aecXML Preliminary Specification

Working draft 0.81
Aug-99

aecXML.org

Send feedback on this specification to: feedback@aecxml.org

1 Scope

aecXML is an XML-based language used to represent information in the Architecture, Engineering and Construction (AEC) industry. This information may be resources such as projects, documents, materials, parts, organizations, professionals, or activities such as proposals, design, estimating, scheduling and construction. It is intended to be used as an XML namespace and to facilitate information exchange of AEC data on the Internet.

The following types of AEC information are within the scope of aecXML. This is meant to be a representative rather than an exhaustive list.

Documents
- RFP, RFQ, RFI, drawings, specifications, ASI, addenda, bulletins, change orders, contracts, building codes, purchase orders

Building Components
- Items from a catalog, custom manufactured items, assemblies, materials

Projects
- Design, construction, decommissioning, operations & maintenance, facility management

Professional Services and Resources
- Engineers, architects, contractors, suppliers, specialties

Organizations
- Standards bodies, government agencies

Software
- CAD, estimating, project management, scheduling, document management

2 Use Cases

Use cases are examples of problems we hope to address with aecXML. This is not a comprehensive list; rather it is a general indication of the scope of aecXML. These use cases will be further developed, containing sample code. The evolution of aecXML should be benchmarked against these use cases to confirm that its refinement enables these problems to be solved.

Scenario 1:

An architect is designing an office building and needs to search for availability of window systems that are:

- Aluminum
- Kynar Finish
- Meet ANSI z97.1
- Meet ASTM E283
- Meet ASTM E331
- Have a U value of 0.24 per ASTM C236
- Available as fixed, hopper and casement units
- Self flashing
- Interior glazed

Scenario 2:

An architect wants to determine the performance characteristics of a catalog item that satisfies the query described above.

Scenario 3:

A contractor wants to find out the availability of roof shingles equivalent to a “brand name” 25-year warranty variety, which was recommended in the architects design specifications.

Scenario 4:

A contractor would like to obtain price quotes for procurement of 50 squares of “brand name” 25-year warranty roof shingles for delivery by 12/12/1999.

Scenario 5:

A design tool would like to integrate with an estimating tool
See Appendix B for details

Scenario 6:

A design tool would like to integrate with a scheduling tool
See Appendix B for details

Scenario 7:

An owner would like to find out which architectural firms have done business in New Jersey on hospitals with construction costs greater than 25 million dollars.

Scenario 8:

An HVAC representative would like to query the design documents to get a list of air-handling units and their design parameters which he would like to associate to specific catalog items from his product line that satisfy the design specifications.

Scenario 9:

A window manufacturer would like to query the design documents to get a list of window assemblies and their design parameters, which he would like to associate to specific catalog items from his product line that meet the relevant design specifications.

Scenario 10:

To prepare a bid document, a contractor would like a product manufacturer to send him the estimation data the manufacturer obtained in a “take-off”, of their building components from the design documents.

Scenario 11:

A design-build firm would like to transfer information from its CAD drawings to a scheduling software package for construction management.

Scenario 12:

A mechanical engineer would like to query the Web to find out about CAD modeling software that integrates with duct sizing tools.

Scenario 13:

A building owner would like to query the Web to find out about software available for Facility Management

Scenario 14:

A contractor would like to extract quantities from an estimating tool and dispatch a procurement request to suppliers over the Web.

Scenario 15:

An employee time-keeping system (i.e. Simplex punch-clock type tracking) would like to exchange information with a scheduling application, to keep track of actual hours worked per task on a project.

Scenario 16:

Exchange of information from a PDA device at a construction site, such as an item being added to a punch list, to an order processing system, or a scheduling system.

Scenario 17:

> A HVAC engineer may wish to query manufacturers catalogs for an outdoor air-handling unit which will supply a total of 20,000 cfm with a constant volume horizontal draw-thru fan, that has an external static pressure of 2.0 inches wg, with the summer design coil air temp. of 80F DB/67F WB with two-inch TA filters and factory mounted controls.

3 Concepts

One of the fundamental design goals of aecXML is to avoid undertaking the nearly impossible task of exhaustively classifying data. Of course, this cannot be completely avoided. There are many elements in aecXML such as <Document> that will over time, get further structured into subcategories, with associated elements. Market pressures should drive these efforts, incrementally. What is presented in this document is an initial, usable framework.

We have taken a building-block approach to the design of aecXML. By creating simple element definitions that are the basis for the language, we can construct higher-level elements. In addition, via the usage of XLinks and XPointers, (which are features of XML), we envision being able to represent the rich and complex universe of AEC data. For example, a Project can contain Scheduling tasks that contain Building Components, with links to Specification Documents. The design goals of aecXML have also been aligned with that of XML. It is useful to reiterate them from the XML specification. See http://www.w3.org/xml for more details.

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML should be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

Elaborating on a few:

> (2) We want to facilitate the flow of information across applications that cover the entire life cycle of an AEC asset, from design to facilities management.

(7) The initial specification of aecXML provides a framework from which various working groups with specific areas of interest can develop the language further.

(8) The AEC Industry spans many groups with vastly varying degrees of sophistication with respect to Information Technology. It is vital that aecXML be simple to adopt and be adapted to existing applications easily.

