NISTIR 7175

XML Schema Design Quality Test Requirement Document

Boonserm Kulvatunyou
K.C. Morris

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7175

XML Schema Design Quality Test Requirement Document

Boonserm Kulvatunyou
K.C. Morris
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

October 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., Director

1. Introduction

Consistent design of XML schema within an organization or single integration project can reduce the number and the severity of interoperability problems. In addition, this consistency makes the XML schema easier to extend, understand, implement, and maintain; and, it paves the way for automated testing and mapping.

Applying best practices is one way to achieve this design consistency. The literatures in the reference section advocate a number of recommended best practices for designing business message standards. In reviewing the recommendations from different references, we discovered that no single agreed upon set of best practices exists.

Using a coherent subset of these recommendations, NIST researchers developed a collection of test requirements. These test requirements are maintained separately and organized according to the original reference documents on which they were based. This paper describes some of these requirements and provides rationale, explanations, examples, and comments for each.

These requirements form part of a framework, which can be used to assess the overall quality of an XML schema. Other parts of the framework include computer executable test cases and test profiles. Briefly, test cases are used to verify the conformance to those requirements and test profiles are groups of test requirements. Test profiles are entry points for executing a set of test cases. More information about test cases and test profiles, which are not described in this paper, can be found in [12].

The audience of this document includes XML architect and systems integration managers who are looking for XML schema guidelines to XML message development. The reader is assumed to have working knowledge of XML and XML Schema.

2. Summary of Test Requirements

This section summarizes design requirements included in this document. The summaries are given for each requirement table in the same order as those included in sections 7-14.

2.1. Requirement from OAGI Design Document

The Open Application Group Inc. (OAGI) design document [1] contains eleven (11) design requirements. These requirements are elementary practices for an XML architecture. Most are fully testable and most are generic across organizations. Consequently, any organization can test against these requirements, although slight variations may be required.

2.2. Requirements from NIST B2B Testbed Recommendation

Design requirements in Section 8 are obtained over the course of the National Institute of Standards and Technology (NIST) Business-to-Business (B2B) Testbed project [22]. Test cases written for a number of requirements included in this recommendation require heuristics. Consequently, the evaluation is approximation. Moreover, they require reference data and organizational specific data. We recommend that XML architect should consider adopting requirements #50, #200, #650 as baseline XML schema design practices.

2.3. Requirements from GCSS-AF BOD Developer's Guide, Version 1.0 Draft

The Air Force (AF) Global Combat and Support System (GCSS) BOD developer's guide [3] is a draft document. When completed, several of the practices described in this document, especially those related to metadata, could be adopted easily by any organization. This document derives a number of practices from the OAGI design document and the ebXML Core Component (ebCC) specification [14].

2.4. Requirements from UBL Naming and Design Rules

The Universal Business Language (UBL) [23] is an XML derivative of the ebCC Specification. The reader of these requirements must have a good working knowledge of the ebCC specification. The UBL approach is (1) to model the data using the ebCC method, and (2) to automatically generate the XML schema from the ebCC constructs (modeled in a class diagram or a spreadsheet). The generated XML schema should normally conform to ebXML and UBL naming and design rules [10]. However, it is not possible to apply some rules to the generation. For example, the rules may be fuzzy, may require more information that is not accessible from the spreadsheet, or may be passive. In addition, final touches or manual changes to the schemas typically occur, particularly in the distributed development environment, and some information needs manual specification in the spreadsheet. Consequently, checking the generated schemas against these test requirements is an important step in quality assurance. For example, requirement #550, *UBL namespaces exclusion*, involves user extension, and requirement #650, *UBL schema location*, checks the schemas Internet accessibility. These two requirements could be tested only after the schema has been generated.

The intent to automatically generate XML schemas from the ebCC constructs makes the UBL name and design rules document very comprehensive. The rules cover recommended as well as disapproved XML schema constructs, XML schema architecture (how schemas are partitioned and modularized), versioning, and detail documentation guidelines. Most of the rules are concrete and testable.

The applicable schema type for UBL has the value of 'LAD' for every test requirement because of the modeling practice promoted by this recommendation. In this practice the low, aggregate, and document level constructs can be mixed together in a single schema.

2.5. XML.GOV Developers Guide

The XML.GOV guideline [11] has thirteen (13) rules, most of which are partially testable. It gives a set of high-level guidelines for developing XML business content specifications for federal agencies.

2.6. KIEC XML Guideline

The Korean Institute of Electronic Commerce (KIEC) XML guideline [2] contains several unique practices that are not organizationally specific. This document identifies several design patterns and controlled vocabularies. KIEC divides information entities into four layers including messages, components (composing messages), basic information entities (composing components), and a code list used by basic information entities.

Some practices conflict with other guidelines. There are also a number of inconsistencies and ambiguities, which must be resolved to improve testability and test coverage. These are described inline in the KIEC requirements table in Section 12.

2.7. Requirements from ASC X12 Reference Model for XML Design

This ASC X12 design guideline [4] contains one of the most comprehensive sets of best practices that are architecturally independent (unlike the UBL naming and design rules which are specific to the ebCC specification). The guideline implements a philosophy that no single practice fits all needs. Its opinion is that what seems an advantageous decision from one viewpoint can be disadvantageous from another.

X12 design rules come in two basic forms: syntax and semantics. The X12 practices seek to accomplish two contradictory design goals: (1) reusability of shared schemas and (2) instance data validation through fully detailed schema specification. X12 guideline recommends that shared document level schemas have as many restrictions as possible yet still create low-level schemas that are reusable. The practice relies on placing many of the restrictions in the higher-level document schemas, which are the ones used in actual business transactions.

Some differing practices between X12 and OAGI design guidelines include its disallowance of the `xsd:substitutionGroup` and `xsd:any` elements, while OAGI allows those practices. X12 guideline also believes that the schema should be as prescriptive as possible for tight validation, while OAGI practice devises multiple stages of validations using the Schematron [9] and leaves the schema as flexible as possible to promote wider adoption.

2.8. Requirements from AEX Guidelines

The Capital Facilities Industry (CFI) has published two documents which serve as guidelines for XML Schema development work within the AEX (Automating Equipment Exchange) project [18]: Using XML Schemas for Facilities Equipment [19] and XML Schema Development Guidelines [20]. The first document contains the initial version of the AEX schemas, called cfiXML. The second contains the guidelines for developing those schemas. They are not completely consistent. When faced with an inconsistency, the later document supersedes. Some requirements in the AEX table are based on our experience from the AEX Testbed project.

3. Test Requirements

Each reference document (design guideline) has a corresponding requirements table, which appears at the end of this document[1].

Each table has the following fields:

1. ID: The ID column indicates the test requirement identification number for referencing purposes.
2. Test Requirement: The Test Requirement column is a short name of the requirement.
3. Test Coverage: The possible values of this column are '*F*' = '*Full*', '*P*' = '*Partial*', '*U*' = '*Unknown*', and '*NA*' = '*Not Applicable*'. Requirements that can have full (*F)* or partial (*P)* coverage are testable. The '*F*' value means that one or more test cases can fully verify the conformance to the test requirement. The '*P*' value means that the associated test cases can verify the conformance to the test requirement only partially. The '*U*' value means that the test coverage cannot be determined. Possible reasons include an external factor or variable scope. The '*NA*' value means that there is no executable test case associated with the test requirement and the requirement is not testable via test cases.
4. Rationale: The Rationale gives one or more reasons why this particular requirement is included. The possible choices are described in Section 3.1.
5. Schema Type: The Schema Type indicates the kinds of schemas applicable to the test requirement. The value will be the combination of the '*L*', '*A*', and '*D*'. Section 3.2 describes this in further detail.
6. Description: The Description explains the practice and general approaches to testing. It also gives examples of the recommended construct.
7. Note: Notes provide extra explanations or opinions that are not contained explicitly in the guideline document. A note can be a comparison with other practices, an explanation of the test coverage value, and further clarification of the description, to name a few.

We note that values for the test-coverage field and schema-type field can change over time. For example, the availability of better reference data may shift the test-coverage value from "P" to "F". Moreover, the suggested values for schema type represent only one way to classify schemas

[1] Note that the NIST B2B Testbed recommendation table does not have a corresponding, published, reference document. The contents of its table are based on our experience with that testbed.

in business data standards such as OAGI and UBL. In addition, the applicable schema type can be test-case dependent.

3.1. Rationale

We used a number of justifications for including specific requirements. They are listed below.

a) *Validation and model clarity* is used for those practices that make the semantics of a construct clear to the user as well as to the machine. Subsequently, an XML parser can better validate the content of the instance against the schema.

b) *Structural clarity* is used for those practices that contribute to a schema's readability, which can facilitate consistent interpretation of a standard and accelerate adoption/implementation.

c) *Clarity* is used for those practices that encapsulate both the structural clarity and validation and model clarity rationales.

d) *Extensibility* is used for those practices that promote reuse through extension. Hence, extensibility also implies reusability.

e) *Common symbolic syntax* is used for those practices that foster the use of common naming conventions. Such practices enable better automation and improve readability and clarity.

f) *Maintainability* is used for those practices that reduce the maintenance burden especially when changes occur. They help minimize repetitious work and potential errors.

g) *Performance* is used for those practices that can reduce computational overhead associated with the XML instance parsing, validation, and other XML processing.

h) *Interoperability* is used for those practices that promote interoperability among partners sharing the same schema. When no other rationale is applicable, this one is used.

i) *Model validity* is used for those practices that ensure the schema's semantic validity (e.g., no duplicate contents). The rules associated with this rationale may overlap with the schema parser or schema semantic checking functionality (the IBM Schema Quality Checker tool offers the schema semantic checking functionality [13]).

3.2. Schema Types

Schema types are categorized based on the level of aggregation of constructs included in the schema. Three values are used in order of increasing level of aggregation: L, A, and D.

1. Low-level schema (L) - the schema typically contains simple types and complex types with simple content definitions. This may map to terms in business content standards. For example, the terms can be core-component types, data types, or basic business entities in the ebCC specification [14]. They can also be fields, meta, or enumeration in the OAGI specification [15]. The types of schema typically contain reusable, context-free vocabulary, and the elements or types included are not by themselves meaningful to a business exchange.
2. Aggregate level schema (A) - the schema typically contains complex type definitions and corresponding global-element declarations. These may map to terms in business content standards such as the aggregate business information entity in the ebCC specification or components in the OAGI specification. The constructs in this schema reuse the constructs from the low-level schema.
3. Document level schema (D) - the schema typically contains only a few definitions of complex types and global element declarations. These may map to terms in business content standards such as the assembly document in the ebCC specification, nouns and business object documents in the OAGI, or the transaction concept in the RosettaNet Implementation Framework specification [21]. The schemas at this level typically do not directly reuse the constructs from the low-level schema but the constructs from the aggregate level schema.

If a schema contains more than one level of construct, it should be tested against the requirements for all assigned values.

4. Future work

A test framework, with a web-based interface, is being developed. A hyperlink will be provided on the NIST Manufacturing Business-to-Business Interoperability Testbed web site [22]. Test cases for a selected set of test requirements will be encoded. Test requirements included in this document may be implemented in the web-based repository. In addition, a set of matrices providing comparison among test requirements coming from different XML guidelines is being developed.

5. References

1. Rowell, M., Feblowitz, M. (2002). *OAGIS 8 Design Document (Draft 0.93)*.
2. Korean Institute for Electronic Commerce. *Guidelines for Development of XML Electronic Messages in Korea (March 2003)*. Available online via <www.xeni.co.kr/support/KIECGuidelineFinal_english_.pdf>.
3. Lockheed Martin Federal Systems (October 2002). *Global Combat and Support System – Air Force BOD Developer's Guide Draft Version 1.1*. Department of Air the Force Headquarters Materiel Systems Group (MSG).
4. ASC X12C Communications and Controls Subcommittee (October 2002). *ASC X12 Reference Model for XML Design*. ASC X12C/2002-61.
5. World Wide Web Consortium (May 2001). *XML Schema Part 0: Primer Structure W3C Recommendation* http://www.w3.org/TR/xmlschema-0/.
6. ebXML Technical Architecture Specification v1.0.4, 16 February 2001.
7. Roger Costello XML Schemas: Best Practices http://www.xfront.com/BestPracticesHomepage.html.
8. Roger Costello XML Schema Versioning http://www.xfront.com/Versioning.pdf.
9. Jelliffe, R., *The Schematron Assertion Language 1.5*, Academia Sinica Computing Center. http://www.ascc.net/xml/resource/schematron/Schematron2000.html.
10. OASIS UBL Naming and Design Rules Subcommittee (November 2003). *Universal Business Language (UBL) Naming and Design Rules*.
11. US Federal CIO Council Architecture and Infrastructure Committee, XML Working Group (April 2002). *Draft Federal XML Developer's Guide*.
12. Kulvatunyou, B., Ivezic, N., Buhwan J. Testing Requirements to Manage Data Exchange Specifications in Enterprise Integration – A Schema Design Quality Focus. *8th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, July 2004, Orlando, Florida.
13. IBM Corporation. The IBM Schema Quality Checker. http://www.alphaworks.ibm.com/tech/xmlsqc.
14. DISA UN/CEFACT. *EbXML Core Component Specification version 1.9* (December 2002). http://webster.disa.org/cefact-groups/tmg/downloads/CCWG/for_review/CCTS_V_1pt90.zip.
15. Open Application Groups Web Site, accessed March 2002. *Open Application Group Integration Specification version 8.0*. http://www.openapplications.org/downloads.
16. University of Edinburgh/W3C. *XSV 2.7, an Open Source XML Schema Validator*. http://www.ltg.ed.ac.uk/~ht/xsv-status.html.

