



# </agcXML>

Information Exchange

## agcXML TECHNICAL FRAMEWORK (v1.0)



# **agcXML Technical Framework (v1.0)**

**National Institute of Building Sciences**

**Yimin Zhu, Ph.D.**

**March 29, 2007**

**Last Revised August 20, 2007**

<b>1. OVERVIEW.....</b>	<b>5</b>
<b>2. AGCXML CONCEPTUAL ARCHITECTURE.....</b>	<b>8</b>
2.1. agcXML Objectives, Scope, and Requirements.....	8
2.2. Contractual Requirements .....	9
2.2.1. agcXML and aecXML/IAI .....	9
2.2.2. Handling Unstructured Content.....	9
2.3. agcXML Conceptual Structure .....	10
2.4. Common Object Library .....	11
2.4.1. COS Conceptual Structure.....	13
2.4.2. The Dictionary.....	24
2.4.3. COS Namespaces .....	27
2.4.4. Handling EXPRESS Definitions .....	27
2.4.5. COS Authoring Recommendations .....	28
2.4.6. Handling Customization and Extensibility for COS Definitions.....	30
2.4.7. Recommended Definitions from the IFC model for agcXML.....	31
2.5. Handling Unstructured Content .....	31
2.6. Process-oriented Design .....	31
2.7. COS Versioning .....	34
<b>3. AGCXML SCHEMA STRUCTURE.....</b>	<b>36</b>
3.1. A High-Level agcXML Schema Structure .....	36
3.2. agcXML Namespace Declaration.....	36
3.3. Schema Header Settings .....	38
3.4. Root Element .....	38

<b>3.5. Naming Convention.....</b>	<b>39</b>
3.5.1. Scope of the Naming Convention.....	39
3.5.2. Authority Establishing Names.....	40
3.5.3. Semantic Rules.....	40
3.5.4. Syntactic Rules.....	41
3.5.5. Lexicon Rules.....	41
3.5.6. Uniqueness Principles.....	42
<b>3.6. Customization and Extensibility .....</b>	<b>42</b>
3.6.1. Extending Using COS Definitions.....	42
3.6.2. Using User-defined Elements.....	43
<b>3.7. Versioning .....</b>	<b>44</b>
<b>4. APPENDIX I: REVIEW OF SOME EXISTING STANDARDS .....</b>	<b>45</b>
<b>4.1. bcXML .....</b>	<b>45</b>
<b>4.2. cfiXML .....</b>	<b>46</b>
<b>4.3. ebXML .....</b>	<b>47</b>
<b>4.4. gbXML .....</b>	<b>48</b>
<b>4.5. ifcXML .....</b>	<b>49</b>
<b>4.6. MIMOSA .....</b>	<b>49</b>
<b>4.7. OSCRE .....</b>	<b>50</b>
<b>5. APPENDIX II: REVIEW OF THE TECHNICAL DESIGN OF SOME EXISTING STANDARDS.....</b>	<b>51</b>
<b>5.1. Namespace .....</b>	<b>51</b>
5.1.1. bcXML .....	51
5.1.2. cfiXML.....	52
5.1.3. gbXML.....	52
5.1.4. ifcXML.....	52
5.1.5. OSCRE .....	53

<b>5.2.</b>	<b>Element and Attribute Naming Issues.....</b>	<b>53</b>
5.2.1.	bcXML .....	53
5.2.2.	cfiXML.....	53
5.2.3.	gbXML.....	54
5.2.4.	ifcXML.....	54
5.2.5.	OSCRE .....	54
<b>5.3.</b>	<b>Elements vs. Attributes .....</b>	<b>54</b>
<b>5.4.</b>	<b>Reuse of Existing Schemas.....</b>	<b>55</b>
5.4.1.	bcXML .....	55
5.4.2.	cfiXML.....	55
5.4.3.	Others .....	56
<b>5.5.</b>	<b>Handling Unstructured Content .....</b>	<b>56</b>
<b>5.6.</b>	<b>Use of “Dictionary” .....</b>	<b>59</b>
<b>5.7.</b>	<b>Extensibility .....</b>	<b>61</b>
<b>5.8.</b>	<b>Process-oriented Design .....</b>	<b>61</b>
<b>5.9.</b>	<b>Versioning and Version Tracking.....</b>	<b>62</b>
5.9.1.	aecXML.....	62
5.9.2.	cfiXML.....	62
5.9.3.	Others .....	63
<b>6.</b>	<b>APPENDIX III: STEPS FOR ABBREVIATING NAMES .....</b>	<b>64</b>
<b>7.</b>	<b>REFERENCES.....</b>	<b>65</b>

## 1. OVERVIEW

A NIST (National Institute of Standards and Technology) study (Gallaher et al. 2004) shows that it costs the capital facility industry in the United States about \$15.8 billion annually because of inadequate interoperability among software systems such as CAD, engineering and other software tools. As a result, many XML-based standards has been developed and applied to design, engineering, eBusiness, and eCommerce applications in the AEC industry as a solution to the problem of interoperability. In this section, some of these standards are summarized and compared. A detailed discussion of these standards and their relationship to agcXML can be found in Appendix I and II.

In addition, in order to uniquely identify the concepts in an eBusiness and eCommerce operation, several initiatives have been launched to provide unique and unambiguously identifiable definitions of terms used in the description of products and services. These are often referred to as dictionary or classification servers. A forerunner in the construction industry is the LexiCon project, developed by the Dutch organization Samenwerkingsverband Tussen de Partners in de Bouwnijverheid, commonly known by its Dutch initialism, STABU [[www.stabu.org](http://www.stabu.org)]. (The English translation of the organization's name is "The Cooperation Bond between the Partners in the Construction Industry.") LexiCon [[www.stabu-lexicon.nl/](http://www.stabu-lexicon.nl/)] is regarded as a neutral set of building object definitions. It consists of a structured set of terms and definitions about building products, materials, and services. There are several other dictionary servers, both open and commercial, available for the construction industry, including the International Framework for Dictionaries (IFD) [[www.ifd-library.com](http://www.ifd-library.com)], a recent collaboration between STABU and three other organizations. The first is the Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology, known by its Norwegian initialism SINTEF [[www.sintef.no](http://www.sintef.no)], which had previously developed the Building and Construction Reference Data Library, known by the Norwegian intialism BARBI. The second is the U.S.-based Construction Specifications Institute, and the third is Construction Specifications Canada (CSC). Other examples include the industry-wide open technical dictionary (eOTD) of the Electronic Commerce Code Management Association (ECCMA) [<http://eccma.org>] and the eCl@ss International Standard for the Classification and Description of Products and Services, developed by the German organization of the same name [[www.eclass.de/](http://www.eclass.de/)].

**Table 1:  
A Comparison of Some of the Existing Standards**

	<b>bcXML</b>	<b>cfiXML</b>	<b>gbXML</b>	<b>ifcXML</b>	<b>OSCRE</b>
<b>Namespace</b>	<ul style="list-style-type: none"> <li>• Two levels: meta-schema for taxonomy and building and construction specification schema</li> <li>• Target namespace defined</li> <li>• elementFrom-Default and AttributForm-Default not defined</li> </ul>	<ul style="list-style-type: none"> <li>• A hierarchical design</li> <li>• targetNamespace is the default namespace</li> <li>• attributeForm-Default set to “unqualified” and elementForm-Default set to “qualified”</li> </ul>	<ul style="list-style-type: none"> <li>• Target namespace defined</li> <li>• elementFrom-Default and attributForm-Default set to “unqualified”</li> </ul>	<ul style="list-style-type: none"> <li>• Very detailed namespace design</li> <li>• elementFrom-Default and attributForm-Default set to “qualified”</li> </ul>	<ul style="list-style-type: none"> <li>• Target namespace not defined</li> <li>• elementFromDefault set to “qualified” and attributFormDefault set to “unqualified”</li> </ul>
<b>Element and Attribute Naming</b>	<ul style="list-style-type: none"> <li>• Mixed use of UCC<sup>2</sup> and LCC<sup>1</sup> for element definitions</li> <li>• UCC<sup>2</sup> for type definitions</li> <li>• LCC<sup>1</sup> for attribute definitions</li> <li>• No specification for abbreviations and acronyms</li> </ul>	<ul style="list-style-type: none"> <li>• UCC<sup>2</sup> for complex or simple type names</li> <li>• LCC<sup>1</sup> for elements and attributes</li> <li>• Use of abbreviations and acronyms specified</li> <li>• Allowing the use of alpha-numeric characters for names</li> </ul>	<ul style="list-style-type: none"> <li>• UCC<sup>2</sup> for elements and LCC<sup>1</sup> for attributes</li> </ul>	<ul style="list-style-type: none"> <li>• Mapping between the EXPRESS format and the XML schema</li> </ul>	<ul style="list-style-type: none"> <li>• UCC<sup>2</sup> for elements, complex types and attributes</li> </ul>
<b>Elements vs. Attributes</b>	<ul style="list-style-type: none"> <li>• Object or entity attributes mapped as XML elements</li> <li>• XML attributes used in limited cases</li> </ul>	<ul style="list-style-type: none"> <li>• Object or entity attributes mapped as XML elements</li> <li>• XML attributes used in limited cases</li> </ul>	<ul style="list-style-type: none"> <li>• XML attributes applied extensively</li> </ul>	<ul style="list-style-type: none"> <li>• Object or entity attributes mapped as XML elements</li> <li>• XML attributes used in limited cases</li> </ul>	<ul style="list-style-type: none"> <li>• XML attributes applied extensively</li> </ul>
<b>Reuse of Existing Schemas</b>	<ul style="list-style-type: none"> <li>• Direct include such as ifcXML</li> </ul>	<ul style="list-style-type: none"> <li>• Mapping</li> </ul>	<ul style="list-style-type: none"> <li>• Possible through include or import</li> </ul>	<ul style="list-style-type: none"> <li>• Possible through include or import</li> </ul>	<ul style="list-style-type: none"> <li>• Possible through include or import</li> </ul>

	<b>bcXML</b>	<b>cfiXML</b>	<b>gbXML</b>	<b>ifcXML</b>	<b>OSCRE</b>
<b>Handling Unstructured Content</b>	<ul style="list-style-type: none"> <li>• Not defined<sup>4</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Not defined<sup>4</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Not defined<sup>4</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Not defined<sup>4</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Not defined<sup>4</sup></li> </ul>
<b>Use of Dictionary</b>	<ul style="list-style-type: none"> <li>• A well-structured library</li> </ul>	N/A <sup>3</sup>	N/A <sup>3</sup>	<ul style="list-style-type: none"> <li>• Category of definitions</li> <li>• ifcLibraryReference</li> </ul>	<ul style="list-style-type: none"> <li>• A description in natural language for each tag name</li> </ul>
<b>Extensibility</b>	<ul style="list-style-type: none"> <li>• Basic mechanisms defined by XML Schema</li> </ul>	<ul style="list-style-type: none"> <li>• Basic mechanisms defined by XML Schema</li> <li>• Additional mechanisms defined for enumeration data types, elements, schemas</li> </ul>	<ul style="list-style-type: none"> <li>• Basic mechanisms defined by XML Schema</li> </ul>	<ul style="list-style-type: none"> <li>• Basic mechanisms defined by XML Schema</li> </ul>	<ul style="list-style-type: none"> <li>• Basic mechanisms defined by XML Schema</li> </ul>
<b>Process-Oriented Design</b>	<ul style="list-style-type: none"> <li>• ebXML</li> </ul>	<ul style="list-style-type: none"> <li>• Independent to any specific process specifications</li> </ul>	N/A <sup>3</sup>	<ul style="list-style-type: none"> <li>• Independent to any specific process specifications</li> </ul>	N/A <sup>3</sup>
<b>Versioning and Version Tracking</b>	<ul style="list-style-type: none"> <li>• Version number</li> </ul>	<ul style="list-style-type: none"> <li>• A four-phase schema release strategy</li> <li>• Versioning style specified</li> <li>• Schema changes and approval management procedures specified</li> </ul>	<ul style="list-style-type: none"> <li>• Version number</li> </ul>	<ul style="list-style-type: none"> <li>• Version and version compatibility specified</li> </ul>	<ul style="list-style-type: none"> <li>• Version number</li> </ul>

**Notes:**

1. LCC – Low Camel Case
2. UCC – Upper Camel Case
3. N/A – Reviewed documents do not specifically discuss the corresponding issue.
4. “Not Defined” – the corresponding issue is not addressed by the specific standard.

## **2. AGCXML CONCEPTUAL ARCHITECTURE**

### **2.1. agcXML Objectives, Scope, and Requirements**

agcXML intends to produce a set of schemas that can facilitate data/information interoperability, integration, and exchange for building design, construction, project management, and facility management operations throughout the lifecycle of a building facility. The data set within the agcXML scope of work, however, consists primarily of data commonly generated for transactional and communication purposes during the building design and construction process, the initial phase of a building facility's lifecycle. The primary goal of the agcXML project is to facilitate and streamline communication and data exchange among the parties to the building design and construction process. Only a subset of this data—that portion having a material relationship to the completed facility—may be of value beyond the design and construction period. The scope of the agcXML project does not include identifying this subset of information or defining how it might be preserved for future use in the continuing lifecycle of a building facility. It is presumed, however, that the availability of this information in the defined, structured, and documented agcXML format will facilitate its preservation and enhance the potential of its use in later phases of the facility lifecycle.