Adding a couple more:

11. aecXML is neutral to multiple classification schemes.
    The strategy adopted by aecXML is that there can be more than one classification standard by which data can be organized and queries on them made. The language should be flexible enough to facilitate that, as well as easily allow for the creation of new classification standards. It provides an element <Classification>, to provide this mechanism.
12. aecXML will inter-operate with other XML schemas.
    E-commerce schemas for Purchase Orders, STEP related XML schemas, Accounting and Financial schemas, will be relevant in an AEC context and may be used in conjunction with aecXML.

There are various tools used by AEC professionals over the lifecycle of a project. These tools come from various vendors. aecXML is intended to be the transport mechanism by which information can flow seamlessly between them. Each aecXML element can contain specific private information that can be appended to the data in order to maintain a connection back to the application. For example both a CAD and an estimating software application may want to exchange <BuildingComponent> data. They can both add information specific to linking the aecXML element data to objects in their software in order to be able to track and update information on it.

AEC industry professionals, especially in the areas of design and construction, usually use classifications by which they organize their data and activities. In the United States, MasterFormat ™ and UNIFORMAT are such standards. There are various international classification standards as well. There are tools, such as estimating software, that have incorporated their workflow around these classification categories. A product manufacturer may choose to organize its catalogs around such classifications as well. When it comes to data definitions, however, a more flexible approach is to allow the data to contain multiple classification entries. These classifications can be industry standards, submitted to the aecXML organization, or private classifications generated by content providers. Tools that work with aecXML data should be able to work with either

classification types. In cases where the classifications are approved standards, with namespaces allocated by the aecXML organization, there will be unambiguous categories available for a query. In the case of private classification schemes, there may be a <Request> <Response> dialog where the valid classification categories are obtained first, and then queries constructed.

There is a communication framework in aecXML that contains two models of communication. There is a two-way <Request>-<Response> mechanism, that involves data exchange in which <Request> has a unique ID associated with it, and the <Response> contains the ID of the corresponding <Request> in the <InResponseTo> element. The <Message> framework is designed for one-way broadcasts of data, with a <Header> and <Body> that are described in the Elements section of this document

Queries in aecXML will be structured via a communication framework. The actual syntax of these queries is not currently addressed. There are currently multiple query language efforts such as XML-QL and XQL under review by the W3C. The query can be encapsulated into the communication framework of aecXML.

aecXML is intended to be an open schema model, where user-defined elements can be added to data. This will provide flexibility when two parties want to carry on a private conversation but want to use the aecXML framework to exchange their data.

4 Building Components

A <BuildingComponent> represents a component in a building. These components have an identity that can be represented by <ApplicationData>, <Name> and <Description>. They also have information about them that can be classified in terms of the life cycle of the component. Applications can exchange <BuildingComponent> information, populating or making queries about the specific portions of this life cycle that are of interest to them.

For example, an architect may submit a query populating the design information about a Door, requesting <DesignationData> that satisfies the design parameters. He may then populate the <DesignationData> information when submitting the data to an estimating system. A scheduling system can enter specific scheduling task information about the component in the <ScheduleData> category, and transfer that information to a construction management database, which can then query the database as to when certain items need to arrive at a site.

An important property of <BuildingComponent> is that it can be used to represent a hierarchy of building components. This is achieved via the

<DesignationData><Assembly> element that is recursively defined to be able to contain <BuildingComponent> items. This allows for assemblies to contain sub-assemblies, which contain discrete items or custom items. Each nested <BuildingComponent> can contain individual information for estimation and scheduling, or the <Assembly> can carry this information as well.

<DesignData>
This element is meant to contain information that is used about the component during its design. There can be further namespace elements used here such as <DoorDesignData> or <DuctDesignData> that can be used to represent specific design information about the component. For instance <DoorDesignData> can contain <Dimension> information as well as <Material> information.

<DesignationData>

A <DiscreteItem> is a <BuildingComponent> that can be obtained from a catalog. Doors, windows, air-handling units, furniture, light fixtures, sprinklers are examples of such items. The <DiscreteItem> will contain adequate information to permit online procurement requests to be made for it. It will contain <PerformanceData> to enable queries for equivalent items to be made.

A <CustomItem> is a <BuildingComponent> that needs to be manufactured from certain specifications. Clarifiers, mill work and cast stone work are examples of such items.

An <Assembly> is an aggregation of <BuildingComponent>. This can be a cavity wall, sprinkler system or any aggregate view of components.

<Material> like concrete, wood or paint has properties that are of interest.

<EstimationData>
Information about cost and quantities for the component is organized here.

<ScheduleData>
Scheduling tasks that the component participates in, along with task information such as <BeginDate> are represented here.

<ConstructionData>
This element is meant for construction status information about the component.

<MaintenanceData>
Maintenance information about the component, such as next scheduled maintenance date, is represented here.

There are at least two ways that <BuildingComponent> can evolve. There can be further namespace hierarchies like <Concrete> and <Wood> created under <Material>. Or, since <Material> contains <Classification>, there can be classification-based salient attribute queries done on <Material> elements, obviating the need for further sub-dividing the <Material> namespace. These are not mutually exclusive evolutions and what may occur is probably a combination of both approaches.

