to restrict that evolutionary tendency in a specification, by restricting extension specifications. A good specification will thus list the things a standardization target has to do, but will never list things that a standardization target might want to do above and beyond the current design requirements.

Req 25 **A specification conformant to this standard shall never restrict in any manner future, logically-valid extensions of its standardization targets.**

The above requirement should not be interpreted as a restriction on quality control. Any efforts by a specification to enforce a level of quality on its standardization targets, when well and properly formed; do not interfere with the proper extension of those targets. So, the specification may require its standardization targets to behave in a certain manner when presented with a logical inconsistency, but that inconsistency must be fundamental to the internal logic of the model, and not a possible extension. Thus, a specification may require a standardization target to accept GML as a feature specification language, but cannot require a standardization target to not accept an alternative, such as KML, or GeoJSON, as long at that alternative can carry viable information consistent with the fundamental intent of the specification.

6.5.6 Optional requirements are organized as requirements classes

Req 26 **The only optional requirements acceptable in a specification conformant to this standard shall be expressible as a list of conformance classes to be passed.**

NOTE Standards and implementations are restricted by this, but not instances of schemas. For example, a XML schema standard can specify an optional element, which data instances may use or not., However schema-aware processors claiming conformance to the standard should be able to handle all elements defined in the schema, whether they are required by the schema or not.

Requirements of the form "if the implementation does this, it must do it this way" are considered to be options and should be in a separate requirements class.

6.5.7 Requirements classes intersect overlap only by reference

Req 27 The common portion of any two requirements classes shall consist only of references to other requirements classes.

This implies that each requirement is directly in exactly one requirements class and all references to that requirement from another requirements class must include its complete "home" requirements class. This means that requirements for dependencies will often result in conformance test cases which require the execution of the dependency conformance class. See for example Annex A.2.1.

NOTE All general recommendations are in the core requirements class. The core conformance test class contains tests that all other conformance classes must pass.

7 Mapping this standard to types of models

7.1 Semantics

The previous section defined requirements for conformance to this standard, but testing for that conformance may depend on how the various forms and parts of a candidate conformant standard are viewed. This clause will define how those views are to be defined in most of the common cases. Standards that take an alternative mechanism to the ones listed here must be tested solely on the structure of their conformance test suite, until an extension to this standard is defined for that alternate mechanism.

Standards are often structured about some form of modeling language, or implementation paradigm. This clause looks at some common types of models and defines a mechanism to map parts of the model (language, schema, etc.) to the conformance classes used as the specification model from Clause 6.1.

As suggested in Clause Req 22, the structure of the normative clauses in a standard should parallel the structure of the conformance requirements classes in that standard. The structure of the normative clauses in a well written specification will follow the structure of its model; this means that all three are parallel.

7.2 Data Models

7.2.1 Common modeling issues

If a data model is to be used to define the parameters of operational interfaces, then that model should belong in the core since it can be considered as part of a common reference model and vocabulary.

If a data mode is to be used to create "data transfer" elements, the issue is more complex. In the use of parameter names and types in the operational model above, the definition of a common

vocabulary in the core is justifiable. In the case where data transfer elements are being defined, it may be that some types and elements are a defining separator between conformance classes and have to exist independently of such data elements defined for non-dependent classes. **For these reasons, care should be taken in creating separable data transfer schemas across requirements.** Dependencies in the schemas will have to parallel the dependencies in the requirements classes. The mechanism for enforcing this is dependent on the schema language.

7.2.2 Requirements class: UML model extends The Specification

UML is an OMG™ and an ISO standard for expressing object models. If the organizing mechanism for the data model used in the specification is an object model, then the mapping from parts of the model to the requirements classes should follow the logical mechanism described here.

The terminology used here is common to all versions of the UML standard, and may be applied to any such version.

First, by the requirements above, the extension relationship of this conformance test class to the core will be made explicit.

Req 28 An implementation passing the UML conformance test class shall first pass the core conformance test class.

The things in a UML Structural Model are:

1. Packages can be contained in one another. A subpackage of a package A is either A or a packages contained in a subpackage of A. A leaf package is one that contains no subpackages other than itself, but does contain classifiers.

2. Class or more properly "Classifier" (classes, interfaces, types, data types, enumerations, code lists[6]), are contained in a package or (rarely) another classifier. In all cases, they are eventually contained in a package.

3. Attribute of a classifier, consisting of a name and a type for its value (possibly modified with a cardinality), contained in the classifier.

4. Operation, consisting of parameters and a return type, contained in its classifier.

5. Parameters of an operations consisting of a name and a type for its value (just like attributes).

6. Return parameters containing the results of the operation, considered to be part of the operation (just like the other parameters).

7. Associations between classes, consisting of a role name (if it is visible from its containing class) and a target type (possibly with cardinality); if navigable, the role is considered to be in its source class and dependent on its target class; if not navigable,

[6] Code list is an ISO TC 211 extension to UML.

it is neither (a non-navigable role while still part of an association, does not affect class implementations. All association must have at least one navigable role).

8. Constraints, which may be scoped to any of the elements 1-5 above. Constraints may be expressed in text, or using a symbolic notation (usually but not necessarily OCL).

