All Fields marked with * are mandatory.

Change Request #:	**204**
Assigned OGC Document #:	**12-017**
Name:	*Adrian Custer
Organization:	*independent
Email:	*ac@pocz.org
Document Name/Version:	*The Specification Model --- A Standard for Modular specifications / 1.0.1
OGC Project Document:	*08-131r3

If this is a revision of a previous submission and you have a Change Request Number, then check here: ☐

Enter the CR number here:

Enter the Revsion Number that you are revising here:

Title:	*Editorial rewrite of The Specification Model -- A Standard for Modular specifications
Source:	*Individual Member Adrian Custer
Work item code:	
Category:	* D (Editorial modification)

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
Clauses affected:	* All
Additional Documents affected:	None
Supporting Documentation:	The text of the change request as a zip archive containing two versions, one in PDF and another in Word format.
Comments:	This change request is sufficiently extensive that it would warrant several working meetings when the authors are ready to tackle the work.
Status:	Assigned
Assigned To:	Policy 1.0 SWG (archived)
Disposition:	Referred and Posted

Open Geospatial Consortium

Date: 2012-02-14

External identifier of this OGC® document: <>

Internal reference number of this OGC® document: YY-nnnrx

Version: 0.1

Category: OGC® Change Request

Editor: Adrian Custer

Change Request for

The Specification Model (08-131r3)

Table of Contents

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 C*ertificates 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 of ....a 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 is 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* releveant 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 proceeding 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 proceedure, 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 'desriable' 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 classses: ...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 will need more work if it is deemed by the authors to be a way to make progress. It is included to give a flavour of the new organization. To be completed, first it must be reviewed to ensure the requirements systematically describe the elements of the model and systematically cover the relations between those elements. Next it would need review to ensure it includes all the good requirements from the existing standard. Finally, the language of each requirement would need very careful review to ensure correctness.

This clause suggests a more systematic approach to the presentation of the requirements, working from the document, through the target types, to requirements and tests, their grouping and the testing of target instances.

Conformant spec docs shall **be text documents** (possibly with associated normative resources) that allow structuring of the text content into numbered clauses and allow structuring those clauses into hierarchies.

(This will be tested by later tests so maybe this is an assumption.)

=> is the spec a text document? Does it allow structuring into hierarachical clauses?

docs shall **list all the Standardization Target Types** for which the doc makes normative statements and shall assign to **each a name label, a URI, and a description** of the type and its possible instances.

> => is there a list? Does each type have a name label, URI, and description? Are any requirements made against a target that is not part of that list?

docs shall have as their **normative text only** requirements *{and tests? and recommendations?}*. Requirements **define atomic testable injunctions** made of a single Target, a particular instance of a single Standardization Target Type. Req. are **visibly identified** as separate entities. Req. **have a name label, *(a summary,)* a URI, and the normative statement**. Requirements may link to external normative resources. The **injunctive language** of requirements uses ... SHALL, ...) (Req. should be in form <target> <verb> (<state>) <injunction>)

> => find all normative text, establish that it is either in a req. or linked from one.

> => find all requirements, for each see that it is visually separate, that it has a name label, (summary,) URI, and normative statement.

> (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.)

Docs shall have, **for every requirement, at least one Test**, a procedure that evaluates conformance of a particular Target instance with the injunction made by the requirement. (To the extent possible, the test or tests evaluating each requirement should be comprehensive, that is together they evaluate conformance thoroughly. (it is not always possible to affirm conformance but sometimes only to negate it --- in that case we need to try to negate it in many ways before we can believe it is true)).

> => find all requirements, for each find at least one test. (N.B. this should be easy given the req. for parallelism made later.)

Docs shall have only tests that are **visibly identified** as separate entities. Tests have a **name label, a summary, a URI, and the statement** of the procedure.

> => find each Test, for each see that it is visually separate, that it has a name label, summary, URI, and statement of the procedure.

Okay, you get the picture. We start with the document, work our way through the entities of the model, here only the key pieces Next we would turn to grouping elements of the UML model, then to testing.

GROUPING

- group req. in to req.mod. and req.cl. Each has name and URI. Each has only req. that have the same std. target type.

-for each target type, declare on rec.class as 'core'.

-group tests into test mod. and test cl. that parallel the req. in name and structure but which are all together in a single clause, the test suite. Each has a name and URI. (the natural consequence is that each will have tests that eval the same std. target type.) Each test cl shall have an associated certificate of conformance; discuss parameters.

-req.cl. can depend on others either b/c target must conform (direct) or b/c the target of the dependency is used somehow (indirect). If req.cl depends, then the conf.cl must also depend. Can't redefine requirements (but can characterize/restrict them).

-dependencies on non-conformant standards (prob. needs to come through a req.cl wrapper that deals with testing, optional stuff (a different class).

TESTING

docs declare a testing process by which a target can be evaluated for conformance with a conformance class. The process must run through the tests of the conformance class and failure in one test is failure to conform with the class. If the target passes all then gets a certificate of conformance, possibly parameterized.

Optionality is merely a consequence of this structure and needs only mentioning rather than formal treatment. We have banned it at any level below Conf.cl therefore that's the level where it comes in and specs should adapt accordingly.

=> find that the process is declared and that it said to work this way. (It does seem there should be a way for us to declare this process and have other standards simply import it somehow.)