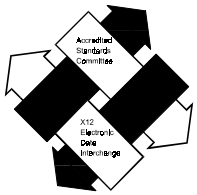


ASC X12C/TG3/2002-xxx

ANSI ASC X12C Communications and Controls Subcommittee
Technical Report Type II

ASC X12 REFERENCE MODEL FOR XML DESIGN

July 2002

ABSTRACT

This Reference Model was motivated by the action item that X12's Communications and Controls Subcommittee (X12C) took at the August 2001 XML Summit to develop "draft design rules for ASC X12 XML Business Document development." Acting on that action item, X12C's EDI Architecture Task Group (X12C/TG3) determined that XML design rules could not be developed without a basis for determining which XML features to use and how to use them. Thus the group also set about developing a philosophical foundation and putting forth some general design principles. This Reference Model covers those topics in addition to a preliminary set of design rules.

The approach discussed herein is a work in progress. It is intended to be the foundation for X12's future XML development, and will become the basis for XML equivalents to the X12 syntax based X12.6 and X12.59, and XML Design Rules. It is consistent with the decisions of X12's Steering Committee to develop its XML work within the ebXML framework.

.

TABLE OF CONTENTS

PREFACE

GENERAL

This document is a Technical Report Type II, commonly referred to as a Reference Model. It was developed by X12C, the Communications and Controls subcommittee.

This technical report was prepared under the guidance of the Accredited Standards Committee (ASC) on Electronic Data Interchange, X12. Organized under the procedures of the American National Standards Institute, ANSI ASC X12 was charged with the development of transactions and structures for use in an Electronic Data Interchange (EDI) environment. The Secretariat is the Data Interchange Standards Association, Inc. ANSI ASC X12 has the following subcommittees:

- ASC X12A Education Administration
- ASC X12C Communications and Controls
- ASC X12F Finance
- ASC X12G Government
- ASC X12H Materials Management
- ASC X12I Transportation
- ASC X12J Technical Assessment
- ASC X12M Procurement/Distribution
- ASC X12N Insurance

In developing X12 technical reports, it is the aim of the X12 subcommittees to facilitate the use and understanding of the standards developed by X12. These technical reports present information that is not currently suitable for standards but that is intended to assist the users of the standards.

VERSION AND RELEASE

This Reference Model is neither based on nor dependent on any particular version of the ANSI ASC X12 Standards; it forms the foundation for X12's future work in XML.

This Reference Model represents the initial version of the architecture and related concepts. It is anticipated that these will evolve with future versions of this document or with standards that are based on it.

COMMENTS

Comments, questions, and suggestions for improvement of this document may be submitted in writing to the Secretariat who will forward them to the appropriate ANSI ASC X12 Subcommittee. ASC XI2 Standards are available for purchase from the ASC XI2 Secretariat.

Director, Publications & Standards

ANSI ASC X12 Secretariat
Data Interchange Standards Association
333 John Carlyle Street - Suite 600
Alexandria, Virginia 22314-2852
Phone: (703) 548-7005 FAX: (703) 548-5738
Publications Order Desk 1 (888) 363-2334
Email: ***publications@disa.org*** Internet: ***http://www.disa.org***

PURPOSE AND SCOPE

Introduction

The Extensible Markup Language (XML), developed by the World Wide Web Consortium (W3C), is a specification designed for Web documents that enables the definition, transmission, validation, and interpretation of data between applications and between organizations. It is a freely available and widely transportable approach to well-controlled data interchange that is open and accessible to the business community. The technology itself allows the design of languages that suit particular needs and harmonious integration into a general infrastructure that is extensible enough to meet requirements and adaptable enough to incorporate emerging new technologies.

The extensibility of XML is the main advantage of this technology as well as its main disadvantage. The ability to create custom-tailored markup languages can lead to a proliferation of languages within business entities. This may not be critical in simple browser-to-web-server solutions, but in business-to-business exchanges it is very undesirable and costly. The development of document definition methodologies and XML design rules is of paramount importance to stem the flow of divergent XML solutions and ensure smart and efficient use of technology resources.

Much work has been done in the document definition and core components arena by ebXML, ANSI ASC X12, and UN/CEFACT Work Groups. Every effort has been made to build on that foundation. However, all of this work is ongoing and some issues of alignment with other standards efforts have yet to be resolved. In its current state some areas of this report are incomplete at the detail level. It is likely that the continuing work to address issues of completeness and alignment will result in changes to some of the recommendations of this report. The X12C Communications and Controls Subcommittee responsible for the preparation of this report anticipates the release with the next year of ANSI ASC X12 Standards based upon the architectural concepts and XML syntax representation presented in this report, and upon the maturing work of the other standards efforts.

The XML syntax design presented in this Reference Model is based on design decisions reached through a process of issue identification, presentation of examples, and evaluation of the pros and cons of each available action according to W3C approved specifications. It provides a set of best practices that define the creation of XML representations of standard business messages.

Target Audience

The X12 XML initiative is targeted at every sector of the business community, from international conglomerates to small and medium sized enterprises (SME) engaged in business-to-business (B2B), business-to-consumer (B2C) and application-to-application (A2A) trade. With that audience in mind, the X12 XML initiative is committed to developing and delivering products that will be used by all trading partners interested in maximizing XML interoperability within and across trading partner communities.

The motivation to develop common standards for document interchange is to enable independent business entities to communicate with minimal additional cost and effort across a wide range of business opportunities. One way organizations can gain advantages of interoperability is by establishing a common set of “good” XML and XML Schema guidelines. The current W3C XML specifications were created to satisfy a very wide range of diverse applications and this is why there may be no single set of “good” guidelines on how best to apply XML technology.

Although this document is created by X12 for its own use, it seeks a wider audience. While standards developers comprise most of the people who attend X12 standards development meetings, the majority of implementers may never participate in development of standards. SMEs are largely unrepresented at standards development meetings but their needs can be served by products resulting from those efforts. Industries or associations who choose not to participate within the X12 environment can nevertheless follow these guidelines and position themselves to meet interoperability demands of the next generation of e-business standards.