Packages have dependencies upon one another based on usage. Any of the following conditions will produce a direct package dependency from package **A** to package **B**:

a) A classifier in **A** uses in its definition a classifier in **B** by using it in or as:

- an explicit dependency relation
- the type of an attribute
- the target of a navigable association role
- the link class of an association that has one of these navigable roles
- the key class for a navigable association role
- the type of a parameter used in an operation
- the return type of an operation
- a generalization or realization association

b) A subpackage of **A** is dependent on a subpackage of **B**.

All of these are statements about the validity of the dependency graph of a UML model. Not all UML editors enforce the validity of the dependency graphs of classes or packages, so this must be checked separately.

Assuming all legitimate direct package dependencies are included in the UML model, the conformance/requirements class associated to a package **A** will directly reference the conformance/requirements class associated to another package **B**, if and only if **A** is dependent on **B**. Indirect dependencies will be dealt with as the conformance classes cascade.

This clause uses UML terminology, but other object modeling languages and, if needed, the object code itself can be mapped to a UML model. An ERA model can be considered as a partial Object Oriented Programming model, and can easily be recast as UML.

Req 29 **To be conformant to this UML conformance class, UML shall be used to express the object model, either as the core mechanism of the standard or as a normative adjunct to formally explain the standard in a model.**

Req 30 **A UML model shall have an explicit dependency graph for the leaf packages and external packages used by the standard consistent with the way their classifiers use those of other packages.**

NOTE External packages having predated the current version of the standard will not normally have dependencies to packages within the standard.

Other dependencies (indirect) will arise from the transitive closure of the above direct dependencies. That means if **A** depends on **B**, and **B** depends on **C** then **A** depends on **C**. Since these indirect dependencies will show up in the cascade of included conformance tests based solely on the direct dependencies, we can ignore them for effects on structure.

Since a package can consist solely of other packages, there is always the capability to define in UML a single package that will correspond to a particular requirements class and its associated conformance class.

Req 31 A UML leaf package shall be associated directly to only one requirements class.

Req 32 Each requirements class shall be associated to a unique package in the model and include either directly or by a dependency any requirement associated to any of its subpackages.

The class definitions are the "requirements" of a UML expressed standard. So the logical consequence of the above is to organize requirements modules based on leaf packages.

Req 33 A requirements class shall be associated to some number of complete leaf packages and all classes and constraints in those packages.

If a requirement uses or refers to elements of more than one package, then one of the packages will be called the source of the requirement, and the other targets. The requirement with the same source package will always be associated to same requirements module and class.

Req 34 Classes that are common to all requirements classes shall be in a package associated to the core conformance/requirements class.

This is actually a derived requirement and is repeated here for emphasis.

This dependency of requirements classes will be consistent with the usual mechanism for describing the source and target of dependencies between packages. By Clause Req 22, only classes in the source requirements class will be affected by the requirement.

In UML source and target of dependency relation are defined in such a way that the source of the relation is dependent on the target of the relation.

Req 35 In the UML model, if a "source" package is dependent on a "target" package then their requirements class shall be equal or the source package's class shall be an extension of the target package's class.

This means that the dependency graph of the UML packages parallels in some sense the extension diagram of the requirements/conformance classes. If we move all leaf packages of the model into "requirements class packages" containing their corresponding modeling packages we get a model which satisfies the following recommendation: **Each requirements class in a conformant specification should be associated to one and only one UML package (which may contain sub-packages for a finer level of structure). If the core requirements class contains only recommendations, it may be an exception to this.**

Req 36 If one leaf package is dependent on another leaf package, then the requirements class of the first shall be the same or an extension of the requirements class of the second.

Req 37 If two packages have a two-way dependency (a "co-dependency"), they shall be associated to the same requirements class.

For example, if two classes have a two-way navigable association, then these two classes must be (transitively) in the same conformance requirements class package.

The hierarchical structure of a UML model is based on UML classes, residing in UML packages. UML packages can then reside in larger UML packages. Although there is nothing to demand it, it is a common practice to move all classes down the hierarchy to leaf packages. Leaf packages are those that contain only classes (that is, contain no smaller subpackages). Classes can act as packages in the sense that a UML class can contain a locally defined class whose scope is the class itself. For our purposes, we will consider a class and its contained local classes to all be in the package of the original class.

Req 38 The UML model shall segregate all classes into leaf packages.

7.2.3 Requirements class: XML schema extends The Specification

This requirements class covers any specifications which has as one of its purposes the introduction of a new XML schema. Such a specification would normally define the schema, all of its components and its intended uses.

First, by the requirements above, the extension relationship of this conformance test class to the core must be made explicit.

Req 39 An implementation passing the XML schema conformance test class shall first pass the core specification conformance test class.

Req 40 An implementation passing the XML schema conformance test class shall first pass the W3C Recommendation for XML schema.

XML Schema is defined by W3C, see [8] and [9]. Each XML schema file (usually *.xsd) carries a target namespace specification, recorded in the `targetNamespace` attribute of the `<schema>` element in the XML representation. In its implementation, this namespace is represented by a globally unique identifier, a URI. All schema components defined with that URI as its namespace designation are part of the same module in XML schema.

The XML Schema specification lists those resolution strategies for namespace and schema that a schema-aware process may use. They work in a predictable fashion independent of the choice of strategy if and only if the schemas are in a one to one correspondence to their namespace. A schema may be dependent on another schema and may contain "import" directives that load all such schemas whenever it is loaded.

