

All Fields marked with * are mandatory.

Change Request #:	205
Assigned OGC Document #:	12-017r2
Name:	*Adrian Custer
Organization:	*independent
Email:	*ac@pocz.org
Document Name/Version:	*The Specification Model --- A Standard for Modular specifications / 1.0.0
OGC Project Document:	*08-131r3
If this is a revision of a previous submission and you have a Change Request Number, then check here: <input type="checkbox"/>	
Enter the CR number here:	<input type="text" value="12-017"/>
Enter the Revision Number that you are revising here:	<input type="text" value="0.1"/>
<hr/>	
Title:	*Editorial rewrite of The Specification Model -- A Standard for Modular specifications
Source:	*Individual Member Adrian Custer
Work item code:	
Category:	* <input type="text" value="D (Editorial modification)"/>
<hr/>	
Reason for change:	* The standard, while exceedingly useful, is poorly structured and written.
Summary of change:	* Reexamine the goal of the document, for a better title, better organized introductory text, and more systematic organization of the content. Review the language of the normative text to fix the language and order. Fix errata.

Consequences if not approved: ?	Without this editorial work, the standard will remain exceedingly difficult to use since most of the requirements are wrong---a reader has to interpret them to figure out what they are trying to say---and because the material has poor internal organization.
<hr/>	
Clauses affected: ?	* All
Additional Documents affected: ?	None
Supporting Documentation: ?	
Comments: ?	This is version 2.0 (aka 12-017r2). The last clause, clause 7, has been expanded greatly.
Status: ?	Assigned ▾
Assigned To: ?	Policy 1.0 SWG (archived) ▾
Disposition: ?	Referred and Posted ▾

Open Geospatial Consortium

Date: 2012-02-18

External identifier of this OGC® document: <12-017r2>

Version: 2.0

Category: OGC® Change Request

Editor: Adrian Custer

Change Request for *The Specification Model (08-131r3)*

Table of Contents

1. Introduction.....	2
2. Big issues.....	5
3. Evaluation of the Requirements.....	8
4. The UML model.....	23
5. Editorial comments.....	24
6. A general outline of a new text.....	28
7. A new set of requirements.....	30

1. Introduction

This Change Request presents an extensive analysis of the OGC Standard

The Specification Model — A Standard for Modular specifications 08-131r3

which aims to clarify the goals, organization, and realization of the document. Through this analysis, it becomes apparent that the standard could use extensive revision in order to:

- edit the text for clarity,
- state the purpose of each requirement class,
- organize like requirements together,
- systematize the requirements for parallelism between the various injunctions,
- edit the requirement language to specify the target and phrase the injunction against that target,
- drop unenforceable, burdensome, or useless requirements and requirement classes,
- clarify the role of the UML model of 'Well-Formed Standards' in the document,
- fix the UML,
- improve the tests from the trivial and vague "Inspect the document to verify the above,"
- harmonize or clarify the differences between the UML and XML requirement classes,
- fix errata,

but much of that work will only make sense after clarifying the conceptual model behind the document, the goals of each requirement class and requirement, and the purpose and organization of each section of the text.

The document *The Specification Model* contributes greatly to improving the quality and content of OGC specifications and the work on that standard has been invaluable in getting the consortium working together towards this goal. The criticism in this document should be seen in a context of praise for the existing work and recognition that this critique is only possible due to the existence of that work: it is much easier to improve a conceptual effort than to conceive of it anew. Therefore, in an Orwellian twist, this criticism is actually praise: my willingness to invest work on the improvement of the standard testifies to the value I see in that document.

The standard, unfortunately, is confused. As a quick illustration of its confusion, it nowhere states what Standardization Target Types it tackles. An exercise for the reader of this document illustrating the confusion of the standard would be to name the Target

Types. If the only Target is a 'specification document' as appears clear from the tests then how do you restate Requirement 10 to target a specification document explicitly?

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

(Update: actually this is possible as shown below but requires whole new concepts.) Another example of the lack of self-conscious focus comes from the stress of 'modularization' as the key contribution of the standard. The greatest contribution of the standard to the OGC standardization process is actually not a framework for modularity but the rule that 'Formal OGC Requirements' must be injunctions made against 'Targets' and injunctions which are testable. The process of phrasing a testable injunction against a target entity reveals so much about our intent and its possibility that it helps us write better standards. Indeed, much of the critique in this document comes simply from the effort below of rephrasing each requirement in the form:

<standardization target> SHALL (NOT) <injunction>
coupled with building a test for those injunctions. The effort reveals directly some of the confusion in the Requirements themselves. I have personally decided that all my injunctions must be in that form; it's boring but exceedingly helpful to me as an author. A second trivial tell tale sign of disorganization is that the 'OGC Requirement Classes' in the document occur at such different levels: §6 for core, §7.2.2, §7.2.3, §7.2.4, and §7.2.5 for the others, something which is not disorganized inherently but suggests the possibility of being so.

The root of the confusion appears to come from the incomplete conceptual model against which the standard is written. The clearest phrasing of the 'world view' against which the standard is written comes in §6.2, ¶1 which states:

The primary difficulty in speaking of specifications (or candidate standards) as a group is their diverse nature. Some specifications use UML to define behavior, others use XML to define data structures, and others use no specific modeling language at all. However, they all must model the standardization target to which they apply since they need to use unambiguous language to specify requirements. Thus, the only thing they have in common is that they define testable requirements and recommendations against some model of an implementation of the ~~specification (the standardization target)~~ type. *{ending modified}*

This could use editorial clarification to become a statement of intent but it gets to the core of what the standard clearly realizes it is attempting to do. The standard aims, among other things, to develop a model of a 'well formed specification' and develop rules which, if respected, would ensure any specification document would be 'well formed'. Conceptually that is at the root of the standard but the standard actually rests on a bigger conceptual foundation than that. The key pieces of that bigger foundation seem to be: target entities, injunctions, evaluations to determine if a target entity fulfills the injunctions, a process to perform the evaluations, and a result of the evaluating process. In other language, the key conceptual elements are: *Targets*, *Requirements*, *Tests*, an unnamed testing processor, and *Certificates of Conformance*. The unnamed processor might be a review by the OAB (running the tests of The Spec. Model standard) or might be a CITE test suite (running the tests of some service implementation specifications) but the processor has no generic representation in the language of the The Spec. Model

standard. The Requirements in The Spec. Model standard make injunctions against these elements: against *Targets* (documents shall group requirements in clauses), against *Requirements* (each requirement has a single target type), against *Tests* (only that they have a URI), against the *processor* (the processor shall only issue a certificate if the target passes all the tests in a conformance class), and against the *certificate* (the certificate shall list any parameters of the testing process). The first three of these can actually be rephrased as requirements of a specification document but the latter two, seemingly, can not. (*Update: well we can, as shown tackling req. 10 below, but this requires introducing the concept of the 'testing process' so the requirement can force spec documents to require their 'testing process' to behave in a particular way.*) The requirements that specify the grouping mechanism I see as fulfilling a secondary purpose: first the standard is aiming to make rules for well written specs, then the standard is aiming to ensure those specs are modular by developing rules for grouping. (A tertiary aim is to define the use of UML and XML schema systems; those have their own issues as we will see later.)

A note on terminology since that also is confused. I will attempt systematically to capitalize and italicize entities which are part of the UML model, *i.e.* the 'classifiers,' and to capitalize and label other entities. We have many 'specifications' floating around: we need to distinguish a 'specification' which is any document which makes injunctions, from a '*Specification*' which is the classifier in the UML model, from a 'Conformant OGC Specification Document' which is a specification which meets all the requirements of the core requirement class (and could potentially become an 'OGC Standard'). Similarly, there are many 'requirements' so I speak of 'injunction' which is a rule against some entity to be some way, of '*Requirement*' which is the UML classifier, and 'Formal Requirement' which is an injunction stated formally in the text to enjoin a Target to be a certain way (although I mix up the last two a bit). Also, I will attempt to systematically call *The Specification Model* 'the standard', call this change request in which this sentence appears 'this document,' and otherwise speak only of 'specifications' since 'standards' are defined as 'specifications' that have undergone some approval process.

This change request document has the following structure.

- The first clause tackles the 'big issues': self-definition of the document in which it is self-conscious of what it is doing, a formal listing of the target types, the role of the UML model in the document, the need to split internal OGC policy from what is otherwise a general document, and the recognition that there are several types of requirements.
- The next clause works systematically through the published requirements to evaluate each one.
- The following clause tackles all the issues with the UML model.
- The final clause works through the text of the document with editorial comments and errata.

2. Big issues

Several major issues related to the standard document must be resolved prior to productive work improving the document.