Design rule decisions presented here are intended to balance the needs of all users of the standards. What seems like an advantageous decision from one viewpoint can be disadvantageous from another, but the intent was to produce guidelines to serve the common good.

High Level Design Principles

The following overall principles govern this design.

- Alignment with other standards efforts -- We shall align with other standards efforts where possible and appropriate. Such efforts include but are not limited to UN/CEFACT and OASIS ongoing ebXML work, World Wide Web Consortium, and OASIS UBL.
- Simplicity -- We shall keep components, interactions, use of features, choices, etc. to a reasonable minimum.
- Prescriptiveness -- This means that, for example, schemas shall be as specific as possible for their particular intended usage, and not generalized. When applied to schemas, this leads to more schemas, each with fewer optional elements and with fairly tight validations. This means that schemas actually used by anyone (rather than template schemas for starting points) would tend to be analogous to an Implementation Guide of a transaction set rather than the full standard definition of the transaction set.
- Limit Randomness -- When applied to processing an electronic business document, this means that when the document is being processed there are a limited number of variations that may occur in the data. It is related to optionality and prescriptiveness. We shall keep randomness to a practical minimum. NOTE: This provides a good philosophical basis for disallowing things like substitution groups and the "ANY" content model when designing document schemas.

Background

The Extensible Markup Language (XML) history and ebXML Business Process and Core Components have been part of the development that has brought us to where we are today. Annex D contains a more detailed review of each of these.

1. INTRODUCTION

1.1 Introduction

This Technical Report addresses the semantic and syntactic representation of data assembled into business messages. The semantic representation defines an overall architectural model and refines the model to an abstract level of detail sufficient to guide the message development process. The syntactic representation utilizes features of the target syntax, while imposing semantic-to-syntax mapping rules and syntax constraints intended to simplify the task of interfacing business messages to business information systems and processes.

The large-scale structure of this architecture has six discrete levels of granularity. Each level builds on the levels below it in manners particular to their differing natures. The six levels are:

DOCUMENT
TEMPLATE
MODULE
ASSEMBLY
BLOCK
COMPONENT

The first three levels, Document/Template/Module, provide features that promote interoperability between national cross-industry standards and proprietary user communities. The remaining three levels, Assembly/Block/Component have characteristics expressly designed around a rational semantic model for granularity. Specifications of optionality and repetition are supported for all levels. Special attention has been paid to the differing needs of senders and receivers in expressing the use of optionality and repetition required by their particular business practices. 'Documents' are the implementable resulting specifications, which are formed when the Slots in a Template are united with a set of Context specific Modules.

The six-level structure of this architecture is designed to provide useful granularity, while at the same time preserving a useful semantic clarity.

Design rules come in two basic forms:

- Syntactic, and
- Semantic

An example of a syntactic design rule in X12 would be the basic data types, i.e. alphanumeric, date, etc. An example of a semantic design rule in X12 would be the general prohibition against duplication. These two aspects of design cannot stand alone. The existing X12 design rules are a direct outgrowth of the particular X12 syntax and the history that created it.

For the ASC X12 XML Reference Model, a semantic design approach has been selected, breaking the EDI lexicon into units for re-use. This approach avoids some pitfalls that result from a decomposition of EDI issues using only syntax as a guide.

1.2 Support for Proprietary Efforts

A primary requirement for this effort has been to meet a need first expressed at the first XML Summit in August 2001. This was a desire for non-X12 participants to contribute and make use of X12 work but in a manner that didn't require an all-or-nothing commitment to either the X12 process or X12 conclusions in every detail.

The top three layers, Document\Template\Module, directly support this need by allowing the mixed use of standardized and proprietary data descriptions.

A proprietary Document can be constructed by combining ASC X12 standardized Templates & Modules with proprietary Modules.

An external entity, corporation, organization, or individual can contribute proprietary Modules for consideration by ASC X12. The level of conformance applied to these contributions would be two-fold. First, does it meet the function and purpose expressed for a particular Slot in a Template? Second, does it conform to the purely syntactic design rules established? A "cross-industry usefulness" test would not be applied. A "duplication of existing item" test would not be applied. Adherence to the X12 philosophical structuring of the bottom three layers (Assembly, Block, Component) would not be required.

2. RESOURCES

The following documents provided resources for this document.

http://www.xfront.com/ - XML Schema Best Practices as maintained by Roger L. Costello

http://www.ibiblio.org/xml/ - Café Con Leche

http://www.w3.org/XML – XML Schema Specifications

http://xml.coverpages.org/sgmlnew.html – Archive of Robin Cover's XML Cover Pages at OASIS

http://www.ietf.org/rfc/rfc2119.txt?number=2119 – Internet Engineering Task Force Request for Comments 2119

http://www.tibco.com/products/extensibility/resources/index_best.htm – Tibco's XML Resources Center Best Practices

http://www.ebxml.org – ebXML Project

http://www.ebtwg.org – UN/CEFACT electronic business temporary work group

Ducket, Jon, Oliver Grffin, Stephen Mohr, Francis Norton, Ian Stokes-Rees, Kevin Williams, Kurt Cagle, Nikola Ozu, and Jeni Tennison; *Professional XML Schemas;* Wrox Press, Birmingham UK, 2001

Dodds, Leigh, "Designing Schemas for Business to Business E-Commerce", http://www.xml.com/lpt/a/2000/06/xmleurope/ecommerce.html

Gregory, Arofan T. "XML schema design for business-to-business e-commerce", XML Europe Conference, 2000

http://www.ebxml.org, Core Components Overview Vestion 1.05, May 10, 2001.

http://quickplace.hq.navy.mil/QuickPlace/navyxml/Main.nsf/057A71D114B95B0D85256AF5006CAD86/1921E59CBABDEE2D85256AFB00605CB3**,** Initial DON XML Developer's Guide, October 29, 2001

Walmsley, Priscilla; Definitive XML Schema; Prentice Hall PTR, 2001

3. MESSAGE ARCHITECTURE

3.1 The Vision -- An Analogy

Imagine a horizontal bar containing a set of seven (7) wheels, each having eight (8) surfaces, a "Wheel Diagram." As the wheels are rotated on the horizontal bar, different surfaces are revealed [as illustrated in Figure 1], and on each surface is a different word.

