OGC® DOCUMENT: 08-131R3

Open Geospatial Consortium

THE MODSPEC MODEL - A STANDARD FOR DESIGNING AND WRITING MODULAR STANDARDS

DRAFT STANDARD

DRAFT

Version: 1.1.0
Submission Date: 2024-12-07
Approval Date: 2029-11-01
Publication Date: 2029-11-01
Editor: Carl Reed, John Herring, Chuck Heazel

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I PREFACE

This OGC member developed and approved document defines a model and related requirements and recommendations for writing and structuring modular standards documents. Further, this model is designed to enable the consistent and verifiable testing of implementations of a standard that claim conformance.

The goal is to ensure that a standard specifies requirements in a common and consistent manner and that these requirements are testable.

NOTE 1: For OGC only: Any new OGC Standard, abstract specification that contains requirements, or major revision of an existing OGC Standard shall comply with the requirements stated in this document (see Requirement 0).

NOTE 2: Historically, this document has been known and abbreviated as the “ModSpec”. For continuity and ease of understanding this document may also be refered to as the “OGC ModSpec”.

Suggested additions, changes, and comments on this this document are welcome and encouraged. Such suggestions may be submitted through the OGC Change Request System (http://www.opengeospatial.org/standards/cr) or by creating an issue in the GitHub repository for this document (https://github.com/opengeospatial/ogc-modspec).

II SECURITY CONSIDERATIONS

No security considerations have been made for this document.

III SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Carl Reed, Charles Heazel, ImageMatters

IV 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.

V DOCUMENT EDITORS

The following OGC Members participated in editing this document:

PERSON	ORGANIZATION REPRESENTED
Carl Reed	Carl Reed & Associates
Chuck Heazel	Charles Heazel inc.

VI DOCUMENT CONTRIBUTORS

The following OGC Members contributed and particpated in developing Version 2 of the ModSpec.

PERSON	ORGANIZATION REPRESENTED
Simon Cox	CSIRO and OGC Fellow
Chuck Heazel	Charles Heazel inc.
Clemens Portele	interactive instruments GmbH
Jeff Yutzler	ImageMatters

VII REVISION HISTORY

This is the second normative version of this document. It includes updates and refinements necessary to accomodate changes in the Standards landscape.

VIII FUTURE WORK

Improvements to this document will be made based on implementation and changing technical requirements. Planned extensions include:

- ModSpec Part 2 providing requirements and recommendations for specifying requirements and conformance tests using JSON.
- ModSpec Part 3 providing requirements and recommendations for specifying requirements and conformance tests using RDFS, SHACL, and OWL.
- ModSpec Part 4 providing requirements and recommendations for specifying requirements and conformance tests using XML and Schematron.

IX FOREWORD

This OGC document (aka the ModSpec) specifies a formal structure for standards documents but does not supply specific content. Where possible, this document is conformant with itself (with respect to the core conformance test class, Clause 9 and the Conformance Test Suite Annex A.1).

X ACKNOWLEDGEMENTS

The following OGC Members were key contributors to Version 1 of the ModSpec

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

1

SCOPE

1 SCOPE

The ModSpec defines characteristics and structure for the specification of Standards that will encourage implementation by minimizing difficulty determining requirements, mimicking implementation structure and maximizing usability and interoperability.

NOTE: For OGC Standards work, the word “standard” in this document applies to all OGC draft standards, approved standards, draft Abstract Specifications, and approved Abstract Specifications. The exceptions are OGC Abstract Specifications that originate in ISO or Community Standards that are developed external to the OGC and then submitted to the OGC.

[Annex-B] defines the UML model upon which the ModSpec is based. Annex B also 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

2 CONFORMANCE

Conformance to the ModSpec by technical implementation standards can be tested by inspection. The test suite is in Annex A.

There are five (5) conformance classes for this document:

1. Standards documents in general (the core) — see Clause 9 and Annex A.1
2. Standards using UML to state requirements, extending the core — see Clause 10.2.2 and Annex A.2
3. Standards using XML schema to state requirements, extending the core — see Clause 10.2.3 and Annex A.3
4. Standards using Schematron to state requirements, extending XML schema — see Clause 10.2.4 and Annex A.4
5. Standards 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 10.2.5 and Annex A.5.

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

- Clause 9 specifies the core requirements which shall be met by all conformant standards.
- Clause 10 gives information on how the ModSpec is to be applied to requirements, conformance clauses, UML models, or XML Schemas.

The various subclauses of Clause 10 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 9.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC: ISO/IEC 10000-1, ISO, IEC

ISO/IEC DIR 2, *ISO/IEC Directives, Part 2*. https://www.iso.org/sites/directives/current/part2/index.xhtml.

ISO: ISO 19105, *Geographic information — Conformance and testing*. International Organization for Standardization, Geneva https://www.iso.org/standard/76457.html.

ISO/IEC: ISO/IEC 19501, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*. International Organization for Standardization, International Electrotechnical Commission, Geneva https://www.iso.org/standard/32620.html.

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

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

ISO/IEC: ISO/IEC 19757-3:2006, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2006). https://www.iso.org/standard/40833.html.

W3C: W3C xmlschema-1, *XML Schema Part 1: Structures Second Edition*. World Wide Web Consortium https://www.w3.org/TR/xmlschema-1/.

W3C: W3C xmlschema-2, *XML Schema Part 2: Datatypes Second Edition*. World Wide Web Consortium https://www.w3.org/TR/xmlschema-2/.

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

For the purposes of this document, the following terms and definitions shall apply. Terms not defined here take 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 [cls-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. building block

a requirements class or a requirements module with no direct dependencies on other requirements classes or modules and their associated compliance test class or compliance test module.

4.3. certificate of conformance

evidence of conformance to all or part of a standard, awarded for passing one or more of the *conformance test classes* (Clause 4.7) specified in that standard

Note 1 to entry: "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 https://www.ogc.org/resources/certified-products/.

Each certificate of conformance is awarded to a *standardization target* (Clause 4.26).

4.4. conformance test

test, abstract or real, of a single *requirements* (Clause 4.21) contained within a standard, or set of standards

4.5. conformance test case

test for a particular requirement or a set of related requirements

Note 1 to entry: When no ambiguity, the word “case” may be omitted. i.e. *conformance test* (Clause 4.4) is the same as *conformance test case* (Clause 4.5).

4.6. conformance test module

set of related tests for a given requirements module all with the same standardization target

Note 1 to entry: When there is no ambiguity, the word “test” may be omitted. i.e. *conformance test module* (Clause 4.6) is the same as **conformance module**. Conformance modules may be nested in a hierarchical way.

4.7. conformance test class

conformance test level ALTERNATIVE

set of **term `conformance tests`, display `conformance test` not resolved via ID `conformance-tests`** that must be passed to receive a single *certificate of conformance* (Clause 4.3)

Note 1 to entry: When no ambiguity is possible, the word “test” may be left out, so *conformance test class* (Clause 4.7) maybe called a *conformance class*.

In the ModSpec, the set of *requirements* (Clause 4.21) tested by the conformance tests within a *conformance class* is a *requirements class* (Clause 4.22) and its dependencies. Optional *requirements* (Clause 4.21) will be in a separate *requirements class* (Clause 4.22) with other *requirements* (Clause 4.21) that are part of the same option. Each *requirements class* (Clause 4.22) corresponds to a separate conformance class

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* (Clause 4.22) *requirements* (Clause 4.21).