2.1 Self-definition

The standard must be explicitly clear as to its **intent**: it aims for better specification documents. To do that it will develop a model of well structured specifications, and will develop rules for realizations of the model and model elements, for the expression of that model in text, for the evaluation of the result, and for the declaration of successful evaluations. The standard hopes that better specification documents will lead to all sorts of good benefits (see the Introduction and §1, ¶1) but those benefits are not tackled directly by the standard.

The standard must be explicitly clear as to what it **is**: it is a specification for specification documents. While the standard calls itself a model (*The Specification Model ...*) that is only one part of what it aims to be as I have argued above. Really the model is only a means by which the standard does what it aims to do (even if it claims that modeling of the target is an inherent necessity of what all specifications do). The issue of modularity is an extra that exists to facilitate reuse, correction, and extension of pieces of these specifications.

If we agree on the goals and the nature of the standard, we can evaluate our work writing the clauses of the text, phrasing the injunctions of the requirements, and grouping the requirements into classes by comparing our work against those two.

2.2 Types of targets

Critical to the process of writing good specifications, and in my mind the single biggest contribution of the formalized approach to specifications contributed by the standard, is forcing the specification documents to define the entities which are being enjoined by the text of the Formal Requirements. Conversely, the single biggest failure of the standard itself is that it does not formally list its own *Standardization Target Types* and does not phrase its *Requirements* as injunctions made of *Targets* of those *Types*.

Clearly, one *Standardization Target Type* targeted by the standard is "specification documents". Some requirements clearly target such documents by expressing injunctions on the text of the specifications (e.g. related to the content and organization of 'clauses'). For example we have Req. 6's "The requirements shall be grouped together in clauses..." and Req. 7's "structure of the document". All the tests appear to assume that a 'specification document' is the target.

While the phrasing of other *Requirements* suggests a further group of *Standardization Target Types* which are the elements of the UML model (*i.e.* "classifiers"), it seems that the language of those injunctions could be rephrased as injunctions of a specification document containing such classifiers. Keeping existing phrasing would bring on the cost of introducing a *Target Type* for each classifier. However, regardless of the formal

Target Type used, it still serves us to organize these *Requirements* by kind of classifier because it makes the testing procedure simpler to express and apply. Instead of the mixed requirement

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

and the test of inspecting the document and seeing that this is true for all "components", in my work on WMS 2.0, I develop a *Requirement* for each classifier, e.g. that it have an HTTP URI. My test then becomes that every occurrence of a text element represented by a classifier should be inspected for the presence of the URI. In my test procedure, all the tests of each classifier are together so that checking each element is a simple run down of the list of requirements. The organized approach grouping *Requirements* by classifier also allows one to see clearly the parallels between the *Requirements* for the different kinds of elements at the minor expense of having more *Requirements* (but which involve less work in assessing them).

A second type of *Standardization Target Type* appears to be an assessment process, but this only emerges from the examination of the language of certain *Requirements*. The second sentence of Req. 1 seems to be of a different genre from the first sentence. The first could be rephrased to target the document as shown below, the second is a criterion for the testing process to decide on the failure by a specification document to conform with a *Conformance Class*.

A third type of *Standardization Target Type* appears in Req. 10 which targets a 'certificate of conformance'. I see no way to modify the language of that requirement to target either a 'specification document' or an 'assessment process'. (Update: I actually did below with some conceptual evolution.)

This leads me to view the core of the standard as actually targeting three different *Standardization Target Types*: specification documents, a conformance testing process, and a certificate of conformance.

Beyond the core, the targets get more confusing. The UML requirement class seems to target 'specifications which use UML to model their target' but it could also be targeting 'specifications that use UML in any way' I am not sure. The XML requirement classes seem to target XML schemas either in general or only those used to model 'data' (see Req. 41). That needs to be clarified by the authors and would be the subject of a future discussion. Note in passing that Req. 39 is against something that could pass the testing of the core conformance class, i.e. a document, whereas Req. 40 is against something that could pass the W3C recommendation, i.e. an XML schema probably expressed as a text '.xsd' file. The two are totally different things and so the language should not call each using the same 'an implementation' language.

2.3 Role of UML model

Given the centrality of the UML model of specifications to the requirements of the standard, I do not understand how it could be squirreled away in an 'informative' annex. The text of Annex C1, despite its conflicts with the rest of the standard, stands as the best textual expression the model of a standard, the UML graph (which seems to be required by Req. 29, claimed as normative in the 'Scope' clause and referenced in clause 6.1) appears to be the only listing of the classifiers, and the classifier by classifier writeup seems to posit the existence of attributes but these are not required elsewhere.

Since I suspect the model should be central to the text of the standard, the UML model should be folded into clause 6 as a section (possibly a formal 'Module') which discusses the model, its entities, and their relations. Alternatively this will have to be done, piece by piece, in the language of requirements.

2.4 Separation of OGC specifics

The standard is both a specification for specifications and is a base requirement for all future OGC Standards. However, this second part is trivial in the doc: that the naming authority issuing URIs be the OGC NA and that the standard be an annex to the Policies and Procedures of the Technical Committee. These must be separated from the rest of the standard, it is too useful as a general help to writing specifications to be tied to the OGC.

2.5 Types of requirements

Identifying the different types of requirements gives us a way to organize them.

Obviously, requirements differ by the *Standardization Target Type* of their *Target*. Properly those are to be in different *Requirement Classes*. However, even within a single class, other characteristics can distinguish Formal Requirements within the core. The different kinds of Formal Requirements for core seem to be:

- Formal Requirements enjoining documents to define model elements in a particular way, e.g. Req.4 "Each requirement in a conformant specification shall have a single standardization target type {sic}."
- Formal Requirements enjoining documents to structure model elements in a particular way, e.g. the first sentence of Req. 15, "Each requirement in the standard {sic} shall be contained in one and only one requirement class",
- Formal Requirements enjoining documents to a particular form, e.g. Req. 6 "The requirements shall be grouped together in clauses (numbered sections) of the document..."
- Formal Requirements enjoining the conformance process, e.g. second sentence of Req. 1 "Failure to meet any part of any requirement shall be a failure to pass

the associated conformance test class."

- Formal Requirements enjoining the certificate of conformance, e.g. Req. 10 "A certificate of conformance shall specify all parameter values used to pass the tests in its conformance test class."

These kinds of Formal Requirements give us an organizational system, especially for the first three kinds which could occur in the same *Requirements Class*.

(Update: Formal Requirements of the latter two kinds actually can be rephrased to have the same target as the first, and therefore all actually could be in the same *Requirements Class*.)

3. Evaluation of the Requirements

In this clause, I examine each Formal Requirement in turn. Part of the analysis involves rephrasing the language of the Formal Requirements into a phrasing where the target is explicit in the language. It is also worth specifying exactly what the testing is to be.

CORE

3.1 Analysis of Req. 1

The text reads:

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

Starting the requirements with testing seems problematic. You are just done saying 'we assume a spec. document has these parts: ...' It would be better to start with defining the rules for each part before getting to how an evaluating process will use that whole structure.

The first sentence needs refactoring. Requirements are supposed to be 'atomic' (see p. 10, paragraph before 1st note.); if so, speaking of their parts, "parts of a requirement," is wrong. And Requirement Modules or Requirement Classes are only tested via their Requirements so that if the language of 'parts of' is supposed to be transitive to 'module' and 'class', "All the parts ofa requirements class", then those 'parts' are actually the requirements and we are back to testing only single, atomic *Requirements*, which, I suspect, is what this requirement should be doing.

Is this language trying to say, via 'parts of requirements' that the Tests targeting a requirement must collectively, fully evaluate that requirement? That's unenforceable and also impossible in certain cases where our injunctions can only ever be invalidated not affirmed. So that sounds like we should be saying:

A conformant specification document SHALL have all of its Requirements matched by one or more Tests that evaluate the Target of the Requirement for conformance with that Requirement.

A conformant specification document SHOULD include Tests that evaluate as completely as possible the conformance of the Target with the Requirement.

The second sentence of the requirement is about how the testing procedure fails a specification document. I can see how to write this if the target is the procedure:

A conformance evaluating procedure SHALL evaluate the conformance of a Target to a Requirements Class by applying to the Target each Conformance Test in the Conformance Class associated with the Requirements Class. The conformance evaluating procedure SHALL interpret the failure of the Target to pass any Conformance Test in the Conformance Class as a failure to pass the Conformance Class itself.

The 'associated with' needs to become a stronger language, probably it should be 'the requirements class declared as its "own" by the conformance class' or some similar formal declaration.