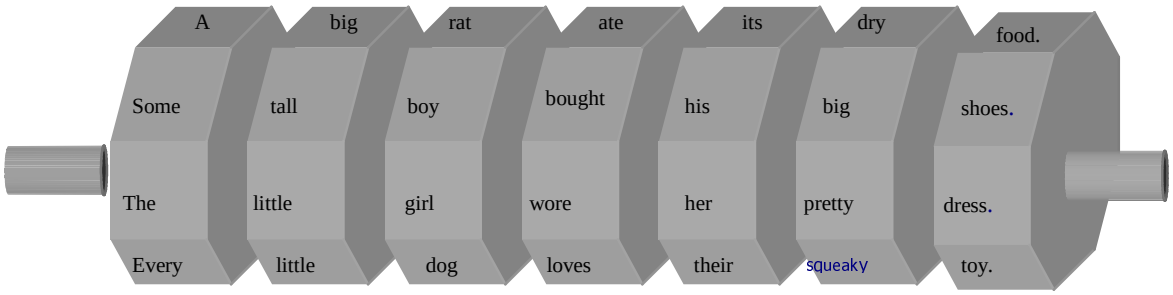


Figure 1

Each wheel rotates independently, thus the number of potential combinations grows exponentially with each additional wheel. With each possible combination of wheel surfaces, a complete and meaningful sentence is constructed.

The Wheel Diagram works because of grammar. The logical placement of the wheels [noun phrase + verb phrase], with each wheel representing a part of speech [article, adjective, noun, verb, etc.], enables each combination to yield a meaningful sentence – some rather silly, but still valid.

This analogy illustrates some concepts which, when applied to electronic business message design, offers an innovative design approach to electronic message design. It leads to a solution that meets the highest level objective, implementable messages, and strikes a balance between the interoperability achieved through standardization and industry needs for timely solutions – autonomy.

Business documents are much more constrained than natural language. Artistic representations are wanted. Further, like natural language only with far more regularity, business documents contain information to communicate *who, what, when, where* and *why*.

- Who answers which parties participate in the business transaction and the actors involved in the exchange.
- What answers the primary subject or purpose of the message.
- When answers event/timing details.
- Where answers location details.
- Why is typically answered by the message type itself, along with accompanying reference information.

3.2 Context Inspired Component Architecture -- modularly flexible Smart Messages

Overview

The Context Inspired Component Architecture, “CICA”, is based on the results of many years of critical analysis within the EDI/E-Business standardization efforts. This architecture leverages the best ideas to date in e-Business development, and applies new semantic rules and levels of abstraction.

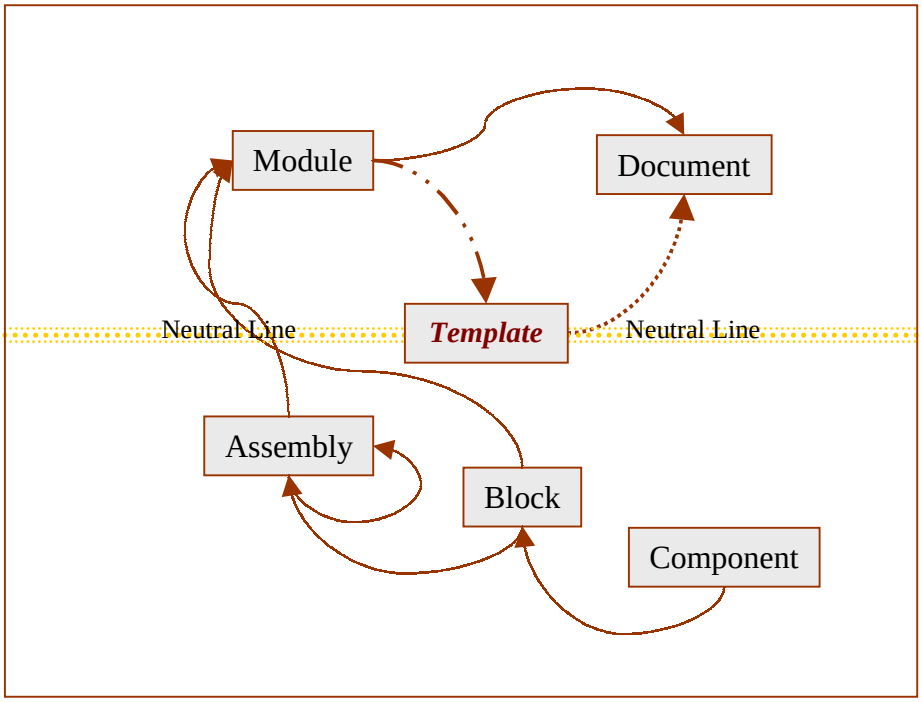


Figure 2

The architecture of CICA includes six (6) layers, the relationship amongst the layers is as illustrated in Figure 2.

The **Template** is conceptually the focal point of the architecture, bridging between Neutral constructs below the line, and Implementable constructs above the line, as shown in figure 2. The Template contains a set of Slots that specify the high level content requirements for the Business Document. The Template is linked with Modules, and the relationship is conditional in that the Module is only used when Context conditions are met.

The **Module** is physically separate from the Template, but associated with the template on a Context case basis, this association relationship is depicted in figure 2 with dashed lines. In other words, the Module is loosely associated with the Template, and only bound with the Template when a predetermined condition is met, Context. Modules answer, at the document level, *Who, What, When, Where* or *Why*. Modules are made up of one or more Assemblies and/or Blocks.