The deliverables included in the base scope of work consist of not only a set of agcXML schemas, but also the necessary data infrastructure (e.g., documented use cases) to support the development, use, maintenance, promulgation, and future extension of the schemas. Schemas will be developed to support the business processes that now depend on the data commonly found in the following types of construction contract documents:

1. Owner/Prime Contractor Agreements
2. Contractor/Subcontractor Agreements
3. Schedules of Values
4. Requests for Information
5. Requests for Pricing/Proposals
6. Architect's Supplemental Instructions
7. Construction Change Directives
8. Submittals
9. Change Order Requests/Approvals

## 10. Application for Payment Requests/Approvals

## 11. Addendum Notifications

In addition, there are nine additional types of contract documents specified in a sliding scope of work, which also may be included in the final deliverables depending on the availability of resources and time.

## **2.2. Contractual Requirements**

### **2.2.1. agcXML and aecXML/IAI**

In accordance with the agreement between the Associated General Contractors of America (AGC) and the National Institute of Building Sciences (NIBS) for this project, “the agcXML project will be executed as part of the buildingSMART Initiative of the IAI, within the framework of the IAI’s aecXML Domain (Committee), under the auspices of the International Alliance for Interoperability-North America (IAI-NA), a Council of the National Institute of Building Sciences.” agcXML also will be “a part of the aecXML family of schemas,” but in recognition of AGC’s authorship will be explicitly distinguished from other aecXML contributions. Conformance to these contractual requirements is demonstrated by the agcXML namespace design, discussed in Section 4.2 below, and the overall technical framework detailed in corresponding sections.

The relationship between agcXML and aecXML is also reflected by the design of the agcXML framework, which suggests that agcXML, as a group of domain-specific schemas, reuses definitions from the aecXML Common Object Library (COL) (see Figure 1).

### **2.2.2. Handling Unstructured Content**

The AGC/NIBS Agreement further specifies that NIBS, in the development of agcXML, “will develop a technical framework for agcXML development” that “will include the development of agcXML architectural design and associated tasks to define...the handling of unstructured documents...” However, the scope of work explicitly states that agcXML “...shall not include unstructured data...”

The NIBS agcXML Project Team interprets the explicit use of the phrases “unstructured documents” and “unstructured data” in these contract provisions to mean that the design of

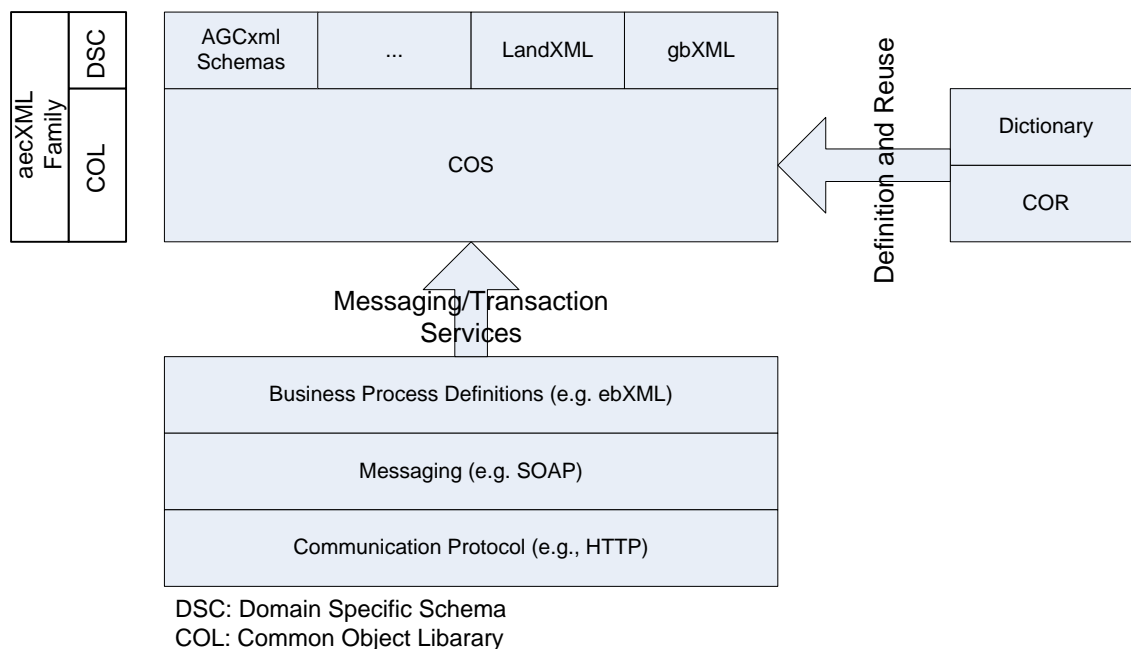
agcXML will consider required features of unstructured content at the document level only, not the data level.

### **2.3. agcXML Conceptual Structure**

Figure 1 illustrates the conceptual structure of agcXML, where agcXML schemas as a part of the aecXML schema family are developed by reusing definitions from a Common Object Library, or, to be more specific, the Common Object Schema, as well as using other customized definitions.

The existing aecXML design recommends two tiers to handle common and reusable objects, namely a Common Object Repository (COR) and a Common Object Schema (COS). The COR is a repository of a collection of third-party schemas, while the COS is “a single schema with a separate namespace under aecXML” (aecXML Framework, <http://www.iana.org/aecxml/>). The COS schema is developed after the consolidation of definitions in COR; thus it is possible that, initially, COS may be empty. On the other hand, according to the existing aecXML design document, schemas such as ifcXML, BLIS-XML, cXML and xCBL can be candidates for the COR.

The concept of a Common Object Library (COL) was proposed by the aecXML community. The intent of the COL effort is to extract common tags and their definitions from ifcXML and related North American and International efforts wherever possible. The COL includes COS, COR, and a dictionary of terms in COS.



**Figure 1 agcXML Conceptual Structure**

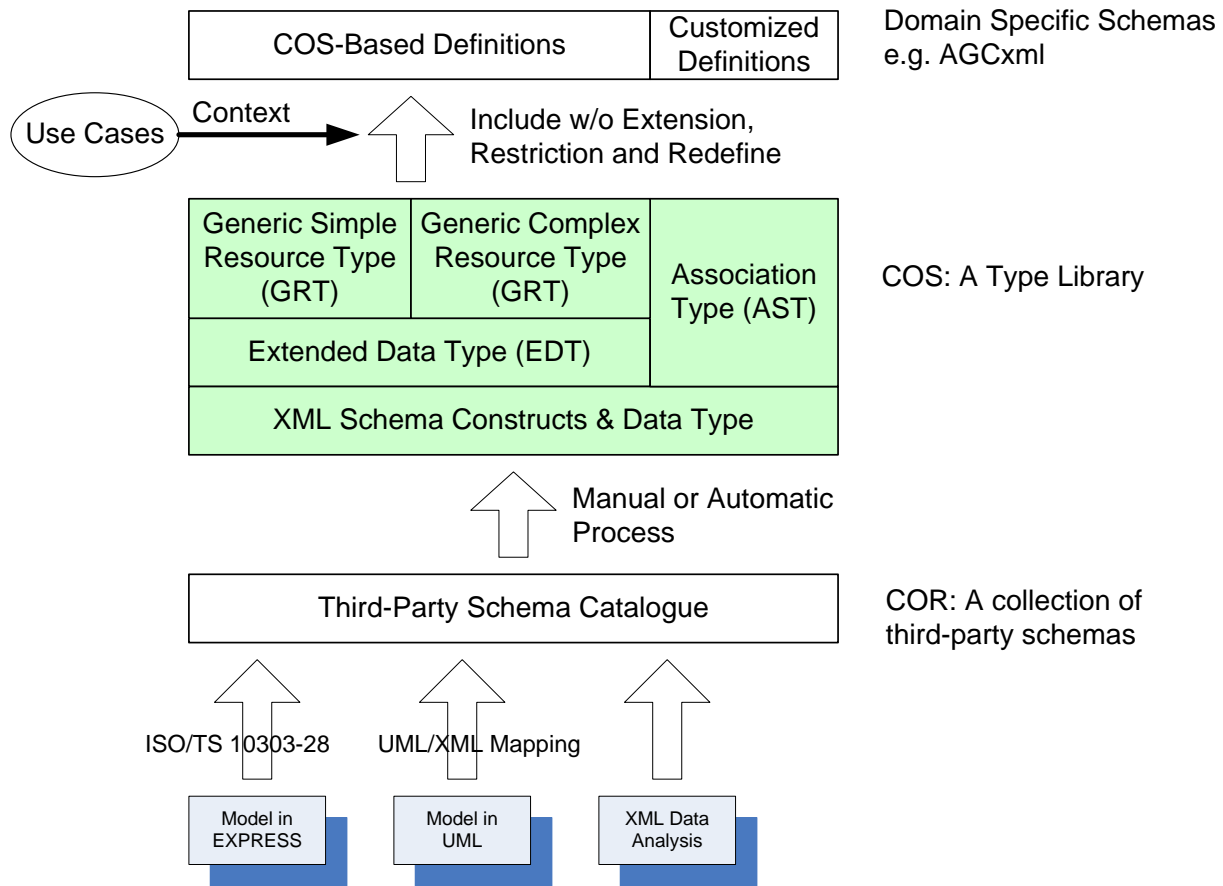
The COS provides reusable definitions for items that represent simple data types, user-defined data types, attributes, and entities/objects. Each item has a corresponding entry in the dictionary, which includes information such as the tag name, a concise description of the item, sources, and so on. Since the existing aecXML document on COS is fairly conceptual, this design interprets the idea of “a single schema with a separate namespace” as 1) allowing multiple physical schema files although conceptually all schemas belong to COS; and 2) allowing multiple namespaces for different schemas.

The agcXML schemas will use generic process-oriented standards such as ebXML, SOAP, and HTTP for defining and executing collaborative business processes as well as messaging. This conceptual structure demonstrates a coherent relationship between ifcXML, aecXML, and agcXML. The following section on Common Object Library (COL) provides more information about the relationships among them.

## 2.4. Common Object Library

The design of a COL intends to address the requirements of reusability and scalability, including reusing third-party definitions, being reused by domain-specific schemas, extending existing definitions, and using customized definitions. The term “scalability” is interpreted as providing

capability for extending or customizing definitions. Figure 2 illustrates the conceptual relationship between COS, COR, and domain-specific schemas (DSS) such as agcXML.



**Figure 2 Conceptual Structure of DSS, COS, and COR**

COR, as a repository, is simply a collection of schemas that are deemed to have value to the AEC-FM community. Each schema keeps its own namespace and authoring style, since the COR may contain XML definitions with many different backgrounds. For example, some schemas may be mapped from EXPRESS definitions, some may be derived from UML models, while others may be directly developed using XML schema authoring tools.

COS has four major components that are built upon existing components: the XML Schema constructs and data types. The four components are Generic Simple Resource Type (GSRT), Generic Complex Resource Type (GCRT), Extended Data Type (EDT) and Association Type (AST). Definitions from third-party schemas may be converted automatically or manually into corresponding definitions in COS, depending on their original modeling and authoring

styles. (See the following section, “COS Conceptual Structure,” for a discussion of those components.) The domain-specific schemas are then built—according to business-specific use cases—by reusing and extending definitions in GSRT, GCRT, EDT, and/or AST and other customized definitions.

## **2.4.1. COS Conceptual Structure**

### ***2.4.1.1. Overview***

The COS conceptual design follows major design principles of ISO\DTS 15000-5: 2006 Core Components Technical Specifications, 2<sup>nd</sup> Edition (UN/CEFACT version 2.2); concepts of type libraries in the W3C XML Schema specification; and the ifcXML specification. Although currently the ISO\DTS 15000-5 is a working draft, some of its design ideas for reusable common definitions can be applied to this project.

The U.S. Department of Navy (DOV) XML Naming and Design Rules, version 2.0, provides a framework that also leverages the concepts specified by ISO\DTS 15000-5. The type library technique is used by W3C, providing a type library of types such as text, array, list, and math. Since many definitions used by agcXML may come from ifcXML, it is desirable to take ifcXML design rules into consideration.

There are two key concepts in the Core Components Technical Specification: core components and business information entities (ISO\DTS 15000-5, 2006). Core components are basic semantic concepts that can be used to compose more complex and business-oriented concepts. Business information entities are context-specific and customized core components.