17. The Apache Software foundation. *Xerces2 Java Parser Release 2.6.2*. http://xml.apache.org/xerces2-j/index.html.
18. Automated Equipment Exchange (AEX) project http://www.fiatech.org/projects/idim/aex.htm.
19. Teague, T. L., Turton, R. W., and Palmer, M. E., *Using XML Schemas for Facilities Equipment*, July 2004. http://www.fiatech.org/projects/idim/aexresources.htm.
20. Palmer, M.E., Burns, M., and Teague, T.L., *FIATECH XML Schema Development Guidelines*, March 2003.
21. The RosettaNet Web Site, accessed May 2003. *RosettaNet Implementation Framework (RNIF) Specification version 2.0*. http://www.rosettanet.org/RosettaNet/Doc/0/TAO5O8VV3E7KLCRIDD1BMU6N38/RNIF2.1.pdf.
22. The Manufacturing Business-to-Business Interoperability Testbed Web Site http://www.mel.nist.gov/msid/b2btestbed/.
23. Unified Business Language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl.
24. United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport – 3055 Code List, Responsible Agency Code. http://www.unece.org/trade/untdid/d03a/tred/tred3055.htm.
25. Metadata Standards Organization Web Site, *ISO/IEC 11179, Information Technology – Metadata Registries (MDR)*. http://metadata-stds.org/.
26. XML Common Business Library. http://www.xcbl.org/ .
27. Open Travel Alliance. http://www.opentravel.org/.
28. United Nation Directories for Electronic Data Interchange for Administration, Commerce, and Transport. http://www.unece.org/trade/untdid/welcome.htm.

6. Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

7. OAGI Design Document

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Check for non-determinism.**	**LAD**	**P**	**Validation and model clarity, Extensibility**
	Description: This includes the check for data structures that fall into "type by attribute" category such as assigning party type as an attribute (i.e., a `Party` element with a `type` or `qualifier` attribute). This unnecessarily limits the extensibility and model (data structure) expression because types are hidden. Separate types should be defined with relationships (e.g., `ShipToParty` and `ShipFromParty` as subtypes of `Party`). In this case, one can associate any unique property to the two subtypes. In addition, whenever only the `ShipToParty` type is appropriate, it can be explicitly indicated in the model and validated by the parser. These cannot be done if the type is hidden in the attribute. **Note:** Example: If there are two types of parties required within an element, this non-determinism pattern cannot be indicated clearly if both types are specified. Because there is little structure/model associated with simple types, the non-determinism should apply to only complex Types. The Venetian blind approach suggests that the user should define as many types as necessary and use them to specify the semantics of elements like the example about Party. In addition to the `ShipToParty` and the `ShipFromParty`, `DeliveringParty` and `ReceivingParty` may be defined. First, this helps relegate the long name to the type definition leaving the element name (which is sent over the wire) succinct. Second, it makes the semantics of the element clearer which leads to better validation.			
100	**Check for conformance to naming conventions.**	**LAD**	**P**	**Common symbolic syntax**
	Description: OAGIS 8.0 recommends use of upper camel case. To ensure consistency, the upper-camel-case tag should be parsed to spell-check each sub-string and to ensure the validity of each substring – words, acronyms, or abbreviations. An all-upper-case substring may be recognized as a specific acronym and ignored or checked against a list of allowable acronyms. **Note:** ebXML Technical Architecture specification [6] provides guidelines for upper-camel-case convention. Cross-reference of allowable acronyms and abbreviation should be specified. This allows for information recognition and more efficient processing. Long tag names have become the convention because of the availability of increased computing power. Upper-camel-case convention is also adopted for such purposes. This test helps ensure common usage of the upper camel case across the organization and will help facilitate some automation as well. OAG recommends that any tag name longer than 31 characters be approved by the consortium.			

150	**Check for improper use of anonymous type.**	**AD**	**F**	**Extensibility**
	Description: Content model of an anonymous type is defined locally within an element. It cannot be referenced outside of that element definition; hence, it cannot be reused. On the other hand, a globally defined type allows it to be referenced and reused. OAGIS adapts the Venetian Blind [7] design approach -- global types should be defined where necessary, but global elements should be declared only for extensible components. See also #300. **Note:** In XML schema, type definitions can be viewed as a content model, but the element definitions are viewed as document structure. Content/data model has tight relationship with functional requirements or functional model. Hence, software components should be developed corresponding to the content/data model rather than to the document structure. The content model represents entities that are used and reused; therefore, software components developed around it can also be reused. Although the use of global types can cause name-clashing problem, the availability of namespace mechanisms reduce this problem drastically. The same term with different concepts (perhaps in different domains) can be defined in different namespaces. Anonymous type may be used for company's specific terms and terms that have a very specific and succinct semantics.			
200	**Check for use of weak typing.**	**AD**	**F**	**Validation and model clarity, extensibility**
	Description: Contents (both element and attribute content) within complex structured elements/types (those having these patterns `complexType/sequence` and `complexType/complexContent`) that are typed as XML Schema primitive Data Type are regarded as weak typing. This pattern is prone to violate the Venetian Blind [7] design (i.e., it is more like a Russian Doll [7] design). **Note:** XML schemas, which provide high degree of information aggregation, should not have their structure based on primitive data types -- this provides little semantics to the users and the integration software. Weak typing also limits the validation capability.			
250	**Check for feature regression.**	**LAD**	**F**	**Maintainability, extensibility**
	Description: Deriving a new complex type by restriction of another complex type requires that the derived type repeat all the components within the base type. This can cause inconsistency within the schema especially when changes are needed. When changes are applied to the base type, all derived types must be revised. **Note:** Alternatives to this functionality are the substitution group or an external restriction specification like Schematron [8]. This can also be viewed as XML Schema limitation.			

300	**Check for global element definition**	**LA**	**P**	**Extensibility**
	Description: OAGIS uses global element definition only when it is intended to be inline extensible. OAGI views `substitutionGroup` extensibility as a design time and declarative extension approach. The use of `xsi:type` extension approach is viewed as a non-declarative and hidden approach, which does not represent well the intention of the schema designer. **Note:** Global elements should be used only in an aggregate-level schema or in component-level information. Global elements exist without context, so they should be very self-contained. In the aggregate level, elements (or schemas) may reference other aggregate-level elements as well as declaring local elements associated with global types of a low-level entity. Low-level element is unlikely to be extended or substituted; hence, global element is not necessary. In fact, the `xsi:type` can indicate the designer's intention to use the `abstract` attribute to indicate whether or not it can be instantiated. The use of `xsi:type` requires that an XML processor be more intelligent and prepared for variations that could occur at runtime. In other word, `xsi:type` is not declarative.			
350	**Improper use of enumeration type.**	**LAD**	**P**	**Extensibility**
	Description: Enumeration is not easily extensible using the XML schema construct. OAGIS practice indicates that enumeration should be used for a stable code list only (not change in years). That means, for example, value list associated with business strategy should not use the enumeration construct. **Note:** We conclude that this practice may be applicable only to definitions in the standard. Elements used only internally may rely on the enumeration construct even if the value list is changeable. In addition, enumeration may be extended using the XML-Schema union construct. The testability is limited because it is hard to determine whether the semantics of a type is appropriate for the enumeration construct. However, references to standard code lists can be created and used for partial detection. UBL and KIEC have identified such a collection of code lists (these are summarized in Section 15).			
400	**Non-recommended extension**	**LA**	**P**	**Performance**
	Description: Some type definitions are not intended to be extensible. OAGIS practice states that its fields and compounds should not be extended due to too much overhead associated with the extensions from these lightweight type definitions. **Note:** The test coverage may be partial because it is impossible to determine the real intention absolutely. If a schema is identified as a low-level type, a test case may assume that all included terms are not extensible. This may not be true for the aggregate-level schema.			
450	**Non-recommended use of default namespace**	**LAD**	**F**	**Structural clarity**
	Description: Imported schemas are those schemas that come from different namespaces. Industry practice is not to use default namespace with the imported schema, so that the user/reader can always recognizes that entities[2] with namespaces are from different namespace than the current (target) namespace.			

[2] The term 'Entity' used in this document generally refers to both 'element' and 'type'.

500	**Recommended use of default namespace, i.e., non-recommended use of no target namespace**	**LAD**	**F**	**Structural clarity**
	Description: Use of default namespace is recommended in an XML schema because it prevents the use of no target namespace schema. **Note:** In the standards arena, all elements should have an associated namespace. The chameleon namespace approach described in [7] is not applicable to the standard development.			
550	**Enforce other/other pattern in the enumeration type**	**LAD**	**F**	**Extensibility**
	Description: OAGIS recommends that the other/other enumeration pattern be an easy (and temporary) enumeration extension approach. OAGIS recommends that such an extension should be considered into the next release of the OAGIS. **Note:** In a not-so-stable code list, OAG also recommends a semantically named design pattern as an alternative to enumeration. This approach recommends that a set of types be created for each enumerated value. Since this approach is based on type definitions, the user can always extend the definition with new types.			

8. NIST B2B Testbed Recommendation

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Check for improper use of no namespace schema.**	**LAD**	**F**	**Extensibility**
	Description: A schema without a namespace can hardly be used across organizations because entities cannot be uniquely identified. In addition, a schema will have limited extensibility before name clashing occurs. The schema's attribute `targetNamespace` must be present.			
100	**Check the schema for its referential ability to facilitate the use of existing standards.**	**LAD**	**P**	**Interoperability**
	Description: This test requirement will try to do semantic matching and suggest a reference to an existing standard. Such concepts should be structured so that their attributes allow specification of meta-data (pointing to associated standards) or they should be typed based on enumerations. **Note:** A table indicating the cross-referencing between concepts and standards is necessary. UBL and KIEC have identified a collection of enumerations (code lists, see Section 15). The test coverage is partial, because we may not be able to collect all the existing references.			
150	**Check for organizational specific look and feel.**	**LAD**	**U**	**Structural clarity**
	Description: The criteria of this test will be organization dependent, although the patterns will be the same. Some of the issues include specific target namespace, namespace abbreviation, and consistent use of qualified or non-qualified element and attribute forms. **Note:** The test coverage is unknown because the scope of the test cannot be predetermined.			
200	**Detect unnecessarily long tag name.**	**LAD**	**F**	**Maintainability, Performance**
	Description: Detect child elements that repeat the name of a parent element (i.e., its context). **Note:** This rule recommends that the name of child elements not be repeated. On the other hand, if the Venetian blind approach is used, long tag names can be deferred to the type definition while keeping the element name short.			
250	**Enforce Id design pattern**	**LAD**	**P**	**Interoperability**
	Description: The Id design pattern recommends that the assigning organization and revision number be specified. This design pattern allows an object to have multiple Ids. **Note:** The test coverage is partial, because the test may not be able to recognize all the constructs that should adopt the Id design pattern.			

300	**Code list design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: Elements definitions that are based on an enumerated list should allow metadata about the code so that an alternative code list can be used. For example, a currency element using currency code should have an attribute such as code list name and/or agency. **Note:** The test coverage is partial, because not all constructs based on the code list may be tested without user indication. The test may not be able to recognize all the constructs that should adopt the code design pattern. However, all enumeration types can be verified if they have the code list construct.			
350	**Indicator design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: An indicator type should include an indicator –type format attribute. **Note:** The test coverage is partial, because not all constructs based on the indicator type may be tested without user indication. The test may not be able to recognize all the constructs that could adopt the indicator design pattern.			
400	**Datetime design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: Datetime type should include a Datetime-type format attribute. **Note:** See also `DateTime. Type` core component type of ebCC specification. The test coverage is partial, because not all constructs based on the datetime type may be tested without user indication. The test may not be able to recognize all the constructs that could adopt the Datetime design pattern.			
450	**Measure design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: Measure type should have a unit attribute and metadata about the unit code list (e.g., name, and agency). **Note:** See also `Measure. Type` core component type of ebCC specification. The test coverage is partial, because not all constructs based on the measure type may be tested without user indication. The test may not be able to recognize all the constructs that could adopt the Measure design pattern.			
500	**Quantity design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: Quantity type should have a unit attribute and metadata about the unit code list (e.g., name, and agency). **Note:** See also `Quantity. Type` core component type of ebCC specification. The test coverage is partial, because not all constructs based on the quantity type may be tested without user indication. The test may not be able to recognize all the constructs that could adopt the Quantity design pattern.			