The **Document** is the user implementable business document, where the Template is joined with the context specific Modules. The Document is derivable from the Template, linked with Modules. This relationship is depicted in figure 2 with an evenly dashed line, linking the Document with the Template.

Assemblies are reusable aggregations of Blocks, which are neutral in terms of usage. For example, a neutral assembly might include a Party, who has a first, middle and last name, and a Location with a shipping address.

Blocks, like assemblies, are neutral. Therefore they are reusable, semantically constrained units of information. Blocks are constrained in size to specify either a Party, Resource, Event, Location or Condition.

The **Component** Data elements are defined within Components, and Components are placed within Blocks. Components specify either identity information or characteristics for the given Block. Primitives are subordinate to the Component.

Smart Messages

The foundation of the CICA architecture is the Business Document Template, “Template”. The Template is divided into Area’s, header, detail and summary, and within each Area are one or more Module Slots, “Slots”. The Slots are abstracted usage descriptions of the Business Document contents and act as placeholders for the context specific Modules which contain details.

The CICA architecture achieves flexible modularity with this detachment between the Slot and the Modules. Context, which specifies a set of business conditions, is used to link the appropriate Module with a Slot. This serves as a foundation for Modularly flexible messaging. For each Slot, there are one or more Modules, the use of which is determined by a specific pre-established context. For example, consider Figure 3, which depicts a Template with three (3) slots in the header, and two (2) slots in the detail. If the cube shaped Slot in the detail section of the Template represents Line Goods Item, there are three (3) sector specific Modules that fit into the Slot, each of which is used according to the context

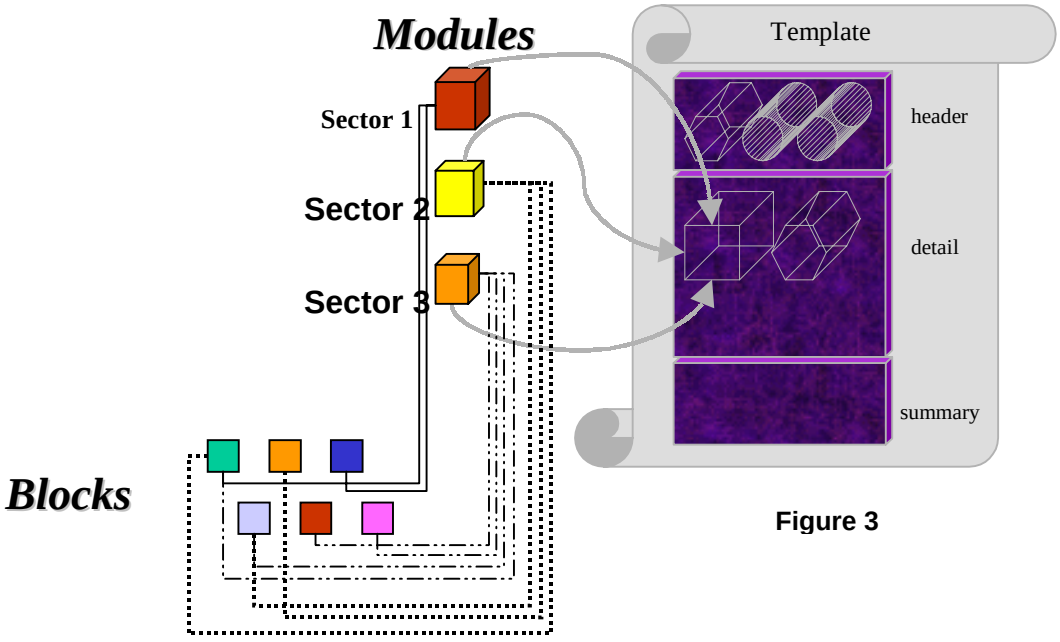


Figure 3

that specifies its use. For each Slot, one or more Modules are created in order to fulfill the purpose specified by the Slot. Each Module is either associated with the Slot as the default Module, or with a specific usage Context. In order to generate a Business Document, Context is specified for each Slot and the requisite links are drawn upon to compile the Context Specific Business Document.

3.3 Relationship between Vision and the CICA Architecture

The vision, presented in Section 3.1 and Figure 1, is recast to present the key CICA constructs.

Each Wheel Diagram represents a single Template, Figure 4a.

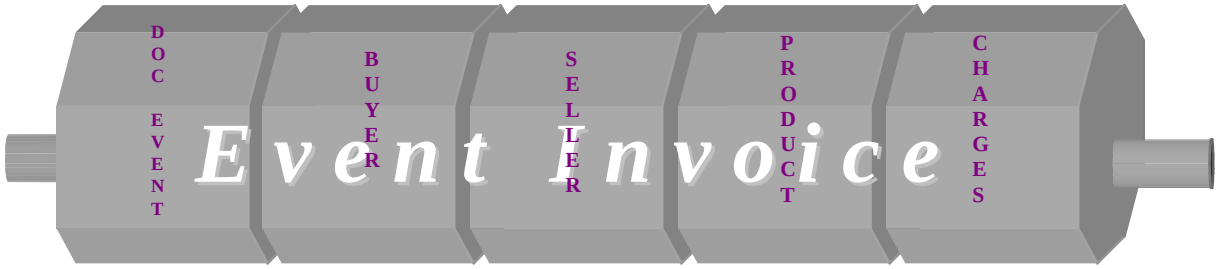


Figure 4a

A Template is determined based on business process circumstance triggering a unique condition. For example, in Invoicing, there are two distinct trigger events that result in entirely different arrangements and organizations of information – event based and account based. Event based Invoices are sent in response to a trigger event [Order, Shipment, etc.] Account based Invoices are sent at regular time intervals, with no specific precursor event. Therefore, Templates are defined for Account based and Event based Invoices.

Each Wheel in the Wheel Diagram for the Event Invoice, figure 4a, is a template slot specifying in abstract terms the Slot purpose. For example, one Slot might be for the "Buyer", a neutral term easily understood by interested parties, and independent of whether various industries have different content and terminology used in place of "Buyer".