The core components include the Aggregate Core Component (ACC), the Basic Core Component (BCC), the Association Core Component (ASCC) and the Core Data Types (CDT). An ACC is “a collection of related pieces of information that together convey a distinct meaning, independent of any specific context” (ISO\DTS 15000-5, 2006) which may serve as the representation of an object/entity. A BCC represents the property of an ACC. An ASCC represents the association between two ACCs, where one ACC is the property of the other ACC. A CDT is a data type that specifies allowed value spaces for values that can be taken by a property such as a BCC.

Business information entities (BIE) are always derived from core components based on a specific business context. The BIE includes four types of components: the Aggregated Business

Information Entity (ABIE), The Basic Business Information Entity (BBIE), the Associated Business Information Entity (ASBIE) and the Business Data Types (BDT). These four types of business information entity components—ABIE, BBIE, ASBIE and BDT—are counterparts of the core components, ACC, BCC, ASCC, and CDT, respectively. Context is used to customize generic definitions, such as those in ACC, BCC, ASCC, and CDT, to business-specific definitions: ABIE, BBIE, ASBIE, and BDT. The current draft of ISO/DTS 15000-5 further specifies eight approved context categories, including business process, product classification, industry classification, geopolitical, official constraints, business process role, supporting role and system capabilities. These types of context can be instantiated according to specific use cases.

The relationship between the concepts in this design and the ISO/DTS 15000-5: 2006 Core Components Technical Specification, 2<sup>nd</sup> Edition, is summarized in Table 1.

**Table 2:**

**The relationship between the concepts in this design and that in ISO/DTS 15000-5**

<b>ISO/DTS 15000-5 Concepts</b>	<b>Concepts in this Design</b>
<b>Core Data Type (CDT)</b>	<b>Extended Data Type (EDT), XML schema built-in data types</b>
<b>Basic Core Component (BCC)</b>	<b>Generic Simple Resource Type (GSRT)</b>
<b>Aggregated Core Component (AGC)</b>	<b>Generic Complex Resource Type (GCRT)</b>
<b>Association Core Component (ASCC)</b>	<b>Association Type (AST)</b>
<b>Context</b>	<b>Use cases and associated definitions</b>
<b>Business Information Entity (BIE)</b>	<b>Domain-specific Schemas</b>

The benefit of such a design is that various schema components, such as data types, attributes, and elements, can be reused more readily because of the discrete nature of the definitions. Consequently, schemas derived by using the reusable components may also be more interoperable. In addition, the dichotomy of core components and business information entities through context may further improve the capability of applications to resolve heterogeneity, because the source of heterogeneity may be traced by definitions in context.

The application of such a design, together with a dictionary (detailed in Section 3.4.2), can also satisfy the agcXML requirement that “the groundwork starts from vocabulary,” without growing into a LexiCon-type of project that is beyond the scope of the agcXML project.

The concept of type libraries is introduced by W3C to improve the reusability of simple and complex types that can be used as building blocks for developing new schemas. Each library may contain one or several related types, such as an address type. XML Schema has developed type libraries for many simple types such as text, list, array, math, quantity, and binary. An obvious benefit of using type libraries, instead of using a monolithic XML Schema file that includes all definitions, is that high-level domain-specific schemas can select only those definitions that are needed.

#### ***2.4.1.2. The Common Object Schema Structure***

The COS has four components, starting with basic XML Schema constructs such as “element” and data types such as “string.” The Extend Data Type (EDT) includes data types that are customized for the purpose of specific applications, but still generic enough. The customization is accomplished by using XML simpleType as shown in Example 1, below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.aecXML.org/COS/EDT/"
  xmlns:edt="http://www.aecXML.org/COS/EDT/"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="GlobalID" type="edt:GlobalID"/>
  <xs:simpleType name="GlobalID">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```

#### **Example 1**

In this example, “GlobalID” has extended a string definition by “restriction”. EDTs are used to construct data types that are not supported by XML built-in data types, such as a globally unique ID. The XML Schema constructs, built-in data types, and extended data types, considered as

equivalent to the Core Data Types (CDT) in the ISO/DTS 15000-5, may be reused by higher level schema definitions.

The Generic Resource Type (GRT) has two types of definitions, those for reusable attributes of simple data types such as BCC, and those for reusable elements or types such as ACC. The former is called a Generic Simple Resource Type (GSRT) and the later is called a Generic Complex Resource Type (GCRT).

GSRTs are declared as XML simpleTypes. For example, “Description” of an object, which may appear in many complex type definitions for objects or entities, has a simple data type and can be a good candidate of GSRT for reusability. In Example 2, a reusable element “Description” is defined:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.aecXML.org/COS/GSRT/"
  xmlns:gsrt="http://www.aecXML.org/COS/GSRT/"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xs:element name="Description" type="gsrt:Description"/>
  <xs:simpleType name="Description">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```

## Example 2

GCRT definitions include generic definitions for business related concepts, such as Person, Organization, Project, Document, and Cost. Many of the IFC definitions, especially those in its resource layer, can be utilized as GCRTs. GCRTs are declared as XML complexTypes. The Person definition, for example, has attributes of simple data types, such as xs:string and xs:date, and of complex types, such as PostalAddress and TeleCommunicationAddress, that are objects themselves. Attributes of simple data types may be directly included in an XML complex type definition as XML elements. For attributes of complex types, both containment and reference are

supported in COS. However, domain-specific schemas should have a consistent style, either containment or reference.

Reference can be implemented using ID/IDREF. Similar to the ifcXML implementation, elements of complex types will be derived from a base element that contains an “id” attribute of ID type and a “href” attribute of IDREF type. The use of those attributes is optional, as shown in Example 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.aecXML.org/COS/GCRT/"
  xmlns:gcr="http://www.aecXML.org/COS/GCRT/"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xs:element name="Element" type="gcr:Element"/>

  <xs:complexType name="Element">
    <xs:attribute name="id" type="xs:ID" use="optional"/>
    <xs:attribute name="href" type="xs:IDREF" use="optional"/>
  </xs:complexType>
</xs:schema>
```

### Example 3

Example 3 shows the definition of the base element. Other complexType definitions can extend the based element to include the two attributes, as shown in Example 4:

```
<xs:element name="Document" type="gcr:Document"/>
<xs:complexType name="Document">
  <xs:complexContent>
    <xs:extension base="gcr:Element">
      <xs:attribute name="referenceType">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="ContainmentOnly"/>
            <xs:enumeration value="ReferenceOnly"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```

        <xs:enumeration value="Choice"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:element name="Person" type="gcr:Person"/>
<xs:complexType name="Person">
    <xs:complexContent>
        <xs:extension base="gcr:Element">
            <xs:sequence>
                <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
                <xs:element name="LastName" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:element name="PostalAddress" type="gcr:PostalAddress"/>
<xs:complexType name="PostalAddress">
    <xs:complexContent>
        <xs:extension base="gcr:Element">
            <xs:sequence>
                <xs:element name="StreetLine" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="City" type="xs:string" minOccurs="0"/>
                <xs:element name="State" type="xs:string" minOccurs="0"/>
                <xs:element name="ZipCode" type="xs:string" minOccurs="0"/>
                <xs:element name="Country" type="xs:string" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="schemaVersion" type="xs:decimal" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

```

## Example 4

Example 4 has two complex types. Each extends the base element. In this way when domain-specific schema is developed, the reference between Person and Address can be achieved by specifying that the href value in Person equals the id value in Address in XML instance documents, as shown in Example 5:

```

<gcr:Person>
  <gcr:FamilyName>Bond</gcr:FamilyName>
  <gcr:FirstName>James</gcr:FirstName>
  <ast:PersonAddressRel xsi:nil="true" href="ccc1"/>
  <ast:PersonAddressRel xsi:nil="true" href="ccc2"/>
</gcr:Person>
<gcr:Address id="ccc1">
  <gcr:StreetLine>Somewhere in the world</gcr:StreetLine>
  <gcr:PostalBox>007</gcr:PostalBox>
</gcr:Address>
<gcr:Address id="ccc2">
  <gcr:StreetLine>Anywhere in the world</gcr:StreetLine>
  <gcr:PostalBox>007</gcr:PostalBox>
</gcr:Address>

```

### Example 5

The associations between objects/entities are implemented by using an association type. The association type is named by concatenating the names of the two objects and a string “Rel.” In Example 4, for example, the Person type is referencing the Address type. The name of the relation type is “PersonAddressRel.” The sequence of the object/entity names also indicates the reference direction. In addition, type safety may also be handled by using KEY/KEYREF in the domain-specific schemas.

```

<xs:element name="PersonPostalAddressRel" type="rfi:PersonPostalAddressRel" nillable="true"/>
<xs:complexType name="PersonPostalAddressRel">
  <xs:complexContent>
    <xs:extension base="gcr:Element">
      <xs:sequence>
        <xs:element ref="rfi:PostalAddress"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```

    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

## Example 6

In order to support both containment and reference, a switch is embedded in the element declaration in Example 6. The switch is “nillable,” which is set to “true,” meaning the content of the PersonAddressRel may be empty. This switch can be turned on in its XML instance document. In the instance document, the “xis:nil” attribute of PersonAddressRel can be set to “true,” then reference is used. Otherwise, if the “xis:nil” attribute is set to “false,” then containment is applied. However, a domain-specific schema such as a Request for Information (RFI) schema needs to determine the reference style, either containment or reference. Example 7 includes a simplified example of an RFI schema, a Person type schema, an Address type schema, and a PersonAddressRel type schema. In addition, a sample XML document for RFI is also provided.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:agcxml="http://www.aecXML.org/AGC/agcXML-20070305"
  xmlns:cos="http://www.aecXML.org/COS"
  targetNamespace="http://www.aecXML.org/AGC/RFI-20070305"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.1">
...
<!-- The RFI Definition -->
<xs:element name="RFI" type="agcxml:RFI"/>
<xs:complexType name="RFI">
  <xs:complexContent>
    <xs:extension base="agcxml:Document">
      <xs:sequence>
        <xs:element name="DocumentID" type="xs:string" minOccurs="0"/>
        <xs:element name="CreationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="DocumentTypeNumber" type="xs:string" minOccurs="0"/>
        <xs:element ref="agcxml:RFISubjectRel" minOccurs="0"/>

```

```

        <xs:element ref="agcxml:RFIProjectRel" minOccurs="0"/>
        <xs:element ref="agcxml:RFIContractRel" minOccurs="0"/>
        <xs:element ref="agcxml:RFIReferenceRel" minOccurs="0"/>
        <xs:element ref="agcxml:RFIContentRel" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="schemaVersion" type="xs:decimal" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
...

<xs:element name="Person" type="agcxml:Person"/>
<xs:complexType name="Person">
    <xs:complexContent>
        <xs:extension base="gcrt:Person">
            <xs:sequence>
                <xs:element name="Signature" type="xs:string" minOccurs="0"/>
                <xs:element ref="edt:RoleInProfession" minOccurs="0"/>
                <xs:element ref="agcxml:PersonPostalAddressRel" minOccurs="0"/>
                <xs:element ref="agcxml:PersonTeleCommunicationAddressRel" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
...

<xs:element name="PostalAddress" type="agcxml:PostalAddress"/>
<xs:complexType name="PostalAddress">
    <xs:complexContent>
        <xs:extension base="gcrt:PostalAddress"/>
    </xs:complexContent>
</xs:complexType>
...

<xs:element name="PersonPostalAddressRel" type="agcxml:PersonPostalAddressRel" nillable="true"/>
<xs:complexType name="PersonPostalAddressRel">
    <xs:complexContent>

```

```

    <xs:extension base="gcrt:Element">
      <xs:sequence>
        <xs:element ref="agcxml:PostalAddress"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
...
</xs:schema>

```

## Example 7

In Example 7, the RFI schema includes some requirements that need to be followed by implementation. The RFI definition extends the “Document” definition (See Example 4), which has declared an attributed called “referenceType”. The “referenceType” is an enumeration of three choices: “ContainmentOnly,” “ReferenceOnly,” and “Choice.” Such design can also be found in ifcXML. In this example, only one attribute is included. The actual agcXML schemas may use attributeGroup instead to include more attributes, depending on use cases. If “ContainmentOnly” is selected, then a containment style is applied to all instance data files. For example, the “PostalAddressRel” element is contained by the “Person” element and the details of the “PostalAddress” definition are also contained with the scope of “Person” (See Example 8). Otherwise, if “ReferenceOnly” is selected, then a reference style is applied to all instance data files. For example, the “xis:nil” attribute in the “PersonPostalAddressRel” is set to “true” and the “PersonPostalAddressRel” is empty. The “Person” no longer contains the postal address information, instead it references such information through id and idref matching (See Example 9).