650	**Check for Null content model**	**LAD**	**F**	**Validation and model clarity**
	Description: A document should not contain all optional elements because then the document can be empty. A complex type entity (type or element) must not contain an empty content. That is it should contain either at least one required child element or allow text content. Simple type entity should enforce non-empty text content.			
700	**Proper use of the plural element/attribute names**	**AD**	**P**	**Validation and model clarity**
	Description: There are generally two interpretations associated with the plural form. The first interpretation is that it is a container of multiple, different information entities or objects. The second interpretation is that it is a container of the same information entity or object, which can be represented in multiple ways (for example, an Item Id could be specified with a customer's version and a supplier's version). We recommend that a plural form be used for the second interpretation while a singular form associated with a multiplicity cardinality be used for the first interpretation (without a container element). **Note:** This test requirement may be tested through assumed schema patterns. An attribute name should not be a plural form unless its type is token. A plural element name should be followed an element with multiplicity cardinality, which must be also a head of a substitution group or based on a type for which various ways of instantiation is possible (e.g., more than one non-abstract extensions exist). The test coverage is partial because it is uncertain whether the assumed pattern covers all the cases.			
750	**Duplicative content (hidden non-deterministic content model)**	**AD**	**P**	**Model validity**
	Description: This test looks for implicitly duplicative content. For example, two child elements with different names pointing to the same complex type. **Note:** It should be noted that some parsers and grammar checker like the IBM Schema Quality Checker already validate explicit cases such as when an element contains references to the same element.			

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Enforce camel case convention**	**LAD**	**P**	**Common symbolic syntax**
	Description: GCSS-AF document reference ebXML technical architecture recommendation. This requirement is the same as that of OAGIS (see #100 in the OAGIS table).			
100	**Usage of acronyms and abbreviations**	**LAD**	**F**	**Common symbolic syntax**
	Description: GCSS-AF recommendations are slightly different from OAGIS in this category. (a) Abbreviations must not be used. (b) Acronyms should be avoided unless very common; but, when used, they must be in all upper case overriding the camel case convention. (c) Acronyms must be spelled out in the associated annotation element. **Note:** In this test, the test coverage has been specified as Full; however, this is a theoretical value. The reason is that the test can always flag unknown values but the flag may or may not be a negative indicator depending on the user's literary reference.			
150	**XML component metadata requirements**	**LAD**	**N/A**	**Interoperability, Common symbolic syntax**
	Description: GCSS-AF requires that component definition (its type definition) must have pointers to information in the registry including the definition, URL, and registry identifier. This would require that all global complex types (which are classified as components) be checked that the metadata is specified. **Note:** This requirement is not testable because the guide recommends multiple, possibly unstructured, ways of capturing the metadata. In addition, it does not normatively specify the metadata needed.			
200	**Check for conformance to ebXML naming convention**	**LAD**	**P**	**Common symbolic syntax**
	Description: This test requirement references the ISO 11179 naming convention. It needs to verify, for example, that no article, adjectives, or plural forms have been used in the name. **Note:** The test coverage in this case is partial because the name may or may not be parsed into separate substrings correctly and words have multiple functionalities resulting in several false alarms and some false positives.			
250	**Check for documentation**	**LAD**	**P**	**Interoperability**
	Description: All element and type definitions must have an annotation child element. **Note:** This only partially checks the documentation. We can never check that the documentation is sufficient.			

300	**Enumeration reference**	**LAD**	**F**	**Interoperability**
	Description: All elements or types based on an enumerated value must provide pointers to where the code lists are derived. **Note:** This requirement is similar to that of #250 in the NIST Recommendation table. However, the scope is limited to only check that all types and elements based on enumeration have placeholders for metadata attributes.			
350	**Versioning**	**LAD**	**F**	**Configuration control**
	Description: Version information must be provided in XML Schemas within the annotation. For a DTD, this information should be captured in a fixed attribute of the root element. For XML Schema, use `<xsd:appInfo>` tag under the annotation of the root element. **Note:** GCSS-AF also recommends that version number of the schema be provided within the XML instance. Every BOD has a 'version' attribute in the root element.			
400	**Metadata in the header of schemas and documents**	**LAD**	**P**	**Maintainability, Interoperability**
	Description: The following metadata indicated in Appendix F must be provided in the headers of both schema and instance*: Schema name, schema version, DOD namespace(s), functional data area, URL to the most current version, a description of the purpose of the schema, name of the application or program of record that created and and/or manages the schema, among others. **Note:** These fields cannot be normatively verified for schema, because no normative tags have been specified to date. At this point we may verify that the first `<xsd:appInfo>` element in the schema contains those information objects in a text format. **See more in Appendix F of the guideline document.** *****This document focuses on requirements for schema only.			
450	**Correct use of attribute (attribute vs. element)**	**LAD**	**P**	**Extensibility**
	Description: The followings characteristics (rules) apply to attribute: (a) Attribute should be based on code list (enumeration). (b) Test through instance. An Attribute value should be a short, simple, single token (no white space). Attribute cannot have a complex structure. (c) Attribute is metadata about information. (d) Attribute should only be used to describe information units that cannot or will not be extended or subdivided. (e) Attributes of an element should generally be applicable to all of its child elements. **Note:** The test coverage is partial because the verifying conditions require information that is not available in the schema and knowledge that is hard to generalize. Particularly, (a) can be verified; (b) can be verified through instances; (c), (d), and (e) cannot be verified because the knowledge about these conditions cannot be captured in a generic fashion.			

10. UBL Naming and Design Rules

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Schema look and feel**	**LAD**	**P**	**Structural clarity**
	Description: The structure of a UBL compliance schema has been specified in Section 3.1 of its guideline. It indicates the order of the schema declarations and the order in which different types of components should be specified. **Note:** The section covered by this requirement is 3.1, rule GXS1. The test coverage is specified as partial because it is uncertain that the different types of components can be identified and checked.			
100	**One root element is defined in a schema**	**D**	**F**	**Structural clarity**
	Description: Each UBL ControlSchema MUST identify one global element declaration that defines the overall business process being conveyed in the Schema expression. That global element MUST include an `xsd:annotation` child element which MUST further contain an `xsd:documentation` child element that declares "This element MUST be conveyed as the root element in any instance document based on this Schema expression." A ControlSchema is a schema that conveys a specific business document. Each name will have one and only one control schema. **Note:** The section covered by this requirement is 3.1.1, rule ELD2.			
150	**Pointer from a dictionary entry name to an element or attribute**	**LAD**	**F**	**Clarity**
	Description: Each dictionary entry name (see ebCC specification for the definition of the dictionary entry name) must have one and only one Fully Qualified Path (FQP) to an associated element or attribute. UBL uses truncated names for element and attribute names, while the dictionary entry name retains the full name. Dictionary entry names are maintained in the UBL documentation dictionary. **Note:** The section covered by this requirement is 3.2.1, rule NMC1.			
200	**UBL Classes**	**LAD**	**NA**	**Model validity**
	Description: UBL Models must define classes based on `ccts:BasicBusinessInformationEntities` and `ccts:AggregateBusinessInformationEntities`. **Note:** The section covered by this requirement is 3.2.2.1, rule MDC1. The Model validity rationale is from the UBL modeling perspective. The testability is identified as not possible because it is uncertain how UBL models are linked to the `ccts:BasicBusinessInformationEntities` and `ccts:AggregateBusinessInformationEntities`.			

250	**Core component type restriction**	**LAD**	**F**	**Model validity**
	Description: Each `ccts:BasicBusinessInformationEntity` must be associated with ebXML Core Component approved `ccts:CoreComponentTypes`. **Note:** The section covered by this requirement is 3.2.2.2, rule MDC2. The testability and test coverage reflect the assumption that the approved `ccts:CoreComponentType` schema is available and `ccts:BasicBusinessInformationEntities` can be discretely identified from annotation.			
300	**Consistent business function in customization**	**LAD**	**NA**	**Model validity**
	Description: Customization of UBL document must retain the original business function of that document. **Note:** The section covered by this requirement is 3.2.2.3, rule MDC3. The testability is identified as not possible because there is no methodology available to measure the customization relative to the business function. The model validity rationale for this particular requirement is only from the UBL perspective.			
350	**Mixed content model**	**LAD**	**F**	**Interoperability**
	Description: Mixed content must not be used to pass business data. Mixed content should be disallowed. **Note:** The section covered by this requirement is 3.2.2.4, rule MDC4. Mix content occurs when an XML element can have both children and character data.			
400	**Reusability**	**LAD**	**F**	**Extensibility**
	Description: All element declarations must be global except for ID and Code. The section covered by this requirement explains in detail the rationale that UBL adopts this architecture. When global elements are used, software code can be reused without a change. This is not possible when using global type definition. **Note:** The section covered by this requirement is 3.3, rule ELD2. Although the test coverage is full, a number of false negatives may be identified because it may not be possible to correctly identify the exception cases for ID and Code.			
450	**Target namespace declaration**	**LAD**	**F**	**Model validity**
	Description: Every schema created (called UBL-defined schemas) and used (called UBL-used schemas) must have a target namespace. **Note:** The section covered by this requirement is 3.4.1, rule NMS1. The model validity rationale for this particular requirement is only from UBL perspective (i.e., it does not mean that a schema without a target namespace has an invalid model).			

500	**Namespace uniqueness and schema version**	**LAD**	**P**	**Structural clarity, Maintenance**
	Description: UBL uses the namespace scheme to group schemas into Schema Sets. All schemas in a schema set must have the same version. Different versions must also have different target namespaces. **Note:** The section covered by this requirement is 3.4.1, rule NMS2. The test coverage is partial because we can only checked that schemas with the same namespace have the same version. We cannot verify a correct target namespace, unless a schema set is identified first. The namespace uniqueness cannot be tested for a similar reason.			
550	**UBL namespaces exclusion**	**LAD**	**F**	**Structural clarity**
	Description: Only UBL maintained schemas could use UBL namespaces; any extension from a UBL library must define its own namespace. **Note:** The section covered by this requirement is 3.4.1, rule NMS3. The full test coverage is based on the assumption that the extensions to the UBL library can be recognized. This requires a complete set of UBL schemas from its library.			
600	**UBL namespace patterns**	**LAD**	**P**	**Structural clarity, Maintenance, Common symbolic syntax**
	Description: UBL uses URN format (a subset of URI, IETF RFC 2396) for its namespaces. The UBL draft schemas must have this pattern, urn:oasis:names:tc:ubl:schema:<name>:<major>:<minor>[:<revision>], and the UBL schemas holding OASIS standard status must follow this pattern, urn:oasis:names:specification:ubl:schema:<name>:<major>:<minor>. The <name>, <major>, <minor>, and <revision> are variables, while the revision is optional. The major version must be a positive integer. The minor version must be a non-negative integer. For example, the first version would start at 1:0. The minor version is incremented for every new release that is backward compatible. The name field should remain the same throughout versions unless a new name is needed. This indicates that any change in a schema module requires a new namespace association. UBL takes the approach that such association should not be altered once published. This ensures that an importing schema always imports the intended content and that such content is always available (unchanged). **Note:** The sections covered by this requirement is 3.4.2 and 3.5, rules NMS4, NMS5, VER1, VER2, VER3, VER4, VER5, another VER4 and VER5 on lines 1054 and 1056 (typo in the document), VER6. The status, draft or standard, of a schema must be identified in order to verify the namespace pattern. The rules VER5 and the duplicated VER4 and VER5 could only be checked if a repository of schemas is kept. The rule VER6 is the same as the rule NMS2 and is directed to the test requirement #500. The revision number will be assumed to be a non-negative integer when presents (this is not explicitly specified in the guideline).			
650	**UBL schema location**	**LAD**	**P**	**Structural clarity**
	Description: UBL schemas must be hosted at OASIS UBL TC web site and are available at this URL pattern http://www.oasis-open.org/comittees/ubl/schema/<schema-mod-name>.xsd. **Note:** The section covered by this requirement is 3.4.4, rule NMS6.			