The strategies used by schema-aware applications using XML schema make the same basic assumption, and is not required to reload the contents of a schema document if it has already loaded the namespace in any previous process (note the independence of this with the schema

location also found in many XML schema files). For this reason, there must be a one to one correspondence between the schema and its namespace URI. When a process loads a schema as defined by its namespace URI identity, it must always get a linkage to all components in that namespace. If not, then at sometime in the future, the process will fail when it finds a reference to such a component that it missed. To prevent this sort of failure, when a schema-aware process first encounters a namespace URI it must always be associated to a schema location (a file) that contains or includes all schema components having the URI as their namespace. This is referred to as the "all-components schema document".

So in defining a XML schema (either completely, or partially in a specification) the fundamental component or module of XML schema is always the namespace and its associated schema; which is designated by a URI.

Req 41 If a specification conformant to the XML schema conformance class defines a set of data schemas, all components (e.g. elements, attributes, types …) associated with a single conformance test class shall be scoped to a single XML namespace.

Req 42 The all-components schema document for an XML Schema shall indicate the URI of the associated conformance test class in the schema/annotation/appinfo element.

The mechanism for dependencies may either be by importation or by inclusion of schema components.

Example: In GML 3, the spatial schema (ISO 19107) and the general feature model (ISO 19109) are both satisfied by elements within the single GML namespace. A viable alternative would to have put the schema components for spatial schema and feature schema in separate namespaces.

This is a choice of design, and at the level of this standard, the trade-off factors cannot be prejudged because the details of such cost-benefit trade-offs are not constant. Either of the above approaches may be used.

Req 43 If a specification conformant to the XML schema conformance class defines a direct dependency from one requirement class to another, then a standardization target of the corresponding conformance test class shall import a schema that has passed the associated conformance test class (dependency) or shall itself pass the associated conformance test class.

NOTE This implies that the value of the schemaLocation on the <import> element will refer to the all-components schema document.

An all-components schema document may be assembled by inclusion of documents that describe subsets of the components associated with the conformance test class. This allows schema designers to do some modularization within a namespace for convenience, notwithstanding the requirement for an all-components schema document.

NOTE A namespace variable is used if the requirements class is not defining a schema, but defining requirements for a schema to be the target of its conformance class. For example, GML defines requirements for application schemas, but does not a priori know the namespace of any application schema. The namespace-for the application schema becomes a variable in the conformance test cases.

Req 44 **No requirements class in a specification conformant to the XML schema conformance class shall modify elements, types or any other requirement from a namespace to which it is not associated.**

Requirements may add constraints. This allows extensions to restrict:

(a) Usage of existing elements, but only if their use was originally optional. This is similar to the rules for inheritance (such as in UML or other object models), where a class can eliminate an attribute from a superclass only if the superclass attribute includes a "0" in its multiplicity range.

(b) The type of existing elements, to sub-types of the original elements. This is similar to the rules for inheritance, where a class can re-define an attribute or association role from a superclass so that its type or class is a specialization of the original.

In summary, effective modularization is enabled by following the pattern of one conformance class per XML namespace; i.e. the set of components in an XML namespace should be referred to as a whole. Subsetting of components in a single XML namespace for conformance purposes is not permitted.

7.2.4 Requirements class: Schematron extends XML schema

Schematron (ISO/IEC 19757-3:2006) provides a notation with which many constraints on XML documents can be expressed. This requirements class covers any specification that uses Schematron to create patterns or constrains for an XML Schema. First the schema must be defined within the bounds of the XML schema requirements class.

Req 45 **A specification passing the Schematron conformance test class shall also define or reference an XML schema that shall pass the XML schema conformance class from this standard.**

Within a Schematron schema, the "pattern" and "schema" elements may be used in a way that corresponds with conformance tests and a conformance test class as follows:

Req 46 **Each sch:pattern element shall implement constraints described in no more than one requirement. Each requirement shall be implemented by no more than one sch:pattern.**

Req 47 **Each sch:pattern element shall be contained within one sch:schema element.**

Req 48 **The value of the sch:schema/@fpi attribute shall be a URI that identifies this implementation**

Req 49 **The value of the sch:schema/@see attribute shall be the identifier for the requirements class that contains the requirement(s) implemented by the schema**

Req 50 **The value of the sch:schema/@fpi attribute shall be used on only one Schematron schema.**

7.2.5 Requirements class: XML meta-schema extends The Specification

This requirements class covers any specification which has as one of its purposes the introduction of a new type of XML schema. Such a specification would normally define the characteristics of such schema, how its components are created and its intended uses.

First, by the requirements above, the extension relationship of this conformance test class to the core must be made explicit.

Req 51 A specification passing the XML meta-schema conformance test class shall first pass the core specification conformance test class.

Since the target specification will be defining requirements for XML schemas, it will require that this standard be used.

Req 52 A specification passing the XML meta-schema conformance test class shall require that its specification targets (XML schema) pass the XML schema conformance class from this standard.

Annex A
(normative)
Abstract Conformance Test Suite

A.1 Conformance Test Class: The Specification (Core)

A.1.1 Requirements are atomic and tests cover all the parts of each of the requirement

All the parts of a requirement, a requirement module or requirements class shall be tested. Failure to meet any part of any requirement shall be a failure to pass the associated conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.1; Req 1

d) Test Type: Conformance.