5 Documents

** elaborate **

6 Projects

** elaborate **

7 Elements

This is the glossary of elements used in aecXML

7.1 <aecXML>

This is the outermost element in the aecXML document, or the namespace for aecXML elements.

7.2 <Acronym>

This element can be used in many aecXML elements to provide a shorthand string by which to refer to the item.

7.3 <AdditionalInformation >

This element is provided as a vehicle by which any data element can encapsulate additional information about it, which is unstructured, or there are no appropriate elements defined for yet.

7.4 <Address>

The address element will define the standard for address information. It will reflect the ISO international standard ** elaborate **

7.5 <ApplicationData>

This element provides the framework by which applications can insert links such as ID’s and project names about the data being described. It is designed to allow data to exchanged between applications, and maintain links back to the objects in the application.

7.6 <AreaCode>

7.7 <Assembly>

This is an element for aggregation of <BuildingComponent> items. Since this is defined in a recursive manner, it allows for assemblies to contain subassemblies.

7.8 <AssemblyCost>

7.9 <AssemblyPrice>

7.10 <Attribute>

7.11 <BallInCourt>

This item in a document represents the current entity operating on the document.

7.12 <BeginDate>

This element specifies the start date of an activity.

7.13 <BuildingComponent>

This element is one of the key components of aecXML. It is a flexible structure meant to represent elements in a facility. One particular hierarchy proposed reflects the life cycle of a building component.

A component goes from a design to specification, then estimating, scheduling, construction/installation, maintenance and decommissioning. The structure of building components should enable the goal of information flow between applications that span this life cycle. These elements may have links into relevant portions of specification documents, or a project they belong to via the use of XPointers.

This component is described in detail in Section 5 of this document.

See <DesignData>, <EstimationData>, <ScheduleData>, <ConstructionData>, <MaintenanceData>, <DesignationData>.

7.14 <Charter>

This element defines the mission of an <Organization>. It is a free format character string.

7.15 <City>

7.16 <Classification>

This is an element meant to provide data with the ability to classify itself. Data can belong to multiple classifications. The attributes of classification are domain and value, which are both strings.
e.g. <Classification domain=”MasterFormat ™”, value=”08340”></Classification> classifies the item being described.

7.17 <Color>

7.18 <Communication>

This element provides the framework under which <Request> <Response>, and <Message> data is described.

7.19 <Company>

7.20 <CompletionDate>

7.21 <ConstructionData>

7.22 <ContactInformation>

7.23 <ContactPerson>

7.24 <Country>

7.25 <CountryCode>

7.26 <CurrentPhase>

7.27 <CustomInformation>

7.28 <CustomItem>

This is a <BuildingComponent> that has to be made-to-order from some specifications. It contains the elements in <DiscreteItem> along with <CustomInformation> regarding specifications for its manufacturing or assembly.

7.29 <Density>

7.30 <Description>

7.31 <DesignationData>

There is an identity hierarchy under <BuildingComponent>. Its sub-elements are <DiscreteItem>, <CustomItem>, <Assembly>, and <Material>.

A door can be thought of as a <DiscreteItem> if it is to be purchased through the process of selection from a catalog. It can be thought of as an <Assembly> in the design process in terms of its frame, door and hardware set.

A cavity wall can be considered an <Assembly> consisting of <DiscreteItem> such as metal studs, <CustomItem> such as lintels and a sub <Assembly> consisting of a CMU wythe, and <Material> such as paint.

7.32 <DesignData>

This hierarchy under the element of <BuildingComponent> is for element design parameters. The namespace for this will be partitioned over time as the schema is developed. The intended usage for this namespace is to exchange appropriate design information about building elements. For example, an architect may want to research possible door configurations with a Web-based tool. The <DoorDesignData> namespace will contain the design parameters for the door such as <Dimension><SizeOfOpening> and <Material>, and the query may result in a set of <DiscreteItem> that satisfy the request. This is an open schema at this level. The benefits are that two parties can agree to use private keywords between them and exchange data.

7.33 <DiscreteItem>

This is an element under <DesignationData> to identify an element that can be bought from a catalog. It can contain the following elements <Name>, <ManufacturerName>, <ManufacturerID>, multiple <Classification> entries, <PeformanceData>, <UnitCost>, <Attribute>, <Keyword>, and <AdditionalInformation>

7.34 <Document>

This category contains the structure for documents interchanged in the AEC industry. There is a vast number of document and document sets that are exchanged in the AEC Industry. A non-exhaustive list is RFP, RFQ, RFI, drawings, specifications, ASI, addenda, bulletins, change orders, contracts, building codes, purchase orders, material safety data sheets and insurance certificates.

It is envisioned that there will be logical hierarchies defined to allow for intuitive navigation to certain document types.

7.35 <DocumentSet>

7.36 <DocumentState>

7.37 <DocumentType>

7.38 <Email>

7.39 <Equipment>

7.40 <Estimate>

7.41 <EstimationData>

7.42 <Extension>

7.43 <Expertise>

7.44 <Fax>

7.45 <Finish>

7.46 <FirstName>

7.47 <From>

7.48 <Header>

7.49 <HighPrice>

7.50 <InstallationPrice>

7.51 <Keyword>

This element can be used for various types of data. These are words that can be chosen to be able to select the data for query purposes.