```

<?xml version="1.0" encoding="UTF-8"?>
<agcxml:agcXML xmlns:agcxml="http://www.aecXML.org/AGC/agcXML-20070305"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gcrt="http://www.aecXML.org/COS/GCRT/"
  xsi:schemaLocation="http://www.aecXML.org/AGC/agcXML-20070305
F:\AGCXML\RFISchemas\agcXML.xsd">
  <agcxml:Person>
    <gcrt:FirstName>James</gcrt:FirstName>

```

```

    <gcr:LastName>Bond</gcr:LastName>
    <agcxml:PersonPostalAddressRel xsi:nil="false">
      <agcxml:PostalAddress>
        <gcr:StreetLine>10555 West Flagler Street</gcr:StreetLine>
      </agcxml:PostalAddress>
    </agcxml:PersonPostalAddressRel>
    <agcxml:PersonPostalAddressRel xsi:nil="false">
      <agcxml:PostalAddress>
        <gcr:StreetLine>1006 Maryland Street</gcr:StreetLine>
      </agcxml:PostalAddress>
    </agcxml:PersonPostalAddressRel>
  </agcxml:Person>
</agcxml:agcXML>

```

### Example 8

```

<?xml version="1.0" encoding="UTF-8"?>
<agcxml:agcXML xmlns:agcxml="http://www.aecXML.org/AGC/agcXML-20070305"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gcr="http://www.aecXML.org/COS/GCRT/"
  xsi:schemaLocation="http://www.aecXML.org/AGC/agcXML-20070305
F:\AGCXML\RFISchemas\agcXML.xsd">
  <agcxml:Person>
    <gcr:FirstName>James</gcr:FirstName>
    <gcr:LastName>Bond</gcr:LastName>
    <agcxml:PersonPostalAddressRel xsi:nil="true" href="cc1"/>
    <agcxml:PersonPostalAddressRel xsi:nil="true" href="cc2"/>
  </agcxml:Person>
  <agcxml:PostalAddress id="cc1">
    <gcr:StreetLine>10555 West Flagler Street </gcr:StreetLine>
  </agcxml:PostalAddress>
  <agcxml:PostalAddress id="cc2">
    <gcr:StreetLine>1006 Maryland Street </gcr:StreetLine>
  </agcxml:PostalAddress>
</agcxml:agcXML>

```

### Example 9

Following the design philosophy of ISO\DTS 15000-5, it is recommended that in COS definitions, referencing to other objects/entities should be limited to those situations where such a referencing is proven to be truly generic, because in many cases referencing to another object depends on use cases and should be modeled in domain-specific schemas. Another issue is context. The ISO\DTS 15000-5: 2006 Core Components Technical Specifications currently specifies eight categories of context:

1. Business Process
2. Product Classification
3. Industry Classification
4. Geopolitical
5. Official Constraints
6. Business Process Role
7. Supporting Role
8. System Capability

Use cases can be used to capture some of the context that is deemed to be useful for this project. On the other hand, many existing models such as IFC have more sophisticated object-oriented designs that the existing XML specification may not be able to handle, or achieve exact mapping. However, as discussed in the document, “flat ifcXML”, by Dr. Thomas Liebich, optimization seems necessary to make ifcXML more XML-alike. In the end, XML is a data representation language, not an object-oriented modeling language.

#### **2.4.2. The Dictionary**

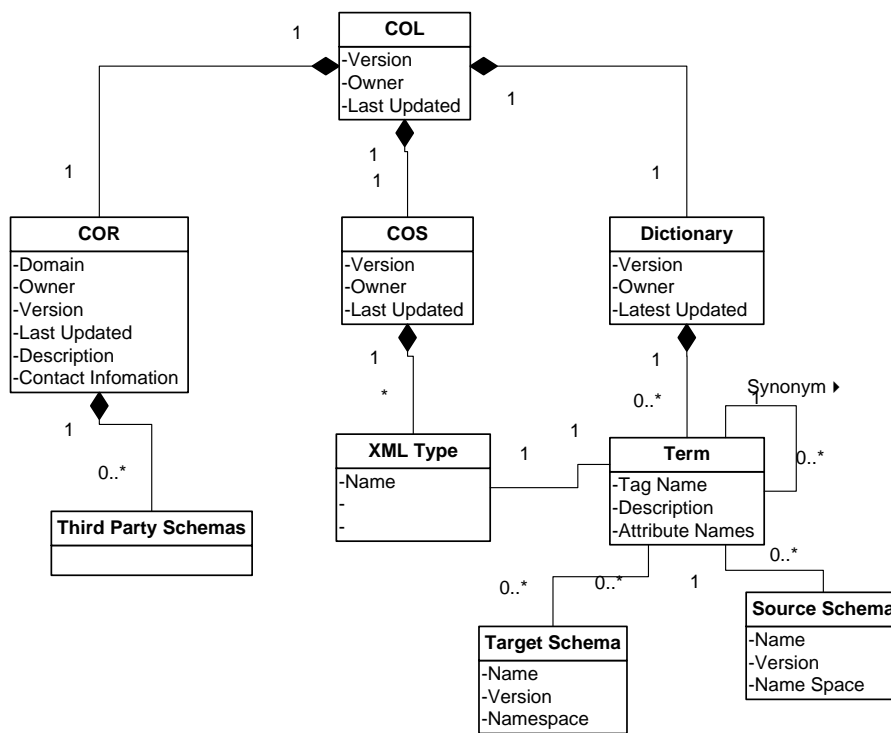
It is not the intention of the agcXML project to develop ontological definitions for terms and vocabularies in the manner of LexiCon. Due to time and resource constraints, this project will only develop an alphabetic list of terms in COS that are selected from third-party schemas and used in agcXML. Terms in the dictionary will be organized alphabetically at this stage.

However, each term should have the following descriptions:

1. Name: An official English name for every term using words as they appear and are defined by the Oxford English Dictionary.
2. Tag name: The XML tag name used in COS. The tag name shall follow the naming conventions set by aecXML.

3. Source schema: the schema where the original XML definition is derived.
4. Target schemas: the schemas that an XML definition in COS is applied to, extended by, or redefined by.
5. Description: a short description of what the term is, the context in which it is used, and other important information.
6. Attributes: the names of attributes that are modeled for COS terms.
7. Synonyms: identified synonyms developed and maintained by other standardization efforts.
8. Antonyms: identified antonyms developed and maintained by other standardization efforts.

The following diagram shows a conceptual structure of the dictionary.



**Figure 3**

The following is an example. The COS may contain a Person definition similar to the definition in the ifcXML stage 2 schema.

```
<xs:complexType name="Person">
```

```

<xs:complexContent>
  <xs:extension base="Element">
    <xs:sequence>
      <xs:element name="Id" type="Identifier" minOccurs="0"/>
      <xs:element name="FamilyName" type="Label" minOccurs="0"/>
      <xs:element name="GivenName" type="Label" minOccurs="0"/>
      <xs:element name="MiddleNames" minOccurs="0">
        <xs:complexType>
          <xs:choice maxOccurs="unbounded">
            <xs:element name="String" type="Label"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:element name="PrefixTitles" minOccurs="0">
        <xs:complexType>
          <xs:choice maxOccurs="unbounded">
            <xs:element name="String" type="Label"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:element name="SuffixTitles" minOccurs="0">
        <xs:complexType>
          <xs:choice maxOccurs="unbounded">
            <xs:element name="String" type="Label"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>

```

## Example 10

*The sample definition may look like the following,*

*Name: Person*

*Tag Name: Person*

*Description: 1. an individual human being (Source: IFC)*

*Attributes: Id*

*FamilyName*

*GivenName*

*MiddleNames*

*PrefixTitles*

*SuffixTitles*

*Source Schema:*

*Name: ifcXML Stage 2*

*Version: 1.0*

*Applications:*

*Target Schema*

*Name: agcXML*

*Version: 1.0*

*Synonyms:*

*Tag Name: ctx:Person*

*Source Schema:*

*Name: cfiXML*

*Antonyms:*

*None.*

### **2.4.3. COS Namespaces**

The COS has four components, the extended data types, the generic simple resources types, the generic complex resource types and the association types, whose namespace declarations need to be determined. The namespace URI for extended data types is <http://www.aecXML.org/COS>. If a prefix is needed, the prefix can be “cos” in low case.

### **2.4.4. Handling EXPRESS Definitions**

Since IFC definitions, which are in EXPRESS format, will be reused to populate COL definitions—especially those in COS—the design principles of ISO/TS 10303-28 and the IAI document “ifcXML Language binding of EXPRESS” should be referenced.

## 2.4.5. COS Authoring Recommendations

### 2.4.5.1. Authoring Schema Settings

The schema header settings of COS definitions shall be similar to those in the following example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:cos="http://www.aecXML.org/COS"
           targetNamespace="http://www.aecXML.org/COS"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           version="1.1">
```

#### Example 11

Definitions from the W3 XSD namespace is declared by using the following form:

```
xmlns:xs=http://www.w3.org/2001/XMLSchema
```

In addition, a prefix “xs” is applied (See the first line in Example 11). The target namespace is declared by using the following form:

```
targetNamespace=http://www.aecXML.org/COS
```

and the targetNamespace uses the following declaration to declare a prefix:

```
xmlns:cos=http://www.aecXML.org/COS
```

Default namespace is not used; all xmlns declarations use prefixes.

Throughout the COS definitions, element names shall be qualified and attribute names shall not be qualified. This is declared by setting the two attributes `elementFormDefault` and `attributeFormDefault` to “qualified” and “unqualified” respectively:

```
elementFormDefault="qualified"
attributeFormDefault="unqualified"
```

### 2.4.5.2. Naming Conventions

Naming conventions for COS definitions are the same as those for agcXML, which are discussed in section 3.5.

#### ***2.4.5.3. Authoring Extended Date Types***

The Extended Data Types are used to handle the extension of built-in XML simple datatypes. The extension is handled by using XML simpleType declaration with either restriction or extension to a base, which is XML simple datatypes. Naming for the extended data types shall follow the naming conventions specified in section 3.5.

#### ***2.4.5.4. Authoring Attributes***

Attributes of objects are declared as XML elements. Only a very limited number of attributes such as ids or hrefs are authored as XML attributes. These types of attributes are specified in this document. Naming for object attributes that are XML elements in XML schemas shall follow the naming conventions for XML elements specified in section 3.5, while naming for object attributes that are XML attributes in XML schemas shall follow the naming conventions for XML attributes specified in section 3.5. COS elements may contain following XML attributes:

1. id of W3 XML Schema ID type with optional use
2. href of W3 XML Schema IDREF type with optional use
3. objGuid of GUID type with optional use

#### ***2.4.5.5. Authoring Elements***

It is strongly recommended that every AEC-FM related element have a type definition. In addition, the element and type pair shall use the same name. Elements without type definitions should be limited for use. The naming of elements and element types shall follow the naming conventions for XML elements specified in section 3.5.

#### ***2.4.5.6. Authoring References***

When a reference from one object to another is modeled in COS as AST (Association Type), the naming of the association type should follow the following pattern:

Name of the referencing object + Name of the referenced object + Descriptive Name + “Rel”

The following rules should be applied when naming references,

- 1) The descriptive name is optional;
- 2) The name of the referencing object and the name of the referenced object must appear in pair. However, they are optional as well;

- 3) If the descriptive name is not applied, then the pair of the referencing object name and the referenced object name must appear. Vice verse.
- 4) The “Rel” is also optional.

The naming of the referencing and the referenced objects shall follow the naming conventions for XML elements/types specified in section 3.5.

#### 2.4.6. Handling Customization and Extensibility for COS Definitions

Customization of COS definitions is not allowed at the COS level. Extension shall be very limited in order to maintain the reusability and the interoperability of COS definitions. Changes to COS definitions shall be handled through versioning. Customization and extensibility shall be handled mostly at the domain-specific schema level.

To support user definitions in domain-specific schemas, a Custom type is declared and defined. When declaring and defining the Custom type, the following issues need to be noted (see <http://www.xml.com/pub/a/2004/10/27/extend.html?page=2>):

- using “##any” in the declaration for namespace
- using “lax” for processCenters

The Custom definition, therefore, takes the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:cos="http://www.aecXML.org/COS"
  targetNamespace="http://www.aecXML.org/COS"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="Custom"/>
  <xs:complexType name="Custom">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

#### Example 12

#### **2.4.7. Recommended Definitions from the IFC model for agcXML**

In general, the need for reusing a particular type can be identified from high-level domain-specific applications. However, some existing definitions in IFCs are generic enough to be considered as candidates for COS definitions for the development of agcXML. These definitions include but are not limited to:

1. Actor Resource
2. Date Time Resource
3. External Reference Resource
4. Cost Resource
5. Quantity Resource
6. Measure Resource
7. Approval Resource
8. Constraint Resource

#### **2.5. Handling Unstructured Content**

To fulfill contractual requirements, a “DocumentMetaData” entity needs to be created using use cases to determine required and optional attributes of the entity. Some of the attributes may come from existing metadata models such as ISO 82045:5 2005 and the IFCs.

Each document, based on specific use cases, can include the DocumentMetaData definition, which is defined as a Generic Complex Resource Type in COS. The definition of DocumentMetaData in COS contains only those required attributes for all documents. Additional attributes may be added according to use cases by extending the COS DocumentMetaData definition in domain-specific schemas.