A.1.2 All components have an assigned URI

Each component of the standard, including requirements, requirements modules, requirements classes, conformance test cases, conformance modules and conformance classes shall be assigned a URI as specified by the OGC naming authority or its equivalent.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.1; Req 2

d) Test Type: Conformance.

A.1.3 Requirements on vocabulary are appropriately placed

Requirements on the use and interpretation of vocabulary shall be in the requirements class where that use or interpretation is used.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.1; Req 3

d) Test Type: Conformance.

A.1.4 Requirements have a single target

Each requirement in a conformant specification shall have a single standardization target type.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.2; Req 4

d) Test Type: Conformance.

A.1.5 Conformance test classes have a single target

All conformance tests in a single conformance test class in a conformant specification shall have the same standardization target.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.2; Req 5

d) Test Type: Conformance.

A.1.6 Requirements are organized by clauses

The requirements shall be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements modules and requirements classes.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.3, Req 6

d) Test Type: Conformance.

A.1.7 Conformance test classes are consistent with requirements classes

The requirements structure of the document shall be in a logical correspondence to the test suite structure.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.3, Req 7

d) Test Type: Conformance.

A.1.8 Requirement classes and the Conformance Test classes in correspondence

The requirements classes shall be in a one-to-one correspondence to the conformance test classes, and thus to the various certificate of conformance types possible for a candidate implementation.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.4, Req 8

d) Test Type: Conformance.

A.1.9 No Optional Elements in Requirements classes

A Conformance class shall not contain any optional conformance tests.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 9

d) Test Type: Conformance.

A.1.10 Certificate of conformance specifies all parameters used

A certificate of conformance shall specify all parameter values used to pass the tests in its conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 10

d) Test Type: Conformance.

A.1.11 Conformance class tests only one requirements class

A Conformance class shall explicitly test only requirements from a single requirements class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 11

d) Test Type: Conformance.

A.1.12 Conformance class specifies all dependencies

A Conformance class shall specify any other conformance class upon which it is dependent and that other conformance class shall be used to test the specified dependency.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 12

d) Test Type: Conformance.

A.1.13 Imported Conformance class tests are consistent with the specification

If a requirements class is imported from another standard for use within a specification conformant to this standard, and if any imported requirement is "optional," then that requirement shall be factored out as a separate requirements class in the profile of that imported standard used in the conformant specification. Each such used requirements class shall be a conformance class of the source standard or a combination of conformance classes of the source standard or standards.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 13

d) Test Type: Conformance.

A.1.14 Naming consistency

For the sake of consistency and readability, all requirements classes and all conformance test classes shall be explicitly named, with corresponding requirements classes and conformance test classes having similar names.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.1, Req 14

d) Test Type: Conformance.

A.1.15 Requirements in one and only one requirements class

Each requirement in the standard shall be contained in one and only one requirements class. Inclusion of any requirement in a requirements class by a conformance class shall imply inclusion of all requirements in its class (as a dependency).

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.2, Req 15

d) Test Type: Conformance.

A.1.16 Co-dependent Requirements are in the same requirements class

If any two requirements or two requirements modules are co-dependent (each dependent on the other) then they shall be in the same requirements class. If any two requirements classes are co-dependent, they shall be merged into a single class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.2, Req 16

d) Test Type: Conformance.

A.1.17 Modularity in implementation is possible

There shall be a natural structure on the requirements classes so that each may be implemented on top of any implementations of its dependencies and independent of its extensions.

All general recommendations shall be in the core.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.2, Req 17

d) Test Type: Conformance.

A.1.18 Requirements follow rules of inheritance

No requirements class shall redefine the requirements of its dependencies, unless that redefinition is for an entity derived from but not contained in those dependencies.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.2, Req 18

d) Test Type: Conformance.

A.1.19 Profiles are expressed as sets of conformance classes

The conformance tests for a profile of a specification shall be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.3, Req 19

d) Test Type: Conformance.

A.1.20 There is a named Core requirements class

Every specification shall define and identify a core set of requirements as a separate conformance class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.4, Req 20

d) Test Type: Conformance.

A.1.21 General conditions are in the core

Every other requirements class in a specification shall have a standardization target type which is a subtype of that of the core and shall have the core as a direct dependency.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.4, Req 22

d) Test Type: Conformance.

A.1.22 Every extension has a consistent target type

Every other requirements class in a specification shall have a standardization target type which is a subtype of that of the core and shall have the core as a direct dependency.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.4, Req 22

d) Test Type: Conformance.

A.1.23 Specification is a core plus some number of extensions

Each specification conformant to this standard shall consist of the core and some number of requirements classes defined as extensions to that core.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.5 Req 23

d) Test Type: Conformance.

A.1.24 Conformance to this standard is required for any extensions

A specification conformant to this standard shall require all conformant extensions to itself to be conformant to this standard.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.5 Req 24

d) Test Type: Conformance.

A.1.25 Future extensions cannot be restricted

A specification conformant to this standard shall never restrict in any manner future, logically-valid extensions of its standardization targets.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.5 Req 25

d) Test Type: Conformance.

A.1.26 Optional requirements are organized as requirements classes

The only optional requirements acceptable in a specification conformant to this standard shall be expressible as a list of conformance classes to be passed.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.6, Req 26

d) Test Type: Conformance.

A.1.27 Requirements classes intersect overlap only by reference

The common portion of any two requirements classes shall consist only of references to other requirements classes.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 6.5.7, Req 27

d) Test Type: Conformance.