All *requirements* (Clause 4.21) in a *conformance class* will have the same *standardization target* (Clause 4.26).

4.8. conformance suite

conformance test suite ALTERNATIVE

abstract test suite ALTERNATIVE

set of *conformance classes* that define tests for all *requirements* (Clause 4.21) of a standard or abstract specification

Note 1 to entry: The *conformance suite* (Clause 4.8) is the union of all *conformance classes*. It is by definition the *conformance class* of the entire standard or abstract specification.

In this Policy, each *requirement* (Clause 4.21) is mandatory within its conformance class and each *requirement* (Clause 4.21) is tested in at least one *conformance test* (Clause 4.4).

4.9. core requirements class

unique *requirements class* (Clause 4.22) that must be satisfied by any conformant *standardization targets* (Clause 4.26) associated to the standard

Note 1 to entry: The core *requirements class* (Clause 4.22) is unique because if it were possible to have more than one, then each **core** would have to be implemented to pass any *conformance test class* (Clause 4.7), 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* (Clause 4.22), its associated *conformance test class* (Clause 4.7) or the software module that implements that requirements class.

4.10. direct dependency (of a requirements class)

another *requirements class* (Clause 4.22) (the dependency) whose *requirements* (Clause 4.21) are defined to also be *requirements* (Clause 4.21) of this *requirements class* (Clause 4.22)

Note 1 to entry: A *direct dependency* (Clause 4.10) of the current *requirements class* (Clause 4.22) will have the same *standardization target* (Clause 4.26) as the current *requirements class* (Clause 4.22). This is another ways of saying that the current *requirements class* (Clause 4.22) extends, or uses all the aspects of the *direct dependency* (Clause 4.10). Any tests associated with this *dependency* (Clause 4.10) can be applied to this *requirements class* (Clause 4.22).

When testing a *direct dependency* (Clause 4.10), the *standardization target* (Clause 4.26) is directly subject to the test in the specified *conformance test class* (Clause 4.7) of the *direct dependency* (Clause 4.10).

4.11. indirect dependency (of a requirements class)

requirements class (Clause 4.22) with a different *standardization target* (Clause 4.26) which is used, produced or associated to by the implementation of this *requirements class* (Clause 4.22)

Note 1 to entry: In this instance, as opposed to the *direct dependency* (Clause 4.10), the test against the consumable or product used or produced by the *requirements class* (Clause 4.22) does not directly test the *requirements class* (Clause 4.22), 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* (Clause 4.10) test the same *standardization target* (Clause 4.26), but *indirect dependencies* (Clause 4.11) test related but different *standardization targets* (Clause 4.26).

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* (Clause 4.3) of a particular kind.

4.12. extension (of a requirements class)

requirements class (Clause 4.22) which has a *direct dependency* (Clause 4.10) on another *requirements class* (Clause 4.22)

Note 1 to entry: Here *extension* (Clause 4.12) is defined on *requirements class* (Clause 4.22) so that their implementation may be software extensions in a manner analogous to the extension relation between the *requirements classes* (Clause 4.22).

4.13. general recommendation

recommendation applying to all entities in a standard

4.14. home (of a requirement or recommendation)

official statement of a *requirement* (Clause 4.21) or *recommendation* (Clause 4.20) that is the precedent for any other version repeated or rephrased elsewhere in a standard

Note 1 to entry: Explanatory text associated with 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.

4.15. model

abstract model ALTERNATIVE

conceptual model ALTERNATIVE

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

4.16. module

one of a set of separate parts that can be joined together to form a larger object

4.17. optional requirements class

An optional requirements class may or may not be implemented or specified in a profile or extension. However, if a profile, extension, or implementation specifies the use of an optional requirements class, then every requirement in that requirements class *shall* be implemented.

4.18. **permission**

uses “may” and is 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.

4.19. **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* (Clause 4.7), conforming subsets, options and parameters of those base standards, or profiles necessary to accomplish a particular function.

Note 1 to entry: In the usage of this Policy, 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* (Clause 4.26) being conformant to a profile implies that the same **target** is conformant to the standards referenced in the *profile* (Clause 4.19).

[**SOURCE:** ISO/IEC 10000-1]

4.20. **recommendation**

expression in the content of a standard 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

Note 1 to entry: Although using normative language, a *recommendation* (Clause 4.20) is not a *requirement* (Clause 4.21). The usual form replaces the “shall” (imperative or command) of a *requirement* (Clause 4.21) with a “should” (suggestive or conditional).

Note 2 to entry: Recommendations are **not** tested and therefor have no related conformance test.

[**SOURCE:** ISO/IEC DIR 2]

4.21. requirement

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

Note 1 to entry: Each *requirement* (Clause 4.21) is a normative criterion for a single **type of standardization target**. In the ModSpec, requirements are associated to *conformance tests* (Clause 4.4) that can be used to prove compliance to the underlying criteria by the *standardization target* (Clause 4.26).

The implementation of a *requirement* (Clause 4.21) is dependent on the type of standard being written. A data standard requires data structures, but a procedural standard requires software implementations. The view of a standard in terms of a set of testable *requirements* (Clause 4.21) allows us to use set descriptions of both the standard and its implementations.

Requirements (Clause 4.21) use normative language and 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.

[**SOURCE:** ISO/IEC DIR 2]

4.22. requirements class

aggregate of all **term `requirements`, display `requirement` not resolved via ID `requirements`** with a single standrdization target that must all be satisfied to pass a *conformance test class* (Clause 4.7)

Note 1 to entry: 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).

4.23. requirements module

collection of **term `requirement class`, display `requirements classes` not resolved via ID `requirement-class`**, *recommendations* (Clause 4.20) and *permissions* (Clause 4.18) with a single *standardization target* (Clause 4.26)

4.24. specification

document containing *recommendations* (Clause 4.20), *requirements* (Clause 4.21) and *conformance tests* (Clause 4.4) for those *requirements* (Clause 4.21)

Note 1 to entry: This definition is included for completeness. See Clause 7.4.

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

4.25. standard

document that has been approved by a legitimate Standards Body

Note 1 to entry: This definition is included for completeness. *Standard* (Clause 4.25) and *specification* (Clause 4.24) can apply to the same document. While *specification* (Clause 4.24) is always valid, *standard* (Clause 4.25) only applies after the adoption of the document by a legitimate standards organization.

4.26. standardization target

entity to which some *requirements* (Clause 4.21) of a *standard* (Clause 4.25) apply

Note 1 to entry: The *standardization target* (Clause 4.26) is the entity which may receive a *certificate of conformance* (Clause 4.3) for a *requirements class* (Clause 4.22).

Note 2 to entry: Need to add examples! The standardization target of the CDB version 2.0 CRS Requirements Classes is to ensure that an implementation clearly defines (with metadata) the CRS for a CDB compliant datastore.