7.52 <LastName>

7.53 <Location>

7.54 <LowPrice>

7.55 <MaintenanceData>

7.56 <ManufacturerName>

7.57 <ManufacturerNumber>

7.58 <Material>

This is a <BuildingComponent> that has properties and a classification.

7.59 <MaterialPrice>

7.60 <MedianPrice>

7.61 <Message>

This protocol is designed for one-way transfer of data.

7.62 <MiddleName>

7.63 <Name>

7.64 <Nickname>

7.65 <Organization>

Professional Organizations, Standards bodies and Government Agencies can structure their data with this element.

7.66 <OrganizationType>

(Non-Profit, Profit, Government, Professional)

7.67 <Owner>

7.68 <Participant>

7.69 <PercentComplete>

7.70 <PerformanceData>

7.71 <PerformanceStandard>

7.72 <Person>

7.73 <Phone>

7.74 <PhoneNumber>

7.75 <PostalAddress>

7.76 <PostalCode>

7.77 <Prefix>

7.78 <Price>

7.79 <Product>

7.80 <ProductName>

7.81 <ProfessionalServices>

Engineers, Architects, Contractors, Suppliers, Specialties and Resellers are elements proposed for further refinement.

7.82 <Project>

Design, Construction, Decommissioning, Operations & Maintenance and Facility Management can be hierarchy under which further structure can be imposed on AEC project data.

** elaborate here **

7.83 <ProjectNumber>

7.84 <Quantity>

7.85 <Reflectivity>

7.86 <Region>

7.87 <Request>

Request-Response is a two-way communication between applications. It could be between a Web search tool and the Internet, or between two applications. The <Request-Response> framework has <Request>, which contains a globally unique ID as an identifier. <Response> contains a <InResponseTo>, which returns the same ID.
The <Request> keyword can further embed an XML query specified in a query protocol that is supported by the W3C for XML. Currently the two that seem to be garnering the most interest are XML-QL and XQL.

7.88 <Resource>

7.89 <Response>

Request-Response is a two-way communication between applications. It could be between a web search tool and the Internet, or between two applications. The <Request-Response> framework has <Request>, which contains a globally unique ID as an identifier.<Response> contains a <InResponseTo>, which returns the same ID
The <Request> keyword can further embed an XML query specified in a query protocol that is supported by the W3C for XML. Currently the two that seem to be garnering the most interest are XML-QL and XQL.

7.90 <ScheduleData>

A building component may belong to a certain task that is being scheduled. There may be start and end dates to activities in this element.
*** elaborate here ***

7.91 <Size>

7.92 <Software>

Design, Estimation, Scheduling, Document Management, Facilities Management, MRO (Maintenance, Repair and Operations) can be one hierarchy for searches for AEC software.

7.93 <State>

7.94 <Task>

7.95 <Texture>

7.96 <Title>

7.97 <To>

7.98 <TotalPrice>

7.99 <TradeName>

7.100 <Type>

7.101 <UnitCost>

7.102 <UnitOfMeasurement>

7.103 <UnitPrice>

7.104 <Url>

7.105 <Vendor>

7.106 <Version>

7.107 <Warranty>

8 Conclusions

This aecXML specification is intended to be the initial proposal of XML grammar for Architecture, Engineering and Construction data. It is intentionally broad in scope, to enable working groups from various industry segments to develop portions of the language that relate to their areas of interest and expertise. XML is rapidly evolving as the standard for electronic communication. It is hoped that aecXML will become the lingua franca for AEC conversations on the Internet. The industry needs to actively participate, rapidly adopt, and incrementally refine aecXML in order to achieve this goal.