A.2 Conformance test class: UML model extends The Specification

A.2.1 Dependency on Core

An implementation passing the UML conformance test class shall first pass the core conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 28

d) Test Type: Conformance.

A.2.2 The UML model is normative

To be conformant to this UML conformance class, UML shall be used to express the object model, either as the core mechanism of the standard or as a normative adjunct to formally explain the standard in a model.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 29

d) Test Type: Conformance.

A.2.3 Dependency graph must be explicit

A UML model shall have an explicit dependency graph for the leaf packages and external packages used by the standard consistent with the way their classifiers use those of other packages.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 30

d) Test Type: Conformance.

A.2.4 Leaf packages in only one requirements class

A UML leaf package shall be associated directly to only one requirements class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 31

d) Test Type: Conformance.

A.2.5 Requirements class associated to a unique package

Each requirements class shall be associated to a unique package in the model and include either directly or by a dependency any requirement associated to any of its subpackages.

e) Test Purpose: Verify that this requirement is satisfied.

f) Test Method: Inspect the document to verify the above.

g) Reference: Clause 7.2.2, Req 32

h) Test Type: Conformance.

A.2.6 A requirements class contains complete leaf packages

A requirements class shall be associated to some number of complete leaf packages and all classes and constraints in those packages.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 33

d) Test Type: Conformance.

A.2.7 Classes common to all requirement classes are in the core

Classes that are common to all requirements classes shall be in a package associated to the core conformance/requirements class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 34.

d) Test Type: Conformance.

A.2.8 Package dependencies become requirements class extensions

In the UML model, if a “source” package is dependent on a “target” package then their requirements class shall be equal or the source package’s class shall be an extension of the target package’s class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 35.

d) Test Type: Conformance.

A.2.9 Dependencies in packages are reflected in dependencies in requirements classes

If one leaf package is dependent on another leaf package, then the requirements class of the first shall be the same or an extension of the requirements class of the second.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 36.

d) Test Type: Conformance.

A.2.10 Co-dependent packages are in the same requirements class

If two packages have a two-way dependency (a "co-dependency"), they shall be associated to the same requirements class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 37

d) Test Type: Conformance.

A.2.11 All classes are in leaf packages

The UML model shall segregate all classes into leaf packages.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.2, Req 38

d) Test Type: Conformance.

A.3 Conformance test class: XML schema extends The Specification

A.3.1 Dependency on Core

An implementation passing the XML schema conformance test class shall first pass the core specification conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 39

d) Test Type: Conformance.

A.3.2 Dependency on W3C XML schema

An implementation passing the XML schema conformance test class shall first pass the W3C Recommendation for XML schema.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 40

d) Test Type: Conformance.

A.3.3 A requirements class corresponds to a single XML namespace

If a specification conformant to the XML schema conformance class defines a set of data schemas, all components (e.g. elements, attributes, types …) associated with a single conformance test class shall be scoped to a single XML namespace.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 41

d) Test Type: Conformance.

A.3.4 A reference to the URI of the test suite in AppInfo

The all-components schema document for an XML Schema shall indicate the URI of the associated conformance test class in the schema/annotation/appinfo element.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 42

d) Test Type: Conformance.

A.3.5 Direct dependencies become schema imports

If a specification conformant to the XML schema conformance class defines a direct dependency from one requirement class to another, then a standardization target of the corresponding conformance test class shall import a schema that has passed the associated conformance test class (dependency) or shall itself pass the associated conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 43

d) Test Type: Conformance.

A.3.6 No requirements class modifies or redefines elements in another namespace

No requirements class in a specification conformant to the XML schema conformance class shall modify elements, types or any other requirement from a namespace to which it is not associated.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.3, Req 44

d) Test Type: Conformance.

A.4 Conformance test class: Schematron

A.4.1 Dependency on XML Schema conformance test class

A specification passing the Schematron conformance test class shall also define or reference an XML schema that shall pass the XML schema conformance class from this standard.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 45

d) Test Type: Conformance.

A.4.2 Each schematron pattern element implements one requirement

Each sch:pattern element shall implement constraints described in no more than one requirement. Each requirement shall be implemented by no more than one sch:pattern.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 46

d) Test Type: Conformance.

A.4.3 Each schematron pattern is in one schematron element

Each sch:pattern element shall be contained within one sch:schema element.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 47

d) Test Type: Conformance.

A.4.4 Each schematron @fpi attribute is used only once

The value of the sch:schema/@fpi attribute shall be used on only one Schematron schema.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 48

d) Test Type: Conformance.

A.4.5 Each schematron @see attribute is used only once

The value of the sch:schema/@see attribute shall be the identifier for the requirements class that contains the requirement(s) implemented by the schema

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 49

d) Test Type: Conformance.

A.4.6 Each schematron fpi attribute is used only once

The value of the sch:schema/@fpi attribute shall be used on only one Schematron schema.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.4, Req 50

d) Test Type: Conformance.

A.5 Conformance Class: XML meta-schema

A.5.1 Supports the Specification class

A specification passing the XML meta-schema conformance test class shall first pass the core specification conformance test class.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.5, Req 51

d) Test Type: Conformance.

A.5.2 Each XML schema is conformant with the XML Schema class

A specification passing the XML meta-schema conformance test class shall require that its specification targets (XML schema) pass the XML schema conformance class from this standard.

a) Test Purpose: Verify that this requirement is satisfied.

b) Test Method: Inspect the document to verify the above.

c) Reference: Clause 7.2.5, Req 52

d) Test Type: Conformance.