4.27. standardization target type

type of entity or set of entities to which the *requirements* (Clause 4.21) of a *standard* (Clause 4.25) apply

Note 1 to entry: For example, the standardization target type for The OGC API – Features Standard are Web APIs. The standardization target type for the CDB Standard is "datastore". It is important to understand that a standard's root standardization target type and can have sub-types and that there can be a hierarchy of target types. For example, a Web API can have sub types of client, server, security, and so forth. As such, each requirements class can have a standardization target type that is a sub-type of the root.

4.28. **statement**

expression in a document conveying information

Note 1 to entry: Includes all statements in a document not part of the normative *requirements* (Clause 4.21), *recommendations* (Clause 4.20) or *conformance tests* (Clause 4.4). Included for completeness.

[**SOURCE:** ISO/IEC DIR 2]

6

INTRODUCTION

6 INTRODUCTION

NOTE 1: Reading the Terms and Definitions clause and Clause <TBD> will help understanding the content and requirements stated in this document.

This OGC document, also known as the `ModSpec`:

- Specifies rules for the internal structure and organization of a standard.
- Defines requirements for specifying the structure of a standards document as organized sets of criteria, those that are to be tested (“requirements”) and those that are not tested (“recommendations” and “permissions”).
- Designed to enable the clear and concise specification of requirements (the *shalls* or *musts* in a standard) that fully supports the ability to define implementable conformance tests.

The goal of this approach is to enable implementations of a standard to be tested and deemed *conformant* or not.

NOTE 2: Please note that the ModSpec has been approved by the OGC Membership as a policy directive for the development and revision of any OGC Standard or Abstract Specification that has requirements. However, the ModSpec is written to be non-OGC specific and can be used by any Standards Development Organization (SDO) as a formal guide for structuring a standards document.

A standard that follows the rules specified in the ModSpec presents requirements organized in requirements classes which must be satisfied by passing the tests defined in a conformance suite (also known as the Abstract Test Suite in an OGC Standard). These tests are organized into conformance classes, each of which represents a mechanism for partial satisfaction of the standard. This results in a standard having a modular structure, where each requirements class has a corresponding conformance (test) class. In a well written standard, the normative clauses and any model or schema are organized in a manner that parallels the requirements and conformance clauses. A goal of the design pattern is the ability to define requirements classes and associated conformance classes that can be used across multiple standards.

There are numerous examples of requirements/conformance classes that can be used not only in OGC Standards, but for geospatially focused standards defined by other organizations and Standards Development Organizations (SDOs). Some OGC examples can be found in the OGC API — Common Part 1: Core Standard and in the CDB 2.0 Standard CRS Requirements Module. By formally implementing the requirements specified in the ModSpec, reusable, modular standards can be developed.

NOTE 3: The approach modelled in the ModSpec has been referred to as the “core and extension model” due to its insistence on a modular structure throughout all parts of a standard and its implementation.

6.1. ModSpec document structure

Version 2.0 of the ModSpec will be split into a Core standard and multiple Parts. This document includes the following parts:

- Core: contains all the core requirements and informational text that define the model and internal structure of a standard.
- Part 1: UML Model requirements

Future Parts to the ModSpec Standard will include:

- Part 2: Specifying requirements and conformance tests using JSON.
- Part 3: Specifying requirements and conformance tests using RDFS, SHACL, and OWL
- Part 4: Specifying requirements and conformance tests using XML and Schematron.

7

CONVENTIONS

7 CONVENTIONS

7.1. 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.

7.2. Identifiers

The normative provisions in this standard are denoted by the URI namespace

https://www.opengis.net/spec/modspec/1.1/

All requirements that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of "req" in a requirement URI denotes:

https://www.opengis.net/spec/modspec/1.1/

An example might be:

/req/core/crs

All conformance tests that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of "conf" in a requirement URI denotes:

https://www.opengis.net/spec/modspec/1.1/

The same convenstion is used for permissions (per) and recommendations (rec).

7.3. Abbreviated terms

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”)
OCL	Object Constraint Language (part of UML)
OGC	Open Geospatial Consortium (http://www.opengeospatial.org/)
OMG	Object Management Group (http://www.omg.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

7.4. Finding requirements and recommendations

Each normative statement in the ModSpec is stated in one and only one place, in a standard format, with an unique label, such as REQ001, REC001, or PER001. A requirement, recommendation, or permission may be repeated for clarification. The statement with the unique label is considered the official statement of the normative requirement or recommendation.

In this document, all requirements are associated with tests specified in the test suite in Annex A. The reference to the requirement in the test case is done by a requirements label Recommendations are not tested although they still are documented using a standard template and have unique identifiers.

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 link between requirements classes and conformance test classes.

8

STANDARDS FUNDAMENTALS

8 STANDARDS FUNDAMENTALS

8.1. Building Blocks

In software development technology, there is a concept called *building block*. In software development, building blocks are used to support the software build process where source code files/libraries can be accessed from multiple sources, converted into executable code, and linked together in the proper order until a complete set of executable files is generated. The same concept can be applied to OGC Standards development: Requirements classes and/or modules can be linked together from one or more standards to create a new standard not originally envisioned when the requirements were originally defined.

The Open Group suggests that building blocks have the following characteristics:

1. A building block is a package of functionality defined to meet business or domain needs.
2. A building block may interoperate with other, inter-dependent, building blocks.
3. A good building block has the following characteristics:
   a) Considers implementation and usage, and evolves to exploit technology and standards.
   b) May be assembled from other building blocks.
   c) May be a subassembly of other building blocks.
   d) Ideally a building block is re-usable and replaceable, and well specified.
4. A building block may have multiple implementations but with different inter-dependent building blocks.

These characteristics are slightly modified from the Open Group definitions to accommodate the use of the building block concept in standards work.

8.2. Standardization Context — Goals and Targets

NOTE: Don't forget to add the requirements cited.

Every OGC Standard document shall include a Standardization Goal (see requirement TBD). This is a concise statement of the problem that the Standard helps address and the strategy envisioned for achieving a solution. This strategy typically identifies real-world entities that need to be modified or constrained. At the abstract level, those entities are the Standardization Target Types. These are the classes of entities to be standardized. A Standard defines the requirements levied on one or more Standardization Target Types.

Instances of a Standardization Target Type are the Standardization Targets. These are the real-world manifestations of the Standardization Target Type. In summary:

- Standardization Goal – identifies the problem and identifies the actors and entities involved in solving that problem
- Standardization Target Type – An abstract representation of one of the actors or entities identified in the Standardization Goal
- Standardization Target – an implementation of a Standardization Target Type. These are the real-world entities which can be tested for conformance with the requirements documented in the Standard.

Standardization Target Types can be hierarchical. The Conceptual, Logical, Physical hierarchy is one example where the Standardization Target Types are information models. Another example would be implementations of OGC API — Features Part 2 which support XML data exchange.

Notice that the Standardization Targets and Standardization Target Types no longer form a simple taxonomy. The Standardization Target Types, Standardization Targets, and Standardization Goal provide a well-defined context for the standard. This will help users of standards to quickly understand the scope of a Standard and to select those Standards appropriate for their needs. It also will help keep Standards developers focused on the intended use of their standards, avoiding standards which are overly broad and/or unfocused.

8.3. Conformance, Requirements, and key information

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 mandatory requirement specified in the standard. 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 standard. The ModSpec requires a standards document 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, the ModSpec separates them into sets (called "requirements classes"), each of which has no optional components. Since the normative statement of each requirement is only declared once and is uniquely identified as such, each requirement will be in a clause associated to its requirements class.