However I don't quite see how to rephrase that as a character of the 'conformant specification document' itself (which is why I think there is more than one Target for the core of the standard). Perhaps this becomes

A conformant specification document SHALL specify that the conformance evaluating procedure MUST evaluate the conformance of a Target to a Requirements Class by applying to the Target each Conformance Test in the Conformance Class associated with the Requirements Class.

A conformant specification document SHALL specify that the conformance evaluating procedure MUST interpret the failure of the Target to pass any Conformance Test in the Conformance Class as a failure to pass the Conformance Class itself.

but this posits that we have formalized that the spec. is developing a 'conformance evaluating procedure'---something we have not yet done.

3.2 Analysis of Req. 2

The text reads:

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

What are the 'components of the standard'? Are they the classifiers that are listed (*i.e.* 'including only') or are they the listed classifiers plus others (*i.e.* 'including, amongst others, ')? If the former, then simply enumerating the classifiers would be more clear.

Why, in the standard, do we care who generates the URIs. If they are Universal then it's not our beef. As stated above, for OGC Standards we can mandate that the URIs are issued by the OGC NA but that is a separate issue. This language should probably change to

"Conformant specification documents SHALL assign a URI, *preferably issued by a proper IANA naming authority*, to each occurrence in the document of a requirement, requirements module, requirements class, conformance test case, conformance module or conformance class."

and this should be grouped with the requirement that, at least some of, these classifiers have names. The italic passage may or may not be a good idea.

3.3 Analysis of Req 3

The text reads:

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

I am not sure how this relates to the Terms and Terminology section of our standards. Would this part be above and beyond? If so, we should take it to heart and define XML stuff in the XML classes; that would provide an example for future writers.

This could be phrased:

Conformant specification documents SHALL, when using new or modified vocabulary in a particular Requirement Class, define that vocabulary or present the rules for interpretation of that vocabulary in the Requirement Class where that vocabulary or usage occurs.

However, the qualifying clause of that sentence brings us new issues. The recommendation at the end of clause 6.5.6 would seem to take this phrasing to push us to make this a separate, and optional, Requirement Class. Perhaps a different phrasing would avoid that difficulty.

3.4 Analysis of Req 4

The text reads:

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

This is straight forwards except that requirements enjoin instances not types.

This could be phrased:

Conformant specification documents SHALL only include formal requirements which enjoin a Target of a single Standardization Target Type.

3.5 Analysis of Req 5

The text reads:

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

Again, this grouping seems to come too early in the list of requirements and we should at least first define the grouping for Requirements into Requirements Classes, then define the grouping of Conformance Tests into Conformance Classes. Finally we can place constraints on the groupings.

This could be phrased:

Conformant specification documents SHALL only contain conformance test classes which group conformance tests that evaluate Target instances of the same Standardization Target Type.

3.6 Analysis of Req 6

The text reads:

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

This is the first formal requirement whose text enjoins the textual structure of the specification document. This type of requirement would seem to merit its own cluster which might even be a formal Module.

The decision to include requirements which enjoin the textual structure raises the question about why the 'assumptions' of clause 6.1 are not simply requirements of this kind. What distinguishes the assumptions from this requirement?

This could be phrased:

Conformant specification documents SHALL structure their text with hierarchical clauses (numbered sections) such that each Requirement Classes is in a separate clause and contains as sub-clauses all its Requirement Modules and those, in turn, contain all their requirements.

It would also be worth considering allowing an 'implicit' Requirements Module for all the requirements classes we are writing which contain only a single module. That is, we would allow Requirement Classes to hold their requirements directly. Either way, this Module system must be clarified because it is not used by the standard itself leaving future authors with no example of the usage of this module level of organization.

3.7 Analysis of Req 7

The text reads:

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

The concept 'requirement structure' is close to meaningless since requirements are atoms. What this is perhaps going for is the 'structuring of the requirements within the document' but then we should really be talking about the groups, the modules and classes, not the atoms.

The phrase 'logical correspondence' is also about a vague as we could get. Is it only talking about matching the hierarchies of each Requirement Class with the hierarchy of its corresponding Conformance Class?

So what this probably means is that the structure of the Requirement Classes and their sub-hierarchies in the core of the document must match the structure of the Conformance Classes and their sub-hierarchies in the separate section containing the Conformance Test Suite, possibly an annex. We must be aware however, that the parallelism breaks down at the level of tests and requirements since they are not one to one.

This could be phrased

Conformant specification documents SHALL structure their text to have the Conformance Test Suite in a separate section which MUST organize its Conformance Classes to follow the organization of the Requirements Classes in the document, and each Conformance Class MUST organize its Conformance Modules to follow the organization of the Requirements Modules in the corresponding Requirements Class in the document.

where this includes assumption 5 as a starting point.

3.8 Analysis of Req 8

The text reads:

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

The one to one part of this should probably be linked with or at least next to the requirement for parallelism of their names. What logical thread gets us to the 'and thus'? We have not seen yet the notion of 'certificates of conformance' nor their link to conformance classes let alone that it is one to one. The UML only mentions certificates as attributes of the Target instance.

This could be phrased

Conformant specification documents SHALL have one conformance test class for each requirements class.

but really this should be establishing the strong link from each conformance test class to a requirements class that the conformance class declares to be its 'injunctive base'.

3.9 Analysis of Req 9

The text reads:

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

This raises an unfortunate trend in the standard to talk of two things that do not exist 'optional requirements' and 'optional tests'. It would seem better to carefully avoid any linguistic construction that uses those terms.

This could be phrased

Conformant specification documents SHALL NOT have any conformance tests which are optional.

followed by explanatory text that shows how optionality is done in conformant specs.

This immediately calls for

Conformant specification documents SHALL NOT have any requirements which are optional.

with a similar explanation.

3.10 Analysis of Req 10

The text reads:

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

As stated above, phrasing this as a character of the document is hard.

Conformant specification documents SHALL require that their conformance testing process only issue certificates of conformance which specify all the parameter values which were used during the application to a particular Target instance of all the tests in the conformance class for which the certificate is being issued.

Surprisingly, this is actually possible reversing my earlier belief that it was not. With this language, we do need to introduce the 'conformance testing process' however. Is this what we should be doing? Ideally we would define the common characters of this 'conformance testing process' (i.e. that it runs all tests in the class or that it work its way through dependencies) so that all conformant specs would not need to.

3.11 Analysis of Req 11

The text reads:

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

Again, the injunction should not just focus on it being a 'single' class but on it being 'the' associated class, so we need a stronger relation. And since conformance classes do not 'test' but only hold tests we need to be more precise.

This might be phrased

Conformant specification documents SHALL only contain Conformance Classes whose Conformance Modules contain only Tests which evaluate conformance of a Target with the requirements of the Requirement Class associated with the Conformance Class containing the tests.

with the 'associated with' remaining a bit vague---we need a name for this association that we can then reference in the requirements.

But wait! Is this **testable**? I suspect it is not. This seems to me to be a 'halting' level problem. We certainly have to say it is our intent but I am not sure we can make it into a testable requirement. How do we determine exactly what a test is testing? Maybe this could be that the tests 'claim to test requirements' and we could use a strong association between each test and the 'requirements whose injunctions the test claims to evaluate.'

3.12 Analysis of Req 12

The text reads:

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

This is clearly a two part requirement.

For the first, how does the specification happen? We probably want to force the Conformance Class to list the name and URI of their dependencies (but the naming requirement only happens below). Again this suggests we should restructure things.

For the second we run into trouble because we have two kinds of dependencies, direct and indirect. The direct dependency is easy, the testing can recurse on the same Target instance; however for the indirect dependency we need to establish the Target of that indirect dependency. For example, if the target were a web service, the direct dependency could be an OWS Common level Conformance Class but the indirect dependency could be towards a data model Conformance Class and what we would want is that the service be tested against the OWS Common level Conformance Class but that the service's output be tested against the data model Conformance Class. Unfortunately we don't know

in general what the role played by the indirect dependency. Do we drop testing of indirect dependencies? (It actually seems like these should be part of specific requirements.)

This could be phrased

Conformant specification documents SHALL only contain Conformance Classes which formally list by name and URI each Conformance Class on which the former Conformance Class depends.

Conformant specification documents SHALL require their testing procedure to include in the evaluation of a Target for conformance with a Conformance Class the evaluation of that same Target for conformance with any directly dependent Conformance Classes.

which only works if this notion of 'direct dependency' makes sense outside the requirement. (As stated elsewhere, the name is unfortunate suggesting the distance of the dependency not that the dependency shares a the same target type.) We might need to make that explanation directly part of the language of the requirement.