700	**Namespace persistence**	**LAD**	**P**	**Structural clarity, Maintenance**
	Description: UBL namespaces must never be changed (while the actual location of schemas may change). **Note:** The section covered by this requirement is 3.4.4, rule NMS7. The requirement is partially testable based on an approximation that two schemas contain similar set of definitions.			
750	**Schema's version attribute**	**LAD**	**F**	**Common symbolic syntax, clarity**
	Description: UBL schemas must be versioned with the following representations: 1) <*major*>:<*minor*>:[<*revision*>] where the <*revision*> is optional. 2) <*major*>:<*minor*> The value of <*major*> version begins with 1. The value of the <*minor*> begins with 0. The first minor version of a major version would have a value of 1. **Note:** The section covered by this requirement is 3.5. The version number should follow what has been described in #600.			
800	**Minor versions compatibility**	**LAD**	**P**	**Interoperability**
	Description: In a minor version, the name of the version construct must not change (the short name), unless the intent of the change is to rename the construct. Minor versions must import their immediately preceding minor version and maintain backward compatibility. The backward compatibility is maintained by the restricted use of only `xsd:extension` and `xsd:restriction` mechanisms and by the guarantee of semantic compatibility. The `xsd:redefine` must not be used. New constructs may be added. **Note:** The section covered by this requirement is 3.5, rules VER5, VER8, VER9, and VER10. While the rule VER5 may be flagged for name changes between minor revisions, we interpret the last part of the rule "intent of change is to rename the construct" as the intention to change the semantics. Such intentions cannot be recognized completely. However, the likelihood of false negatives increases. While rule VER8 is testable in full, testing the rule VER9 is not possible because the semantic compatibility with the previous version cannot be determined. For this reason, the test coverage is partial.			
850	**Unambiguous schema dependency**	**LAD**	**F**	**Model validity**
	Description: UBL classified the schemas into control schemas, internal schema modules (same namespace), and external schema module. The control schema module includes (depends on) the internal schema modules. UBL requires that any external dependency should only import the control schema of that namespace and not the internal schemas of that namespace. **Note:** The section covered by this requirement is 3.6.1.1, rule SSM2 and SSM3.			
900	**Internal schema module name**	**LAD**	**F**	**Common symbolic syntax, clarity**
	Description: UBL internal schema module name must have the following format <ParentSchemaModuleName/ControlSchema><InternalSchemaModuleFunction><SchemaModuleName>. The internal schema module must be in the same namespace as the control schema. **Note:** The section covered by this requirement is 3.6.3, rule SSM6 and SSM7.			

950	**Data Type documentation**	**LAD**	**F**	**Clarity**
	Description: The data type must contain a structured set of documentation in the following pattern. `UniqueIdentifier` (M), `CategoryCode` (M), `DictionaryEntryName` (M), `Definition` (M), `Version` (M), `ObjectQualifierClass` (O), `ObjectClass` (M), `QualifierTerm` (M), `UsageRule` (O+). A usage rule is a constraint describing specific conditions that are applicable to the data type. A data type that is defined using restriction on the Content Component must include the following documentation pattern. `RestrictionType` (M), `RestrictionValue` (M), and `ExpressionType` (O). A data type that is defined using restriction on the Supplementary Component must include the following documentation pattern. `SupplementaryComponentName` (M), `RestrictionValue` (M) and repetitive. The `CategoryCode` is the category to which the object belongs. For example, ABIE for Aggregate Business Information Entity, BBIE for Basic Business Information Entity, ASBIE for Association Business Information, RT for Representation Term. **Note:** M = Mandatory, O = Optional, O+ = Optional with repetitive. The occurrence of the `ObjectClass` is not specified in the guideline. It is assumed mandatory here. The section covered by this requirement is 3.7, rule DOC1, DOC2, and DOC3. Although not specified in the guideline, the `CategoryCode` for Data Type is assumed DT.			
1000	**Basic business information entity documentation**	**LAD**	**F**	**Clarity**
	Description: The basic business information entity (BBIE) must contain a structured set of documentation in the following pattern. `UniqueIdentifier` (M), `CategoryCode` (M), `DictionaryEntryName` (M), `Version` (M), `Definition` (M), `Cardinality` (M), `QualifierTerm` (O), `UsageRule` (O+), `ConstraintLanguage` (O+), `BusinessTerm` (O+), `Example` (O+). A usage rule describes specific conditions that are applicable to the basic business information entity. The Cardinality indicates whether the BBIE represents a not-applicable, mandatory, optional, or repetitive characteristic of the Aggregate Business Information Entity (ABIE). The `QualifierTerm` qualifies the Property Term of the associated Core Component Property in the associated Aggregate Core Component. The `ConstraintLanguage` formally indicates how the BBIE is derived from the Core Component. The `Example` contains example values of the BBIE. The `CategoryCode` is always BBIE in this case. **Note:** The section covered by this requirement is 3.7, rule DOC4.			
1050	**Aggregate business information entity documentation**	**LAD**	**F**	**Clarity**
	Description: The Aggregate Business Information Entity (ABIE) must contain a structured set of documentation in the following pattern. `UniqueIdentifier` (M), `CategoryCode` (M), `DictionaryEntryName` (M), `Version` (M), `Definition` (M), `QualifierTerm` (O), `UsageRule` (O+), `ConstraintLanguage` (O+), `BusinessTerm` (O+), `Example` (O+). The `UsageRule` describes specific conditions that are applicable to the ABIE. The `QualifierTerm` qualifies the Object Class Term of the Aggregate Core Component. The `ConstraintLanguage` formally indicates how the ABIE is derived from the stored Core Component and Business Context. The `Example` is example of a possible value of the ABIE. The `CategoryCode` is always ABIE in this case. **Note:** The section covered by this requirement is 3.7, rule DOC5.			

1100	**Association business information entity documentation**	**AD**	**F**	**Clarity**
	Description: The association business information entity (ASBIE) must contain a structured set of documentation in the following pattern. `UniqueIdentifier` (M), `CategoryCode` (M), `DictionaryEntryName` (M), `Version` (M), `Definition` (M), `Cardinality` (M), `QualifierTerm` (O), `UsageRule` (O+), `ConstraintLanguage` (O+), `BusinessTerm` (O+), `Example` (O+). The `Cardinality` indicates whether the ASBIE represents a not-applicable, mandatory, optional, or repetitive characteristic of the ABIE. The `UsageRule` describes specific conditions that are applicable to the ABIE. The `QualifierTerm` qualifies the Property Term of the associated Core Component Property in the associated Aggregate Core Component. The `ConstraintLanguage` formally indicates how the ASBIE is derived from the stored Core Component and Business Context. The `Example` is example of a possible value of the BBIE. The `CategoryCode` is always ASBIE in this case. **Note:** The section covered by this requirement is 3.7, rule DOC6.			
1125	**Core Component documentation**	**LAD**	**F**	**Clarity**
	Description: The core component (CC) must contain a structured set of documentation in the following pattern. `UniqueIdentifier` (M), `CategoryCode` (M), `DictionaryEntryName` (M), `Version` (M), `Definition` (M), `ObjectClass` (M), `PropertyTerm` (O), `UsageRule` (O+), `BusinessTerm` (O+). A `UsageRule` is a constraint that describes specific conditions that are applicable to the BBIE. The `CategoryCode` is always CCT in this case. **Note:** The section covered by this requirement is 3.7, rule DOC7. Although the M/O of the `ObjectClass` and `PropertyTerm` are not indicated in the guideline, mandatory are assumed.			
1150	**Element's documentation**	**LAD**	**F**	**Clarity**
	Description: Any element declaration must contain a structured set of documentation in the following pattern, `<documentation>`Dictionary Entry Name`</documentation>`. The Dictionary Entry name is an official name not the tag name. **Note:** The section covered by this requirement is 3.7.1, rule DOC8.			
1200	**Code list used documentation**	**LAD**	**F**	**Interoperability, Clarity**
	Description: For each UBL construct containing a code, the UBL documentation MUST identify zero or more code lists that MUST be minimally supported when the construct is used. The following documentation must be specified. `Prefix` (M), `CodeListQualifier` (M), `CodeListAgency`, and `CodeListVersion`. The `Prefix` is for example, '`cnt`' for Country Code List. The `CodeListQualifier` is for example '`ISO 3166-1`'. The `CodeListAgency` indicates maintainer of the code list such as '`6`'. The `CodeListVersion` is for example '`0.3`' **Note:** The section covered by this requirement is 3.7.1, rule DOC9. Although a test can be performed to validate the presents of these annotations, it may not be able to validate the values.			

1250	**XML names (element/attribute/type names)**	**LAD**	**F**	**Clarity**
	Description: XML names must follow the Oxford English dictionary. Element and type names must follow upper camel case, and attribute names must follow lower camel case as described in the ebXML architecture specification document [6]. **Note:** The section covered by this requirement is 4.1, rule GNR1, GNR9, and GNR10. From the run-time testing perspective, those names can be validated lexically with Oxford English dictionary. However, some valid Oxford words may not be appropriate for this purpose.			
1300	**XML names and dictionary entry name**	**LAD**	**F**	**Clarity**
	Description: UBL XML names must be taken from ebCC specification conformant dictionary entry name. However, any character used in the dictionary entry name that is not allowed in the XML name, as specified by W3C standard, must be omitted. **Note:** The section covered by this requirement is 4,1, rule GNR2 and GNR3. UBL allows some deviations from of the XML name from the dictionary entry name. If such deviation does not have a formal pattern, many false negative flags may result.			
1350	**Used of abbreviations or acronyms**	**LAD**	**F**	**Clarity**
	Description: Only abbreviations or acronyms documented in the Appendix B of the guideline are allowed in UBL XML names. **Note:** The section covered by this requirement is 4.1, rule GNR4.			
1400	**Singular form of XML names**	**LAD**	**F**	**Clarity**
	Description: UBL XML names must be in singular form unless the concept itself is of plural form by default, e.g., goods. **Note:** The section covered by this requirement is 4.1, rule GNR8.			
1450	**`ComplexType` name of ABIE**	**LAD**	**F**	**Clarity**
	Description: `ComplexType` name associated with an ABIE must follow the ABIE's dictionary entry name with separators omitted and 'Details' replaced with 'Type'. For example, an ABIE Dictionary Entry Name (DEN) 'Transport_ Equipment Seal. Details' is converted into a `complexType` name `TransportEquipmentSealType`. **Note:** The section covered by this requirement is 4.2.1, rule CTN1. This test will be performed based on the DEN provided in the annotated documentation.			
1500	**`ComplexType` name of the BBIE**	**LAD**	**F**	**Clarity**
	Description: `ComplexType` name associated with a BBIE must follow the BBIE's DEN with separators and Object Class Term omitted and 'Type' suffix to the Representation Term. For example, a BBIE DEN 'Account. Charge. Indicator' is converted into a `complexType` name `ChargeIndicatorType`. **Note:** The section covered by this requirement is 4.2.2, rule CTN2. This test will be performed based on the DEN provided in the annotated documentation.			

1550	**`ComplexType` name of the Primary Representation Term**	**LAD**	**F**	**Clarity**
	Description: `ComplexType` name associated with a Primary Representation Term must follow the name of the corresponding Core Component Type (CCT) with separators omitted and 'Type' suffix to the Primary Representation Term name. For example, a Primary Representation Term 'Amount' is based on a CCT 'Amount. Type', the corresponding `complexType` name is `AmountType`. **Note:** The section covered by this requirement is 4.2.3, rule CTN3. This guideline does not require annotated documentation for the Primary Representation Term. The DEN for checking this conformance may need to be obtained from another source.			
1600	**`ComplexType` name of the Secondary Representation Term**	**LAD**	**F**	**Clarity**
	Description: `ComplexType` name associated with a Secondary Representation Term must follow the name of the Secondary Representation Term with a 'Type' suffix to the Secondary Representation Term name. For example, the corresponding `complexType` name of the Secondary Representation Term 'Value' is `ValueType` (the Primary Representation Term of 'Value' is 'Numeric'. **Note:** The section covered by this requirement is 4.2.3, rule CTN4. The guideline does not require annotated documentation for the Secondary Representation Term. The DEN for checking this conformance may need to be obtained from another source.			
1650	**`ComplexType` and `simpleType` name of the CCT**	**LAD**	**F**	**Clarity**
	Description: A CCT may be mapped to a `simpleType` or a `complexType`. XML names associated with a CCT must follow the DEN of the CCT with the separators removed. For example, the `complexType` name of the CCT 'Quantity. Type' is `QuantityType`. **Note:** The section covered by this requirement is 4.2.4, rule CTN5 and STD2. See also #1950.			
1700	**Element name of the ABIE**	**LAD**	**F**	**Clarity**
	Description: Element name of an ABIE must follows the type to which it bounds with the 'Type' suffix omitted. For example, an ABIE 'Transport_ Equipment Seal. Details' would have an element name `TransportEquipmentSeal`. **Note:** The section covered by this requirement is 4.3.1, rule ELN1.			
1750	**Element name of the ASBIE**	**LAD**	**F**	**Clarity**
	Description: An ASBIE does not have its own `complexType`, rather it is an association between an element and `complexType` (which represents the corresponding ABIE). The element name associated with an ASBIE must be its DEN Property Term and Qualifiers; and, the Object Class Term and Qualifiers of its associated ABIE. All separators must be removed. Redundant words in the ASBIE property terms and qualifiers and the associated ABIE Object Class Term and Qualifiers must be dropped. For example, an ASBIE 'Person. Home_Address. US_ Address' is associated with the ABIE 'US_ Address. Details'. The corresponding global element name of the ASBIE is `HomeAddressUS`. **Note:** The section covered by this requirement is 4.3.3, rule ELN4. This test will be performed based on the DEN provided in the annotated documentation. There is no example provided in the UBL guidelines. It is formulated in this document. The truncation applies on the word 'Address' in the example.			