Annex B
(normative)
Changes required in the OGC Standards;
Transition plan for incorporation into the OGC procedures

B.1 Existing standards

This standard does not apply to existing standards or to those adopted before this standard's adoption.

B.2 New standards and specifications; during the first year

Any new standard, specification or major revision of an existing standard shall comply with this standard in the structure of its internal models and its conformance tests.

Failure to conform by a candidate standard to this standard should be specifically noted and reasons given for such non-compliance in the conformance clauses of any new or new version of such candidate standards.

The adoption of such documents not compliant with this standard shall be considered as an authorized exception to the requirements of this standard by the OGC. This "exception by majority vote" will be a valid approach only before the 1-year anniversary of the adoption of this standard.

B.3 New standards and specifications; after the first year

After the first anniversary of the adoption of this standard, the adoption of such documents not compliant with this standard shall be considered as an exception to the requirements of this standard by the OGC and shall be considered as an exception to the rules of the OGC and will require a two-thirds ($^2/_3$) majority ("Robert's Rules") or as specified in the current OGC Policy and Procedures for an exception to procedure. A similar vote is required within the Planning Committee or as specified in any Policy and Procedure document used by this committee.

This vote is a vote to override the rules of the consortium, and not a vote on the merits of the proposed specification which has failed to conform to this standard. Voting to override the rules carries no intellectual property implications in addition to that consistent with a non-participating member role for the proposed specification as provided in the TC Policy and Procedures.

Annex C
(informative)
Definitions

C.1 Semantically ordered definitions

Clause 4 formally defines the terms used in the conformance tests in alphabetical order. It may be easier to understand the more significant terms in the following less formal definitions arranged in a bottom-up order:

1. a **standardization target type** is a type of entity about which a standard is written. An instance of a **standardization target type** is a **standardization target.** A standard may address multiple targets in separate **conformance classes**.

2. a **requirement** is a statement of a condition to be satisfied by a single **standardization target type,** and it must be stated in "normative" language.

3. a **conformance test** checks if a set of **requirements** are met (**pass**) or not met (**fail**) by a **standardization target**. The relationship between **conformance tests** and **requirements** is many-to-many.

4. all **conformance tests** are graded as **pass** or **fail** against each instance of the **standardization target.**

5. a **requirement** is associated to at least one **conformance test**.

6. a **recommendation** is a suggestion and is not associated to any **conformance test**.

7. a **requirements class** is a set of one or more **requirements** all on the same **standardization target type**.

8. a **conformance (test) class** is a collection of **conformance tests** that are associated to and only to the requirements in a corresponding **requirements class**.

9. a **conformance (test) class** is also collection of **conformance test modules** that group **conformance tests** on a single **standardization target type.**

10. a **conformant implementation** is a **standardization** t**arget type** that has successfully passed all tests in a specified **conformance (test) class** and received a **certificate of conformance**

11. the **core requirements class** of a standard is the minimal set of **requirements** which must be supported by all **conformant implementations**. If a specification addresses multiple **standardization target types**, it may have a **core** for each **target type**.

12. an **extension** of a **requirements class** is a second **requirements class** (the extension) that adds additional **requirements** to the first **requirements class** (the

base requirements class being extended). The extension is said to be dependent on the **base**. Any **conformance test class** must identify all of its dependencies during the execution of conformance tests against a candidate **standardization target**.