Therefore, the ModSpec defines a “requirements class” as a set of requirements that must all be passed to achieve a particular conformance class (see Clause 4.7). This document also includes a “middle” structure called a conformance test module. Requirements modules parallel the conformance test modules. A standard written to the ModSpec may use this “module” structure in any manner consistent with the rest of this Policy.

A standard may have mandatory and optional requirements classes. This allows the options in the testing procedure to be grouped into non-varying mandatory and optional conformance classes. Each requirement within an optional requirements class is mandatory when that requirements class is implemented. When needed, a particular requirements class may contain only a single requirement.

However, care must be taken, since the requirements classes may not always in a one-to-one correspondence to conformance classes in other standards which may be the source of requirements for a standard conformant to the ModSpec. If other standards are used, their options shall be specified to be useable within a standard conformant to this policy, see Clause 9.4.1.

Conformance classes may 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.

As defined in the ModSpec, 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).

In the ModSpec, each conformance requirement is separated in its own labeled paragraph, such as Table 2 above.

The distribution of the information in a standard is not restricted. The only requirement is that requirements be grouped in a manner consistent with the conformance test classes, see Table 14 and Table 15.

8.4. Documenting the Standard

NOTE: For OGC Standards, the assumptions is that documents are in a commonly used logical form (template).

This form should be specified by the following descriptions:

1. A standards document 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 standard 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). In OGC Documents, this is Annex A – Abstract Test Suite.
3. All requirements, recommendations, permissions, and models are introduced and defined first in the numbered Clauses.
4. All requirements are identifiable as requirements.
5. All requirements in a document are uniquely numbered.
6. All tests for conformance to those requirements are defined in the Conformance Test Suite.
7. Tests are be grouped for convenience into conformance test classes and if desired the classes are grouped into conformance test modules.
8. The tests, if conducted, determine to some degree of certainty whether an implementation meets the requirements which the tests reference.
9. The tests are organized into some number of conformance "classes" where each conformance class has a one to one relationship with a requirements class. If a standard does not do this, it is has by default only one "conformance class".
10. Certificates of conformance (see Clause 4.1) are awarded by a testing entity based on these conformance classes.
11. There is a clear distinction between normative and informative parts of the text.
12. Examples and notes are informative, and do not use "normative" language.

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.

The ModSpec defines a "requirement" of a standard as an atomic testable criterion. See the formal definition of requirement in Clause 4.21

A UML representation of important properties of this model is given in [annex-B-2].

9

REQUIREMENTS CLASS: CORE

9 REQUIREMENTS CLASS: CORE

The following requirements specify the rules for the content and structure of a modular standard. These requirements are also known as the `core` of the ModSpec.

Table 1

REQ000	**/req/core/ogc-compliance** Any new OGC Standard, abstract specification that contains requirements, or major revision of an existing OGC Standard *SHALL* comply with the requirements stated in this document.

Table 2

REQ001	**/req/core/reqs-are-testable** All the parts of a requirement, a requirements module, or requirements class *SHALL* be testable. Failure to pass any part of any requirement shall be a failure to pass the associated conformance test class.

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

Table 3

REQ002	**/req/core/all-components-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. For OGC standards documents, these URIs SHALL be conformant with the OGC Naming Authority policies.

NOTE: In the OGC, the enforcement of this requirement and its associated recommendation is the purview of the OGC Naming Authority or its equivalents.

Table 4

REC001	**/req/core/uri-external-use** 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.

While a requirement may be referenced in more than one place in a standard, the normative definition of a requirement shall be its “**home**” (see Clause 7.4) and will be the only place where full normative language is used.

The following permissions relate to possible content specified in the core of a standard.

Table 5

PER001	**/per/core/informational-content-in-core** The informational and structural universals of the standard *MAY* be included in the core text and its associated models without violations of the ModSpec. This is true if 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 standard.

Table 6

PER002	**/per/core/core-may-contain-schema-terms** The core *MAY* contain the definition and schema of commonly used terms and data structures for use in other structures throughout the standard.

Table 7

PER003	**/per/core/core-names-of-operations** 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 standard. 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.

The following states how and where vocabularies are specified in relation to a requirement or requirements class.

Table 8

REQ003	**/req/core/vocabulary-and-parent-req-class** Requirements on the use and interpretation of vocabulary *SHALL* be in the requirements class where that use or interpretation is used.

Table 9

PER004	**/per/core/external-vocabs-core**

	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.

9.1. Using the model

The primary difficulty in speaking about standards (or candidate standards)[1] as a group is their diverse nature. Some standards 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 against some model of an implementation of the standard (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.

NOTE: 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 possibly separated from the authors of the standard, leading to issues of separate interpretations of the same standard.

The assumption is that each standard has a single (root) standardization target type from which all extensions inherit. If this is not true, then the standard can be logically factored into parts each corresponding to a "root" standardization target type, and that the standard addresses each such part separately (see the definition of requirements class in Clause 4.22). In this sense, the next requirement divides standard into parts more than restricting their content.

Table 10

REQ004	**/req/core/single-standardization-target** Each requirement in a conformant standard *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.

[1]This is purposely written as "as not yet adopted" standards, since it is during the authoring process that the ModSpec must be considered, not *post facto*.

Table 11

REQ005	**/req/core/modspec/test-class-single-standardization-target** All conformance tests in a single conformance test class in a conformant standard *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 Clause 4.11).

Table 12

PER005	**/per/core/repeated-requirements** 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 standard 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".

Table 13

PER006	**/per/core/abstract-superclass** The standard may introduce an abstract superclass of all affected standardization target types and use this for the requirements common to all of the affected target types. This is diagrammed in Figure 1.

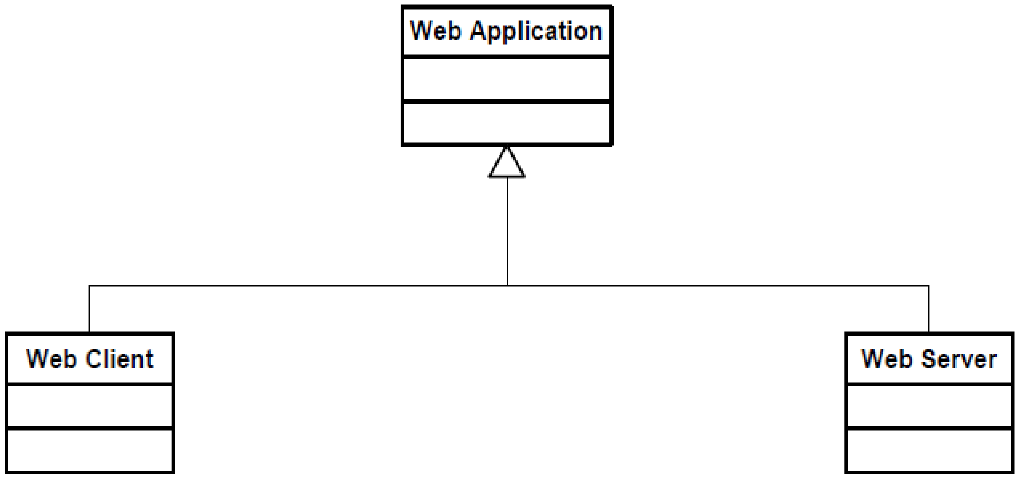

Figure 1 — Abstract superclass example

Example — Abstract Superclass:

9.2. The “standards” document

Each standard document is comprised of a set of requirements and their associated conformance tests.