The Template with Modules linked, shown in figure 4b, uses the Wheel Diagram to present the modularly flexible invoice example. In this example, each Slot or wheel is linked with one or more Modules, shown on Wheel surfaces. The first wheel is that of the Document Event,

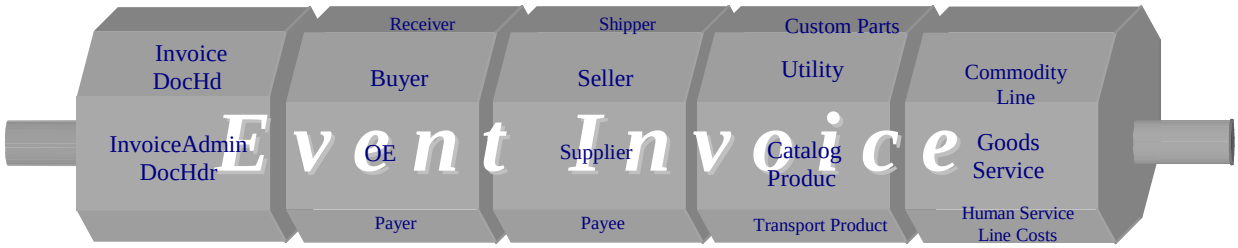


Figure 4b

and two surfaces have Module entries, InvoiceDocHdr and InvoiceAdminDocHdr. Thus, there are two [2] levels of agreement demonstrated, the abstract level depicted by the Slot/Wheel, and the specific level shown in the Wheel surface/Module.

The power of this architecture is twofold. First, the modular flexibility provides structured flexibility, maintaining stability at the context specific level. The second is the underlying layers of semantics, which provide for levels of agreement. For example, it may not be possible to agree to the details of how to specify a product, but it is possible to agree that this is the place where the product must be specified. The layers provide a means to achieve agreement and harmonization that are practical.

3.4 Templates

Overview

Templates are the focal point construct in the architecture, and play a pivotal role bridging between neutral and specific in achieving modularly flexible messages. Modularly flexible messages are an important innovation in that the resulting Business Document is semantically concise, yet the Template provides a mechanism through Module substitution for flexibility. The result is to accomplish both of which would otherwise be considered mutually exclusive objectives – flexibility & autonomy for responsive industry solutions and cross industry interoperability with semantically concise messages.

Templates are established for each business process specific use of a message, figure 5. As an example, there are two fundamental invoicing models, event and time based. Event based procurement involves a buyer and seller, where the buyer places a specific Order with the seller, the seller delivers in accordance with the Order, and the buyer is Invoiced in accordance with that delivery event. Examples of this include catalog orders, trips to the store, traditional healthcare plans, etc. In contrast, time based procurement involves an arrangement whereby the buyer and seller have an ongoing relationship, the goods/services are routinely made available and used as desired, and invoicing occurs according to a time schedule. This Invoicing style includes any statement/time based invoicing methods, specifically: utilities, credit card, telephone, etc.

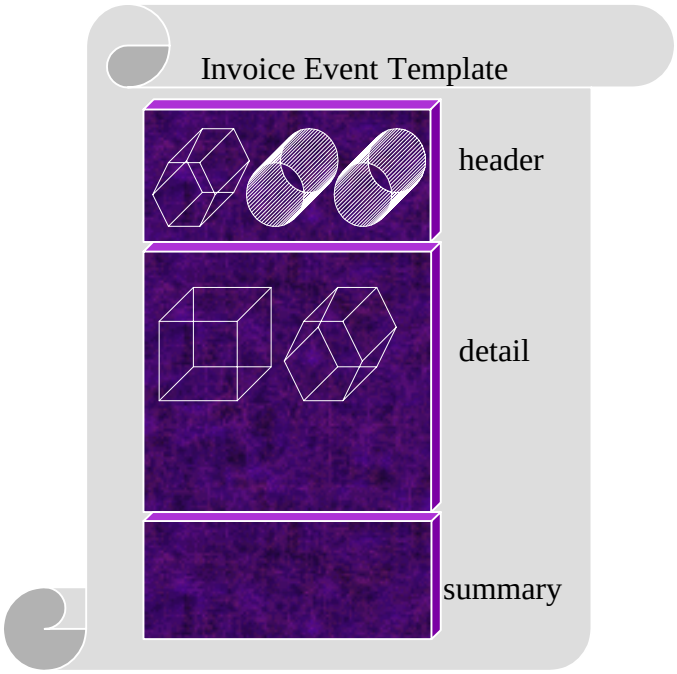


Figure 5

Contents

Shown in Figure 5 is a Template. A Template is divided into three logical areas, “Area”: header, detail and summary. These subdivisions have semantic significance in that header information applies to the entire Business Document and specifies the business context and parties to the business exchange, the detail contains the subject of the message, and the summary contains summarized information about the detail [use of this section is generally discouraged].

Business documents also need to explicitly specify the relationships among their components, to reflect the appropriate structure of those components during assembly. Knowing how the pieces fit together in the overall structure encourages reuse of the components in other documents or processes. In some cases the structure will be simple, but where documents represent a large volume of different items, or multiple references (e.g. a ship notice containing items requested in separate purchase orders), the structure can easily become more complex. The Template specifies this logical arrangement of information.

Within each Area are zero or more Slots shown in figure 5 with wire frame geometric shapes. Slots, depicted in Figure 6 by various 3D wire frame shapes, identify in abstract terms the

logical composition of the message at the business process level. Slots are absolutely specific in terms of the logical business purpose that they identify. The Slots are abstract in that they use a neutral term, such as Seller, although various industries/sectors might use Supplier or Provider. The abstraction is in harmonized, generally recognized terms and independent of industry or sector specific terminology. This aids in the reuse of the Templates, which are developed around Business Process requirements.

Slots do not contain contents, in other words data elements. They serve as a logical break between the purpose of information, Slot, and the detailed Context specific contents, Modules.