3.13 Analysis of Req 13

The text reads:

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

This raises the issue of external dependencies which could be either standards that are conformant or documents which are not.

In the case that the specs that have 'requirements classes' presumably that means that they follow the standard so they do not have any 'requirement {that} is "optional" ' which saves us from the factoring work.

In the case that the imported specs have 'requirement{s that are} "optional",' then we do need to separate out the required from the optional. However, I would not take the approach of this requirement. I suspect the cleanest way to allow for the importing of external specs is to require that they be 'wrapped' in one or more formal Req. Classes. This would formalize both the obligatory and optional injunctions and formalize the tests. This is so preferable to any other mechanism that I would personally require the approach.

I do not attempt to rephrase this requirement. Without clarification on how 'another standard' could have a 'requirements class' I will not attempt to rewrite this. The second sentence is unclear 'each such used requirements class' is the 'imported class'? The

sentence also has "{e}ach such requirements class shall be a conformance class" which is contradictory to the definitions so I don't know how to fix this.

3.14 Analysis of Req 14

The text reads:

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

This could be phrased

Conformant specification documents SHALL explicitly assign a unique name to each requirements class and each conformance test class, with the name of the conformance test class being similar to that of its corresponding requirements class.

but, again, the 'corresponding' should be a stronger language.

3.15 Analysis of Req 15

The text reads:

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

The second sentence is erroneously structured "Inclusion of any requirement in a requirements class {so far so good} **by a conformance class**" makes no sense. It seems what is wanted is that if a **test** (which is 'associated with' a requirement) is included in a conformance class, then all the tests which are associated with all the requirements in the same requirements class as the requirement associated with the first test must be included in the conformance class.

These could be phrased

Conformant specification documents SHALL only have requirements which are contained in one and only one requirements class.

Conformant specification documents SHALL organize their tests into conformance classes in such a way that if a test is included in a conformance class, then all the tests associated with all the requirements in the same requirements class as the original test must be included in the conformance class.

It is a mouthful. Repeating myself the 'associated with' needs to be defined and could use better language. This also suggests that we need a requirement that a test can only 'be associated with' requirements in the same requirements module (class); maybe that exists already, I am getting confused.

3.16 Analysis of Req 16

The text reads:

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

The phrasing of the second sentence introduces something that cannot exist and then applies a procedure to fix things. It is sufficient to prohibit the state. An explanation can be added to the descriptive text showing how to resolve a situation which might arise in developing a standard but is not permitted.

These could be phrased

Conformant specification documents SHALL place in the same requirements class any requirements or requirements modules which are mutually dependent.

Conformant specification documents SHALL NOT have mutually dependent requirements classes. (They logically form a single requirements class).

Can a requirement be mutually dependent with a requirements module? If not, the first of these should be split into two requirements or a compound requirement. Again, the whole requirements *module* system needs explanation.

3.17 Analysis of Req 17

The text reads:

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

This should be dropped as a requirement. First of all it is untestable. How, by "Inspect{ing} the document" could we ascertain this is so? Secondly "natural structure" has no meaning. Thirdly, we go to great lengths to avoid dealing with implementation. Yes, this can be a goal of modularity but that should be stated in the text not become a normative injunction.

3.18 Analysis of Req 18

The text reads:

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

Since the whole goal of this game is modularity and substitutability, it seems simpler to simply ban re-definition of requirements. Anyone needing a requirements class

substantially similar to, but not identical to, an existing requirements class should simply write a new one, include all the identical requirements by reference and add in their own modified requirements. Since the requirements have URIs this cross-referencing should be easy. (Well except that we are banned from doing this by the injunction that each requirement appear in only one requirements class.)

I do not understand the meaning or intent of the second part of the sentence.

This could be phrased

Conformant specification documents SHALL NOT include any requirement which redefines another requirement, but this requirement does not imply that requirements cannot constrain other requirements.

Leading to the difficulty of distinguishing 'redefinition' from 'constraint'.

3.19 Analysis of Req 19

The text reads:

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

Profiles, what are profiles? Profiles are specifications. Specifications have this requirement so why would we need a new requirement targeting profiles?

Not sure how to handle this one. Personally, I think that 'profiles' are the new 'standards'. For me, specs simply define lots of requirements classes; profiles then aggregate a coherent set of requirements classes for a particular purpose.

3.20 Analysis of Req 20

The text reads:

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

Well you can't identify a set of requirements as a conformance class; you could as a requirements class. And this needs to be expanded for the various target types. And actually this is a bad idea: I might have a standard that defines two text data structures that work totally differently (say one is line structured, another structured by markup). Why would we force on others the requirement to have an empty requirements class just to have a 'core'. Having 'core' is a good idea where it makes sense but not something to mandate.

This could be phrased

Conformant specification documents SHALL, for every standardization target type for which the document defines a requirement, define one requirements class

to be the 'core requirements class' for that standardization target type and the conformant specification document SHALL require that every target of that standardization target type conform with the 'core requirements class' for the target to conform to ...?... any requirements class derived from the core? any requirements class enjoining targets of that standardization target type?

I'm not even sure how to complete this language.

3.21 Analysis of Req 21

The text reads:

Req 21 All general recommendations shall be in the core.

This breaks down upon looking up the definition of 'general recommendation': "recommendation applying to all entities in a specification model." "All entities" seems a bit broad: all classifiers in the UML? All targets of a given target type?

So what is the point and why would sensible authors not know where to make their recommendations?

3.22 Analysis of Req 22

The text reads:

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

By 'other requirements class' ('other' to what?) I presume we are talking of 'non-core' classes.

We run into issues with multiple target types again. If we don't mandate core as I suggest we cannot above, we could not mandate this.

This could be phrased

Conformant specification documents SHALL only have requirements classes which are not defined as a 'core requirements class' if the non-core requirements classes have a direct dependency, possibly transitive, on a 'core requirements class'.

Here we see the problem with the language 'direct dependency' which suggests the distance of the dependence not the similarity of target types. Calling them 'base dependencies' or 'root dependencies', as against 'associated dependencies' or something similar, might work better. The sub-type constraint of the requirement in the standard follows naturally from the definition of direct dependency.

3.23 Analysis of Req 23

The text reads:

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

Wow. This requirement language is well structured. It would be perfect, except that we need to allow for multiple target types, and the 'shall consist of' is problematic. Oh and the standard will have a whole lot more than just requirements classes.

This could be phrased

Conformant specification documents SHALL only contain requirements classes which are either a 'core requirements class' or are extensions, possibly transitively, to a 'core requirements class'.

But at this point, the requirement is highly similar to the one above so they might be merged. (And I am still against this concept of 'core'---I think we give authors tools to write good specs not injunctions to make them write them in our particular way).

3.24 Analysis of Req 24

The text reads:

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

Ha! This recursion sucks. This was the first glimpse I had that WMS was going to have to define a second target type, a 'specification document extending this WMS specification'. Why? Extensions are either conformant to the standard or not; the work of extension is no business of a conformant specification.

The phrasing is pretty good except for the "to itself" which is not unambiguous: in French it would refer to 'this standard' in English usually to the conformant spec. (Reading anglo-saxon possessives is delicate work---in the documentation of the GNOME project we highly discourage their use since our translators and readers are world wide.)

3.25 Analysis of Req 25

The text reads:

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

Well structured language but conflicts with the requirement immediately before it, req. 24. Of questionable utility. The 'in any manner' seems vague enough to be untestable. Since the requirements are what make normative injunctions, this should probably be phrased as a constraint on requirements (that must not restrict) rather than on standards.

3.26 Analysis of Req 26

The text reads:

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

Here are the non-existent 'optional requirements' again; no requirements are optional. What would an example be of 'an optional list of conformance classes'? This seems to be a natural result of banning the optionality of requirements rather than a requirement in its own right.

3.27 Analysis of Req 27

The text reads:

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

I'm not sure how references are made so I can not evaluate this effectively. Requirements classes have many "common portions" such as definitions and target types. If the 'references' were made in some normative statement then those normative statements could be shared, possibly by reference. Absent that, references would always be made anew and so we are not talking about 'common' elements but merely 'parallel' elements and we get into trouble.

That takes care of the requirements in the 'core requirements class' of the standard. At this point, I am giving up on this work pending feedback from the authors. I leave the text below in case we eventually build on this particular document.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. The UML model

The UML model could use a lot of work to improve it but the details of that will need to await feedback from the authors on the issues raised in this change request.

There are two errors in the diagram.

The type of the 'reference' element in the ConformanceTest element is 'boolean' but in C.11 it is 'String' which is probably the type required if it is to hold a URI.

The multiplicity of the 'module' relation for the RequirementsClass element is missing but should be '1..' to allow for several modules in a class.