Use cases are needed for determining required attributes for the COS definition.

#### **2.6. Process-oriented Design**

agcXML will use other generic standards for process design and messaging such as SOAP, Web Services, and ebXML. These three standards represent three different categories of process-related standards.

SOAP (Simple Object Access Protocol) is a communication protocol, using HTTP and XML. SOAP is also simple and extensible. The required SOAP Body element may contain

actual business messages. If the focus of SOAP is on point-to-point messaging, the Web Services extend the capability of SOAP by allowing collaborating systems to list and discover services by using open protocols. Web Services have three major components: SOAP, Universal Description, Discover, and Integration (UDDI), and WSDL (Web Services Description Language). ebXML, as discussed earlier, has a broader scope to support conducting eBusiness. With such a goal, ebXML, although also relying on SOAP and having similar capability as UDDI, has more sophisticated design and capability than the previous two in supporting eBusiness, such as defining and orchestrating business processes.

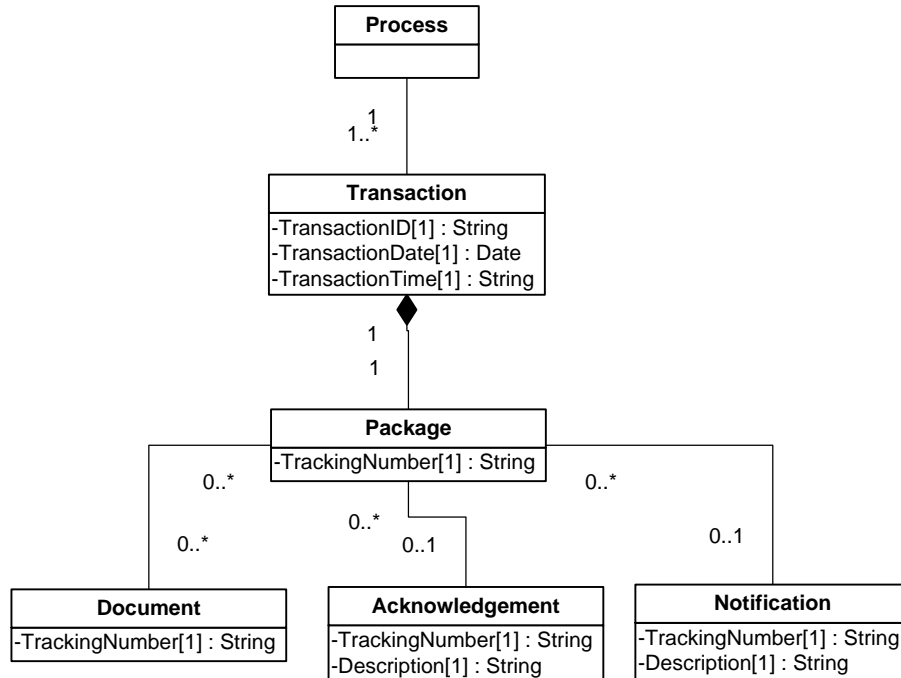
On the other hand, a common feature of those generic process-oriented standards is that they all allow domain-specific data to be included in messaging as payloads. agcXML at this stage leverages this technical design until use cases may suggest the need for further customization for handling AEC-FM related business processes. There might be process-oriented requirements specific to use cases in the AEC industry. Once the requirements are identified, they then may be modeled.

A Package is a collection of documents that need to be delivered together. The Package is an implementation of the concept of a variable content container, which may contain variable content. Figure 4 is a partial definition of a Transaction object, which includes a Package. First, a type “Document” is declared and defined as an “abstract” type, as shown in Example 13:

```
<xs:element name="Document" type="cos:Document" abstract="true"/>
<xs:complexType name="Document" abstract="true">
  <xs:complexContent>
    <xs:extension base="cos:Element">
      <xs:attribute ref="cos:referenceType"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### Example 13

Then, a Package type and element is declared and defined. The Package definition makes reference to the Document definition as defined in Figure 4, below. The business documents such as RFI or Change Order (CO) can substitute the abstract “Document” in data files.



**Figure 4**

In addition, since specific documents such as Request for Information or Request for Payment are the instantiations of the Document type, those specific documents can be included in the package, as shown in Example 14, below. Note that the “maxOccurs” is set to “unbounded.”

```

<xs:complexType name="Package">
  <xs:complexContent>
    <xs:extension base="gcr:Element">
      <xs:sequence>
        <xs:element ref="gcr:Document" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gcr:Notification" minOccurs="0"/>
        <xs:element ref="gcr:Acknowledgement" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="RFI" type="gcr:RFI" substitutionGroup="gcr:Document"/>
<xs:complexType name="RFI">
  <xs:complexContent>
    <xs:extension base="gcr:Document">

```

```

        <xs:sequence>
            <xs:element name="Name" type="xs:string"/>
        </xs:sequence>
    </xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:element name="CO" type="gcr:CO" substitutionGroup="gcr:Document"/>
<xs:complexType name="CO">
    <xs:complexContent>
        <xs:extension base="gcr:Document">
            <xs:sequence>
                <xs:element name="Name" type="xs:string"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:schema>

```

### Example 14

## 2.7. COS Versioning

Definitions in COS shall use the following format for versioning:

**x.y**

where x is the major version number and y is the minor version number. The major version number is separated from the minor version number by a period. New versions with compatible changes will be reflected with upgrade in minor version numbers. The upgrade of major version number usually reflects incompatible changes.

In order to ensure that an XML document is validated against the intended XML schema, versioning of XML documents and corresponding schemas needs to be properly designed and aligned. (The following design follows best practices discussed at <http://www.xfront.com/Versioning.pdf>.) To do so, the schema version is declared in the schema header using the optional “version” attribute, as shown in Example 15:

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.aecXML.org/COS"
    xmlns:cos="http://www.aecXML.org/COS"

```

```
elementFormDefault="qualified"  
attributeFormDefault="unqualified"  
version="1.3">
```

### Example 15

Note that the “version” attribute cannot be used directly by XML parsers for validating instances. In addition, each COS definition shall contain a required attribute, “schemaVersion,” which shows exactly the schema version with which a COS instance shall comply:

```
<xs:attribute name="schemaVersion" type="xs:decimal" use="required"/>
```

There are two ways to use such a design. The first is to add an additional attribute “fixed” to the “xs:attribute” definition:

```
<xs:attribute name="schemaVersion" type="xs:decimal" use="required" fixed="1.0"/>
```

In this case, an XML document must contain the attribute “schemaVersion” with a value of “1.0” in order to pass validation. The advantage of this method is that an instance is guaranteed to comply with the schema version. The disadvantage is that does not permit an instance to indicate that it is valid using multiple versions of a schema.

The second approach, aiming at relieving the strict constraint of the first use, does not declare the “fixed” attribute. The schemaVersion only reflects the version number with which an instance is compatible. The schemaVersion thus can be a list of numbers with which an instance is deemed to be compatible. Such a design provides some flexibility. In addition, the “version” attribute is still used to indicate the schema version. However, conventions need to be set up to specify acceptable compatibility. For example, the conventions may say that an instance with a version of “1.1” is compatible with schemas having a higher version number, as shown in Example 16, in which a “schemaVersion” set to “1.1” is still valid with schemas having “version” set to “1.2” or “1.3”:

```
<Address xmlns="http://www.aecXML.org/COS"  
xmlns:cos="http://www.aecXML.org/COS"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```

xsi:schemaLocation="http://www.aecXML.org/COS/GCRT/.xsd"
schemaVersion="1.1">
  <StreetLine>Any place</StreetLine>
  <PostalBox>007</PostalBox>
</Address>

```

### Example 16

The second use of the design is more flexible. Instance documents, for example, may not have to change if they remain valid with the newer schema version, and the design offers an alternative to using schemaLocation as a means to point to the correct schema version. The implementation of this design, however, requires extra processing by an application.

It is up to applications to determine the use of the design. However, the “version” and the “schemaVersion” attribute are required for COS definitions. The “fixed” attribute is optional, depending on implementation.

## 3. AGCXML SCHEMA STRUCTURE

### 3.1. A High-Level agcXML Schema Structure

Figure 5 illustrates a high-level agcXML schema structure, indicating that the agcXML definitions may come from two types of sources, COL definitions and user definitions.

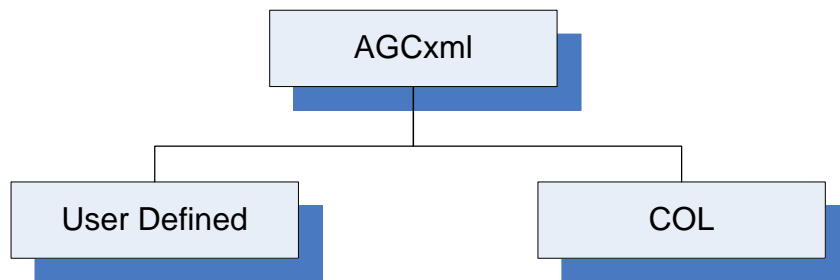


Figure 5

### 3.2. agcXML Namespace Declaration

aecXML™ recommends a namespace reference scheme as:

```

http://www.aecXML.org/schemas/[Level]/[WorkingGroupName]/[NamespacePrefix]-[ISODATE]

```

[Level] can be one of the following:

- WD (Working Drafts): submitters can put their schemas into this category for public use and review.
- CR (Candidate Recommendation): a schema in this category has been raised to Candidate Recommendation status for final public review and is put into this namespace by the aecXML™ Technical Committee (submitters cannot put schemas in this level).
- REC (Recommendation): a schema in this category has been raised to Recommendation status and is put into this namespace by aecXML™ Technical Committee (submitters cannot put schemas in this level).

[WorkingGroupName] is specified by aecXML™. Note that the working group name does not have to be one of the six existing working groups. [NamespacePrefix] is usually a domain-specific name under a working group (for example, LandXML). Finally, [ISODATE] specifies the date of the schema (for example, October 18, 2000 is represented by 20001018).

Namespaces of agcXML schemas will be declared using the following rules:

- [Level] shall correspond to the status of an aecXML recommendation, either WD, CR or REC;
- [WorkingGroupName] shall be “AGC” in upper case;
- [NamespacePrefix] shall be a concise name, or abbreviation, of the document to be developed; and
- [ISODATE] shall be the date of the schema.

For example, if an agcXML schema is adopted by aecXML/IAI as a working draft on June 1, 2007, then the namespace URI declaration would be as follows:

<http://www.aecXML.org/AGC/agcXML-20070305>

This namespace declaration is a simplified version that does not contain the “level” part. The status of a document, which is specified by [Level], can be indicated outside the namespace declaration.

### 3.3. Schema Header Settings

Each agcXML schema should declare a targetNamespace designated to the schema under development as shown in Example 17. The W3 XML Schema namespace should be set as shown and use “xs” as a prefix. No default namespace is declared. All names have prefixes. The elementFormDefault is set to “qualified” and the “attributeFormDefault” is set to “unqualified.” Since domain-specific schemas of agcXML will use different namespaces from those used by COS type libraries, “import” statements are used in order to reuse COS definitions.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:agcxml="http://www.aecXML.org/AGC/agcXML-20070305"
  xmlns:cos="http://www.aecXML.org/COS"
  targetNamespace="http://www.aecXML.org/AGC/RFI-20070305"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.1">
```

#### Example 17

### 3.4. Root Element

agcXML has one root element called “agcXML”. It is a collection of all definitions in the agcXML namespace (See Example 18). To provide flexibility, minOccurs and maxOccurs are set to “0” and “unbounded”. It is up to individual implementers to determine the business constraints.