1800	**Attribute name**	**LAD**	**F**	**Clarity**
	Description: Use of attribute is very restricted in UBL. Attribute is used restrictively with the Supplementary Component of the CCT. As such, a UBL attribute name must be the Property Term and Representation Term of the DEN of one of the Supplementary Component with separators omitted. For example, the CCT 'Quantity. Content' would have attributes such as `unitCode`, `unitCodeListID`, `unitCodeListAgencyID`, and more. **Note:** The section covered by this requirement is 4.4, rule ATN1. Not all Supplementary Components need to map to attributes (see #1950). Either the example given in UBL is wrong or the DENs of the Supplementary Component given in the CCTS version 1.9 is wrong. In the CCTS, the 'Unit' is not part of the Property Term, so it should not be part of the UBL attribute name. See also #2550.			
1850	**Anonymous type**	**LAD**	**F**	**Extension, Maintenance (versioning)**
	Description: All types must be named. **Note:** The section covered by this requirement is 5.1.1, rule GTD1. This is equivalent to OAG recommendation #150 for the use of anonymous type.			
1900	**Typed by `xsd:any`**	**LAD**	**F**	**Interoperability**
	Description: The type, `xsd:any`, must not be used. **Note:** The section covered by this requirement is 4.1.1, rule GTD2.			
1950	**CCT mapping to XML Schema construct**	**LAD**	**P**	**Extensibility**
	Description: Generally, CCT must be mapped to the `complexType` construct with attributes reflecting its Supplementary Components. However, if the Supplementary Components is already encapsulated by the XML Schema's built-in data type, then the CCT must be mapped to a `simpleType`. For example, the CCT 'Date Time. Content' which has 'Date Time. Format. Text' as a Supplementary Component can map both the Content and Supplementary Components to the `xsd:dateTime` XML Schema data type. CCT `complexType` must be defined with `xsd:simpleContent`, which is an extension of the XML Schema built-in data type for its Content Component. The Supplementary Components that are represented by `xsd:attribute` must be associated with XML Schema built-in data type or user-defined `xsd:simpleType`. The user-defined `xsd:simpleType` must only be used when it is based on UBL standardized code list. The `use` attribute either `optional` or `required` must be specified in the schema for all the Supplementary Components. **Note:** The section covered by this requirement is 5.1.2, 5.1.3.4, and 5.1.3.5, rule STD1, CTD9 (both rule are the same), CTD10, CTD11, CTD12, CTD13, CTD14, and CTD15. See also #1650, #2250. The test coverage is partial because the test case cannot always determine whether the Supplementary Component should be a `simpleType` or a `complexType`. The example given on line 2025 and 2045 is wrong, it uses `xsd:restriction` instead of `xsd:extension`.			
2000	**CCT and xsd:restriction**	**LAD**	**F**	**Model validity**
	Description: CCT should not be constructed in XML Schema using `xsd:restriction` based on `xsd:simpleType`. **Note:** The section covered by this requirement is 5.1.2, rule STD3.			

2050	**CCTS constructs mapping to the `complexType`**	**LAD**	**U**	**Extensibility**
	Description: Certain types of Core Component constructs must be mapped to `complexType` with a name. **Note:** The section covered by this requirement is 5.1.3, rule CTD1. This rule is still fluid. The rule says that all classes in UBL must map to `complexType` constructs, but there is a comment indicating that that is not always the case.			
2100	**ABIE mapping to XML Schema construct**	**LAD**	**P**	**Extensibility**
	Description: Every corresponding `xsd:complexType` definition of a ABIE must use the `xsd:sequence` element with appropriate global element references, or local element declarations in the case of ID and Code, to reflect each property of its class as defined in the corresponding UBL model. **Note:** The section covered by this requirement is 5.1.3.1, rule CTD4. If assume that the test case does not linked to UBL model for validation, the testability is partial. The test case will only validate that the ABIE uses the `xsd:sequence` and relies on the annotated documentation in the schema itself to recognize that the construct is an ABIE.			
2150	**BBIE mapping to XML Schema construct**	**LAD**	**F**	**Extensibility**
	Description: Every BBIE `xsd:complexType` definition content model must use the `xsd:simpleContent` element, and the `xsd:simpleContent` element must consist of an `xsd:extension` element. The `base` for extension must be derived either from the CCT Primary Representation Term or from the data type of the Secondary Representation Term. **Note:** The section covered by this requirement is 5.1.3.2 rule CTD5, CTD6, and CTD8. This extension is a re-naming.			
2200	**Representation Term maps to `complexType`**	**LAD**	**F**	**Extensibility**
	Description: An `xsd:complexType` must be defined for Primary as well as Secondary Representation Term. **Note:** The section covered by this requirement is 5.1.3.3, rule CTD2 and CTD3. The test case would check that if the annotation indicates that the term is a Primary/Secondary Representation Term then the `xsd:complexType` must be used. The rule CTD3 is still fluid.			
2250	**ABIE and BBIE element declaration**	**LAD**	**F**	**Extensibility**
	Description: Classes in UBL model (i.e., ABIEs) must be defined as global elements bound to their corresponding `xsd:complexType`. **Note:** The section covered by this requirement is 5.2.2, rule ELD3. The test case would assume that classes mean ABIEs.			

2300	**Use of the CCT XML Schema construct**	**LAD**	**F**	**Model validity**
	Description: `xsd:simpleType` and `xsd:complexType` constructs of CCTs must only be bound to elements that represent Basic Core Component (BCC) or Basic Business Information Entity (BBIE). **Note:** The section covered by this requirement is 5.2.2, rule ELD4. This rule does not seem to match with #2150, which indicates that an BBIE has its own `xsd:complexType` definition which is an extension of the CCT. Hence, the BBIE does not need to associate directly with the CCT schema type definitions. This rule is still fluid.			
2350	**ASBIE element declaration**	**LAD**	**F**	**Model validity, Extensibility**
	Description: A global element based on ASBIE must be declared bound to the associated ABIE xsd:`complexType` definition. **Note:** The section covered by this requirement is 5.2.2,1 rule ELN3. The rule name should be changed to ELDx (an element declaration rule).			
2400	**Elements bound to CCT**	**LAD**	**F**	**Model validity**
	Description: For each CCT `xsd:simpleType,` an `xsd:restriction` element must be declared. **Note:** The section covered by this requirement is 5.2.2.3, rule ELD5. This rule enforces a grammatically ambiguous schema, an `xsd:restriction` cannot appear by itself. It also conflicts with #2000, which indicates that `xsd:restriction` must not be used with CCT.			
2450	**Code list import**	**LAD**	**F**	**Model validity**
	Description: The code list `xsd:import` element MUST contain the namespace and schema location attributes. **Note:** The section covered by this requirement is 5.2.3, rule ELD6.			
2500	**Empty element**	**LAD**	**F**	**Interoperability**
	Description: Empty element must not be declared in an XML schema. **Note:** The section covered by this requirement is 5.2.4, rule ELD7.			
2550	**User-defined attribute**	**LAD**	**F**	**Extensibility**
	Description: User-defined attribute should not be used. When used, it should only convey the CCT Supplementary Component. **Note:** The section covered by this requirement is 5.3.1, rule ATD1. See also #1800.			
2600	**Global attribute**	**LAD**	**F**	**Model clarity**
	Description: If a UBL `xsd:SchemaExpression` contains one or more common attributes that apply to all UBL elements contained or included or imported therein, the common attributes must be declared as part of a global attribute group. If the CCT's Supplementary Component `xsd:attribute` is common to all UBL elements, it must be declared as part of the XML schema global attribute group. **Note:** The section covered by this requirement is 5.3.2, rule ATD2 and ATD3.			

2650	**Supplementary component attribute declaration**	**LAD**	**P**	**Model validity**
	Description: For the CCT schema construct that uses `xsd:extension` element, the `xsd:extension` element must have `xsd:attribute` declared for each of its Supplementary Component. However, for those CCT XML Schema constructs that are based on `xsd:restriction` of `xsd:simpleType`, an `xsd:base` attribute must be declared and set to the appropriate `xsd:datatype`. **Note:** The section covered by this requirement is 5.3.3, rule ATD3, ATD4, and ATD5. The test coverage is partial because the rule ATD5 is not testable. The test case cannot determine whether an appropriate `xsd:datatype` is used to define the Supplementary Component.			
2700	**Schema location**	**LAD**	**F**	**Model validity**
	Description: The `xsd:schemaLocation` attribute must contain a persistent and resolvable URL. The URL must be an absolute path. **Note:** The section covered by this requirement is 5.3.4, rule ATD7 and ATD8. Although this rule is testable, the result is transient.			
2750	`xsd:nil` **attribute**	**LAD**	**F**	**Interoperability**
	Description: The XML Schema built-in nillable attribute (`xsd:nil`) must not be used for any declared UBL element. **Note:** The section covered by this requirement is 5.3.5, rule ATD9.			
2800	`xsd:any` **attribute**	**LAD**	**F**	**Interoperability**
	Description: The `xsd:any` attribute must not be used. **Note:** The section covered by this requirement is 5.3.6, rule ATD10.			
2850	**Code list creation and maintenance**	**L**	**NA**	**Reusability**
	Description: All UBL Codes must be part of a UBL or externally maintained Code List. The UBL Library should identify and use external, standardized code lists rather than develop its own UBL-native code lists. The UBL Library MAY design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists. If a UBL code list is created, it should be globally scoped (designed for reuse and sharing, using named types and namespaced Schema Modules) rather than locally scoped (not designed for others to use and therefore hidden from their use). All UBL-maintained or used Code Lists must be enumerated using the UBL Code List Schema Module. **Note:** The section covered by this requirement is 6, rule CDL1, CDL2, CDL3, CDL4, and CLD5. These are code list design principles rather than design rules.			
2900	**UBL Code list schema module name**	**L**	**P**	**Structural clarity**
	Description: The name of each UBL Code List Schema Module must be of the form: {Owning Organization}{Code List Name}{Code List Schema Module} **Note:** The section covered by this requirement is 6, rule CDL6. The partial test coverage is given for this requirement because the information necessary for the test may not be available, i.e., the test case would need to ask the user for Owning Organization, Code List Name, etc.			

2950	**UBL Code list has its own namespace**	**L**	**P**	**Extensibility**
	Description: An `xsd:import` element must be declared for every code list required in a UBL schema. **Note:** The section covered by this requirement is 6, rule CDL7 and NMS19. This requirement can also be interpreted as each UBL code list must has its own unique namespace. The test coverage is partial because the test case many not be able to fully verify the namespace uniqueness.			
3000	**UBL Code list namespace pattern**	**L**	**F**	**Model validity**
	Description: The namespace of UBL code list schema module must conform to this pattern - urn:oasis:ubl:codeList:<Code List. Identification. Identifier>:<Code List. Name. Text>:<Code List. Version. Identifier>:<Code List. Agency. Identifier>:<Code List. AgencyName. Text>. The Agency Identifier must be derived from UN/EDIFACT Data Element (DE) 3055 [24]. However, roles defined in DE 3055 must not be used. The token comprising the namespace must adhere to these rules: (1) Not white space, (2) Use only characters in the range 0-9, a-z, and A-Z, no special character, (3) If the version identifier has minor version identified, the minor version must be separated from the major version with a period (.). **Note:** The section covered by this requirement is 6, rule CLDX, CLDXX.			
3050	**UBL Code list importation**	**LAD**	**F**	**Validation and model clarity**
	Description: When UBL code list is imported, the `xsd:schemaLocation` attribute of the `xsd:import` statement must specify the complete URI identifying the code list schema. **Note:** The section covered by this requirement is 6, rule CLDXXX. That URI is the namespace.			
3100	**Maximum use of `xsd:simpleType`**	**L**	**NA**	**Interoperability**
	Description: The `xsd:simpleType` should be used as much as possible. **Note:** The section covered by this requirement is 7.1, rule GXS3.			
3150	**W3C schema namespace abbreviation**	**LAD**	**F**	**Structural clarity**
	Description: The ‘xsd’ must be used as namespace abbreviation for the W3C meta-schema. That is the following must be declared in the `xsd:schema` element “`xmlns:xsd="http://www.w3.org/2001/XMLSchema"`”. **Note:** The section covered by this requirement is 7.2, rule GXS4.			
3200	**No `xsd:substitutionGroup`**	**LAD**	**F**	**Interoperability**
	Description: The `xsd:substitutionGroup` is inconsistent the UBL guiding principle; hence, it must not be used. **Note:** The section covered by this requirement is 7.3, rule GXS5.			