Some of the names of the relations seem problematic, e.g. the 'requirements' relation of the ConformanceClass element should be 'requirementsClass'.

Many new elements could be added such as 'certificate' and many of the relations could be qualified, e.g. all the 'requirements' in the relation from the ConformanceTest.

5. Editorial comments

In general there are some issues with vocabulary and terms. The pluralization of the descriptive adjective in *Requirements Class* seems odd and lacks parallelism with Conformance Class.

5.1 p.0 Editor:

Is there really a Policy 1.x SWG as exists on the twiki or is this actually a product of the OAB / TC as a whole?

5.2 p0 Title:

The title, as explained above, misses out on what the standard is doing. The model is only part of what it's doing, modularity is only one aspect of better specs.

Constraints on the title mean that it not only needs to reflect the contents of the standard document but needs to be easily expressible in speech ('the mod spec' is current) and in text where *The Specification Model* leaves something to be desired. Also, since 'standard' is defined to be a 'spec' that has been approved, I would imagine it is bad policy to have documents calling themselves 'standards'.

Possible improvements include *The OGC Specification Specification --- A framework for rigorous, testable, and modular specifications*. It could be called the 'Spec Spec' and the main title would make for easy reading when referenced in text. Alternatives for the main title could be *The OGC Specification Standard*, with only a slight ambiguity of what noun the OGC adjective attaches to, or *The OGC Specification for Specifications*. The 'framework' could be replaced with 'approach' or 'rule set' or 'model, requirements, and tests'.

5.3 p xi Preface

In first sentence, drop "for writing standards to be used".

5.4 p xi Document terms ...

When referencing other documents, include their name, in this case OWS Common, so that a reader does not have to stop and look up the reference.

5.5 p xiii Foreword

First paragraph: s/principals/principles

Second paragraph: drop the injunction to specify the source. We don't care. Serious people will. It is not worth saying. And if we put OGC in the title, anyone referencing the document will be acknowledging the source.

Paragraph 3 "structures of other *specifications*" and throughout the document. The target is 'specification documents;' that some become standards is irrelevant.

Paragraph 3 drop "that would normally claim conformance" it is not needed and confuses things. possibly bring in the from the first sentence in the Preface, "for software, services, or data structures."

Paragraphs 4 and 5. Drop. This belongs in a separate document, say the P&P. Anyhow, it was probably only needed early in the lifetime of the document before everyone heard about 'the modular spec'.

Paragraph 7 (last) also in Template. This needs a contact info or web page so people know how to submit stuff.

5.6 p xiv Introduction

line 3 s/standard/specification

add to list 'read and understood' to the list. A spec. is useful if people can quickly decide that it is **not** relevant to their work.

first paragraph, last sentence. It might make more sense, but keep the play of the paragraph by integrating the capitalization stressed in the footnote, to end with '...conductive to being a Really Useful Specification.'

The first sentence of the last paragraph belongs with the preceding paragraph.

The remainder of the last paragraph seems to be about "This standard" If so, this should be a quick summary of the internal model of the spec. rules for targets, tests, a testing procedure, and certificates. all split into modules.

The note ends with "and its implementation" which seems to enter territory that our specs are not allowed to tread. Implementations details are not our purview.

5.7 Scope p1

This section lacks clarity. The text "specifies some desirable" mixes two ideas 'specify' (which implies 'required') and 'desirable' which is optional. "implementation structure" is out of scope for our document---whether implementations do or do not reflects all sorts of engineering decisions outside our knowledge.

The model is made to seem normative but then the UML diagram and text are all in the informative annex: confusing.

A better intro might be along the lines of:

This specification aims to improve the quality and utility of specification documents both by developing set of requirements for such documents that improve their formalism and structure and by coupling the requirements to testable procedures which can determine that the documents conform to, or at least do not deviate from, the requirements.

The requirements of this specification ensure the normative text of those documents is identifiable, that the normative text uses language which follows the formalism of clearly identifying the entity enjoined by the language, and that the normative text only contains testable injunctions since the tests are developed along with the normative text.

The requirements of this specification also ensure that conformant specification documents could be developed as modules, possibly dependent on other similar modules. The goal of modularization is to facilitate coherency and reuse between specifications thereby enabling the construction of a coherent system of interdependent specifications without getting lost in the complexity.

This specification develops its requirements first by constructing a conceptual model of specification documents and then by making its injunctions against the components of the conceptual model. The conceptual model is presented graphically in the diagram [ref] and verbally in [ref]. This specification also develops rules for a testing procedure based on the inspection of a candidate specification document. This specification also tackles injunctions for the use of the Unified Modeling Language (UML) (?classifer?) diagrams and for the use of schema definition and constraint languages.

5.8 Conformance p1

Drop the first paragraph, it should be in OGC TC internal rules.

The paragraph listing target types appears to claim to be listing conformance classes: "There are 5 conformance classes: ...Specifications documents in general..." Probably each entry should be in the form "<conf.cl. name>, which targets/enjoins ..."

5.9 Normative References pp2-3

Move ref [7] to end since we discuss XML schema before schematron.

5.10 Terms and definitions pp3-8

certificates are not awarded per standard but per conformance class so this definition seems wrong.

conformance test case I suspect this multiplicity is problematic; probably it would make more sense to have each test focus on one requirement and have the testing of multiple requirements emerge only from the language of some requirement, either one of the original ones or a new one.

conformance test class Drop the 'level' I have not seen it elsewhere. Does the defn. need to address parameterization of the testing? The note discusses "optional requirements" which are defined not to exist; optionality must be discussed with different language.

conformance suite "It is by definition the conformance class of the entire specification" is false since a spec can have several target types and therefore several, separate root conf.classes.

conformance test what is an "abstract" test? The tests merely describe the procedure by which to test something.

core requirement class "unique *per target type*" "if it were possible"

direct dependency/indirect dependency the language of the definitions must be aligned. the names should become something more significant say 'underlying' vs. 'related'.

extension Here 'another' is used differently than it was in the two previous definitions; the usage should be aligned.

general recommendation What is an 'entity' of a specification model?

home The term is poor. It sounds like a work around for needing to discuss the 'injunction' throughout the text which allows us to reserve *Requirement* for the statement itself (as proxy for the more complete 'normative statement of the requirement'). The definition makes 'home' which is a place into a statement---that's a bad idea: the term must at least be 'home statement' or some such.

leaf package What's a package, what's a classifier? How much of this are we going to define? the [UML] notation is confusing, it should be "(see [UML])". (this should also be done for all other references.)

model "something, with" => "some thing as a"

profile I would rewrite this as a spec that defines itself primarily by reuse of another, selecting optional choices and adding constraints making corrections or clarifications and adding obligations.

drop "or standard"

"base standard and/or other profiles"=>"dependent specification"

requirement we have to decide which form of this word we are defining, whether a generic normative injunction in any standard, the *Requirement* classifier with name, URI, statement, ... in conformant specs or just the normative injunction in a *Requirement* classifier. It seems that this is closer to the latter; if so, it needs 'testable' 'against a target instance' 'coupled to tests'

specification similarly which one are we defining? In note, s/standard/specification

standard What do we do about de-facto standards, say GeoTIFF which has never been formalized? (well probably we should do the formalization work but that's a separate question).

standardization target needs to include 'instance' add examples servers, clients, specs, data structures...

standardization target type s/type/kind (don't reuse the word in its definition) Drop 'or set of entities'? I am not sure what is added by the 'set' or maybe this is a transitive 'kind of...set'. add examples Is there really a 'Open Web Service' type or do we just wish there were one? A 'GML application schema' is **not** "a specification" (sensu above)---it is not a 'document containing recommendations ...'

5.11 5.3 p8

The "'bold font" home' in the text should not be in red or bold, it distracts from the examples.

"in this sense, all requirements are listed" but it's exactly NOT in that sense. home is in the requirement class (as per the font), the listing is for convenience.

5.12 Conventions

add a section for the normative language SHALL, etc. taken from Clause 6.1, description 4, p10.

(My editorial work stops here. For the rest of the document, more fundamental issues like organization and the requirements must be tackled before we worry about phrasing.

6. A general outline of a new text

This needs more work. The existing text which provides context and transitions between requirements should be reviewed and allocated effectively into this structure. Leaving for now until after feedback on what has been done so far.

General:

- make all req.cl. top level---they are few.
- the decision in par 2 of 6.2 to avoid the complexity of having several target types in a standard is a mistake, it makes the standard easier to write but much harder to read and interpret.

Clause 5 and before: existing structure is OK.