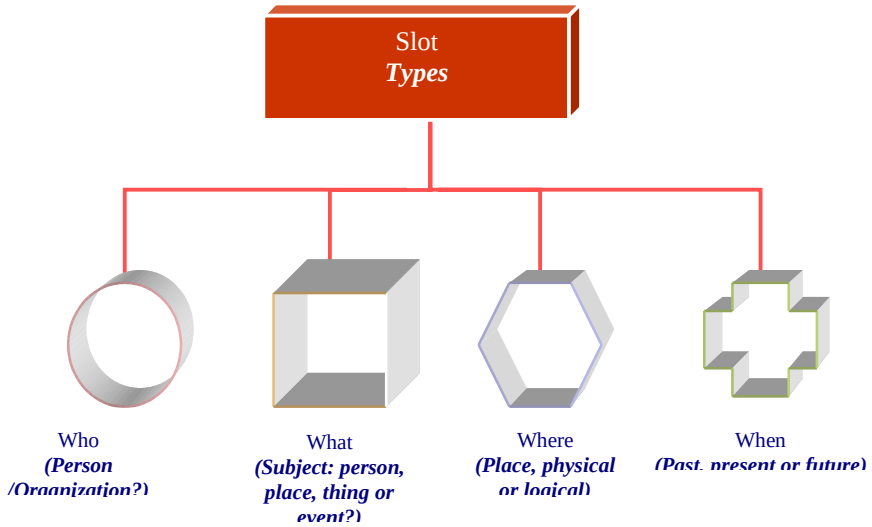


Figure 6

Slots are designed around the Business Document need to express the Who, What, When and Where, [as shown in Figure 6], which when combined detail a Business Document. Each Slot use is to specify a single one of the Who, What, When or Where, at the Business Process level. This subject will be dealt with more in the following sections.

3.5 Modules

Overview

Modules specify details in accordance with the abstract business purpose identified by the Slots in the Template. In general, one or more Modules are defined for each Slot identified in the Template, although it is possible that a single Module can fill more than one Slot in a Template. It is expected that some of the Slots will have only one or a small number possible Modules, such as the case in the Invoice case with the Buyer or Seller Slots. In other cases, there could potentially be many different modules, based on perhaps business sectors.

Figure 3, shown earlier, illustrates a situation where multiple Modules are associated with a single Slot. On the left is a set of Modules that can be plugged into the Slot. Each module in the example has some commonality – shown by the shared red filled box. This commonality in some compositions is not a requirement of peer modules, but what is certain is that there is different composition. Therefore, amongst various industry sectors there are differences in information requirements for modules, e.g. line goods item. The links between the Slots and Modules, shown with arrows, are established for a context. What is meant by Context is a specific business circumstance that unambiguously links the specific Module to the Module Slot in the Template. Context is specified in a prescribed manner, described in 3.6.

At the Template level, for each of the Slots, Context specific links are made between Slots and Modules. Modules can be reused many times across Templates, whether the templates in which the modules are used are:

- Peer Templates: Templates that serve the same general business function, such as Invoice.
- Same Business Process: Templates used within the same business process. It is expected that a single Module could appear in multiple or all Templates used in the business process.
- Same Sector: Modules that are sector specific, such as ones specifying the sector's product/service, could be used in a variety of business processes in which sector members participate.

Types

Modules, like Slots, are formed around the same semantically motivated boundaries. Grammatically speaking, like Slots, Modules specify either a *Who, What, When,* or *Where,* as illustrated in Figure 7. The Slot identifies the need for a Module in terms of the business process purpose or usage, specified in abstract terms. The Module supplies a set of details responding to the prescribed purpose, the Slot.

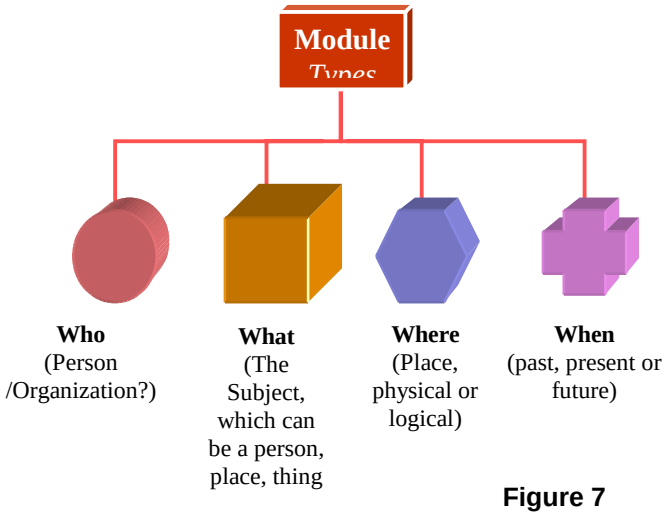


Figure 7

Contents

Modules are made up of reusable constructs, which are either Assemblies or Blocks. A Module can be as small as that being constructed from a single Block, or as complex as constructed from a set of Blocks and Assemblies.

Placing a Block or Assembly into a Module gives it a semantic purpose. Modules can be complex enough to require the use of multiple Blocks and Assemblies, although the primary purpose is singular. For example, within Vital Records exists the need to specify a party who has died, a decedent. The decedent is the Module, as shown in Figure 8, but peripheral to the decedent are the birth/parents, last address, spouse/marriage, and adoption/parents, etc. These are descriptive details about the decedent that involves parties, locations, events, etc.