3250	**The `xsd:final`**	**LAD**	**NA**	**Extensibility**
	Description: The `xsd:final` attribute must be used to control the extension. **Note:** The section covered by this requirement is 7.4, rule GXS6. The testability is not possible, because it is not possible to reason where the `xsd:final` should be used.			
3300	**No `xsd:notation`**	**LAD**	**F**	
	Description: The `xsd:notation` data type must not be used. **Note:** The section covered by this requirement is 7.5, rule GXS7.			
3350	**The `xsd:all`**	**LAD**	**F**	**Interoperability**
	Description: The `xsd:all` element must not be used. The semantics of the `xsd:all` element is inconsistent with the UBL data-centric scenarios/transactions. **Note:** The section covered by this requirement is 7.6, rule GXS8.			
3400	**The `xsd:choice`**	**LAD**	**F**	**Interoperability**
	Description: The `xsd:choice` element compositor must not be used. **Note:** The section covered by this requirement is 7.7, rule GXS9.			
3450	**The `xsd:include`**	**LAD**	**F**	**Model validity**
	Description: The `xsd:include` element must only be used in the Control Schema to avoid circular reference. **Note:** The section covered by this requirement is 7.8, rule GXS10. See also #850. The test case would identify a Control Schema and its child Internal Schema from their namespaces.			
3500	**The `xsd:union`**	**LAD**	**F**	**Extensibility**
	Description: The `xsd:union` must be used only with Code List. **Note:** The section covered by this requirement is 7.9, rule GXS11. The test case would identify a code list from its namespace association.			
3550	**The `xsd:appinfo`**	**LAD**	**F**	**Interoperability**
	Description: The `xsd:appinfo` can cause security threat. Hence, UBL only recommends using it for non-normative (documentation) information for the purpose of wider interoperability. **Note:** The section covered by this requirement is 7.10, rule GXS12. The test case will warn of any use of `xsd:appinfo` without considering the information specified in order to obtain the full test coverage.			

3600	**The `xsd:extension` and `xsd:restriction`**	**LAD**	**NA**	**Extensibility**
	Description: The `xsd:extension` and `xsd:restriction` may be used where appropriate. **Note:** The section covered by this requirement is 7.11, rule GXS13. The testability is not possible, because it is not possible to reason where the two elements are appropriate.			

Note:

Requirements in The section 3.6.4 have not been extracted, because it is uncertain right now whether we need to test these core standard schemas. If these schemas do not evolve that much, it is not worth to write test requirements and test cases.

Rule GXS2 in The section 3.7.2 has not been extract. It indicates that two versions of schema must be provided by UBL, one is a fully annotated schema and the other is run-time schema which has all documentation stripped out. It is expected that only the fully annotated version will be tested for design rules conformance, because the run-time schema can be automatically generated from the fully annotated schema.

Rule GNR5, GNR6, GNR7 in The section 4.1, which talks about creating and maintaining the list of allowable abbreviations and acronyms, have not been extracted.

Rule CTD7 in The section 5.1.3.2, this rule is already captured by the XML schema grammar itself.

Rule CDL8 in The section 6, this is a usage recommendation rather than a schema design rule.

11. XML.GOV Developers Guide

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Naming convention**	**LAD**	**P**	**Common symbolic syntax, Structural clarity**
	Description: The camel case convention should follow the one defined in the ebXML technical architecture document. Element and type names should use the UpperCamelCase convention. Attribute name should use the lowerCamelCase convention. **Note:** The section covered by this requirement is 3.1.			
100	**Acronym and abbreviation usages**	**LAD**	**F**	**Structural clarity**
	Description: The summary of this guideline includes (1) acronym should not be used, when used should be in upper case, (2) abbreviation should not be used, (3) the acronym used must be expanded in the `xsd:documentation` tag, (4) the underscores (_), periods (.) and dashes (-) must not be used. **Note:** The section covered by this requirement is 3.2.			
150	**Use of ISO 11179 data element definition**	**LAD**	**F**	**Structural clarity**
	Description: The XML element, data type, and attribute names should follow the ISO 11179 part 5 conventions [25], i.e., Object Class. Property Term. Representation Term. The guidelines recommend that the ISO 11179 name be concatenated according to UpperCamelCase for the element/type, and to lowerCamelCase for the attribute. **Note:** The section covered by this requirement is 3.3.2.			
200	**Meaningful component name**	**LAD**	**NA**	**Structural clarity**
	Description: This is a recommendation for the process to come up with a meaningful name. **Note:** The section covered by this requirement is 3.3.3. Although trivia test can be performed, for example, to detect verb phrase or prepositional phrase, it is very difficult, if possible at all, to test whether the name is sufficiently meaningful. Hence, we associate the value 'NA' with the test coverage.			
250	**Schema conformance to W3C standard or DTD conformance to ISO 8879**	**LAD**	**F**	**N/A**
	Description: This requirement is considered outside of the boundary of the schema design quality. It is put here for documentation purpose. The requirement, in fact, also includes the recommendation to transition to the XML schema from DTD. However, the rationale to determine the transition is arbitrary and is not testable. **Note:** The section covered by this requirement is 4.1. Generally, any schema or DTD should conform to these two standards.			

300	**Schema development methodology**	**LAD**	**NA**	**Validation and model clarity**
	Description: The guideline recommends object-oriented modeling (in place of relational model) of business process using UML or UMM methodology to drive the schema development. The schema development should be a team effort involving the domain expert, an IT specialist, and the manager. **Note:** The section covered by this requirement is 4.2. This requirement is not testable because the recommended modeling methodologies have arbitrary effect to the resulting schema structures (i.e., the methodologies do not signify schema best practices).			
350	**Capturing metadata**	**LAD**	**P**	**Validation and model clarity, structural clarity**
	Description: The guideline recommends that metadata be recorded as much as possible in the schema using the annotations and restrictions (or other schema constructs) and in the DTD using comments. Optionally, a separate guideline document may replace the schema annotations. In addition, two versions of a schema may be created, a verbose one with all the annotations and a lean without any annotations for validation purpose. **Note:** The section covered by this test requirement is 4.3. Note that this test requirement is subjected to quite limited test coverage. Annotation metadata can be checked for each element. However, other types of metadata can be check in a limited way. OAGIS requirement #200, the use of weak typing, can be considered as part of this requirement. In fact, requirements falling into the validation and model clarity rationale may be considered part of this requirement.			
400	**Capturing application specific metadata**	**LAD**	**P**	**Interoperability**
	Description: An XML payload should not be used to capture the programming statements/expressions. It may capture software initialization parameters but this should be in a physically separate file from the business payload. **Note:** The section(s) covered by this test requirement is 4.3.1. The conformance to this test requirement can be better detected if taking into account sample instances. The test coverage is partial because it may not be possible to detect all programmatic patterns.			
450	**Capturing XML component definition**	**LAD**	**F**	**Interoperability**
	Description: All type definitions and element declarations must have a definition specified in the annotation tag. URL pointer to additional information may be specified. An extension to the documentation tag is a recommended way to associate this information. **Note:** The section covered by this test requirement is 4.3.2. This requirement is a child of the requirement #350. There is no standard way of specifying this information. Until then, the full test coverage means that all type definitions and element declarations are checked to have an annotation child element.			

500	**Enumerations**	**L**	**P**	**Validation and model clarity**
	Description: Stable code lists should be enumerated in the schema. Unstable code list should be annotated with URI pointing to the documentation. **Note:** The section covered by this test requirement is 4.3.3. The test coverage is partial because it may not be possible to identify all the data fields that should be enumerated. See also OAGIS requirement #350 and NIST B2B Testbed requirement #300. A similar approach to these two test requirements can be taken to verify the conformance (using a code list reference table) to this requirement.			
550	**Version documentation in DTD**	**LAD**	**F**	**Interoperability, Maintenance**
	Description: In a DTD, the version number should be specified in the first comment or as a fixed attribute in the root element. **Note:** The section covered by this test requirement 5.1.1.			
600	**Version documentation in Schema**	**LAD**	**F**	**Interoperability, Maintenance**
	Description: In an XML schema, the version number should be specified using the version attribute of the `xsd:schema` element. **Note:** The section covered by this test requirement is 5.1.2. Using this versioning approach, the schema version associated with an instance may not be evident, unless a target schema version is specified in the instance. This is similar to the requirement in the section 5.1.3 that an associated schema version must be specified in the stylesheet. This requirement is not documented here because it is not in the scope of the schema design quality. See also Roger Costello XML Schema Versioning [8].			
650	**Element Vs. Attribute**	**LAD**	**P**	**Model validity, Structural clarity**
	Description: Attributes should be used to convey metadata to understand the business value associated with the element. Attributes should not contain a long string value but rather token or number types of values. Attributes should describe information units that will not be further subdivided or extended. Attributes should contain metadata that is applicable to the whole element's content. Putting both business value and metadata in the attribute will result in ambiguity about which one is the metadata and vice versa. A long string value in the attribute will be white space truncated by the parser. **Note:** The section covered by this test requirement is 6. This test requirement cannot be effectively verified with schema being the only input. Instance data would allow reasoning based on evidence about whether the attribute construct is used according to this guideline.			

12. KIEC XML Guideline

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**XML tag names should follow ISO 11179**	**LAD**	**P**	**Common symbolic syntax**
	Description: Check that the (element) tag names conform to the ISO 11179 recommendation. Generally, ISO 11179 recommendations are, for example, an element name must start with capital letter, an attribute name must start with a lower case letter, names must be typically in singular form unless looping is required, and names should use only Verb, Noun, or Adjective. **Note:** The sections covered by this test requirement are 2.1.1, 2.1.7, 2.1.11, and 2.1.12. The section 2.1.10 states that XML tag names, which are used by XML library, should be unique. This can be difficult for an XML designer. Typically, only global element or type must be unique within a single namespace. The schema validators may not detect this issue. Preliminary experiment indicates that the XSV [16] does validate this issue, while the Xerces [17] does not. Similar to the test requirement #200 in the GCSS-AF guide, the test coverage of this requirement is partial because the name may or may not be parsed into separate substrings correctly and words have multiple functionalities resulting in a lot of false alarms and some false positive.			
100	**UID assignment range**	**LAD**	**P**	**Structural clarity**
	Description: KIEC recommends ranges of UIDs associated with the level of information construct. See the table in section 2.2.2 of the guideline. **Note:** The section covered by this test requirement is 2.2.2. In order to verify this requirement, the test must be able to map the construct in the XML schema to the construct in the ebXML Core Component hierarchy of constructs. For example, identify which construct in XML schema is Basic Business Information Entity, or Basic Core Component. The current version of this document quantifies the test coverage of this requirement as partial because it is uncertain whether such mapping can be done precisely (unlike UBL it is annotated in the schema).			
150	**UID incremental sequence**	**LAD**	**P**	**Structural clarity**
	Description: (a) UID is assigned alphabetically for each layer. In case a new information entity is added, a number, which is in the middle of the before and after the new entity, is assigned. (b) For composite Business Information Entities (BIEs), UID is sequentially assigned based on the composite core component to which the composite business information elements refer. (c) For messages of domestic (Korean) standard electronic documents, UID is first classified based on business process and then assigned sequentially in units of 100. For vertical standard, UID is assigned by adding 5 sequentially to the message number. (d) In case UID is assigned by developing new information entities or components, or updating existing components, the assignment must be made in consultation with KIEC. **Note:** The sections covered by this test requirement are 2.2.3, 2.2.4, 2.2.5, 2.2.6. The test coverage is partial. The requirement in (a) is testable by creating an alphabetical ordering of the terms and verifying the sequential numbers. The requirement in (b) is testable if the relationship between the BIEs and their corresponding Core Component can be identified. The requirements in (c) and (d) are not testable. *** The price we have to pay versus its value could make this requirement not worthwhile to test.			