Clause 6: Core rules for specifications

On the generic nature of specs---§6.2, ¶1

Goal: develop good specs, build model, a

Process: the doc tackles first making good testable injunctions against well defined targets, then deals with the modularity stuff.

The model of specifications: Target, Requirement, Tests, Eval.Process, Cert.

Requirements (following the new organization below)

Clause 7: Rules for specifications using UML (?to model their targets?)

Clause 8: Rules for specifications using XML Schema (?to model data structures?)

Clause 9: Rules for specifications using XML Schematron (for ...?)

7. A new set of requirements

This clause presents the injunctions from the *Requirements* and from the UML of the standard reworked into a more systematic presentation. This approach starts with requirements for the document itself, then requires a list of *Standardization Target Types*, then presents the requirements for *Requirements*, *Recommendations*, and *Tests*, then turns to rules for grouping into *Requirements Classes* and *Modules* and into *Conformance Classes* and *Modules*, and finally describes the conformance testing process which conformant specification documents must declare. The language of these requirements follows the recommendations presented below in being: <Target> <Verb> <Rule>.

There are unresolved issues in this writeup:

- Do we need to enjoin that the structuring of a *Conformance Class* into its *Conformance Modules* necessarily parallel the structuring of the associated *Requirements Class* into its *Requirements Modules*?
- Do we really mandate the existence of a 'core' or does this fall out naturally from allowing dependence and excluding circularity in the dependencies? It seems that standards will naturally create useful 'cores' but mandating this seems silly.
- How do we handle the requirement on the placement of vocabularies of req. 3 in the standard? Req. 3 enjoins the position of specific vocabulary but this would need fleshing out with comparison to the general practice of having that in Clause 4 of the document so this would need clarification.
- Several of the requirements of the standard are not included because they were considered untestable or to be bad ideas.

There may be some way to do facilitate life for conformant specifications by enabling them to satisfy easily the Testing Process requirements. Ideally, those documents would be able to say "my testing process should work the way of that of the Spec Spec's" thereby avoiding them the work of rewriting similar rules each time. I suppose that would come not from the other documents referencing the section in the new Spec. Spec. declaring the requirements but from referencing the section in the new Spec.Spec. declaring the rules for its own Testing Process.

This clause will need more work if it is deemed by the authors to be a way to make progress on the text of the standard. To be completed, first this list of Requirements and Tests must be reviewed to ensure these Requirements systematically describe the elements and relations of the UML model in the standard (C.2). Next this list would need review to ensure it includes all the good requirements from the existing standard. Also, the language of each requirement would need very careful review to ensure correctness. Finally, this writeup will have to be reformatted and names, URIs and lists added so as to comply with its own rules.

Terminology:

- injunctive statement --- is in a Requirement
- suggestive statement --- is in a Recommendation
- evaluation procedure --- is in a Test
- evaluation process --- is the conformance testing mechanism

- clause --- any hierarchical element in a document
- top-level clause --- (like #7 above)(a clause without parent clause)
- sub-clause --- any clause which is not top-level
- parent-clause --- the clause directly containing a sub-clause

- dependent Requirements Class --- required and with same Target type (was 'direct dependent')
- associated Requirements Class --- one used by parts of another, with diff. type (was indirect dependent)

Specification documents

Conformant specification documents SHALL **be text documents** (possibly with associated normative resources) whose content includes a title page and is structured into numbered clauses, some of which are internally structured hierarchically into sub-clauses.

{The rules for the association of normative resources will be developed below.}

{The requirement that some clauses will have an internal hierarchy must be fulfilled since clauses with Req.Cl. must be so structured and one must be present.}

=> is the spec doc a text document?

=> does the spec doc have a title page?

=> is the content of the spec doc organized into numbered clauses?

=> are any of those clauses internally structured into hierarchical sub-clauses?

Conformant specification documents SHALL provide, preferably on or close to their title page, the **name** of the specification, the name of and contact information for the **principal**, that is the author, editor, or responsible party, the **copyright holder** of the specification and the **date** of the document.

{TODO: Look at Dublin Core for what fields are required so others can catalog the resource or name the spec doc by reference. Also, where is the URI for the document itself to be declared?}

(Fuse with previous: this from the UML model)

Conformant specification documents SHALL state a name, authority (a principal, with a name and contact info), date, status, list of dependent specs, clause with test suite, and list of requirement classes.

=> Inspect the document, see that each of these is present and can be readily identified for what it is.

Conformant specification documents SHALL make all of their normative statements, that is their *Requirements* and *Recommendations* (described below), as **visually distinctive blocks** of text in the contents of the document.

{TODO: should this be part directly of the Req. and Rec. set of requirements rather than here? (see the language of the Test req.)}

=> find all of the normative statements, determine that each is visually distinctive.

=> inspect the rest of the document, ensure that it does not contain any normative statements or normative language (SHALL, ...)

=> find all normative resources (diagrams, annexes, schemas), establish that it is declared as normative by one of the normative statements.

Model Entities

Standardization Target Types

Conformant specification documents SHALL provide a **list of all the *Standardization Target Types*** for which the doc makes normative statements and SHALL assign to **each *Standardization Target Type* a name label, a URI, and a description** explaining the type and its possible instances.

{A Target instance will also need a name since that will be used in any Requirement written enjoining the Target. Do we need to formalize that?}

=> does the doc have a list of Standardization Target Types?

=> does each STT in the list have a name label, a URI and a description?

(We don't care that each of these have a Req.Cl. targeting it since a spec, say for an OWS, may wish to declare extra targets, say a client, for use by extension specs and profiles.)

(We test later that all the Requirements target only one of these.)

Requirements

Conformant specification documents SHALL make their **injunctive statements** only in the text of a normative statement which is a *Requirement* because it is labeled "Requirement".

=> find all of the normative statements (established to be visually distinctive by the test above), determine that all injunctive statements (those using SHALL...) are made only in normative statements labeled 'Requirement'.

Conformant specification documents SHALL only contain *Requirements* which have a **name** label, a **URI**, (a summary,) and a block of **injunctive text**.

=> find all normative statements labeled 'Requirement', determine that each has a name label, a URI, (a summary,) and injunctive text.

Conformant specification documents SHALL only contain *Requirements* whose injunctive text **uses the normative language** (SHALL, ...)

=> find all normative statements labeled 'Requirement', check that the verb of the statement is one allowed in the language rules.

Conformant specification documents SHALL only contain *Requirements* which **present the URI of a *Standardization Target Type*** and whose **injunctive text has as the subject a *Target*** which is an instance of the *Standardization Target Type* whose URI is presented.

=> find all normative statements labeled 'Requirement', determine that each has a URI for a *Standardization Target Type* and has normative text that clearly targets an instance of the STT. (*Note that due to the grouping rules for Requirements described below the URI of the STT may be assigned to the group as a whole and therefore might not be presented directly within the text of each Req.*)

Conformant specification documents SHALL only contain *Requirements* whose **Target is an instance of one of the *Standardization Target Types* listed** by the document in the list of *Standardization Target Types*.

=> find the list of STTs targeted by the document and find all normative statements labeled 'Requirement', determine the Target of each *Requirement*, check those Targets are instances of one of the STT listed in the list.

(The summary is very useful for readers when they encounter a reference. "shall follow req. 10" does nothing for a reader whereas "shall follow req. 10 *All requirements have a URI* " makes for easy reading.)

Conformant specification documents SHALL only contain ***Requirements* which have a list of the *Tests*** evaluating conformance of *Targets* with the injunction of the *Requirement*.

{This is from the UML model. However, is this needed since navigation from Tests to Requirement is possible? Two way navigation seems redundant and is not needed further by these requirements.}

=> TODO

Conformant specification documents SHOULD only contain *Requirements* whose injunctive text follows the pattern:

"<target> INJUNCTIVE_VERB (optional_qualifier) <injunction>"

Recommendations

Conformant specification documents SHALL make their **suggestive normative statements** only in a text statement which is a *Recommendation* because it is labeled "Recommendation".

=> find all of the normative statements (established to be visually distinctive by the test above), determine that injunctive statements (those using SHOULD...) are made only in normative statements labeled 'Recommendation'.

Conformant specification documents SHALL only contain *Recommendations* which **have a name label and a block of suggestive text**.

=> find all normative statements labeled 'Recommendation,' determine that each has a name label and a block of normative text making a suggestion

Conformant specification documents SHALL only contain *Recommendations* whose block of **suggestive text identifies the *Target*** to which the suggestion applies.

=> find all normative statements labeled 'Recommendation,' determine that each has a block suggestive text that identifies as the subject a *Target*, an instance of a *Standardization Target Type*.