Table 14

REQ006	**/req/core/requirements-grouped** Requirements SHALL be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements classes.

Table 15

REQ007	**/req/core/requirements-test-suite-structure** The requirements structure of the document SHALL be in a logical correspondence to the test suite structure.

If two requirements are in the same requirments class, they should be tested in the same conformance class in the conformance suite. Each requirement is separately identifiable either by a label as is done in the ModSpec.

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.

9.3. Conformance Test Suite

The requirements specified in this clause 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 are 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.

Table 16

REQ008	**/req/core/requirements-class-correspondence-to-conformance-classes** 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.

9.4. Requirements for Modularity

9.4.1. Each Conformance class tests a complete requirements class

Table 17

REQ009	**/req/core/no-optional-tests** A Conformance class SHALL not contain any optional conformance tests.

This requirement stops conformance classes from containing optional requirements and tests, and, at least as far as the standard is concerned, makes all certificates of conformance mean that exactly the same tests have been conducted. Standards documents may use recommendations for such options, but the conformance test classes do not test recommendations.

Table 18

PER007	**/per/core/conf-class-paramterized** 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. This can be thought of as an "if-then-else" decision tree.

For example, if a conformance class needs to apply tests against a specific data format, such as GML or KML, 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.

Table 19

REQ010	**/req/core/all-parameters-expressed** 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.

Table 20

REQ011	**/req/core/conf-class-single-req-class** 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.

Table 21

REQ012	**/req/core/con-class-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.

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

Example — 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 an OGC Web Feature Service (WFS) implementation instance 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 standard's authors wish to make a particular requirement optional, Table 17 forces them to include it in a separate requirements class (and therefore in a separate conformance test class) which can be left untested.

NOTE: Standards developed outside the OGC may not follow a strict parallelism between requirement specification and testing, so for use within a standard compliant to the ModSpec, special care must be taken in importing conformance test classes from other standards.

Table 22

REQ013	**/req/core/imported-requirements-class**
A	If a requirements class is imported from another standard for use within a standard conformant to the ModSpec, 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 standard.
B	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 tests should be easy if the standards document is non-ambiguous. To insure this, by utilizing the names of the two types of classes the following requirement places a default mapping between the two.

Table 23

REQ014	**/req/core/all-classes-explicitly-named** 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.

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 a requirements class corresponds precisely to a conformance class if they both are related (as described) to the same set of requirements.

9.4.2. Requirements classes contain all requirements tested by a conformance test case

Table 24

REQ015	**/req/core/req-in-only-one-rec-class**
A	Each requirement in the standard *SHALL* be contained in one and only one requirements class.
B	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.

Table 25

REC002	**/rec/core/parallel-structure** If possible, the structure of the normative clauses of the standard *SHOULD* parallel the structure of the conformance classes in the conformance clause.

The above requirement in conjunction with Table 17 means that all requirements in a conformant standard 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.

Table 26

REQ016	**/req/core/co-dependent-requirements**
A	If any two requirements are co-dependent (each dependent on the other) then they shall be in the same requirements class.
B	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).

Table 27

REC003	**/rec/core/circular-dependencies** Circular dependencies of all types should be avoided whenever possible.

Table 28

REQ017	**/req/core/structure-requirements-classes** There *SHALL* be a natural structure to 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 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.

Table 29

REQ018	**/req/core/requirements-and-dependencies** 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.

Clause 9.2 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 standard "by example" that leads to conformance tests. Examples are therefore non-normative, by definition.

9.4.3. Profiles are defined as sets of conformance classes

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

Table 30

REQ019	**/req/core/profile-conformance** The conformance tests for a profile of a standard *SHALL* be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets.

9.4.4. There is a Defined Core

Table 31

REQ020	**/req/core/core-requirements-separate** Every standard *SHALL* define and identify a core set of requirements as a separate conformance class.

Table 32

REQ021	**/req/core/requirements-and-dependencies** All general recommendations *SHALL* be in the core.

Table 33

REQ022	**/req/core/requirements-and-dependencies**
A	Every other requirements class in a standard *SHALL* a standardization target type which is a subtype of that of the core
B	And every requirement class *SHALL* have the core as a direct dependency.

Table 34

REC004	**/rec/core/simple-core** The core *SHOULD* be as simple as possible.

Table 35

PER008	**/per/core/core-type** The core *MAY* be partially or totally abstract.

Table 36

PER009	**/per/core/req-class-another-standard**

	The core requirements class *MAY* be a conformance class in another standard.

Table 37

REC005	**/rec/core/optional-tests** If a requirements class is from another standard, the current standard *SHOULD* identify any optional tests in that conformance class that are required by the current standard's core requirements class. See Table 22.

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.

Table 38

PER010	**/per/core/core-maybe-recommendations** 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."

9.4.5. Extensions are requirements classes

A common mechanism to extend the functionality of a standard is to define extensions, which may be either local or encompass other standards.

Standards should use extensions as required and feasible, but should never hinder them.

Table 39

REQ023	**/req/core/core-and-extensions** Each standard conformant to the ModSpec *SHALL* consist of the core and some number of requirements classes defined as extensions to that core.

Table 40

REQ024	**/req/core/extensions-conformant-to-the-modspec** A standard conformant to the ModSpec *SHALL* require all conformant extensions to itself to be conformant to the ModSpec.

Since software is evolutionary at its best, it would not be wise to restrict that evolutionary tendency by restricting the specification of extensions. A good standard 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.

Table 41

REQ025	**/req/core/restriction-of-extensions** A standard conformant to the ModSpec *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 standard 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 standard 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 standard 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 standard.

9.4.6. Optional requirements are organized as requirements classes

Table 42

REQ026	**/req/core/optional requirements** The only conditional requirements acceptable in a standard conformant with the ModSpec *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.

9.4.7. Requirements classes intersect overlap only by reference

Table 43

REQ027	**/req/core/req-class-overlap-by-reference** 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.

10

MAPPING THIS STANDARD TO TYPES OF MODELS

10 MAPPING THIS STANDARD TO TYPES OF MODELS

10.1. Semantics

The previous section defines requirements for conformance to the ModSpec, but testing for that conformance may depend on how the various forms and parts of a conformant standard are viewed. This clause specifies 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 ModSpec 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 model from [cls-6-1].

As suggested in Clause Table 33, the structure of the normative clauses in a standard should parallel the structure of the conformance test classes in that standard. The structure of the normative clauses in a well written standard will follow the structure of its model. This means that all three are parallel.

10.2. Data Models

10.2.1. 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 model 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.

10.2.2. Optional Requirements class: UML model extension to the core

If the organizing mechanism for the data model used in the standard 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.