C.2 UML Model

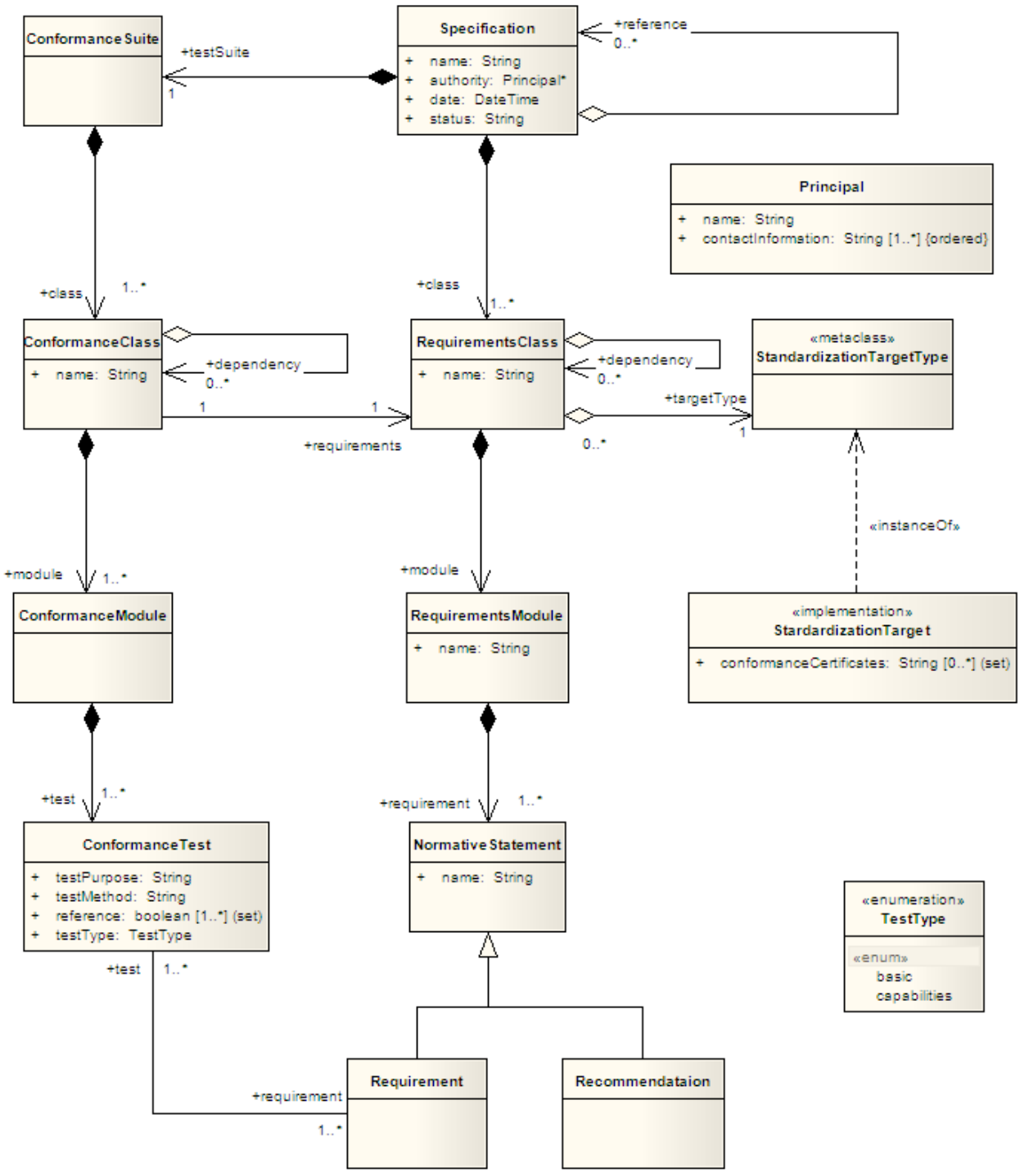

Figure C.1: Specification structure

Figure C.1 represents a UML model consistent with the specification model described in Clause 6.1. The following subclauses describe the classes shown in this UML class diagram.

C.3 Specification

A specification, to become a standard must be approved by an authority, such as ISO or OGC. This gives the specification a namespace corresponding to the standards authority. Thus the name of the specification is usually a local name in this namespace: for example "ISO 19107: Geographic Information – Spatial schema" is an approved specification of ISO TC 211 (the authority as a TC under ISO). The parts of the name that give this are "ISO" and "Geographic Information". The local name within this namespace is "19107: Spatial schema". This syntax is not usual for namespaces where the namespace name is usually either a prefix or suffix of the local name. GenericName, LocalName, Namespace are all types described in "ISO 19103: Geographic information – Conceptual schema language". The attributes of a Specification describe its local name, its authority, the date of publication and its current status (such as CD, DIS, IS in ISO, or Draft, Candidate Standard or Standard in OGC).

```
Class Specification
   {
   name       : String               // name for the standards
   authority  : Principal            // Standards Body & Namespace for name
   date       : DateTime             // publication date
   status     : String               // description of current status
   reference  : Specification[0..*]  // other standards used normatively
   testSuite  : ConformanceSuite     // all tests associated to the standard
   }
```

C.4 Conformance Suite

The unique conformance suite of a specification lists the tests (grouped into conformance test classes consisting of some number of conformance test modules, containing some number of conformance tests) that allow testing of an implementation of the specification for conformance with the specification. Every specification needs one of these suites, or conformance cannot be claimed with proof. In ISO and OGC, the conformance suite included in the specification is usually an abstract description of the tests which will be implemented. Other standards may use a more concrete description. For the purposes of this OGC standard, the precise nature of the conformance suite is not particularly important as long as it is not ambiguously stated.

Each conformance test within a conformance class should be against a single standardization target defined for that class. A conformance suite may contain several defined conformances classes for the same standardization target.

```
Class ConformanceSuite
   {
   class  : ConformanceClass[1…*]
   }
```

C.5 Conformance Class

The requirements in the requirements classes of a specification have to be tested and the conformance classes are the containers for these tests' definition. The requirements classes will have interdependencies, and this is reflected in the explicit dependencies between the conformance classes. If class "a" is dependent on class "b", then to pass the test for "a" a

standardization target must also pass the test for "b." The class name is shared with its corresponding requirements class.

```
Class ConformanceClass
   {
   name         : String                  // name of the conformance class
   dependency   : ConformanceClass[0…*]
                // other classes that must be passed if this one is
                // to be passed (consistent with Requirements class)
   requirement  : RequirementClass
   }
```

C.6 Requirements class

The specification requirements classes (usually realized as clauses in the specification document) segment the requirements in the specification in a manner consistent with the conformance classes. Since the requirements class and the conformance class will eventually be referred to in a certification of conformance, they should have names, probably in the namespace defined by the specification's name and authority.

```
Class RequirementsClass
   {
   name        : String                      // name of the conformance class
   dependency  : RequirementsClass [0…*]     // dependent requirements classes
   module      : RequirementsModule[1…*]     // subsets of this requirementClass
   targetType  : StandardizationTargetType
   }
```