```
<!-- The Root Definition-->
<xs:element name="agcXML" type="agcxml:agcXML"/>
<xs:complexType name="agcXML">
  <xs:sequence>
    <xs:element ref="agcxml:RFI"/>
    <xs:element ref="agcxml:Acknowledgement"/>
    <xs:element ref="agcxml:Contract"/>
    <xs:element ref="agcxml:Notification"/>
    <xs:element ref="agcxml:Organization"/>
    <xs:element ref="agcxml:Package"/>
    <xs:element ref="agcxml:Party"/>
```

```

    <xs:element ref="agcxml:Person"/>
    <xs:element ref="agcxml:PostalAddress"/>
    <xs:element ref="agcxml:Project"/>
    <xs:element ref="agcxml:Reference"/>
    <xs:element ref="agcxml:Request"/>
    <xs:element ref="agcxml:Response"/>
    <xs:element ref="agcxml:TeleCommunicationAddress"/>
  </xs:sequence>
  <xs:attribute name="schemaVersion" type="xs:decimal" use="optional"/>
</xs:complexType>

```

### Example 18

## 3.5. Naming Convention

agcXML shall follow the XML 1.0 Recommendation on naming restriction. The naming convention derived in this document follows the principles of ISO 11179 Part 5—Naming and Identification Principles. The objective is to ensure name consistency, name appearance, and name semantics. This section describes in detail the major elements in the naming convention:

- Scope of the naming convention
- Authority establishing names
- Semantic rules
- Syntactic rules
- Lexicon rules
- Rule to maintain unique names

### 3.5.1. Scope of the Naming Convention

The naming convention specified in this document applies to the naming of XML elements, complex types, simple types, attributes and extended data types in aecXML family schemas, such as those in COS and agcXML. The naming convention is prescriptive. Rules for formulating names are specified and such rules will be enforced if a schema is to be considered as an aecXML family schema.

### 3.5.2. Authority Establishing Names

The aecXML/IAI technical committee will enforce the compliance of a candidate aecXML family schema with the naming convention.

### 3.5.3. Semantic Rules

The semantic rules determine the source and content of the XML terms used for XML schema complex types, simple types, elements and attributes. If there is an underlying model for the XML schema, such as an EXPRESS model or a UML model, the names of the XML components are derived from the corresponding model parts. For example, ifcXML applies the following rules for mapping names between XML schema and EXPRESS components:

1. ENTITY names shall be mapped into XML element or complexType names
2. SELECT type names shall be translated into XML group names.
3. TYPE names shall be mapped into XML simpleType names.
4. Attribute names shall be translated into local XML element names.

In general, specifications such as ISO 10303-28 and XMI Mapping shall be used as references for deriving names for XML components. If there is not a specific underlying model, it is recommended to follow the ISO 11179 Part 5 for develop names. According to ISO 11179 Part 5, name parts include object class terms, property terms, representation terms and qualifier terms (ISO 11179 2005). In addition, the following principles shall be followed for COS or agcXML schemas:

- The names of schema components shall be in English.
- The names of schema components shall use common terms used in the AEC industry.
- The names of complexTypes shall represent a set of ideas, abstractions, or things in the real world whose properties and behavior follow the same rules. In addition, a complex type element is an XML element that contains other elements and/or attributes.
- The names of simpleTypes shall represent constraints or information about the values of attributes or text-only elements.
- The names of XML elements shall be derived by using the corresponding complexType or simpleType names. In addition, if the element is an association type element, the naming shall indicate the reference direction.

- The names of XML attributes represent characteristics of elements or associations with other elements.
- Regardless of whether there is an underlying model, each name shall have a dictionary entry (see Section 2.4.2).

#### **3.5.4. Syntactic Rules**

agcXML names are single or compounded English terms that directly represent the meaning of the corresponding XML components, such as complexTypes or attributes. If compounded names are used, it is recommended that the sequence of the terms reflect the following pattern:

- object class terms followed by property terms followed by representation terms, and
- quantifiers shall precede the terms that they quantify.

For example, the name CostBudetPeriodTotalAmount reflects the application of such a pattern (ISO 11179, 2005). In many cases object class terms, property terms, and representation terms do not need to be compounded when naming XML types, element, attributes and so on. If an element represents a reference, the string “Rel” shall be appended to the last position of a name following the name of the referenced element. The format is:

“ReferencingElmentReferenceElementRel”

In some cases when multiple words are used for a single common name representing an object or document in AEC-FM, these words don’t suggest any pattern as discussed above. They are there simply because they are part of an atomic name, such as RequestForInformation. When multiple words are used for a single name, a word is concatenated to the preceding word in this manner without using a space or other characters such as “\_”, “-“, “/”, “\” or “.”.

#### **3.5.5. Lexicon Rules**

The appearance of names shall follow the following rules:

1. Names for complexType are UpperCamelCase
2. Names for group are UpperCamelCase
3. Names for simpleType are UpperCamelCase
4. Names for element are UpperCamelCase
5. Names for attribute are LowerCamelCase

6. Name for attributeGroup are LowerCamelCase

Other rules include:

1. Nouns are used in singular form only.
2. Verbs (if any) are in the present tense.
3. Name parts or words in multi-word terms shall follow the above appearance rule. Not special characters shall be allowed.
4. Abbreviations may be developed by following rules developed by Information Warehouse & Access (IWA), [http://www.ais.vt.edu/IWA/naming\\_stds.html](http://www.ais.vt.edu/IWA/naming_stds.html) (see Appendix III).
5. Acronyms are only allowed when such acronyms, represented all in upper-case letters, are industry norms such as RFI.

### **3.5.6. Uniqueness Principles**

All names in agcXML or COS must be unique by themselves or become unique when they are qualified with a namespace prefix.

## **3.6. Customization and Extensibility**

agcXML domain-specific schemas may be developed by extending COS definitions and/or by using user-defined elements.

### **3.6.1. Extending Using COS Definitions**

Customization of the value space of attributes is allowed only by using XML Schema built-in mechanisms. For example, use “maxInclusive” to restrict the length of an integer, or use “union” to expand an enumeration list, or use “pattern” to specify the input facet for an email address. XML attribute customization or extension is not allowed.

Extending COS element content by using existing COS definitions is straightforward. For example, if the original COS Person definition does not include an attribute called “EmploymentDate” and the attribute is defined in COS as a reusable attribute, then a domain-specific schema that uses the Person definition may extend this definition by including an “EmploymentDate” element.

Adding an association between two elements, which is not specified in COS, is also simple. The domain-specific schema needs to declare a relationship element, which is not

contained in AST. The mechanism is very similar to that discussed in the COS section. In Example 19, it is assumed that the association between Person and Address is not specified in COS, so a “PersonAddressRel” is created in the domain-specific schema:

```
<xs:element name="PersonPostalAddressRel" type="agcxml:PersonPostalAddressRel" nillable="true"/>
<xs:complexType name="PersonPostalAddressRel">
  <xs:complexContent>
    <xs:extension base="gcr:Element">
      <xs:sequence>
        <xs:element ref="agcxml:PostalAddress"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### Example 19

#### 3.6.2. Using User-defined Elements

User-defined elements are allowed to be added to a content model (e.g., sequence, all, and choice). The user-defined elements will always be added as the last element within the content model. Such a mechanism is also applied by the cfiXML design (Palmer et al 2004)

```
<xs:element name="RFI" type="agcxml:RFI"/>
<xs:complexType name="RFI">
  <xs:complexContent>
    <xs:extension base="gcr:Document">
      <xs:sequence>
        <xs:element ref="gcr:Person" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gcr:Address" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="CustomItems">
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="gcr:Custom">
                <xs:sequence>
                  <xs:element name="Note" type="xs:string"/>
                </xs:sequence>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```

        </xs:complexContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

### Example 20

In the above example, the RFI schema extends the Custom element from COS and adds a Note definition. This is just one example. There are other approaches to reusing the Custom element.

### 3.7. Versioning

Versioning of agcXML domain-specific schemas is similar to that of COS schemas, using the following format:

**x.y**

where x is the major version number and y is the minor version number. The major version number is separated from the minor version number by a period. New versions with compatible changes will be reflected by an upgrade in minor version numbers. The upgrade of major version number usually reflects incompatible changes. The schema version is declared in the schema header using the optional “version” attribute, as shown in Example 21, below:

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:agcxml="http://www.aecXML.org/AGC/agcXML-20070305"
  xmlns:cos="http://www.aecXML.org/COS"
  targetNamespace="http://www.aecXML.org/AGC/agcXML-20070305"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0">
...
<xs:attribute name="schemaVersion" type="xs:decimal" use="required"/>

```

### Example 21

The reason for using such a design is similar to that discussed in Section 2.7.

## 4. APPENDIX I: REVIEW OF SOME EXISTING STANDARDS

This section provides an overview of some existing standardization efforts relevant to agcXML. Related technical details such as namespace design, naming issues, the reuse of IFC definitions, the handling of unstructured content, and schema structures will be discussed in Appendix II.

### 4.1. bcXML

The goal of the European eConstruct project is to develop, implement, demonstrate and disseminate a new communication technology—bcXML—for the European Building and Construction Industry (Lima et al, 2002). The eConstruct project focuses on four areas: use cases, modeling, reference architecture, and prototype implementation.

bcXML is designed to support two genres of applications: eCommerce and eBusiness. eBusiness, with a focus on collaboration among project participants throughout the lifecycle of a project, has a larger scope than eCommerce, whose major focus is on buying and selling construction-related products and services.

bcXML currently is available in both “heavy” and “lite” versions. The heavy bcXML meta-schema provides a solution that can integrate bcXML with other product models such as ifcXML and other standards such as Electronic Data Interchange (EDI), and is designed mainly for eBusiness applications. The bcXML lite version, focused on eCommerce applications, defines a simpler and more flexible solution. It makes full use of XML technology, though a few features such as Product Data Technology (PDT) interoperability are compromised.

The bcXML modeling work is done at different levels of abstraction: generic, specification, and occurrence. The meta-schema includes a meta-definition of a template used to define taxonomy for a given industry called eXtensible Taxonomy Definition (XTD). The schema level includes the actual taxonomy definitions, called bcBuildingDefinitions, which can “in turn be instantiated to create product catalogue content and also for the actual specification requirements and the returned solution message.” (Stephens et al, 2002).

The taxonomy of building-construction-related concepts in bcXML is mostly defined through bcDictionary and bcTaxonomy, developed by the LexiCon project. More information about dictionaries is provided in section **Error! Reference source not found.** below.

The bcXML approach has a strong transaction-based flavor because it is based on the ebXML framework (Tolman et al 2001).

According to existing bcXML reference documents, major results of the projects include the XTD Meta-Schema and the bcBuildingSpecification schema. The lite version has received more attention than the heavy version. The European eConstruct project was completed in 2003. The future maintenance of bcXML is uncertain (<http://www.econstruct.org>).

## **4.2. cfiXML**

The FIATECH Automating Equipment Information Exchange (AEX) project, in collaboration with the AIChE Design Institute for Physical Properties (DIPPR) 991 project, ePlantData, and the National Institute for Standards and Technology (NIST) AEX project, focuses on developing a set of XML-based specifications—Capital Facility Industry XML (cfiXML)—to support the automation of information sharing associated with the design, procurement, delivery, operation, and maintenance of engineered equipment (<http://www.cfixml.org/>).

The initial scope of the cfiXML schemas covers equipment that is deemed important to the participants. The list of equipment includes centrifugal pumps and shell and tube exchangers. The participants also recognize five business scenarios in which the application of such a technology may produce significant financial benefits: Request for Quote (RFQ), Quote, Purchase Order (PO), as-built documentation, and Bill of Materials (BOM).

The design of the cfiXML schemas follows an object-oriented approach. The initial design includes objects and properties representing basic equipment information found in various equipment lists or bill of materials documents, equipment datasheets for shell and tube exchangers and centrifugal pumps, and process materials, calculation methods, and experimental property data. However, the design is not limited to these types of equipment. The current cfiXML schemas are intended to handle any equipment items, any engineering documents, and any material properties.

The AEX schemas have four basic parts:

- Core data type schemas that extend the data types included in XML Schema to meet engineering data type requirements;
- Core engineering object schemas that include reusable base engineering object definitions;

- Subject engineering object schemas that provide definitions for specific engineering items; and
- Collection-container schemas that are used to model engineering document schemas.

Many of the design characteristics reflect the best practice of XML design

(<http://www.xfront.com/BestPracticesHomepages.html>). Unlike the design of bcXML, which directly incorporates IFC concepts into its “heavy version” meta-schema, the AEX project applies a mapping approach (Begley et al. 2005). The report uses an example—a centrifugal pump—to demonstrate the possibility of mapping between ifcXML and cfiXML representations. The case study seems to suggest that the data file expressed in cfiXML is less voluminous than the same data file expressed in ifcXML, but this is a single instance.

### **4.3. ebXML**

A joint effort of the United Nations Center for Facilitation of Procedures and Practices for Administration, Commerce and Transportation (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS), ebXML is intended to provide an open XML-based infrastructure that enables global eCommerce in an interoperable, secure, and consistent manner (<http://www.ebxml.org/>). Major ebXML components include:

- Messaging Services that specify the packaging, transport, and routing of ebXML messages. ebXML message format relies on the Simple Object Access Protocol (SOAP) specification, v1.1 with attachments. ebXML Messaging Services 2.0 was accepted by ISO/TS 15000-2 in May 2004, and version 3.0 shows compatibility with web services.
- Trading partner profiles and agreements, including collaboration protocol profile (CPP) and collaboration protocol agreement (CPA), that enable trading partners to establish eBusiness relationships. CPP, derived from business process specification, captures the interaction among collaborating parties and the way that data are exchanged. CPA describes the agreement between two companies on technical characteristics that define features, services, and processes in the eBusiness relationship between the two companies. The Trading Partner Profile and Agreement was accepted by ISO (ISO 15000-01 ISO/TC 154) as a standard in May 2004.