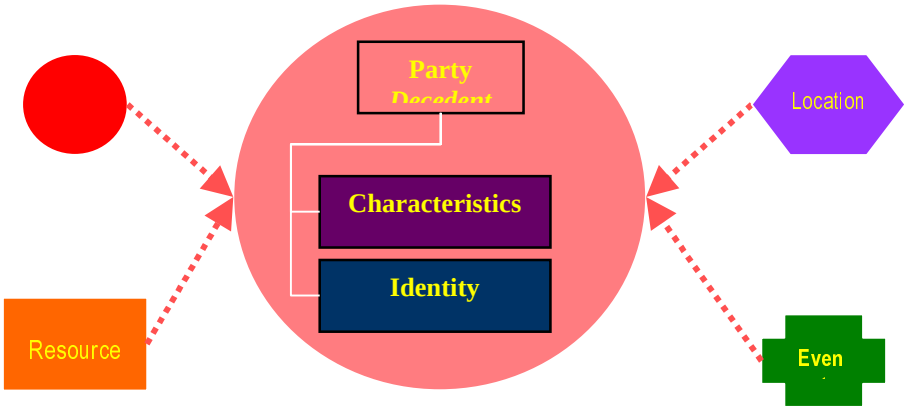


Figure 8

3.6. Context

Overview

Separation between Template with Slots, and Modules, is fundamental to accomplishing modular flexibility. Hand in hand with the separation is the need to predictably establish the link. This is the role of Context.

Context is the set of descriptors that quantify the 'business circumstances' under which a Module is used in a Slot. This is most easily explained with an example.

In our Invoice example [figure 4b], the Template contains a number of Slots, one for Document Event. Two Document Event Modules have been identified, shown in the illustration on the wheel surfaces. The first is the default Doc Event. The second, Administrative Doc Event contains additional reference information used in dealings with the US Government, and could have some other applications. So, the link between the Slot and the Modules are different for the two different Modules, and Context is used to explicitly describe when each Module is to be used.

Types

In order to ensure consistency, Context must be analogous to a highly constrained language, where there are a set of predetermined parts of speech, context categories, and a predetermined set of values. This ensures that if your goal is say Ocean Transportation, there is one and only one way to say it, avoiding ambiguity.

In keeping with our goal to align with the ebXML Core Component work, their section 6.2.2 of the current technical specification, specifies categories of Context, or types. The contents are contained in Annex C of this document.

The URL for the full document is

http://www.unece.org/cefact/ebxml/ebXML_CCTS_Part1_V1-8.pdf,

Contents

A comprehensive list of values must be specified for each context category. For each context category, the ebXML CC specification has identified one or more available sources. These, in addition to X12 selection, are documented in Annex C of this document.

3.7 Documents

Overview

Documents are the unit, which contains a Context specific, details specification upon which transformation rules are applied to generate an XML Schema. These resulting Documents are what are considered "bullet" documents, in that they are semantically concise for implementation.

Documents are produced when Context is applied to the Slots in a Template, given the Context links between Modules and Slots. For example, in Invoicing, there is a Slot for Document Event, linked with two Modules. Thus, at the Slot level there is a decision that must be made to select the proper link, specifying Context. Once the Context for each Slot is specified, the Template is joined with a specific set of Modules – a Document.

The Document is covered in more detail in the Invoice Example, detailed in Annex B.

3.8 Assemblies

Overview

Assemblies are a construct used to create reusable groupings of Blocks. Like Blocks, they are independent of usage, neutral, and fit between Modules and Blocks. Blocks, which are detailed in the following section, are semantically limited to specify a single Party, Location/Place, Resource, Event. Various groupings of these constructs can be very convenient, in order to specify structure or for reuse purposes. For example, party + location are commonly used constructs and an Assembly is a convenient means for managing this reuse.

In our Invoice example, Buyer is a Slot, and a set of Modules is produced to specify the various Context specific contents for Buyer. One such Buyer Module contains six (6) parties: the Buyer, the Buyer Contact, the Paying party, the Paying party Contact, the Ship-to party, and the Ship-to party Contact. In each case, the Contact party has the same composition. This is a candidate for an Assembly.

Types

Assemblies will typically have a primary type. In the example above using party + location, the purpose of the assembly is to specify a Party which has a location. While location information is supplied, the primary purpose is to specify the Party.

Contents

Assemblies are created out of one or more Blocks and/or Assemblies.

3.9 Blocks

Overview

Blocks are constructs created to specify a single Party, Resource, Event, Location or Purpose. Blocks are concise units in that they specify in detail and with all that applies to the Identification and Characteristics of the object being specified.

Types

Blocks specify a single noun, i.e., a Party, Location, Resource or Event [as shown in Fig. 9].

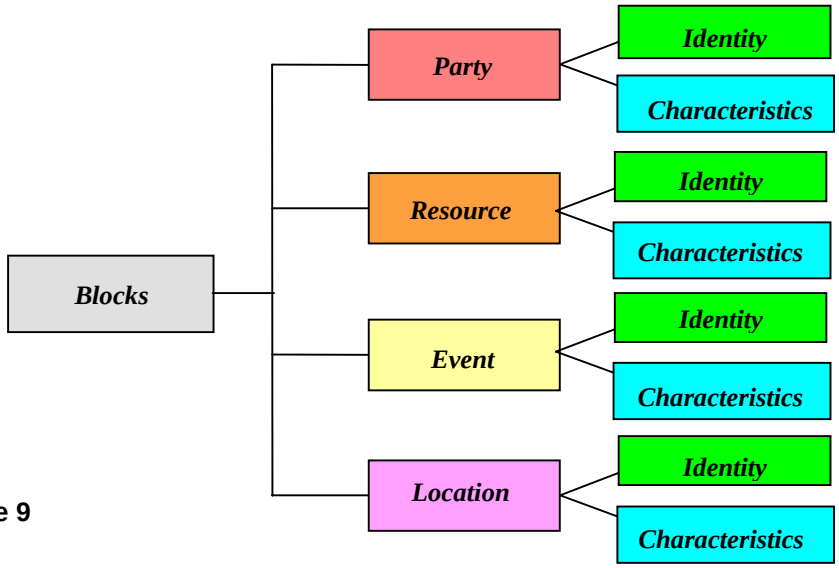


Figure 9

The single noun is a critical element of this architecture's granularity. All Blocks are comprised of a single noun, therefore, semantically speaking, Blocks are predictable and peers at three (3) levels, they convey a single Noun, they convey a Noun of one of four types [party, resource, event or location], and the general contents [identity and characteristics]. This granularity assures that peer Blocks are semantic equivalents. This is a foundation required to achieve modular flexibility.