Appendix A: aecXML schema

```
<?xml version ="1.0"?>
<!--Generated by XML Authority. Conforms to XML Data subset for IE 5-->
<Schema name = "aecxml_prelim_v_0081.xml"
       xmlns = "urn:schemas-microsoft-com:xml-data"
       xmlns:dt = "urn:schemas-microsoft-com:datatypes">
      <ElementType name = "aecXML" content = "eltOnly" order = "many"
minOccurs = "0" maxOccurs = "*">
            <AttributeType name = "version" dt:type = "string" default
= "00.81" required = "yes"/>
            <AttributeType name = "timestamp" dt:type = "dateTime"
required = "yes"/>
            <AttributeType name = "crc" dt:type = "int"/>
            <AttributeType name = "aecXMLId" dt:type = "int" required =
"yes"/>
            <AttributeType name = "language" dt:type = "string"
required = "yes"/>
            <attribute type = "version"/>
            <attribute type = "timestamp"/>
            <attribute type = "crc"/>
            <attribute type = "aecXMLId"/>
            <attribute type = "language"/>
            <element type = "Project"/>
            <element type = "DocumentSet"/>
            <element type = "BuildingComponent"/>
            <element type = "ProfessionalService"/>
            <element type = "Organization"/>
            <element type = "Software"/>
            <element type = "Communication"/>
            <element type = "Task"/>
            <element type = "Equipment"/>
            <element type = "Resource"/>
      </ElementType>
      <ElementType name = "Project" content = "eltOnly" order = "seq">
            <AttributeType name = "projectId" dt:type = "int" required
= "yes"/>
            <attribute type = "projectId"/>
            <element type = "Name"/>
            <element type = "Description" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "ProjectNumber" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "Owner" minOccurs = "0" maxOccurs = "1"/>
            <element type = "Participant" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "Type" minOccurs = "0" maxOccurs = "1"/>
            <element type = "Size" minOccurs = "0" maxOccurs = "1"/>
            <element type = "Location" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "CurrentPhase" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "PercentComplete" minOccurs = "0" maxOccurs
= "1"/>
```

```
            <element type = "BeginDate" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "CompletionDate" minOccurs = "0" maxOccurs
= "1"/>
            <element type = "ApplicationData" minOccurs = "0" maxOccurs
= "1"/>
            <element type = "Document" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "BuildingComponent" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "Project" minOccurs = "0" maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "ProjectNumber" content = "textOnly" dt:type
= "string"/>
      <ElementType name = "Owner" content = "eltOnly" order = "seq">
            <group order = "many" minOccurs = "0" maxOccurs = "*">
                  <element type = "Person"/>
                  <element type = "Company"/>
            </group>
      </ElementType>
      <ElementType name = "Type" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Size" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Location" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "PercentComplete" content = "textOnly"
dt:type = "number"/>
      <ElementType name = "BeginDate" content = "textOnly" dt:type =
"date"/>
      <ElementType name = "CompletionDate" content = "textOnly" dt:type
= "date"/>
      <ElementType name = "CurrentPhase" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "ApplicationData" content = "mixed" model =
"open"/>
      <ElementType name = "BuildingComponent" content = "eltOnly" order
= "seq">
            <AttributeType name = "componentId" dt:type = "int"/>
            <attribute type = "componentId"/>
            <element type = "Name"/>
            <element type = "Description" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "DesignationData" minOccurs = "0" maxOccurs
= "*"/>
            <element type = "DesignData" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "EstimationData" minOccurs = "0" maxOccurs
= "*"/>
            <element type = "ScheduleData" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "ConstructionData" minOccurs = "0"
maxOccurs = "*"/>
```

```
            <element type = "MaintenanceData" minOccurs = "0" maxOccurs
= "*"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "ApplicationData"/>
      </ElementType>
      <ElementType name = "DocumentSet" content = "eltOnly" order =
"seq">
            <AttributeType name = "documentSetId" dt:type = "int"/>
            <attribute type = "documentSetId"/>
            <element type = "Name"/>
            <element type = "Description"/>
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "Document"/>
            </group>
      </ElementType>
      <ElementType name = "Document" content = "eltOnly" order = "seq">
            <AttributeType name = "documentId" dt:type = "int"/>
            <attribute type = "documentId"/>
            <element type = "DocumentType"/>
            <element type = "DocumentState"/>
            <element type = "BallInCourt"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "ApplicationData" minOccurs = "0" maxOccurs
= "*"/>
      </ElementType>
      <ElementType name = "DocumentType" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "DocumentState" content = "textOnly" dt:type
= "string"/>
      <ElementType name = "BallInCourt" content = "eltOnly" order =
"one">
            <element type = "Person"/>
            <element type = "Company"/>
      </ElementType>
      <ElementType name = "DiscreteItem" content = "eltOnly" order =
"seq">
            <AttributeType name = "discreteItemId" dt:type = "int"/>
            <attribute type = "discreteItemId"/>
            <element type = "ManufacturerName" minOccurs = "0"
maxOccurs = "1"/>
            <element type = "ManufacturerNumber" minOccurs = "0"
maxOccurs = "1"/>
            <element type = "TradeName" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "ProductName" minOccurs = "0" maxOccurs =
"1"/>
            <element type = "Classification" minOccurs = "0" maxOccurs
= "1"/>
            <element type = "Keyword" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Attribute" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "PerformanceData" minOccurs = "0" maxOccurs
= "1"/>
            <element type = "Warranty"/>
```

```
            <element type = "Quantity"/>
            <element type = "Price" minOccurs = "0" maxOccurs = "1"/>
      </ElementType>
      <ElementType name = "CustomItem" content = "eltOnly" order =
"one">
            <AttributeType name = "customItemId" dt:type = "int"/>
            <attribute type = "customItemId"/>
            <element type = "DiscreteItem"/>
            <element type = "CustomInformation" minOccurs = "0"
maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "Assembly" content = "eltOnly" order = "seq">
            <AttributeType name = "assemblyId" dt:type = "int"/>
            <attribute type = "assemblyId"/>
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "BuildingComponent"/>
            </group>
      </ElementType>
      <ElementType name = "ManufacturerName" content = "textOnly"
dt:type = "string">
            <AttributeType name = "manufacturerId" dt:type = "int"/>