- Business process specification supports using Unified Modeling Language (UML) and XML schemas for defining business processes. Following ISO 14662, the business process specification has two views: the Business Operation View (BOV) and the Functional Service View (FSV). The BOV, focusing on high-level interactions among companies and related requirements, concerns the semantics of business data as well as accepted conventions and agreements. The FSV addresses system interactions and requirements, including issues such as message syntax, naming conventions, and communications protocols.
- Registries and repositories that store the data objects used in ebXML messages and list potential trading partners, including but not limited to the submission of industry-wide schemas or document type definitions that define industry messages and vocabularies, the submission of industry business process models, the submission of company collaboration protocol profiles, and the discovery of trading partners.
- The ebXML Core Components Technical Specification (CCTS) is a methodology for developing general business data types by using basic and reusable definitions provided by the specification. The core components include common modeling concepts for objects and data, naming conventions for defining generic semantic meaning in dictionary entry names, and reusable data types. The CCTS also was approved by ISO in 2005.

ebXML provides a generic infrastructure for conducting eBusiness and eCommerce. The need for creating a mechanism for sharing domain-specific business process data, if any, must be addressed by domain-specific standardization efforts.

#### **4.4. gbXML**

The green building XML (gbXML) schema is designed to support interoperability between CAD applications and energy and resource analysis tools in pre-design phases of building design. gbXML provides a detailed description for a single building or set of buildings. The descriptions, in XML format, can be used to determine the costs of operation, the pollution produced, operational energy requirements, and health issues (<http://www.gbxml.org/>). The gbXML schema is currently a working draft schema of aecXML.

#### **4.5. ifcXML**

The objective of the Industry Foundation Classes XML (ifcXML) extraction and evaluation project is to provide the internationally agreed-upon content and structure of the IFC2x specification (and any valid subset thereof) to the XML community (Liebich, 2001).

ifcXML enables the exchange of IFC data files in XML format and the reuse of IFC content and structure within XML-based initiatives for data exchange and sharing. The scope of ifcXML includes the ifcXML methodology to generate XML schema from the EXPRESS-based IFC model, the ifcXML schema, and the XML data files.

The ifcXML methodology allows automatic conversion of the EXPRESS-based IFC model into XML format based on the ISO 10303 Part 28 Edition 2. It has been approved by the International Alliance for Interoperability (IAI) as a formal methodology to generate XML definitions for IFC2x and future versions. It has been further approved for the generation of ifcXML for IFC2x2 and IFC2x3 to comply with more recent versions of ISO10303-28.

The current ifcXML schema is generated by an automatic translation of IFC EXPRESS definitions into XML format. The methodology also allows the translation rules to be configured so that a more compressed, “XML-like” language definition can be generated. This approach has been proposed as the optimized ifcXML, or flat ifcXML, and is intended to serve as a Common Object Schema Repository for other standards such as bcXML and aecXML, which reuse the ifcXML definitions (Liebich, 2004). The current ifcXML specification is available at <http://www.iai-international.org/Model/IfcXML2.htm>.

#### **4.6. MIMOSA**

The Machinery Information Management Open System Alliance (MIMOSA) is a trade association dedicated to developing and encouraging the adoption of open information standards for operations and maintenance in manufacturing, fleet, and facility environments (<http://www.mimosa.org/>). MIMOSA's open standards are intended to enable collaborative asset lifecycle management in both commercial and military applications. OpenO&M<sup>TM</sup>, maintained by MIMOSA, mainly supports the following functional areas:

- Collaborative Asset Life-cycle Management (CLAM)
- Universal Identification (UID)
- Condition-Based Maintenance (CBM)

- Condition-Based Operation (CBO)

Major design artifacts include an Open System Architecture for Enterprise Application Integration (OSA-EAI) terminology dictionary, a Common Conceptual Object Model (CCOM), a Common Relational Information Schema (CRIS), CRIS Reference Database Specifications, a Tech-Doc Producer/Consumer V3.0 Specification, a Tech-XML Client/Server V3.0 Specification, a Tech-File XML File Import/Export Specification, and a Tech-Web HTTP Client/Server Specification.

#### **4.7. OSCRE**

The Open Standard Consortium for Real Estate (OSCRE) is an industry organization that promulgates a set of definitions and rules that facilitate automatic data/information sharing among various software systems used in the real property industry (<http://www.oscre.org/>).

As of this writing, OSCRE's published standards have been implemented by the real property industry for some time. OSCRE v6.1 supports business processes including:

- Core/Management
- Valuation
- Discounted Cash Flow (DCF)
- Analysis
- Financial transactions
- Indirect property
- Legal
- Searches

The overall design of OSCRE XML schemas is very XML-like and concise.

## 5. APPENDIX II: REVIEW OF THE TECHNICAL DESIGN OF SOME EXISTING STANDARDS

In this section, the technical design of some existing standards, including namespace, the reuse of existing schemas, extensibility, semantics, the handling of unstructured content, process-oriented design, versioning, and naming conventions are reviewed in detail.

### 5.1. Namespace

A namespace in an XML schema defines a collection of XML elements or definitions identified by a Universal Resource Identifier (URI) reference. It is a solution mainly for resolving naming collisions. One of the key concepts is the `targetNamespace`, which identifies elements and attributes to be developed in a particular schema. In this subsection, namespace designs of aforementioned schemas are reviewed.

#### 5.1.1. bcXML

bcXML has schemas defined at two levels: meta-schema for taxonomy, and a building and construction specification schema for applications such as catalogue. For example, the namespace of the heavy meta-schema is declared as follows:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.bcXML.org/2001/bcXML"
            xmlns="http://www.bcXML.org/2001/bcXML">
...
</xsd:schema>
```

Such a namespace seems to be consistently applied to all other bcXML schemas. The specification schema, for example, uses the same type of declaration. In this case, the namespace identified by the URI, `http://www.bcXML.org/2001/bcXML`, is declared as a default namespace as well as a `targetNamespace`.

Two other attributes of the `xsd:schema` element, `elementFormDefault` and `attributeFormDefault`, are not defined, indicating that all elements and attributes are not qualified; that is to say, it is not necessary to use namespace prefixes for elements or attributes in instance documents.

### 5.1.2. cfiXML

cfiXML schemas have a unique, hierarchical, and consistent namespace design. The root namespace reference for all cfiXML schemas is “http://www.cfixml.org.” Each individual namespace reference is constructed based on the root namespace. For example, the URI namespace reference for the basic equipment schema is “http://www.cfixml.org/mtrl.” The string “mtrl” is also used as a prefix for qualifying elements or attributes within the namespace if needed. In AEX schemas, the targetNamespace is set by using the following declaration statements:

```
targetNamespace="http://www.cfixml.org/mtrl"  
xmlns=http://www.cfixml.org/mtrl
```

The targetNamespace is also the default namespace. The attributeFormDefault is set to “unqualified” and the elementFormDefault is set to “qualified”. However, all global elements must always be qualified.

### 5.1.3. gbXML

The declaration of namespace in gbXML is shown below:

```
targetNamespace="http://www.gbxml.org/schema/0-34"  
xmlns=http://www.gbxml.org/schema/0-34"
```

The two attributes, attributeFormDefault and elementFormDefault are set to “unqualified”.

### 5.1.4. ifcXML

The ifcXML uses the following namespaces declarations in the XML schema header:

```
<xs:schema  
xmlns:xs="http://www.w3.org/2001/XMLSchema"  
xmlns:ex="urn:iso.org:standard:10303:part(28):version(2):xmlschema:common"  
xmlns:xlink="http://www.w3.org/1999/xlink"  
xmlns:ifc="http://www.iai-international.org/ifcXML2/RC2/IFC2X2_FINAL"  
targetNamespace="http://www.iai-international.org/ifcXML2/RC2/IFC2X2_FINAL"  
elementFormDefault="qualified"
```

```
attributeFormDefault="unqualified">...  
...  
</xsd:schema>
```

The targetNamespace is set for definitions of this particular schema, identified by the URI “http://www.iai-international.org/ifcXML2/RC2/IFC2X2\_FINAL”. Since all namespaces have prefixes, there is no default namespace in this declaration. In addition, elements need to be qualified while attributes do not.

### 5.1.5. OSCRE

The declaration of namespace in OSCRE is a little different. OSCRE does not declare a targetNamespace:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
attributeFormDefault="unqualified"  
elementFormDefault="qualified"  
  
...  
</xs:schema>
```

This is also a Chameleon namespace design.

## 5.2. Element and Attribute Naming Issues

### 5.2.1. bcXML

bcXML does not clearly define the naming scheme. Within its schemas, one can find a mixed use of UCC (Upper Camel Case) and LCC (Lower Camel Case) for element definitions. However, type definitions are always UCC and attribute definitions are always LCC. bcXML does not specify the use of abbreviations and acronyms.

### 5.2.2. cfiXML

According to cfiXML reference documents, the AEX project has recommended that:

- Upper Camel Case (UCC) is to be used for XML complex or simple type names; and
- Lower Camel Case (LCC) is to be used for XML element and attribute names.

This design is slightly different from the existing aecXML style guideline, which states that elements use UCC. The aecXML style guideline does not specify naming conventions for simple or complex type names. AEX schemas allow abbreviations and acronyms. In addition, AEX schemas only allow alpha-numeric characters for names.

### **5.2.3. gbXML**

gbXML follows the rules for aecXML schemas: UpperCamelCase for elements and LowCamelCase for attribute.

### **5.2.4. ifcXML**

ifcXML is generated from EXPRESS schemas, therefore there are mapping rules between IFC definitions in EXPRESS format and XML schemas (Nisbet and Liebich 2005). The current rule is that the mapping into ifcXML does not change the name formatting; the particular EXPRESS to XSD configuration naming-convention = "preserve-case" must be used. As a result, the following naming rules apply:

- ENTITY names are mapped into XML element and complexType names in UpperCamelCase.
- SELECT type names are mapped into XML group names in UpperCamelCase.
- TYPE names are mapped into XML simpleTypes in UpperCamelCase.
- Attribute names (comprising attribute and relationship names) are mapped into local XML element names in UpperCamelCase.

### **5.2.5. OSCRE**

OSCRE uses one naming format. All elements, complexTypes and attributes are UpperCamelCase. In OSCRE schemas, simpleType is used mostly for customizing data types.

## **5.3. Elements vs. Attributes**

Another authoring style is the handling of attributes of objects/entities. They can be mapped to XML elements or attributes. Although either approach is fine, it is ideal to have a consistent style. bcXML, cfiXML and ifcXML schemas use the first approach, i.e., attributes of objects are mapped to XML elements. XML attributes are used only in limited cases, such as IDs.

## 5.4. Reuse of Existing Schemas

Although it would be ideal to reuse existing definitions, it seems that only a few standardization efforts are very seriously considering such an idea, such as bcXML and the AEX project.

However, these two projects seem to take two different approaches.

### 5.4.1. bcXML

bcXML explicitly incorporates ifcXML definitions into its meta-schema. For example, in its “heavy version” meta-schema the bcXML “bcTaxonomyItem” makes references to the ifcXML definitions and the bcXML “SpecificationLocation” makes references to the “Organization” of the “Actor Resources” of the IFC model (Böhms et al. 2001). The conceptual design is implemented by using “include” in the XML Schema definitions according to bcXML documentation:

```
<xsd:include schemaLocation="http://www.bcXML.org/2001/IfcXML_COS.xsd"/>  
<xsd:include schemaLocation="http://www.bcXML.org/2001/ActorResource.xsd"/>
```

In addition, the definition of the “bcXML” element includes a reference to “ifcXML\_COS,” which brings ifcXML Common Object Schema definitions to the bcXML schema. However, the bcXML lite version does not address reusing IFC definitions.

### 5.4.2. cfiXML

In the conceptual architecture of cfiXML schemas, there is no specific design for reusing definitions from other standards. The cfiXML schemas are self-contained. According to a NIST study comparing ifcXML and cfiXML, by defining the same object—a pump, for example—many of the definitions in the ifcXML and the AEC schemas can be mapped. There are cases, however, where mappings may not be straightforward (Begley et al 2005), such as:

6. When there are a number of one-to-many mappings. The GlobalID of ifcXML, for example, may possibly be mapped to obj:organizationID and obj:objected in cfiXML schemas, and the FamilyName, GivenName and MiddleName in ifcXML may be mapped to ctx:person or obj:name in cfiXML schemas. To handle one-to-many mappings, additional mechanisms may be needed.

7. When elements take on different sets of values. The ChangeAction in ifcXML, for example, has fewer enumerated values than its counterpart in cfiXML schemas.
8. When the intention of an element may not be sufficiently clear to other schema developers to permit effective mapping. The intention of FlowResistanceRange in ifcXML, for example, is not clear to cfiXML schema developers.
9. When use cases are different. For example, ifcXML does not distinguish between “available” and “required” for NetPositiveSuctionHead, while the cfiXML schemas do.

These differences in the design of the two schemas unavoidably arise from the fact that the two standards were developed separately to address different scopes and business applications. On the other hand, one may question whether it is necessary to maintain a perfect mapping between two schemas based on different use cases and intended to serve different applications.

### **5.4.3. Others**

Other aforementioned schemas, such as MIMOSA, OSCRE, LandXML and gbXML do not explicitly reference or reuse other schema definitions.