200	**UID uniqueness**	**LAD**	**F**	**Interoperability**
	Description: All XML elements in the XML schema can have an associated Id attribute. According to XML schema standard, any schema parser should check for the Id's uniqueness. However, that validation is limited to a single file. This test requirement must check for uniqueness across schemas included in a standard. **Note:** Note that all schemas included in a standard must be supplied for the test coverage to be full.			
250	**Use of default namespace**	**LAD**	**F**	**Structural clarity**
	Description: Default namespace (no-prefix) can be assigned to either the XML schema namespace (option 1) or the target namespace (option 2). **Note:** The section covered by this test requirement is 2.3.1. KIEC indicates that the advantage of option 2 is that it allows a no-target-namespace schema to be created. However, this generally is not a good practice for standard development. The disadvantage of this option as described by KIEC is indicated as a good practice in OAGIS because it differentiates the elements in the same namespace as the target namespace (without prefix) from elements in imported namespaces (with prefixes). KIEC describes that the advantage of option 1 is that it reverses the disadvantage in option 2 (i.e., all referenced elements have prefixes). As stated earlier, the disadvantage of option 2 is viewed as an advantage by others. Then KIEC indicates that the disadvantage of option 1 is that a no-target-namespace schema cannot be created with this option. Again, other organizations suggest that the no-target-namespace schema generally should not be used. However, most industry standards including OAG, XML Common Business Library (xCBL) [26], and Open Travel Alliance (OTA) [26] use option 2. Therefore, option 2 is viewed as the best practice for the default namespace usage.			
300	**Extension**	**LAD**	**P**	**Maintainability**
	Description: Extension should be done with these principles. (a) The extension should be compatible with others. (b) Version upgrades should be done without much extra effort. (c) The user-defined extension should eliminate the need for any core XML schema files modification, and should not be dependent on any of those files. (d) Any extension to the current standard should be done in new namespaces. **Note:** The section covered by this test requirement is 2.4.2. The test coverage is partial. The principle in (a) is too vague for test case generation. The principle in (b) could be linked to other test requirements associated with Maintainability rationale such as the feature regression in the #250 of the OAGI table. The principle in (c) may be checked by verifying the differences between the current (agreed upon) standard specification and the standard specification used by a user. This will notify the user of any unexpected modification. The principle in (d) can be tested.			
350	**Use of UserArea extension**	**LAD**	**F**	**Interoperability**
	Description: The UserArea extension is not recommended. The UserArea extension may be viewed as any element definition that is associated with the XML Schema data type `any`, which allows any arbitrary well-formed XML content to be specified. **Note:** The section covered by this test requirement is 2.4.3. This test requirement maybe suitable for only Business-to-Business (B2B) standards because in the Application-2-Application (A2A) integration there are typically needs for such arbitrary extension.			

400	**Allowable plural tags**	**LAD**	**F**	**Structural clarity**
	Description: KIEC at the present allows 12 plural components including Addresses, Attachments, Contacts, DocumentIds, DocumentReferences, Locations, Parties, PartyReferences, Prices, Properties, Ranges, and Taxes. **Note:** The section covered by this test requirement is 2.5.4. There are two interpretations associated with plural forms as noted in the NIST B2B Testbed recommendation #700. The interpretations associated with these terms are not documented in this version of the guideline.			
500	**Controlled object class term**	**LAD**	**F**	**Structural clarity**
	Description: KIEC defines controlled-object-class terms including Container, Delivery, Location, Message, Order, Organization, Packaging, Party, PaymentTerms, Person, Price, Project, Quote, Shipment, Tax, Transaction, Transportation **Note:** The section covered by this test requirement is 3.2.6. These are quite a limited set of terms comparing with Nouns and Components defined in OAGIS.			
550	**Identifier design pattern**	**LAD**	**P**	**Interoperability**
	Description: This guideline recommends an item identifier, document identifier, document reference identifier, classification code schemes. **Note:** The sections covered by this test requirement are 3.3.5, 3.3.6, 3.3.7, and 3.3.8. See also #250 in the NIST recommendation table. The item identifier, document identifier, and classification code schemes have the same pattern, which require containers for responsible/assigning organization and revision numbers in addition to the code/id itself. The document reference identifier pattern has document date and time, description, name, status, usage, and note. The test coverage is partial because the test case may not be able to recognize all the constructs that should follow these patterns.			
600	**Item property design pattern**	**LAD**	**P**	**Interoperability**
	Description: KIEC recommends an XML structure for representing item property derived from IMD segment of EDIFACT [28]. This recommendation translates into a requirement that any recognized Item or object element having a pattern of arbitrary property assignments to follow the recommended property pattern. **Note:** The section covered by this test requirement is 3.3.9. The recommended pattern is a plural ‘properties’ element containing 0 or more property elements. The property element contains property value, description, effective period, UOM, qualification, note, party reference and a user area. The test coverage is partial because it may not be possible to detect all elements pertaining to this pattern.			

650	**Item design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: KIEC recommends an XML structure for representing an item component. This recommendation translates into a requirement that concepts similar to the Item component follow this pattern. **Note:** The section covered by this test requirement is 3.3.10. This requirement seems to be inconsistent with the requirement #500. The 'Item' is not included in the controlled object class term. It is also inconsistent with the requirement #400 because it contains a 'ClassIds', which is not listed as allowed plural tags. The item pattern should contain item id, status, name, commodity name, class ids, item type, properties, definition, serial number, parent serial number, material details, lot details, attachments, and user area. The test coverage is partial because it may not be possible to detect all elements pertaining to this pattern.			
700	**Transportation unit design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: KIEC recommends an XML structure for representing a physical container/shipment unit. This recommendation translates into a requirement that concepts having similar or same semantics as a physical transportation unit, e.g., packaging in OAGIS to contain information listed in this pattern. **Note:** The section covered by this test requirement is 3.3.11. This requirement maps to the Packing, a controlled object class term in the requirement #500. The transportation unit pattern should contain container Id, sealed container Id, container type, freight item Id, shipper cost, tracking number, material, sequence identification, shipper cost, total Id of freight, temperature of the transportation unit, description of the unit, shipping note associated with the unit, charges, and document references. The test coverage is partial because it may not be possible to detect all elements pertaining to this pattern.			
750	**Transportation information design pattern**	**LAD**	**P**	**Extensibility, Interoperability**
	Description: KIEC recommends an XML structure for representing information about transportation (transit information). This recommendation translates into a requirement that any information entity (e.g., shipment) seeking to indicate its routing and transit information should contain information identified in this pattern. **Note:** The section covered by this test requirement is 3.3.12. This requirement maps to the Transportation, a controlled object class term in the requirement #500. Note an inconsistency with the requirement #400, the 'ShippingInstructions' is not an allowed plural tag. This pattern also contains a 'UserArea' element, which is inconsistent with the requirement #350, which discourages the use of the UserArea extension. The transportation information pattern should contain Id, mode and means of transportation, date time of the transportation, scheduled delivery, carrier information, transit direction, service level, shipping instructions, route taken, location, transportation equipment, and a UserArea extension. The test coverage is partial because it may not be possible to detect all elements pertaining to this pattern.			

13. ASC X12 Reference Model for XML Design

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Versioning**	**LAD**	**F**	**Maintainability, Interoperability**
	Description: This guideline recommends the use of a unique name space for each release. **Note:** The section covered by this test requirement is 7.1.2. Although it is technically possible to test conformance to this requirement, all previous schema releases must be provided in order the make the conformance absolute. The question is whether all schemas use the new namespace regardless of whether there are changes from the previous release. Currently, we take a holistic interpretation. However, when components are maintained individually in the schema this practice may not be practical. In addition, if components are flagged as changed without in fact any changes, this practice could confuse developers or computer agents resulting in increasing software maintenance cost.			
100	**Spelling**	**LAD**	**P**	**Structural clarity, Common symbolic syntax**
	Description: Oxford English is recommended for validating the terms used. **Note:** The section covered by this test requirement is 7.1.3. The test coverage is partial under the assumption that terms may not be accurately parsed and validated with the dictionary.			
125	**Size of document**	**LAD**	**NA**	**Performance**
	Description: Due to concern about the memory usage in software that load the whole document into the memory, it is recommended that the document be of a reasonable size. **Note:** The section covered by this test requirement is 7.1.4. Since the requirement is specified without quantifiable information, this requirement is not testable.			
150	**Deepest level of nested elements**	**LAD**	**F**	**Performance**
	Description: The guideline recommends that the deepest level of child element should be in the neighborhood of 10 from the document root. **Note:** The section covered by this test requirement is 7.1.4. The test coverage is determined to be full if the recommended level is discrete at 10. This practice may not affect the software performance directly, but rather indirectly from the time necessary to implement software for a deep data structure.			

200	**Naming convention**	**LAD**	**P**	**Structural clarity, Common symbolic syntax**
	Description: UpperCamelCase is to be used for element and lower camel case is to be used for attribute. It also recommends other aspects of naming conventions as specified in the section 4.3 of the ebXML Technical Architecture V1.0.4. **Note:** The section covered by this test requirement is 7.1.5. The requirement only indicates the naming convention for element and attribute; hence, the respective test cases will ignore the type definition naming. The test coverage is indicated as partial because the test case may not be able to perfectly parse the name and checking for conformance.			
250	**Verbose attribute name**	**LAD**	**F**	**Performance, Structural clarity**
	Description: An attribute name should not contain name(s) of parent element(s). If applying the attribute from within an `AttributeGroup`, then the contextual value of the attribute name should be contained within the attribute name. An attribute can be viewed as an adjective to the element (which is a noun). This can result in a rule that attribute name must be an adjective. **Note:** The section covered by this test requirement is 7.2.5, under the "Benefits of using Attribute" subsection. Since the names of attributes within an `AttributeGroup` may contain contextual value (e.g., name of the parent element), the attribute names within an `AttributeGroup` will not be checked.			
300	**Use of namespace**	**LAD**	**F**	
	Description: Hierarchical namespaces are recommended. A single namespace holding all common terminologies among all functional subcommittees (or other logical grouping) is recommended. A single namespace should be used for each functional subcommittee. It is ambiguous to whether the document level schema is recommended to have its own namespace or to have the same namespace as the associated functional subcommittee. **Note:** The section covered by this test requirement is 7.3.11. This test requirement suggests that a no target namespace schema should not be used and all schemas should have a target namespace. In order to verify the compliance to this test requirement, the user needs to indicate the type of schema -- core, subcommittee, or document. Hence, the test coverage is full only if all necessary information is specified.			
350	**Processing instruction**	**LAD**	**F**	**Interoperability**
	Description: Processing instructions should not be used both in schema and in instance documents. The rationale is that the processing instruction usually contains information that should normally be included in the XML data. **Note:** The section covered by this test requirement is 7.2.8. Rather than using processing instruction, such information should be captured in the XML data structure itself.			
400	**String type element design pattern**	**LAD**	**F**	**Validation and model clarity**
	Description: A required element of type string should be defined with a restriction to have at least one character. **Note:** The section covered by this test requirement is 7.3.2. Normally, XML schema considers any Unicode character to be a valid string (even a white space). X12 is still discussing whether to require that the one character to be non-white-space. In fact, this constraint should also apply to optional element, because the constraint should be activated whenever the element is present. That is, it should apply to all elements of type string, unless the element is nillable.			

450	**Non-nillable required element**	**LAD**	**F**	**Model validity**
	Description: Mandatory element should not be nillable. **Note:** The section covered by this test requirement is 7.3.2.			
500	**Nillable optional element**	**LAD**	**F**	**Model validity**
	Description: Optional element (minOccur =0) shall be declared nillable. **Note:** The section covered by this test requirement is 7.3.2.			
550	**String type attribute design pattern**	**LAD**	**F**	**Validation and model clarity**
	Description: A required string typed attribute should be defined with a restriction to having at least one character. **Note:** The section covered by this test requirement is 7.3.2.See also #400.			
600	**Use of the mixed content**	**LAD**	**F**	**Validation**
	Description: Mixed content should not be allowed in documents designed solely for data exchange. Mix content allows both textual content and child elements to appear. In this case, the textual content cannot be validated or constrained to a particular data type. **Note:** The section covered by this test requirement is 7.3.3. A 'Description' element may be used in place of the mix content type.			
650	**Use of the wildcard element and attribute**	**LAD**	**F**	**Model clarity and validation**
	Description: The recommendation is not to use the `any` element or `anyAttribute` attribute. The rationale is that this allows invalid data to be inserted. The moderate recommendation is also that the entities instantiated under the `any` or `anyAttribute` should come from other namespaces (than the one in the target namespace). **Note:** The section covered by this test requirement is 7.3.3. The moderate recommendation suggests that whenever the `xsd:any` or `xsd:anyAttribute` is used as type of element, a `namespace` attribute should be used and that the attribute value must not be `##any` or `##local`.			
700	**Use of the abstract types**	**LAD**	**F**	**Model clarity and validation**
	Description: The recommendation is not to use the abstract type -- although it suggests that such functionality may be used for specifying a constraint to be at least one-of. The document cites that this feature makes the instance document ambiguous to what element will actually be exchanged. **Note:** The section covered by this test requirement is 7.3.3. This is analogous to the OAGIS use of `substitutionGroup`. However, OAGIS still uses the abstract type to indicate the intention that such type is not instantiable (not as at-least-one-of constraint). The OAGIS practice may be a preferred method. The testability is indicated as full with the assumption that the abstraction type should not be used without exception.			