Table 44

Requirements Class — UML extension to the core	
/req/core/data-representation	
Target	ModSpec Conformant UML Model
Dependency	OGC ModSpec Version 2 (need proper title and document number)
REQ028	/req/core/uml/conformance-with-core
REQ029	/req/core/uml/object-model
REQ030	/req/core/uml/dependency-graph
REQ031	/req/core/uml/leaf-package
REQ032	/req/core/uml/class-package
REQ033	/req/core/uml/to-leaf
REQ034	/req/core/uml/common-req-classes
REQ035	/req/core/uml/source-target
REQ036	/req/core/uml/leaf-package-dependency
REQ037	/req/core/uml/two-way-dependency
REQ038	/req/core/uml/segregate-into-leaf-packages

Any conformant UML extension shall comply with the ModSpec Core requirements class.

Table 45

REQ028	**/req/core/uml/conformance-with-core** An implementation passing the UML conformance test class *SHALL* first pass the core conformance test class

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.

Table 46

REQ029	**/req/core/uml/object-model** To be conformant to this UML requirements 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

Table 47

REQ030	**/req/core/uml/dependency-graph** 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.

Table 48

REQ031	**/req/core/uml/leaf-model** A UML leaf package *SHALL* be associated directly to only one requirements class.

Table 49

REQ032	**/req/core/uml/class-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.

The class definitions are the “requirements” of a UML expressed standard. Therefore the logical consequence of the above is to organize requirements classes based on leaf packages.

Table 50

REQ033	**/req/core/uml/to-leaf** 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/or class.

Table 51

REQ034	**/req/core/uml/common-req-classe** Classes that are common to all requirements classes *SHALL* be in a package associated to the core 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 Table 33, only classes in the source requirements class will be affected by the requirement.

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

Table 52

REQ035	**/req/core/uml/source-target**
A	In the UML model, if a “source” package is dependent on a “target” package then their requirements class *SHALL* be equal or
B	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 all leaf packages of the model are moved into "requirements class packages" containing their corresponding modeling packages the model then satisfies the following recommendation:

Each requirements class in a conformant standard 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.

REQUIREMENT 1	
STATEMENT	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.

REQUIREMENT 2	
STATEMENT	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.

REQUIREMENT 3	
STATEMENT	The UML model *SHALL* segregate all classes into leaf packages.

10.2.3. Requirements class: The XML schema extension to the core

This requirements class covers any standard which has as one of its purposes the introduction of a new XML schema. Such a standard 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.

REQUIREMENT 4	
STATEMENT	An implementation passing the XML schema conformance test class shall first pass the ModSpec core conformance test class.

REQUIREMENT 5	
STATEMENT	An implementation passing the XML schema conformance test class shall first pass the W3C Recommendation for XML schema.

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.

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".

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

REQUIREMENT 6	
STATEMENT	If a standard 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.

REQUIREMENT 7	
STATEMENT	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 the ModSpec, 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.

REQUIREMENT 8	
STATEMENT	If a standard 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 1: 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 2: 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.

REQUIREMENT 9	
STATEMENT	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:

1. 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.

2. 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.

10.2.4. Optional Requirements class: Schematron extension to the ModSpec Core

Schematron (ISO/IEC 19757-3:2006) provides a notation with which many constraints on XML documents can be expressed. This requirements class covers any standard 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.

REQUIREMENT 10	
STATEMENT	A standard 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:

REQUIREMENT 11	
STATEMENT	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.

REQUIREMENT 12	
STATEMENT	Each sch:pattern element shall be contained within one sch:schema element.

REQUIREMENT 13	
STATEMENT	The value of the sch:schema/@fpi attribute shall be a URI that identifies this implementation

REQUIREMENT 14	
STATEMENT	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

REQUIREMENT 15	
STATEMENT	The value of the sch:schema/@fpi attribute shall be used on only one Schematron schema.

10.2.5. Optional Requirements class: XML meta-schema extension tothe ModSpec Core

This requirements class covers any standard which has as one of its purposes the introduction of a new type of XML schema. Such a standard 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.

REQUIREMENT 16	
STATEMENT	A standard 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 the ModSpec be used.

REQUIREMENT 17	
STATEMENT	A standard passing the XML meta-schema conformance test class shall require that its specification targets (XML schema) pass the XML schema conformance class from the ModSpec.

A

ANNEX A (NORMATIVE) ABSTRACT CONFORMANCE TEST SUITE

A

ANNEX A (NORMATIVE) ABSTRACT CONFORMANCE TEST SUITE

A.1. Conformance Test Class: The 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 2
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 3
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 8
4. Test Type: Conformance.

A.1.4. Requirements have a single target

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 10
4. 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 standard shall have the same standardization target.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 11
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.

2. Test Method: Inspect the document to verify the above.
3. Reference: Clause 9.2, Table 14
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Clause 9.2, Table 15
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 16
4. Test Type: Conformance.

A.1.9. No Optional Elements in Requirements classes

A Conformance class shall not contain any optional conformance tests.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 17
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 19
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 20
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 21
4. 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 standard 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 standard. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 22
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 23
4. 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).

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 24
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.

2. Test Method: Inspect the document to verify the above.
3. Reference: Table 26
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 28
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 29
4. Test Type: Conformance.

A.1.19. Profiles are expressed as sets of conformance classes

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 30
4. Test Type: Conformance.

A.1.20. There is a named Core requirements class

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 31
4. Test Type: Conformance.

A.1.21. General conditions are in the core

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 33
4. Test Type: Conformance.

A.1.22. Every extension has a consistent target type

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 33
4. Test Type: Conformance.

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

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

1. Test Purpose: Verify that this requirement is satisfied.

2. Test Method: Inspect the document to verify the above.
3. Reference: Table 39
4. Test Type: Conformance.

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

A standard conformant to the ModSpec shall require all conformant extensions to itself to be conformant to the ModSpec.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 40
4. Test Type: Conformance.

A.1.25. Future extensions cannot be restricted

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 41
4. Test Type: Conformance.

A.1.26. Optional requirements are organized as requirements classes

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 42
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 43
4. Test Type: Conformance.

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

A.2.1. Dependency on Core

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 45
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 46
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 47
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 48
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 49
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.

2. Test Method: Inspect the document to verify the above.
3. Reference: Table 50
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 51.
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 52.
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 1.
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 2
4. Test Type: Conformance.

A.2.11. All classes are in leaf packages

The UML model shall segregate all classes into leaf packages.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 3
4. 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 standard conformance test class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 4
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 5
4. Test Type: Conformance.

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

If a standard 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 6
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 7
4. Test Type: Conformance.

A.3.5. Direct dependencies become schema imports

If a standard 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 8
4. Test Type: Conformance.

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

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 9
4. Test Type: Conformance.

A.4. Conformance test class: Schematron

A.4.1. Dependency on XML Schema conformance test class

A standard 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 10
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 11
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 12
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 13
4. 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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.

3. Reference: Requirement 14
4. 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 15
4. Test Type: Conformance.

A.5. Conformance Class: XML meta-schema

A.5.1. Supports the Specification class

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

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 16
4. Test Type: Conformance.

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

A standard 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.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Requirement 17
4. Test Type: Conformance.