## **5.5. Handling Unstructured Content**

Most of the existing standardization efforts do not address this issue specifically; it is assumed that they are dealing with structured information only. In document-based information exchange, this is not always the case. Many construction documents are actually a mix of structured and unstructured information.

Meta-data models can be used to describe unstructured information. The application of meta-data can be at the document level or the content level. For example, at the document level Microsoft Word applies a meta-data model for describing each document instance. The meta-data model describes some common features for any Word document, such as title, subject, author, creation date, last modified date, and so on. The agcXML project will only focus on document-level meta-data. It is beyond the scope of this project to address the matter of unstructured data at the content level.

Document meta-data models developed in the AEC-FM industry already exist. The IFC External Reference Resource, for example, has an entity called ifcDocumentInformation

([http://www.iai-international.org/Model/R2x3\\_final/index.htm](http://www.iai-international.org/Model/R2x3_final/index.htm)). The ifcXML definition of the entity, which has basic meta-data for generic documents, is shown below:

```
<xs:element name="IfcDocumentInformation" type="ifc:IfcDocumentInformation"
  substitutionGroup="ex:Entity" nillable="true"/>
<xs:complexType name="IfcDocumentInformation">
  <xs:complexContent>
    <xs:extension base="ex:Entity">
      <xs:sequence>
        <xs:element name="DocumentId" type="ifc:IfcIdentifier"/>
        <xs:element name="Name" type="ifc:IfcLabel"/>
        <xs:element name="Description" type="ifc:IfcText" nillable="true" minOccurs="0"/>
        <xs:element name="DocumentReferences" nillable="true" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="ifc:IfcDocumentReference" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute ref="ex:itemType" fixed="ifc:IfcDocumentReference"/>
            <xs:attribute ref="ex:cType" fixed="set"/>
            <xs:attribute ref="ex:arraySize" use="optional"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="Purpose" type="ifc:IfcText" nillable="true" minOccurs="0"/>
        <xs:element name="IntendedUse" type="ifc:IfcText" nillable="true" minOccurs="0"/>
        <xs:element name="Scope" type="ifc:IfcText" nillable="true" minOccurs="0"/>
        <xs:element name="Revision" type="ifc:IfcLabel" nillable="true" minOccurs="0"/>
        <xs:element name="DocumentOwner" nillable="true" minOccurs="0">
          <xs:complexType>
            <xs:group ref="ifc:IfcActorSelect"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="Editors" nillable="true" minOccurs="0">
          <xs:complexType>
            <xs:group ref="ifc:IfcActorSelect" maxOccurs="unbounded"/>
            <xs:attribute ref="ex:itemType" fixed="ifc:IfcActorSelect"/>
            <xs:attribute ref="ex:cType" fixed="set"/>
            <xs:attribute ref="ex:arraySize" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```

        </xs:complexType>
    </xs:element>
    <xs:element name="CreationTime" nillable="true" minOccurs="0">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="ifc:IfcDateAndTime"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="LastRevisionTime" nillable="true" minOccurs="0">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="ifc:IfcDateAndTime"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="ElectronicFormat" nillable="true" minOccurs="0">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="ifc:IfcDocumentElectronicFormat"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="ValidFrom" nillable="true" minOccurs="0">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="ifc:IfcCalendarDate"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="ValidUntil" nillable="true" minOccurs="0">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="ifc:IfcCalendarDate"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

```

```

        <xs:element name="Confidentiality" type="ifc:IfcDocumentConfidentialityEnum"
            nillable="true" minOccurs="0"/>
        <xs:element name="Status" type="ifc:IfcDocumentStatusEnum" nillable="true"
            minOccurs="0"/>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

In addition, ISO 82045-5:2005, “Application of metadata for the construction and facility management sector,” specifies metadata elements and methods for sharing and exchanging management data (metadata) for documents, to be used with electronic or paper-based document management systems (ISO 82045, 2005).

ISO 82045-5:2005 specifies required and optional metadata for managing the lifecycle of documents, including creating, exchanging and distributing, checking and approving, searching and retrieving, reusing and revising, and archiving. ISO 82045-5:2005 also specifies a basic set of metadata and additional metadata for document sharing, document exchange, and document archiving. One particular metadata that the ISO 82045-5 discusses is the classification metadata “Document class,” which can be used to describe a technical document from different facets, e.g., document kind, document purpose, discipline, element, drawing scale and others.

The ifcDocumentInformation is able to provide most of the metadata that belong to the basic set, but not those for document sharing, document exchange, and document archiving as specified in ISO 82045-5:2005.

## 5.6. Use of “Dictionary”

Several aforementioned standards use the concept of dictionary to record, define, and share the meaning of terms applied in schemas. A bcXML schema is constructed from finer components, including terms, vocabulary, and dictionary. According to bcXML documents, a term is a word/string/phrase used to denote an object, an activity, a property, a measure, or a unit. A vocabulary is a set of terms relevant to a specific domain. A dictionary is made up of vocabularies. A term in a dictionary has a verbal description for human comprehension (Böhms et al. 2001). MIMOSA maintains a simple format of dictionary containing a list of terms and their definitions (MIMOSA 2003).

Many other standards maintain a catalogue of definitions. The catalogue will usually provide information such as the name, a short description, version, and other relevant information of a term. IFC, and ifcXML in consequence, have an access mechanism to external dictionaries, the IfcLibraryReference. The element has the basic meta-data for accessing any dictionary server. The following shows the ifcXML definitions:

```

<xs:element name="IfcExternalReference" type="ifc:IfcExternalReference" abstract="true"
  substitutionGroup="ex:Entity" nillable="true"/>
<xs:complexType name="IfcExternalReference" abstract="true">
  <xs:complexContent>
    <xs:extension base="ex:Entity">
      <xs:sequence>
        <xs:element name="Location" type="ifc:IfcLabel" nillable="true" minOccurs="0"/>
        <xs:element name="ItemReference" type="ifc:IfcIdentifier" nillable="true" minOccurs="0"/>
        <xs:element name="Name" type="ifc:IfcLabel" nillable="true" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="IfcLibraryReference" type="ifc:IfcLibraryReference"
  substitutionGroup="ifc:IfcExternalReference" nillable="true"/>
<xs:complexType name="IfcLibraryReference">
  <xs:complexContent>
    <xs:extension base="ifc:IfcExternalReference">
      <xs:sequence>
        <xs:element name="Language" type="ifc:IfcLanguageId" nillable="true" minOccurs="0"/>
        <xs:element name="ReferencedLibrary" nillable="true" minOccurs="0">
          <xs:complexType>
            <xs:choice>
              <xs:element ref="ifc:IfcLibraryInformation"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```
</xs:complexContent>  
</xs:complexType>
```

## 5.7. Extensibility

Extensibility is a capability that a standard offers to its users to extend definitions specified by the standard. In addition to the extensibility that is provided by XML Schema such as “extension” or “restriction,” many standards also intend to specify additional extensibility conventions. AEX schemas (Palmer et al, 2004) specify that:

- Extensions to enumeration data types are allowed through an AEX specified method, the union data type;
- Attribute extension is not allowed;
- Element extension is allowed by using an “any” type element. A “Custom” element, which is an “any” type, is appended to the last element in a content model. Through such a mechanism, a user-defined extension may be added to an AEX defined element;
- Schema extensions are managed by mirroring an original xsd file with a new file with a new name, i.e., the name of the original file appended with a letter “U”. The original file is “read-only, while the new file can be “read-write”.

Other aforementioned domain-specific schemas do not specifically discuss this subject, indicating that they limit such a capability to whatever capability that XML Schema supports.

## 5.8. Process-oriented Design

In many cases, the concept of process is often referred to as transactions. Domain-specific standards such as bcXML usually rely on domain-independent standards such as ebXML to provide the transaction services (Böhms et al. 2001). The design of cfiXML schemas also suggests the potential application of generic business process services such as RossettaNet and messaging by using SOAP (Teague 2004). Others have not specified a particular design for managing processes. Regardless, many existing domain-specific standardization efforts assume that this issue is out of their scope. Their focus is typically on defining domain-specific content. It is conceivable, however, that some domain-specific process-oriented requirements may be identified when proper use cases are determined. In those cases, the domain-specific standards need to address specific process requirements pertaining to their domains.

## 5.9. Versioning and Version Tracking

Although W3C may provide an example for handling schema versioning and release, many different XML schema development efforts resort to their own schemes.

### 5.9.1. aecXML

Existing aecXML documents do not specify schema versioning and release issue. They only provide a classification of schemas in terms of their status of acceptance (aecXML 2001), including:

- In use—these schemas are in consistent, active use by a reasonably large constituency. They are in active use in commerce, by industry groups, or contain objects that have no aecXML Common Object Repository (COR) or COS equivalent.
- Of conformance class 1—these schemas are assembled by using definitions from the COR of aecXML; and
- Of conformance class 2—these schemas are developed based on COR definitions but with an intention to be promoted as COS definitions.

### 5.9.2. cfiXML

cfiXML has developed its own schema versioning and releasing scheme. The schema releasing scheme includes a four-phase schema releasing strategy (cfiXML 2004), including:

- Trial version;
- Working version;
- $\beta$  public release; and
- Approved public release.

cfiXML also defines schema release designations. For each approved public release, there is a corresponding version number in the following format:

`xx.yyy.zzzz`

where, “xx” represents a major release number; “yyy” represents a minor release number; and “zzzz” represents a build or service pack number. Schema change and approval management procedures are also specified.

### **5.9.3. Others**

The remaining aforementioned domain-specific schema development efforts do not specifically address schema versioning, releasing, and change management issues in their reference documents.

## 6. APPENDIX III: STEPS FOR ABBREVIATING NAMES

A business name should not be abbreviated if it is possible to spell it out. But a technical name should always use standard abbreviations when they are available

([http://www.ais.vt.edu/IWA/naming\\_stds.html](http://www.ais.vt.edu/IWA/naming_stds.html)). The following rules should also be observed:

- Prime words, class words and modifying words may be abbreviated.
- Use the singular form of each word.
- Use the root form of each word (i.e., cancellation = cancel).
- Remove vowels (e.g., course = crs).
- If word begins with vowel or combination of vowels do not remove those vowels (e.g., automate = autmt).
- Drop the second consonant of a word with double consonants (e.g., enroll = enr).
- Drop underscores or separators.
- Drop unimportant modifiers.
- Drop unnecessary consonants from right to left but do not drop consonants in the class word.
- As a last resort, drop the prime word.
- Avoid creating acronyms or abbreviations that result in an English word (e.g., do not use “no” to mean “number”).

## 7. REFERENCES

1. aecXML (2001), “aecXML Schema Acceptance Criteria”, <http://www.iai-na.org/aecXML>.
2. Begley, E.F., Palmer, M.G. and Read, K.A. (2005), “Semantic Mapping between IAI ifcXML and FIATECH AEX Models for Centrifugal Pumps”, <http://www.fire.nist.gov/bfrlpubs/build05/art008.html>
3. Böhms et al. 2001, “bcXML Specification”, eConstruct,
4. cfiXML (2004), “cfiXML Process Document”, <http://www.cfixml.org/>.
5. Gallaher, M.P., O’Connor, A.C., Dettbarn, J.L. Jr. and Gilday L.T. (2004). “Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry.” NIST GRC 04-867, <http://www.bfrl.nist.gov/oa/publications/gcrs/04867.pdf>.
6. ISO11179 part 5, Information Technology – Metadata registries, part 5 Naming and identification principles, ISO, 2005.
7. ISO 82045 part 5, “Document Management Part 5: Application of metadata for the construction and facility management sector,” ISO, 2005.
8. ISO/DTS 15000-5 (2006), “Core Components Technical Specification, 2<sup>nd</sup> edition”, UN/CEFACT version 2.2.
9. Lima, C., Stephens, J. and Böhms, M. The bcXML Prototype: the eConstruct Approach Supporting eCommerce in the Building and Construction Industry. <http://www.eConstruct.org/>. 2002
10. MIMOSA (2003), “MIMOSA OSA-EAI Terminology Dictionary”, MIMOSA, <http://www.mimosa.org/>.
11. Nisbet, N. and Liebich, T. (2005), “ifcXML Implementation Guide”, IAI, <http://www.iai-international.org/>
12. Palmer, M., Burns, M. and Teague, T. (2004), “XML Schema Development Guidelines”, FIATECH, <http://www.cfixml.org/>.
13. Stephens et al. 2002, “eCommerce and eBusiness in the Building Construction Industry: Preparing for the Next Generation Internet”, final report, IST-1999-10303, eConstruct
14. Teague, T. (2004), “XML Schema Architecture” FIATECH AEX Project
15. Tolman, F., Böhms, M, Lima, C., van Rees, R., Fleuren, J. and Stephens, J. eConstruct: Expectations, Solutions and Results. ITCOn. Vol. 6, pp. 175 – 198. 2001.