750	**Use of the group feature**	**LAD**	**F**	**Model clarity and validation, Maintainability**
	Description: Although the uses of complex type group and attribute group promote reuse, the document argues that too much reuse can complicate maintenance. In addition, the functionality of the group feature is very similar to that offered by the type definition. The recommendation is not to use the group feature at all and instead defines as many types as necessary. **Note:** The section covered by this test requirement is 7.3.3. Although the recommendation to avoid using too many schema features to improve understandability is sound, the rationale that reuse can complicate maintenance is unclear and is counter-intuitive.			
850	**Group and Type redefinition**	**LAD**	**F**	**Maintainability**
	Description: The group redefinition feature allows the ‘group’ or ‘type’ definition to be redefined. Since the whole content definition has to be redefined, this can affect other dependencies (e.g., extension). The document also argues that this functionality is under defined in the XML Schema specification. The recommendation is not to use the redefinition functionality. **Note:** The sections covered by this test requirement are 7.3.3. and 7.3.4. This is called feature regression in the OAGIS term. That is the redefinition has to repeat the whole content model. See #250 in OAGIS table.			
800	**Substitution Groups**	**LAD**	**F**	**Structural clarity**
	Description: The substitution group allows the content model to be flexible. Elements can be declared a substitute for the “head” element. Those elements can appear wherever the head element appears in the content model. The document argues that the feature allows the content model to be excessively flexible in that anybody can alter the content model substantially using this feature. Consequently, the use of substitution group is not recommended. **Note:** The section covered by this test requirement is 7.3.3. This is conflicting with the OAGIS extensibility guideline which recommends the use of substitution group to declare extension at design time rather than using the `xsi:type`.			
900	**Type definition**	**LAD**	**F**	**Extensibility**
	Description: The document recommends the use of named type (i.e., global type) instead of the anonymous type (i.e., locally defined type). **Note:** The section covered by this test requirement is 7.3.4. This recommendation is the same as that of OAGIS design document that also recommends the use of globally defined type. See #150 in OAGIS table.			
1000	**Type derivation**	**LAD**	**F**	**Extensibility**
	Description: The document recommends that type derivation based on `restriction` and `extension` be allowed. **Note:** The section covered by this test requirement is 7.3.4. Test cases generated for this test requirement may verify that the `xsd:schema` does not have the `finalDefault` attributed defined and that type definitions do not have the attribute `final` defined. This requirement is less restrictive than the OAGIS recommendation. OAGIS recommends that type restriction be used only with simple type due to feature regression associated with restriction on complex type. See #250 in OAGIS table. It also relates to #850 in this table.			

950	**Built-in simple type**	**LAD**	**NA**	**Interoperability**
	Description: The document recommends that built-in simple types defined by XML schema specification should be used, and that a subset of these types should be used as will be defined in X12's XML equivalent of X12.6. **Note:** The section covered by this test requirement is 7.3.4. The requirement is not possible to test because it may not be possible to identify which type definition should override the built-in simple type. In addition, the XML equivalent of X12.6 data type has not been constructed. This requirement is opposite from the OAGIS recommendation that the built-in schema simple types as weak typing. That is they should not be used in the aggregate and document level schemas.			
1050	**Type substitution**	**LAD**	**F**	**Interoperability**
	Description: Type substitution feature allows base types to be replaced by derived types. The document argues that this can cause problems for to an application processing an instance document in which an unknown derived type is used. Therefore, type substitution should be disallowed. **Note:** The section covered by this test requirement is 7.3.4. The test cases derived from this test requirement will either verify that the `xsd:schema` element has an attribute `blockDefault` equals `#all` or that all type definitions have an attribute `block` equals `#all`.			
1100	**Locally defined element and attribute**	**LA**	**F**	**Maintainability**
	Description: The document recommends that elements and attributes be locally defined except the root element. The rationale is that the global element and attribute can create name-clashing problem, although they are more extensible and reusable. **Note:** The section covered by this test requirement is 7.3.5. This requirement is different from the OAGIS practice. OAGIS uses global elements to satisfy its favored `substitutionGroup` extensibility mechanism. See #300 of OAGIS. It is also opposite to UBL practice which requires that all elements are global.			
1150	**Use of default or fixed value**	**LAD**	**F**	**Interoperability**
	Description: The default value feature used with attribute or element can present problems when a schema is not present (the processor needs to obtain the default value from the schema). The recommendation is to disallow these two features. **Note:** The section covered by this test requirement is 7.3.6. There are different processing behaviors associated with the attribute default and element default. The processor assigns a default value to the attribute when the attribute is not present at all in the instance document. On the other hand, the processor assigns default value to the element when the element appears with empty content in the document. If the element does not appear then it is null.			

1150	**Uniqueness constraint**	**LAD**	**F**	**Interoperability**
	Description: The document describes three different features of similar functionalities (linking entities and uniqueness constraint) including ID with IDREF, Key/KeyRef with Uniqueness, and XLink with XPointer. The document recommends the Key/KeyRef with Uniqueness as of now. The rationale is that the ID/IDREF has major limitations including the strict format of the ID value, which must start with a character and must be alphanumeric character except underscore. In addition, the ID values have to be unique for all elements and attributes within a document. XLink and XPointer are relatively immature and inherit some limitations from the ID/IDREF. Key can be duplicated (e.g., Key/ID of a customer can be the same as Key/ID of an invoice). The uniqueness constraint of Key can be specified with respect to a target element or attribute. **Note:** The section covered by this test requirement is 7.3.7. Two caveats associated with the Key/KeyRef feature are described. Some parsers may not validate the XPATH expressions in the `xsd:field` and `xsd:selector` field. In addition, when the uniqueness and the reference are required the element or attribute specified in the `xsd:field` must be mandatory. Test cases for this requirement should take into account these two caveats.			
1200	**Use of the `xsd:annotation` vs. the XML comment**	**LAD**	**F**	**Structural clarity**
	Description: Although the document argues that extensive use of the `xsd:annotation` can reduce the schema readability and increase processing time, use of the `xsd:annotation` element is recommended for all type definition for clarity. It also recommends that the format and structure of the annotation follow the meta-data described in Section 6 of the guideline. Use of XML comment is not recommended. **Note:** The section covered by this test requirement is 7.3.8. Use of the `xsd:annotation` element is good because it can be processed by the application and XSLT. Test cases associated with this test requirement will flag any use of XML comment.			
1250	**Use of the Notation**	**LAD**	**F**	**Interoperability**
	Description: Since there is no standard way to process a 'Notation', use of the 'Notation' can cause interoperability problem and hence is discouraged. Notation indicates the type of external files associated with the XML document. The document argues that an application must know how to process the file anyway; consequently, the Notation is deemed unnecessary. **Note:** The section covered by this test requirement is 7.3.8. The argument about the unessential of the Notation may be too restrictive. Although the application needs to know how to process the file, it needs to know first what type of file it is and that is what the Notation tells (as analogous to the MIME type).			
1300	**Use of the `xsd:documentation` vs. the XML Comment**	**LAD**	**F**	**Structural Clarity**
	Description: Use of the `xsd:documentation` element is recommended. This is similar to the requirement #1200. The `xsd:documentation` element is a child element of the `xsd:annotation` element. **Note:** The section covered by this test requirement is 7.3.8. Test cases associated with this test requirement will flag any use of XML comment.			

1350	**Use of the `xsd:appInfo` or DTD's `ProcessingInstruction`**	**LAD**	**F**	**Interoperability**
	Description: Te `xsd:appInfo` of the XML schema serves the same functionality as the `ProcessingInstructions` of the DTD. Use of the `xsd:appInfo` is considered risky because there is no mature recommendation yet to how the XML processor will pass the `xsd:appInfo` to the application; hence, this feature is not recommended. **Note:** The section covered by this test requirement is 7.3.9.			
1400	**Use of the Length**	**LAD**	**F**	**Interoperability**
	Description: The document does not recommend the use of fixed or maximum length constraint except for the case of coded value. **Note:** The section covered by this test requirement is 7.3.10. Usage of the 'Length' facet will be rejected because of this test requirement, unless it is recognized that coded value is expected. Although the test coverage is full, a number of false positives are expected.			

14. AEX Design Guidelines

ID	Test Requirement	Schema type	Test Coverage	Rationale
50	**Conformance to naming conventions.**	**LAD**	**P**	**Validation and model clarity, Extensibility**
	Description: Use of upper camel case is recommended. To ensure consistency, an upper-camel-case tag should be parsed to spell-check each sub-string and ensure the validity of each substring – words, acronyms, or abbreviations—against the published table of terms. An all upper case substring may be recognized as a specific acronym and ignored or checked against a list of allowable acronym. **Note:** See "Using AEX" The section 2.11. Along with the cfiXML specifications is a published list of the terms used including abbreviations and acronyms. Names can be parsed to verify that their component terms are included in the published list; however, in some circumstances there may be ambiguity in the parsing. A test can flag these situations, which can then be examined by a person. One convention the AEX project uses which is different from others is that an element name is lower camel case while the type name is upper camel case.			
100	**Conformance to choice ordering conventions**	**LAD**	**F**	**Validation and model clarity, Extensibility**
	Description: Check to see that in choice or sequence lists the items are ordered alphabetically with the exception of those things for which a "logical ordering" makes more sense, such as addresses. **Note:** See "Using AEX" The section 2.11. This requirement is only partially testable because there is no listing of exceptions to the rule of alphabetical listings, yet the exceptions are allowed for. A tool can flag occurrences of exceptions and a person can then determine if they are acceptable exceptions.			
150	**Correct use of object references.**	**LAD**	**F**	**Validity**
	Description: This guideline refers to a technique for referencing objects from an XML instance document. This means that the schema design is that every object that may be referenced by another object, whether externally or internal to the XML instance data, should inherit from the common complex type "Object." **Note:** See "Using AEX" The section 2.12. The test for this guideline is really more of an aid. A program can flag types which do not inherit from the complex type "Object,." and a person needs to determine whether the modeling is correct.			
200	**Use of "custom" item in sequence and choice definitions**	**LAD**	**F**	**Extensibility**
	Description: The AEX Guidelines document recommends a mechanism for customizing sequence or choice type schema constructs. The mechanism requires that every sequence or choice type definition include an element called "custom." This test checks for the existence of the custom element in those definitions. **Note:** See "Using AEX" The sections 2.13 and 5.2c. This practice is similar to OAGI's 'UserArea' element, which is a practice that is adopted but not documented in its design guideline. New releases of OAGI specification intend to deprecate this practice though.			

250	**All Elements Optional By Default**	**LAD**	**F**	**Clarity, Extensibility, Maintainability**
	Description: The cfiXML Schema development process calls for all elements to be optional by default. The position is that required elements will be validated outside of the schema, as they will differ for different transactions. **Note:** See "XML Schema Development Guidelines" The section 5.3.9. This practice is similar to OAGI architectural practice; however, OAGI specification enforces a set of minimal requirements in the schema.			
300	**Check for improper use of anonymous type.**	**LAD**	**F**	**Extensibility**
	Description: Content models of an anonymous type are defined locally within an element. They cannot be referenced outside of that element definition; hence, they cannot be reused. On the other hand, a globally defined type allows it to be referenced and hence reused. cfiXML adapts the Venetian Blind [7] design approach in that global types should be defined where necessary and global elements be declared only for extensible components. **Note:** This principle has yet to be documented; however, it is used in cfiXML Version 1. This practice is similar to that of the OAGI practice #150. The test of this requirement can be partially automated by flagging the use of anonymous types thereby assisting a person in determining the correctness of the usage. Since the test flags all anonymous type definitions, the test coverage value is Full. In XML schema, type definitions can be viewed as a content model, while the element definitions are viewed as document structure. Content/data model has tight relationship with functional requirements or functional model. Hence, software components should be developed corresponding to the content/data model rather than to the document structure. The content model represents entities that are used and reused; therefore, software components developed around it can also be reused. Although the use of global types can cause name-clashing problem, the available of namespace mechanism reduces this problem drastically. The same term with different concepts (perhaps in different domain) can be defined in different namespaces.			
350	**Global elements exist for all extensible types.**	**LA**	**F**	**Extensibility**
	Description: Since the cfiXML approach is based on the idea of "users" defining their own documents as needed using the contents of cfiXML as the building blocks, it is recommended that extensible, globally defined types have corresponding global elements as well. **Note:** This principle has yet to be documented; however, it is used in cfiXML Version 1. The test of this requirement can be partially automated by flagging the global types that do not have a corresponding global element. Since the test flags all global types not having a corresponding global element, the test coverage value is Full.			

15. An Example Code List Reference Table

This is an example code list table that can be used to check whether the schema has taken into account the code list rather than allowing free text when possible. UBL has also made potential code lists available as international standards; however, it does not give specific details about the international standards associated with them.

Data	Standard (s)	Note
Currency	ISO 4217	
Country	ISO 3166-3:1999 Part 1	
Unit	UN/ECE Recommendation No. 20	
Payment method code	ISO 10962:2001, KIEC	
Language code	IETF RFC 3066	
Product code	EAN-13, UPC	EAN-13 is widely used in distribution, UPC is widely used in retail.
Classification code	UNSPSC, HS, NAIC	
Property code	EDIFACT 7081	This is product property code such as dimension.
Message function code	EDIFACT 1225	
Response type code	EDIFACT 4343	This code indicates response type of messages.