            <attribute type = "manufacturerId"/>
      </ElementType>
      <ElementType name = "ProductName" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "TradeName" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Classification" content = "textOnly" dt:type
= "string">
            <AttributeType name = "classType" dt:type = "string"
required = "yes"/>
            <attribute type = "classType"/>
      </ElementType>
      <ElementType name = "Warranty" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Material" content = "eltOnly" order = "seq">
            <AttributeType name = "materialId" dt:type = "int"/>
            <attribute type = "materialId"/>
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "Classification"/>
                  <element type = "Density"/>
                  <element type = "Finish"/>
                  <element type = "Reflectivity"/>
                  <element type = "Color"/>
                  <element type = "Texture"/>
            </group>
      </ElementType>
      <ElementType name = "Price" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "Density" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Finish" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Reflectivity" content = "textOnly" dt:type =
"string"/>
```

```
      <ElementType name = "Color" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Texture" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "CustomInformation" content = "mixed" model =
"open"/>
      <ElementType name = "DesignationData" content = "eltOnly" order =
"one">
            <AttributeType name = "designationId" dt:type = "int"/>
            <attribute type = "designationId"/>
            <element type = "DiscreteItem"/>
            <element type = "CustomItem"/>
            <element type = "Assembly"/>
            <element type = "Material"/>
      </ElementType>
      <ElementType name = "DesignData" content = "mixed" model =
"open">
            <AttributeType name = "designId" dt:type = "int"/>
            <attribute type = "designId"/>
      </ElementType>
      <ElementType name = "EstimationData" content = "eltOnly" order =
"seq">
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "Estimate"/>
            </group>
      </ElementType>
      <ElementType name = "Estimate" content = "eltOnly" order = "one">
            <AttributeType name = "costId" dt:type = "int"/>
            <attribute type = "costId"/>
            <element type = "UnitCost"/>
            <element type = "AssemblyCost"/>
      </ElementType>
      <ElementType name = "UnitCost" content = "eltOnly" order = "seq">
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "Classification"/>
                  <element type = "Description"/>
                  <element type = "Quantity"/>
                  <element type = "UnitOfMeasurement"/>
                  <element type = "UnitPrice"/>
                  <element type = "AdditionalInformation"/>
            </group>
      </ElementType>
      <ElementType name = "AssemblyCost" content = "eltOnly" order =
"seq">
            <group order = "seq" minOccurs = "0" maxOccurs = "*">
                  <element type = "Classification"/>
                  <element type = "Description"/>
                  <element type = "Quantity"/>
                  <element type = "UnitOfMeasurement"/>
                  <element type = "AssemblyPrice"/>
                  <element type = "AdditionalInformation"/>
            </group>
      </ElementType>
      <ElementType name = "UnitOfMeasurement" content = "textOnly"
dt:type = "string"/>
```

```
      <ElementType name = "UnitPrice" content = "eltOnly" order =
"seq">
            <AttributeType name = "wageType" dt:type = "string"/>
            <AttributeType name = "zipCodePrefix" dt:type = "string"/>
            <attribute type = "wageType"/>
            <attribute type = "zipCodePrefix"/>
            <element type = "LowPrice"/>
            <element type = "MedianPrice"/>
            <element type = "HighPrice"/>
      </ElementType>
      <ElementType name = "AdditionalInformation" content = "mixed"
model = "open"/>
      <ElementType name = "LowPrice" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "MedianPrice" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "HighPrice" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "AssemblyPrice" content = "eltOnly" order =
"seq">
            <AttributeType name = "wageType" dt:type = "string"/>
            <AttributeType name = "zipCodePrefix" dt:type = "string"/>
            <attribute type = "wageType"/>
            <attribute type = "zipCodePrefix"/>
            <element type = "MaterialPrice"/>
            <element type = "InstallationPrice"/>
            <element type = "TotalPrice"/>
      </ElementType>
      <ElementType name = "MaterialPrice" content = "textOnly" dt:type
= "number"/>
      <ElementType name = "InstallationPrice" content = "textOnly"
dt:type = "number"/>
      <ElementType name = "TotalPrice" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "ScheduleData" content = "mixed" model =
"open">
            <AttributeType name = "schedulingId" dt:type = "int"/>
            <attribute type = "schedulingId"/>
      </ElementType>
      <ElementType name = "ConstructionData" content = "mixed" model =
"open">
            <AttributeType name = "constructionId" dt:type = "int"/>
            <attribute type = "constructionId"/>
      </ElementType>
      <ElementType name = "MaintenanceData" content = "mixed" model =
"open">
            <AttributeType name = "maintenanceId" dt:type = "int"/>
            <attribute type = "maintenanceId"/>
      </ElementType>
      <ElementType name = "ManufacturerNumber" content = "textOnly"
dt:type = "string"/>
      <ElementType name = "PerformanceData" content = "eltOnly" order =
"seq">
            <AttributeType name = "organizationStandard" dt:type =
"string"/>
            <attribute type = "organizationStandard"/>
```

```
            <group order = "many" minOccurs = "0" maxOccurs = "*">
                  <element type = "Description"/>
                  <element type = "PerformanceStandard"/>
            </group>
      </ElementType>
      <ElementType name = "PerformanceStandard" content = "eltOnly"
order = "seq">
            <AttributeType name = "organizationAcronym" dt:type =
"string" required = "yes"/>
            <AttributeType name = "documentName" dt:type = "string"
required = "yes"/>
            <attribute type = "organizationAcronym"/>
            <attribute type = "documentName"/>
      </ElementType>
      <ElementType name = "ProfessionalService" content = "eltOnly"
order = "seq">
            <AttributeType name = "professionalServiceId" dt:type =
"int"/>
            <attribute type = "professionalServiceId"/>
            <element type = "Participant" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "Expertise"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "Project" minOccurs = "0" maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "Organization" content = "eltOnly" order =
"seq">
            <AttributeType name = "organizationId" dt:type = "int"/>
            <attribute type = "organizationId"/>
            <element type = "Name"/>
            <element type = "ContactInformation"/>
            <element type = "Description"/>
            <element type = "OrganizationType" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "Charter"/>