B

ANNEX B (NORMATIVE) OGC ONLY: CHANGES REQUIRED IN THE OGC STANDARDS

B ANNEX B (NORMATIVE) OGC ONLY: CHANGES REQUIRED IN THE OGC STANDARDS

NOTE: The following is for OGC Standards and Abstract Specifications only: No changes are required to existing OGC Standards

B.1. New OGC standards and revisions to existing OGC standards

Any new standard or major revision of an existing standard *SHALL* comply with the ModSpec in the structure of its internal models and its conformance tests.

Failure to conform by a candidate standard to the ModSpec 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 the ModSpec *SHALL* be considered as an authorized exception to the requirements of the ModSpec by the approporiate authority, such as the OGC or ISO. An exception to the rules of the authority such as the OGC and will require a two-thirds (2/3) majority ("Robert's Rules") or as specified in the authorities Policy and Procedures for an exception to procedure. In the OGC, a similar vote is required within the Executive Planning Committee or as specified in any Policy and Procedure document used by this committee.

C

ANNEX C (INFORMATIVE) DEFINITIONS

C

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* (Clause 4.27) is a type of entity about which a standard is written. An instance of a *standardization target type* (Clause 4.27) is a *standardization target* (Clause 4.26). A standard may address multiple targets in separate *conformance classes*.

2. a *requirement* (Clause 4.21) is a statement of a condition to be satisfied by a single *standardization target type* (Clause 4.27), and it must be stated in "normative" language.

3. a *conformance test* (Clause 4.4) checks if a set of *requirements* (Clause 4.21) are met (**pass**) or not met (**fail**) by a *standardization target* (Clause 4.26). The relationship between *conformance tests* (Clause 4.4) and *requirements* (Clause 4.21) is many-to-many.

4. all *conformance tests* (Clause 4.4) are graded as **pass** or **fail** against each instance of the *standardization target* (Clause 4.26).

5. a *requirement* (Clause 4.21) is associated to one *conformance test* (Clause 4.4).

6. a *recommendation* (Clause 4.20) is a suggestion and is not associated to any *conformance test* (Clause 4.4).

7. a *requirements class* (Clause 4.22) is a set of one or more *requirements* (Clause 4.21) all with the same *standardization target type* (Clause 4.27).

8. a *conformance (test) class* (Clause 4.7) is a collection of *conformance tests* (Clause 4.4) that are associated to and only to the requirements in a corresponding *requirements class* (Clause 4.22).

9. a *conformance (test) module* (Clause 4.6) is also collection of **term `conformance test classes` not resolved via ID `conformance-test-classes`** that group *conformance tests* (Clause 4.4) on a single *standardization target type* (Clause 4.27).

10. a **conformant implementation** is a *standardization target type* (Clause 4.27) that has successfully passed all tests in a specified *conformance (test) class* (Clause 4.7) and received a *certificate of conformance* (Clause 4.3)

11. the *core requirements class* (Clause 4.9) of a standard is the minimal set of *requirements* (Clause 4.21) which must be supported by all **conformant implementations**. If a standard addresses multiple *standardization target types* (Clause 4.27), it may have a **core** for each **target type**.

12. an **extension** of a *requirements class* (Clause 4.22) is an additional *requirements class* (Clause 4.22) (the extension) that adds additional *requirements* (Clause 4.21) to the first *requirements class* (Clause 4.22) (the **base requirements class** being extended). The extension is said to be dependent on the **base**. Any *conformance test class* (Clause 4.7) must identify all its dependencies during the execution of conformance tests against a candidate *standardization target* (Clause 4.26).

C.2. UML Model

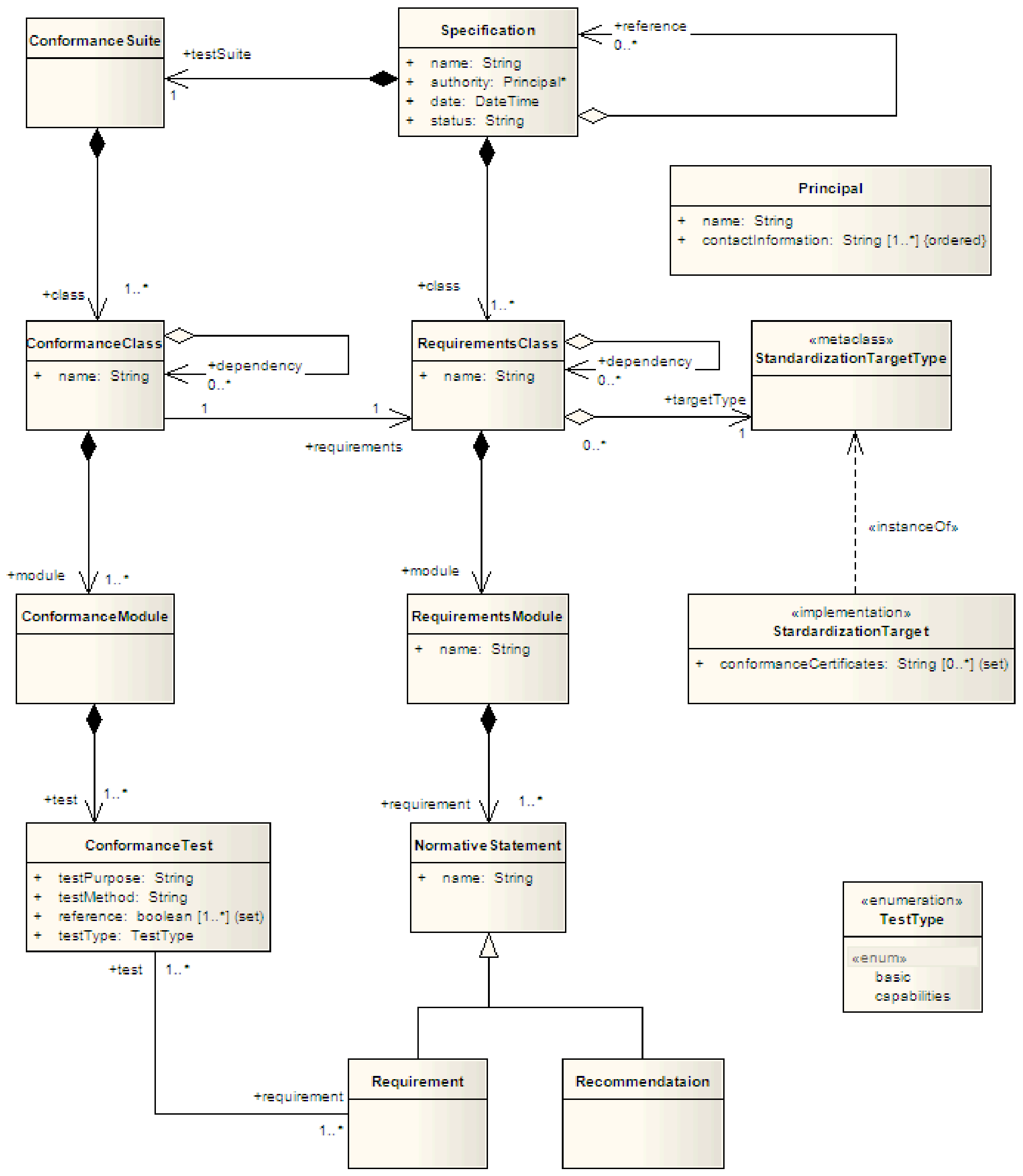

Figure C.1 — Specification structure

Figure C.1 represents a UML model consistent with the specification model described in [cls-6-1]. The following subclauses describe the classes shown in this UML class diagram.