C.7 Requirements module

The specification requirements modules (usually realized as subclauses of the requirements class in the specification document) segment the requirements in the specification in a manner consistent with the conformance test modules.

```
Class RequirementsModule
   {
   name         : String
   requirement  : Requirement[1…*]
   }
```

C.8 Normative Statement

The normative statements, either requirements or recommendations of a standard, are organized into the requirements modules and classes, and may be tested by the conformance tests in their requirements class's corresponding conformance class. If tested, the statement is a "Requirement", and if not tested the statement is a "Recommendation."

```
Class NormativeStatement
   {
   name  : String                // name the requirement
   test  : ConformanceTest[1…*]  // corresponding conformance test
   }
```

C.9 Requirement

Each normative statement which is tested by any conformance test is a requirement.

```
Class Requirement inherits NormativeStatement
   {
```



```
   test  : ConformanceTest[1…*]  // corresponding conformance test
   }
```

C.10 Recommendation

A normative suggestion which will not be directly tested is a "Recommendation." Recommendations have a variety of uses, for example:

1. Legal restriction, such as "not for commercial use" or "for planning purposes." These allow the specification to restrict use of its implementation to standardization targets for which it was designed.

2. Statement of best practices. These are included as suggestions for logical designs that may implement the requirements in the same module.

Regardless of their use, Recommendations are not tested since they are not required of all conformant implementations.

```
Class Recommendation inherits NormativeStatement
```

C.11 Conformance test

The requirements in a specification have to be tested and the conformance test specification contains the test's definition. In this standard, conditional tests, based on a requirement with some precondition, have as part of their execution the task of creating the precondition in which they will be tested. If the condition is not creatable, then the test is not required to be executed.

```
Enumeration TestType
   {
   basic
   capabilities
   }
Class ConformanceTest
   {
   testPupose   : String
   testMethod   : String
   Reference    : String
   testType     : TestType
   requirement  : Requirement[1…*]
   }
```

C.12 StandardizationTarget

Each conformance class (and hence requirements class) is targeted to a particular type of implementation. An implementation testable by a conformance class is a StandardizationTarget of that class, and (once the appropriate test have been passed) can carry a certificate indicating its conformance to a requirements class proved by the tests in the conformance class.

```
Class StandardizationTarget
   {
   conformanceCertificates  : String[0…*]    // conformance classes passed
                                             // by this target
   type                     : StandardizationTargetType
   }
```


C.13 StandardizationTargetType

A Standardization Target Type is the type or collection of standardization targets testable by the conformance class corresponding to a requirements class. This is a «metaclass» in the sense that it defines a type of implementation with members of the type testable by a particular conformance class. This is a “multiple classification” scheme, since each target may be a member of multiple StandardizationTargetTypes.

```
Abstract Metaclass StandardizationTargetType
   // description of the standardization targets in this type
```


Annex D
Bibliography

To preserve a unique numeric identifier for all documents listed as references in this standard, the numbering of references in this annex is continued from the list of normative reference in Clause 3 on page 2.

[10] Object Management Group (OMG), February 2007, Unified Modeling Language: Superstructure, version 2.1.1 , formal/07-02-05, available from OMG.org at http://www.omg.org/cgi-bin/doc?formal/07-02-05

[11] Object Management Group (OMG), February 2007, Unified Modeling Language: Infrastructure , version 2.1.1 , formal/07-02-06, available from OMG.org at http://www.omg.org/cgi-bin/doc?formal/07-02-06

[12] ISO/IEC JTC 1, ISO/IEC 9075:2003 -- Information Technology -- Database Languages – SQL.

[13] ISO/IEC JTC 1, ISO/IEC 13249-3:2006 -- Information technology -- Database languages -- SQL multimedia and application packages -- Part 3: Spatial

[14] W3C, XML Schema 1.1 Part 1: Structures, 31 August 2006, available from W3C at http://www.w3.org/TR/xmlschema11-1/

[15] W3C, XML Schema 1.1 Part 2: Datatypes, 17 February 2006, available from W3C at http://www.w3.org/TR/xmlschema11-2/

[16] ISO/IEC TR 10000: Information Technology — Framework and taxonomy of International Standardized Profiles

Bibliography for examples

The following documents are either standards or draft standards that in general follow the general rules of ISO for conformance test suites. The first two (GeoREL and the OWS5 discussion of WFS) meet most of the requirements of this standard.

[17] ISO 19149: Geographic information - Rights expression language for geographic information – GeoREL

[18] OGC 08-079 OWS5: OGC Web feature service, core and extensions, an OGC discussion paper

[19] ISO 19107: Geographic information — Spatial schema

[20] ISO 19111: Geographic information — Spatial referencing by coordinates

[21] ISO 19119: Geographic information — Services

[22] ISO 19125: Geographic information — Simple features

[23] ISO 19133: Geographic information — Location-based services — Tracking and navigation

[24] ISO 19136: Geographic information — Geography Markup Language (GML)

[25] ISO 19141: Geographic information — Schema for moving features

[26] ISO 19142: Geographic information — Web feature service (WFS)

[27] ISO 19143: Geographic information — Filter Encoding

[28] ISO 19148: Geographic information — Location-based services — Linear referencing systems

[29] ISO 19149: Geographic information — Rights expression language for geographic information — GeoREL

[30] ISO 19153: Geographic information — Geospatial Digital Rights Management (GeoDRM RM)

[31] ISO 19156: Geographic information — Observation and measurements