            <element type = "Expertise" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "Participant" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "Acronym" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Keyword" minOccurs = "0" maxOccurs = "*"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "OrganizationType" content = "textOnly"
dt:type = "string"/>
      <ElementType name = "Charter" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Expertise" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Software" content = "eltOnly" order = "seq">
            <AttributeType name = "softwareId" dt:type = "int"/>
            <attribute type = "softwareId"/>
            <element type = "Vendor"/>
            <element type = "Product"/>
```

```
            <element type = "Version"/>
            <element type = "Description"/>
            <element type = "Keyword" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Attribute" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "AdditionalInformation" minOccurs = "0"
maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "Vendor" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Product" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Version" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Communication" content = "eltOnly" order =
"seq">
            <AttributeType name = "communicationId" dt:type = "int"
required = "yes"/>
            <attribute type = "communicationId"/>
            <element type = "Header"/>
            <group order = "many" minOccurs = "0" maxOccurs = "*">
                  <element type = "Request"/>
                  <element type = "Response"/>
                  <element type = "Message"/>
            </group>
      </ElementType>
      <ElementType name = "Header" content = "eltOnly" order = "seq">
            <element type = "To"/>
            <element type = "From"/>
      </ElementType>
      <ElementType name = "To" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "From" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Request" content = "eltOnly" order = "seq">
            <AttributeType name = "requestId" dt:type = "int" required
= "yes"/>
            <attribute type = "requestId"/>
      </ElementType>
      <ElementType name = "Response" content = "eltOnly" order = "seq">
            <AttributeType name = "responseId" dt:type = "int" required
= "yes"/>
            <AttributeType name = "inResponseToRequestId" dt:type =
"int" required = "yes"/>
            <attribute type = "responseId"/>
            <attribute type = "inResponseToRequestId"/>
      </ElementType>
      <ElementType name = "Message" content = "eltOnly" order = "seq">
            <AttributeType name = "messageId" dt:type = "int" required
= "yes"/>
            <attribute type = "messageId"/>
      </ElementType>
      <ElementType name = "Task" content = "eltOnly" order = "seq"/>
      <ElementType name = "Equipment" content = "eltOnly" order =
"seq"/>
```

```
      <ElementType name = "Resource" content = "eltOnly" order =
"seq"/>
      <ElementType name = "Acronym" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Address" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "AreaCode" content = "textOnly" dt:type =
"int"/>
      <ElementType name = "Attribute" content = "textOnly" dt:type =
"string">
            <AttributeType name = "attributeName" dt:type = "string"
required = "yes"/>
            <attribute type = "attributeName"/>
      </ElementType>
      <ElementType name = "City" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Company" content = "eltOnly" order = "seq">
            <element type = "Name"/>
            <element type = "ContactInformation" minOccurs = "0"
maxOccurs = "*"/>
            <element type = "ContactPerson"/>
            <element type = "Acronym" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Keyword" minOccurs = "0" maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "ContactInformation" content = "eltOnly"
order = "seq">
            <element type = "PostalAddress" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "Phone" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Fax" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Email" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Url" minOccurs = "0" maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "ContactPerson" content = "eltOnly" order =
"seq">
            <element type = "Person" minOccurs = "0" maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "Country" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "CountryCode" content = "textOnly" dt:type =
"int"/>
      <ElementType name = "Description" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Email" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Extension" content = "textOnly" dt:type =
"int"/>
      <ElementType name = "Fax" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "FirstName" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Keyword" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "LastName" content = "textOnly" dt:type =
"string"/>
```

```
      <ElementType name = "MiddleName" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Name" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Nickname" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Participant" content = "eltOnly" order =
"one">
            <AttributeType name = "role" dt:type = "string" default =
""/>
            <attribute type = "role"/>
            <element type = "Company"/>
            <element type = "Person"/>
      </ElementType>
      <ElementType name = "Person" content = "eltOnly" order = "seq">
            <element type = "Prefix"/>
            <element type = "FirstName"/>
            <element type = "MiddleName"/>
            <element type = "LastName"/>
            <element type = "Title" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Nickname" minOccurs = "0" maxOccurs =
"*"/>
            <element type = "ContactInformation" minOccurs = "0"
maxOccurs = "*"/>
      </ElementType>
      <ElementType name = "Phone" content = "eltOnly" order = "seq">
            <AttributeType name = "name" dt:type = "string" required =
"yes"/>
            <attribute type = "name"/>
            <element type = "CountryCode"/>
            <element type = "AreaCode"/>
            <element type = "PhoneNumber"/>
            <element type = "Extension"/>
      </ElementType>
      <ElementType name = "PhoneNumber" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "PostalAddress" content = "eltOnly" order =
"seq">
            <element type = "Address" minOccurs = "0" maxOccurs = "*"/>
            <element type = "Region"/>
            <element type = "City"/>
            <element type = "State"/>
            <element type = "Country"/>
            <element type = "PostalCode"/>
      </ElementType>
      <ElementType name = "PostalCode" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Prefix" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Quantity" content = "textOnly" dt:type =
"number"/>
      <ElementType name = "Region" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "State" content = "textOnly" dt:type =
"string"/>
```

```
      <ElementType name = "Title" content = "textOnly" dt:type =
"string"/>
      <ElementType name = "Url" content = "textOnly" dt:type = "uri"/>
</Schema>
```