Conformant specification documents SHALL only contain *Recommendations* whose ***Target* is an instance of one of the *Standardization Target Types* listed** by the document in the list of *Standardization Target Types*.

=> find the list of *Standardization Target Types* in the document, find all normative statements labeled 'Recommendation,' for each 'Recommendation' find the *Target* in the block of suggestive text, determine that the *Target* is an instance of one of the *Standardization Target Types* listed by the document.

Conformant specification documents SHALL only contain *Recommendations* whose **language uses as normative text (SHOULD...)**

=> find all normative statements labeled 'Recommendation,' determine that each has a block suggestive text that uses as its verb one of the suggestive verbs from the allowed language (SHOULD, MAY...).

Conformant specification documents SHOULD only contain *Requirements* whose injunctive text follows the pattern:

"<target> SUGGESTIVE_VERB (optional_qualifier) <injunction>"

Tests

Conformant specification documents SHALL only **describe a testing procedure**, that is a procedure evaluating that a *Target* of one or more *Requirements* follows the injunctions described in the injunctive text of those *Requirements*, in the text of a visually separated statement which is a *Test* and which is labeled "Test".

=> inspect the document, see that the only place that there is text describing a testing procedure is within the body of a visually separated statement which is labeled 'Test'.

Conformant specification documents SHALL only contain **Tests which have a name label, a URI, (a summary,) and a block of text** describing the testing procedure.

=> find all the Tests, see that each has a name label, URI, (summary,) and testing procedure.

Conformant specification documents SHALL only contain *Tests* which **have a testPropose, testMethod, testType** (an enum (extensible?) basic|capabilities).

=> find all the Tests, see that each has a testPurpose, testMethod, testType, (and type is in the enum/code list)

Conformant specification documents SHALL only contain *Tests* which **list the URI(s) of the Requirement(s)** whose injunctions are evaluated by the testing procedure.

=> find all the *Tests*, see that each has the list of the URIs of the *Requirements* which are evaluated by the testing procedure.

Conformant specification documents SHOULD devise a sufficiently comprehensive *Test* or suite of *Tests* to evaluate as completely as possible that the injunction made by each *Requirement* has been respected by a tested *Target*.

{Note that it is not always possible to affirm that a Target follows the injunction of a Requirement; sometimes it is only possible to have tests which show that a Target fails to follow a given injunction. In those cases, many evaluations must be attempted before conformance can be presumed.}

Conformant specification documents SHALL have at least one *Test* for every *Requirement*.

=> find all the *Requirements* in the document, find all the *Tests* in the document, then determine that each *Requirement* has at least one *Test* in the set of *Tests* with a URI pointing to the *Requirement*. (Note that later *Requirements* make this simpler than stated here due to the injunction mandating parallelism in the grouping, we only need to seek *Tests* in the Conformance Class matched to the *Requirements Class* containing the *Requirement*.)

Groups of Requirements

Conformant specification documents SHALL declare one clause or more to be a *Requirements Class* because it is labeled "Requirements Class".

{Note, 'clause' could be at any level, a top-level clause or a sub-clause.}

*{Note, the pluralization of the qualifier "Requirement*s*" in*

'Requirements class' seems strange in comparison to 'Conformance class'}

=> inspect the document, see that it has at least one clause labeled "Requirements Class."

Conformant specification documents MAY declare one or more clause of a document to be a *Requirements Module* because the clause is labeled "Requirements Module" but each clause so labeled SHALL be a (?direct?) sub-clause of a *Requirements Class*.

=> inspect the document to find all clauses labeled "Requirements Module", determine that all such clauses are contained in a *Requirements Class*.

Conformant specification documents SHALL group their *Requirements* into the *Requirements Classes* of the document in such a way that, if a *Requirements Class* contains one or more *Requirements Modules*, then all the *Requirements* of that *Requirements Class* are in its *Requirements Modules*.

{Note that the Requirements Classes which contain their Requirements directly will be said to have an "implicit Requirements Module".}

=> find all the *Requirements*, find all the *Requirements Classes* determine that all *Requirements* are in a *Requirements Class* clause, possibly in a sub-clause.

=> find all the *Requirements*, find all the *Requirements Classes* and all the *Requirements Modules*, for each *Requirements Class* that contains *Requirements Modules* determine that all *Requirements* are actually in one of the *Requirements Module*.

Conformant specification documents SHALL have all the *Requirements* in a *Requirements Module*, declared or implicit, enjoin *Targets* of the same *Standardization Target Type*.

=> find all the declared *Requirements Modules* and find all the implicit *Requirements Modules* from *Requirements Classes* which do not have any declared *Requirements Modules*, for each *Requirement Module* find all its *Requirements* and determine that the *Target* of each *Requirement* is an instance of the same *Standardization Target Type* as that of all the other *Requirements* in the same *Requirements Module*.

Conformant specification documents SHALL have *Requirements Classes* that have a distinctive name, a URI, and a list either of the *Requirements Modules* it contains, if it has any, or of the *Requirements* which it contains.

=> find all the *Requirements Classes*, for each determine that it has a name which is distinct from the names of all the other *Requirements Classes*, that it has a URI and that it contains a list of *Requirements Modules* if it contains any or a list of *Requirements*.

Conformant specification documents SHALL have *Requirements Modules* that have a distinctive name, a URI, and a list of the contained *Requirements*.

=> find all the *Requirements Modules*, for each determine that it has a name which is distinct from the names of all the other *Requirements Modules*, that it has a URI and that it contains a list of *Requirements*.

Conformant specification documents SHALL declare one clause, possibly in an Annex, to be the *Conformance Test Suite* because it is labeled "Conformance Test Suite" and contains in its clauses all of the *Tests*.

{Do we associate this via a requirement instead?}

=> inspect the document to find a clause with that label

=> determine that all the *Tests* are in sub-clauses within the clause

Conformant specification documents SHALL associate with each *Requirements Class* a sub-clause of the *Conformance Test Suite* which is a *Conformance Class* because it is labeled "Conformance Class" and because it has a URI for the *Requirements Class* associated with the *Conformance Class*.

=> Find all *Requirements Classes* and all *Conformance Classes*, for each *Requirements Class* find the URI of the class determine that there is a *Conformance Class* which lists as the same URI as that of the associated *Requirements Class*.

Conformant specification documents MAY declare one or more clause of a document to be a *Conformance Module* because the clause is labeled "Conformance Module" but each clause so labeled SHALL be a (?direct?) sub-clause of a *Conformance Class*.

=> inspect the document to find all clauses labeled "Conformance Module", determine that all such clauses are contained in a *Conformance Class*.

Conformant specification documents SHALL have each *Conformance Class* declare a name for the class similar to the name of the associated *Requirements Class*, declare a URI for the *Conformance Class*, and provide a list either giving the name and URI of every *Conformance Module* in the *Conformance Class* if there are any or giving the name and URI of every *Test* in the *Conformance Class*.

{A Conformance Class which contains Tests directly will be said to have an "implicit Conformance Module."}

=> find every Conformance Class, determine that each has a name, a URI, and a list either of Conformance Modules or of Tests.

Conformant specification documents SHALL have each explicit *Conformance Module* declare a name for the module, a URI for the module, and a list giving the name and URI of every *Test* in the module.

=> find every Conformance Module, determine that each has a name, a URI, and a list of the name and URI of every Test.

Conformant specification documents SHALL group in the same *Conformance Class* all of the *Tests* which evaluate the conformance of a *Target* with the *Requirements* listed in the *Requirements Class* associated with the *Conformance Class*.

{Phrasing needs work.}

{This will inherently guarantee that all the tests target the same Std. Target Type.}

=>

Groups of Tests

Conformant specification documents SHALL declare one clause, preferably an Annex, to be the **Conformance Test Suite** because it is labeled "Conformance Test Suite".

=> inspect the document, see that it has one clause labeled "Conformance Test Suite."

Conformant specification documents SHALL declare for every *Requirements Class* of the document a **Conformance Class** in the *Conformance Test Suite* which is labeled "Conformance Class" and which has as its name a name similar to the name of the *Requirements Class* with which it is associated.

{The term 'associated' is weak.}

=> find all *Requirements Classes* and their names, find the *Conformance Test Suite*, for each *Requirements Class* determine that the *Conformance Test Suite* contains a *Conformance Class* whose name is similar to that of the *Requirements Class*.

Conformant specification documents MAY declare one or more clause of a document to be a **Conformance Module** because the clause is labeled "Conformance Module" but each clause so labeled SHALL be a (*?direct?*) sub-clause of a *Conformance Class*.

{I do not assume that Conformance Modules need to parallel Requirements Modules but this might be desired.}

=> inspect the document to find all clauses labeled "Conformance Module", determine that all such clauses are contained in a *Conformance Class*.