C.3. Specification

C.4. Specification

For a draft standard (aka specification) to become an international standard, the document must be approved by an authority, such as ISO or the OGC. The attributes of a standard 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).

The attributes of a Standard describe its local name, its authority, the date of publication and its current status (such as Draft, Candidate Standard or Standard in the OGC).

Table C.1 — Specification attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
name	Name of the standard.	M	1	`String`
authority	Standards body or author of this standard.	M	1	`Principal`
date	Publication date of the standard.	M	1	`DateTime`
status	Publication status of this standard.	M	1	`String`

C.5. Conformance Suite

C.6. Conformance Class

C.7. ConformanceClass

The requirements in the requirements classes of a standard must 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.

Table C.2 — ConformanceClass attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
name	Name of the conformance class.	M	1	`String`
dependencies	Conformance classes that this conformance class depends on. These dependent conformance classes must be passed if this one is to be passed.	O	N	`ConformanceClass`
requirements Class	The requirements class that this conformance class aims to test against.	M	1	`RequirementsClass`

C.8. Requirements class

C.9. RequirementsClass

Requirements classes (usually realized as clauses in the standard's document) segment the requirements in the standard 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 standard's name and authority.

Table C.3 — RequirementsClass attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
name	Name of the requirements class.	M	1	`String`
dependencies	Requirements classes that this requirements class depends on. These dependent requirements classes must be satisfied for this requirements class to be satisfied.	O	N	`RequirementsClass`
modules	Requirements modules that make up this requirements class.	M	N	`RequirementsModule`
targetType	Type of standardization target.	M	1	`StandardizationTargetType`

C.10. Requirements module

C.11. RequirementsModule

A requirements modules (usually realized as groups of one or more requirements classes in the standard) group the requirements and recommendations in the standrd in a manner consistent with the conformance test modules.

Table C.4 — RequirementsModule attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
name	Name of the requirements module.	M	1	`String`
requirements	Requirements classes that this requirements module contains.	M	N	`Requirement`

C.12. Normative Statement

C.13. NormativeStatement

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".

Table C.5 — NormativeStatement attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
name	Name of the normative statement.	M	1	`String`

C.14. Requirement

C.15. Requirement

Normative statement that constitutes a requirement.

Table C.6 — Requirement attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
tests	Conformance tests that when passed confirm the satisfaction of this requirement. NOTE: If this requirement is a requirement part, it may or may not have a corresponding conformance test.	M	N	`ConformanceTest`
parts	Collection of requirements that are parts to this requirement. Satisfaction of all requirement parts are necessary for this requirement to be satisfied. Optional.	O	N	`Requirement`

C.16. Recommendation

C.17. Recommendation

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

- 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.
- 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.

C.18. Conformance test

C.19. ConformanceTest

A conformance test aims to satisfy a requirement and can potentially contain multiple test methods.

Table C.7 — ConformanceTest attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
abstract	Whether this test is abstract or concrete. An abstract conformance test is commonly called an abstract test.	M	1	`Boolean`
testPurpose	Purpose of the conformance test.	M	1	`String`
testType	Type of the conformance test.	M	1	`TestType`
testMethod	Method to perform this conformance test. A method is considered a “part” of the test if there are multiple of them.	O	N	`ConformanceTestMethod`
references	References to the specification(s) of the conformance test.	O	N	`RichText`
requirements	Corresponding requirement or requirement part that this conformance test is supposed to test against.	M	N	`Requirement`

C.20. StandardizationTarget

C.21. 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.

Table C.8 — StandardizationTarget attributes

NAME	DEFINITION	MANDATORY / OPTIONAL / CONDITIONAL	MAX OCCUR	DATA TYPE
conformance Certificates	conformance classes passed by this target	O	N	`String`
type	Type of the standardization target type.	M	1	`StandardizationTargetType`

C.22. StandardizationTargetType

C.23. StandardizationTargetType

BIBLIOGRAPHY

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.

[1] ISO/IEC: ISO/IEC 9075:2003, *ISO/IEC JTC 1, ISO/IEC 9075:2003 — Information Technology — Database Languages — SQL.*. ISO, IEC (2003).

[2] ISO/IEC: ISO/IEC TR 10000, *ISO/IEC TR 10000: Information Technology — Framework and taxonomy of International Standardized Profiles*. ISO, IEC

[3] ISO/IEC: ISO/IEC 13249-3:2006, *Information technology — Database languages — SQL multimedia and application packages — Part 3: Spatial*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2006). https://www.iso.org/standard/38651.html.

[4] 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

[5] 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

[6] W3C: W3C xmlschema11-1, *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. World Wide Web Consortium http://www.w3.org/TR/xmlschema11-1/.

[7] W3C: W3C xmlschema11-2, *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. World Wide Web Consortium http://www.w3.org/TR/xmlschema11-2/.

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.

[8] John Herring: OGC 08-079, *OWS5: OGC Web feature service, core and extensions*. Open Geospatial Consortium (2008).

[9] ISO: ISO 19101, *Geographic information — Reference model*. International Organization for Standardization, Geneva https://www.iso.org/standard/26002.html.

[10] ISO: ISO 19107, *Geographic information — Spatial schema*. International Organization for Standardization, Geneva https://www.iso.org/standard/66175.html.

[11] ISO: ISO 19111, *Geographic information — Referencing by coordinates*. International Organization for Standardization, Geneva https://www.iso.org/standard/74039.html.

[12] ISO: ISO 19119, *Geographic information — Services*. International Organization for Standardization, Geneva https://www.iso.org/standard/59221.html.

[13] ISO: ISO 19125, *Geographic information — Simple features*. ISO

[14] ISO: ISO 19133, *Geographic information — Location-based services — Tracking and navigation*. International Organization for Standardization, Geneva https://www.iso.org/standard/32551.html.

[15] ISO: ISO 19136, *Geographic information — Geography Markup Language (GML)*. International Organization for Standardization, Geneva https://www.iso.org/standard/32554.html.

[16] ISO: ISO 19141, *Geographic information — Schema for moving features*. International Organization for Standardization, Geneva https://www.iso.org/standard/41445.html.

[17] ISO: ISO 19142, *Geographic information — Web Feature Service*. International Organization for Standardization, Geneva https://www.iso.org/standard/42136.html.

[18] ISO: ISO 19143, *Geographic information — Filter encoding*. International Organization for Standardization, Geneva https://www.iso.org/standard/42137.html.

[19] ISO: ISO 19148, *Geographic information — Linear referencing*. International Organization for Standardization, Geneva https://www.iso.org/standard/75150.html.

[20] ISO: ISO 19149, *Geographic information — Rights expression language for geographic information — GeoREL*. International Organization for Standardization, Geneva https://www.iso.org/standard/32567.html.

[21] ISO: ISO 19153, *Geospatial Digital Rights Management Reference Model (GeoDRM RM)*. International Organization for Standardization, Geneva https://www.iso.org/standard/32571.html.

[22] ISO: ISO 19156, *Geographic information — Observations, measurements and samples*. International Organization for Standardization, Geneva https://www.iso.org/standard/82463.html.