Appendix B: Sample aecXML scenario

Situation

A contractor has scheduled a project. They would like to begin planning and budgeting for material to be delivered to the construction site.

Currently, scheduling applications are capable of tracking non-labor resources. However, this data is not related to the design model, which contains the actual quantity calculations. To do this they will have to gather material and quantity data from the design model. Pricing data will be generated from an estimating application.

This data will be gathered and analyzed by the scheduling application, which will then generate precise resource requirements. This completed resource data combining schedules, costs and quantities can then be output to a bidding/estimating application. Purchase/Delivery requisitions will be generated by a contract management application.

Task Execution Detail

All examples of data exchange in this example are envisioned as being performed by aecXML.

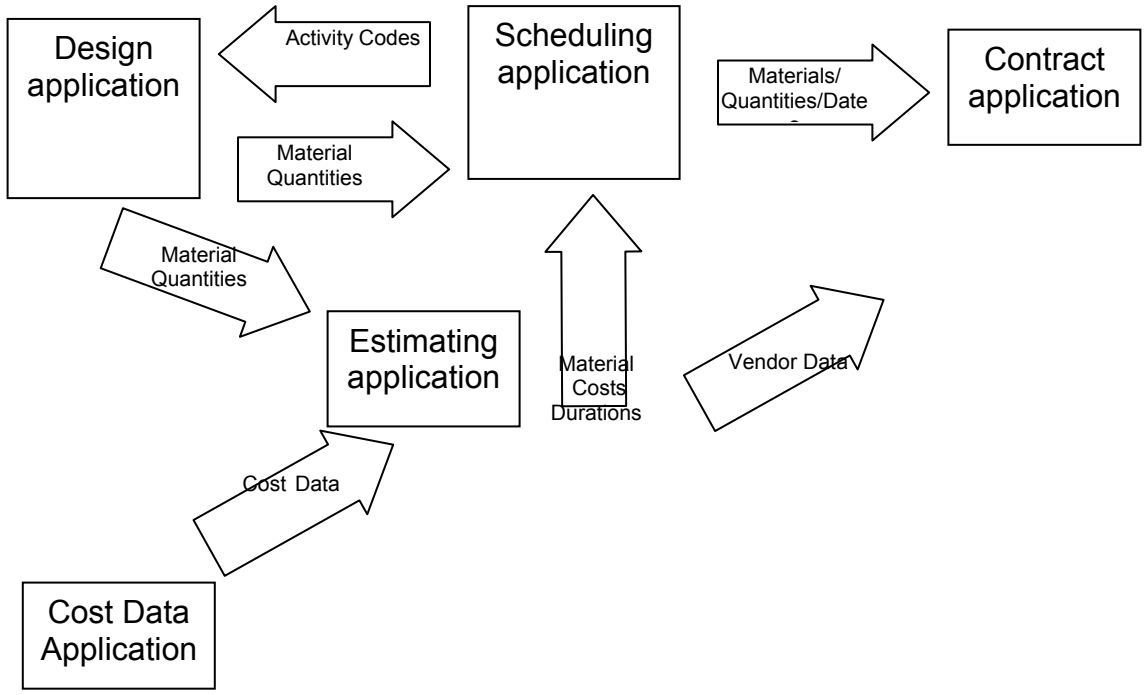
A project schedule is generated containing an activity schedule. Each activity item contains start/duration dates and resource requirements. Resource requirements are typically estimated and manually entered for the purposes of scheduling. To begin this scenario, the scheduling application would output a list of activities to the design application.

The list of activities is imported into the design application. As the design is created, elements are associated to an activity. The design application would calculate material quantities based upon actual design criteria. Currently, this data might only be exported as a spreadsheet in gross quantity totals for the purpose of project cost estimates. The tagging process would allow greater usage of the data.

The tagged design data is exported to the scheduling and the cost estimating applications. The cost estimating applications provide cost extensions, which are

then re-exported to the scheduling application. This data is combined together to provide an activity by activity view of the data.

The scheduling application is now able to generate a precise schedule including the dates that different materials, including quantities are required on the job site. From this information, data can be passed to a contract management application to generate purchase and delivery requisitions.



Sample Transactions:
*** elaborate ****