Conformant specification documents SHALL **group their Tests into the Conformance Classes** of the document in such a way that, if a *Conformance Class* contains one or more *Conformance Modules*, then all the *Tests* of that *Conformance Class* are in its *Conformance Modules*.

{Note that the Conformance Classes which contain their Tests directly will be said to have an "implicit Conformance Module".}

=> find all the *Tests*, find all the *Conformance Classes*, determine that all *Tests* are in a *Conformance Class* clause, possibly in a sub-clause.

=> find all the *Tests*, find all the *Conformance Classes* and all the *Conformance Modules*, determine that all *Tests* which are in a *Conformance Class* that contains *Conformance Modules* are actually in one of the *Conformance Module*.

Conformant specification documents SHALL have **in the same Conformance Class all the Tests** evaluating the *Requirements* of the *Requirements Class* associated with the *Conformance Class*.

=> find all the *Tests*, for each find all the *Requirements* for which the *Test* lists the URI (as that of a tested *Requirement*) and determine the *Requirements Class* in which those *Requirements* are contained, then determine that all the *Tests* which evaluate conformance with a *Requirement* from each *Requirements Class* are all contained in the same *Conformance Class* which is the *Conformance Class* associated with the *Requirements Class*.

Dependencies and Associations between Classes

Terminology: To work around the confusing 'direct' vs 'indirect' dependency language of The Spec.Model document, I refer to 'dependency'/'requisite' (was 'direct dependency') versus 'association' (was 'indirect dependency').

Conformant specification documents MAY include normative resources other than the text statement of a *Requirement* or *Recommendation*, such as formal schemas for structured documents or graphical diagrams of entities and relations, but such resources SHALL only be included through the formal text of a *Requirement*.

{Check 'text statement' language compared to what it is called in the Req. and the Rec. sections.}

{Does this mean the Test Suite Annex must be included normatively by a requirement?}

=> inspect the document to find all the resources included as normative by the document, determine that each is declared to be normative through the text statement of a *Requirement*.

Conformant specification documents MAY include *Requirements Classes* which declare a formal dependency on one or more other *Requirements Classes* which share the same *Standardization Target Type* by including a list which is labeled "Requisite Requirements Classes" and contains the name and the URI of each prerequisite *Requirements Class*.

Conformant specification documents SHALL NOT allow circularity in the dependencies between *Requirement Classes*.

Conformant specification documents SHALL have, if the document includes a *Requirements Class* which declares a formal dependency on another, the *Conformance Class* associated with each such *Requirement Class* include a list labeled "Requisite Conformance Classes" which lists the name and URI of the *Conformance Class* associated with each of the prerequisite *Requirement Classes*.

Conformant specification documents SHALL require that, if the document includes a *Requirements Class* which declares a formal dependency on another, the *Testing Procedure* testing a *Target* for conformance with such a *Requirement Class* ascertain the conformance of the *Target* with each prerequisite *Requirement Class*, possibly by running the *Tests* of the *Conformance Class* associated with the prerequisite *Requirement Class*.

=> Tests: TODO

Conformant specification documents MAY include *Requirement Classes* which declare a formal association with one or more other *Requirement Classes* which have a different *Standardization Target Type* by including a list which is labeled "Associated Requirement Classes" and contains the name and the URI of each

associated *Requirement Class*. Any document including such a *Requirement Class* SHALL have the *Conformance Class* associated with each such *Requirement Class* ... {? parameterized testing?}

{This is in support of the 'parameterization of Tests', for example for an image service req.cl. that declares an association on several image req.cl. and the test of the service returning a valid image would include ensuring that the image returned is valid against the requirements of the image req.cl.}

{How do we test this?}

Conformant specification documents SHALL NOT include any *Requirement* which redefines a *Requirement* in a prerequisite or associated *Requirement Class*, although a *Requirement* may further restrict or constrain an earlier *Requirement*.

{Does this imply that a Requirement can depend on another?}

=> ? Check the language of all requirements?

=> ?find Req.Cl. with prerequisite or associated Req.Cl, check each req. for its language not redefining the language in any of the req. in the prereq. or assoc. Req.Cl.

Conformant specification documents MAY include external specification documents as normative elements of a *Requirements Class* but such specification documents SHALL only be included through the formal text of a *Requirement*.

{Should these Req. only be allowed in a separate Req.Cl.??}

Conformant specification documents SHALL include, for any external specification document declared as normative by a *Requirement*, the title of the document, the author or editor of the document, the publishing organization adopting the specification as standard (if any), and a URL if the document is posted as an Internet accessible resource.

Conformant specification documents SHALL specify if any of the features which are optional in an external specification are considered mandatory by the *Requirement* including the external specification and conformant specification document SHALL NOT include any *Tests* related to the including *Requirement* which evaluates conformance with an optional feature of the external standard which has not explicitly been made mandatory by the including *Requirement*.

=> inspect the document to find any external specification included as normative, determine that each is declared to be normative through the text statement of a *Requirement*.

=> find any *Requirement* including an external specification, determine that it lists the spec title, author, publisher/standardizing organization, and URL.

=> find all *Requirements* that refer to external specifications, find any language of those *Requirements* that make mandatory elements which are optional in the external standard, find all the elements

which are optional in the external specification, determine that none of the optional elements of the external standards are made obligatory by a *Requirement* unless the *Requirement* makes that obligation explicit.

=> find all *Requirements* that refer to external specifications, find any language of those *Requirements* that make mandatory elements which are optional in the external standard, find all the elements which are optional in the external specification, find all the *Tests* in the *Conformance Class* associated with the *Requirements Class*, determine that none of the *Tests* evaluate conformance of a *Target* with an optional feature of the external standard unless the *Requirement* makes that obligation explicit.

Testing

Conformant specification documents SHALL declare (?in a separate clause?) a *Testing Process* for each *Standardization Target Type* for which the document make formal *Requirements*.

{I am not sure we want to call it "Testing Process."}

=> ?how do we test? do we look for a label "Testing Procedure"?

Conformant specification documents SHALL declare that the *Testing Process* evaluates candidate *Targets* at the level of the *Conformance Class* by establishing that the *Target* passes all of the *Tests* of the *Conformance Class* and of any dependent *Conformance Class*, such that either the *Testing Process* declares that the *Target* passes a *Conformance Class* (and therefore that the *Target* "conforms with" the *Requirements Class* associated with the *Conformance Class*) or the *Testing Process* issues no declaration.

{We should note that this holds direct implications for how optionality is handled in conformant specs; only Req.Cl. can be optional not Req. not Tests.}

=> inspect the document, find where the *Testing Process* is declared, determine the document enjoins this process to only issue declaration of conformance at the level of the *Conformance Class*.

Conformant specification documents SHALL declare that the *Testing Process* issue the declaration that a *Target* passes a *Conformance Class* using a "Certificate of Conformance" in which is present the name and URI of the *Requirements Class* associated with the *Conformance Class* which the *Target* passed.

{The URI of the Req.Cl. seems better than that of the Conf.Cl.}

=> inspect the document, find where the *Testing Process* is declared, determine that the document enjoin this process to only issue "Certificates of Conformance" in which the name and URI of the *Requirements Class* tested is listed

Conformant specification documents SHALL declare that the *Testing Process*, when testing a *Target* using *Tests* which require an external resource of possibly different kinds, qualify the name and URI of the *Requirements Class* listed on the "Certificate of Conformance" using a parameter value describing the type of resource used in any *Test* which was part of the *Conformance Class* evaluated.

{This wording needs work.}

=> inspect the document, find where the *Testing Process* is declared, determine that the document enjoin this process to add a parameter value to the name and URI of the *Requirements Class* tested which lists the kind of resource used for that external dependency

{Move the testing of prerequisite and associated Req.Cl. here?}

Rules for OGC Specifications

These should be part of the OGC Policies and Procedures for Specification Documents.

Valid OGC specifications SHALL be documents which follow the requirements and recommendations of the OGC Document Template and associated recommendation.

{And is in ISO Form (e.g. Clause 4 defines vocab.)}

Valid OGC specifications SHALL be documents which conform with the requirements of the Requirement Class "Well-formed Documents" ([http-uri](#)) in "The OGC Spec. Standard".

=> Check the OGC Specification with the tests of the Conformance Class "Well-formed documents" ([http-uri](#)) of "The OGC Spec. Standard."

Valid OGC specifications SHALL only use for document elements URIs which are issued by the OGC-NA.

=> Inspect the document, find all URIs, determine that each of the URIs is issued by the OGC-NA. (is there a list somewhere?)