Contents

Each Block contains Identity information that varies depending on the type of Block, and optionally may have Characteristics. Anything less is not a Block; anything more must be an Assembly if neutral, or a Module is semantically specific.

3.10 Components

Overview

Components are the lowest level contained within this architecture. Components, like Blocks, are formed based on the need for a physical arrangement of information. For example, given two types of Parties, an individual and an organization, the identity information required for the two types of Parties is significantly different, therefore the components used to specify identity are different. The need for different components results in the need for separate Blocks.

Types

Components are used to specify one of two types of information, Identity or Characteristics.

Identity information is going to vary based on the Block type. The details required to identify a person are dramatically different from those details used to identify an event.

Characteristics provide descriptive information, such as physical or demographic details. To make a comparison to natural language, characteristics can be compared to adjectives. Typically, characteristics are one of two forms, 1) have a companion piece specifyied, such as in the case of unit of measure or, 2) are one of a finite list. Some examples of characteristics are height, weight, hair color, class of service, property feature or quality, etc. Primitives have been defined to establish linkages between peer semantics, when represented differently physically.

In the interest of alignment with Core Components, the types of components are aligned. The types are taken from table 6-1 in the current ebXML specification, http://www.unece.org/cefact/ebxml/ebXML_CCTS_Part1_V1-8.pdf, and included in this document in Annex C.

4. COMPONENT ARCHITECTURE

Given the number of industries, organizations and business processes that are involved in making eBusiness standards – there is no shortage in complexity. In this environment, even making the determination that two things are the same is not as straightforward as it sounds. And when they are not the same, how many ways can they be related? And what conclusions and knowledge can be drawn from structural relationships?

This effort relies heavily on a strong semantic foundation for all decisions. Integral with the strong semantic foundation is the need for quantifiable indicators for making decisions, including the ability to quantify precisely the ways in which two things might be considered related and at what point they are to be considered the same.

From these tests, rules can be formulated to reinforce these conclusions.

4.1 Structure Rules Overview

There are three tests that can be applied when comparing two candidate information constructs to determine the level to which they are related. These are Form, Fit and Function, and they are taken from the Parts world where they are used to determine when a new part number needs to be assigned. These tests, while the same for each CICA construct, have slightly different implications depending on the semantic abstraction of the construct. Modules, the most semantically specific construct, are more sensitive to purpose and usage and a little less impacted by structure. In contrast, Blocks contain abstract semantics and are affected more by structure. These details effect how to apply these tests and the resulting rules. The general concepts are presented below.

For eBusiness considerations, Form, Fit and Function are defined as follows:

FORM: Physical – the structure, contents and components of the information structures being specified. For example, parts have names and so do people. People have first, middle and last names, whereas a part has a single name, part name. The difference in Form makes these two types of names different. In contrast, you might have a Student First Name and Student Last Name, compared with a Patient First Name and Patient Last Name. Form-wise, these two examples are the same.

FIT: Identity-Meaning-Specificity – Two organizations or industries that share the common element named Part Number have reason to believe that there is some commonality. Sometimes two uses of an identically named item do not provide the same level of specificity, and therefore these items are not the same thing. In ebXML, a case using a Vehicle Identification Number, “VIN” was used. Different organizations use the VIN, but they may be referring to a different subset of sub-components. Each subsection of the component parts of the VIN, for the same vehicle, is different information. Can they all of these different subsets of the same base number all be called VIN – no! Other examples are found with Part Number, with different levels of specificity found with a construct called Part Number. For these to be considered the same, they must specify the same level of specificity.

FUNCTION: Purpose or how used. – When comparing two information constructs, such as Product, there is a common purpose or usage – which motivates treating them as ‘the same’, even though the detail used to specify various Products can vary widely. In some cases, the various Product descriptions are similar in form, but in many cases, this is not the case. Effort to merge dissimilar definitions results in ambiguity, which later needs to be disambiguated. In the CICA architecture, through the use of abstract layers and links, these Functionally related constructs are associated, without imposing ambiguous merging.

4.2 Detailed Structure Rules

The levels of equality that are true determines how closely related two information constructs are. Consider the following:

4.2.1 Condition 1:

FORM = YES

FIT = YES

FUNCTION = YES

When all three test are true, then with 100% certainty we can determine that the two are the same thing, the constructs are semantically equal. Examples of this situation are Shipper, Seller, or Supplier. These are different industry-specific terms for a semantically equivalent party playing a role. Frequently the descriptive details are exactly the same; and when that case is true, they are semantic equals in every sense.

4.2.2 Condition 2:

FORM = NO

FIT = NO

FUNCTION = YES

When equality is based on function alone, the two information constructs appear below a common parent structure. For example, in the travel industry you have rooms in hotels and passenger seats on flights. Although they are specified with different data elements and are called different things, they are used in the same manner in a business process/message. Thus, the two appear beneath a common parent [at some level], possibly human service products.

4.2.3 Condition 3:

FORM = YES

FIT = NO

FUNCTION = NO

This case is very common in EDI today, and is well supported. The X12 N1 loop specifies the name, id and address of any party, person or organization. The fundamental difference is that in the CICA architecture, Blocks are specified for the various data arrangements [different where a party is an individual versus an organization]. Further, this is independent of whether the construct can represent many purposes, which is the expected case. Therefore, in terms of Blocks, it is expected to have a single block [Party with First, Middle and Last Name] used for many specific parties: Passenger, Patient, or Student.

4.2.4 Condition 4:

FORM = NO

FIT = YES

FUNCTION = NO

This is the case where an information construct serves the same semantics in two different settings/business conditions, but it is used differently and has different components.

4.2.5 Condition 5:

FORM = YES

FIT = NO

FUNCTION = YES

In the automotive industry, Part Number is used to specify the desired product.

Ford has a significant digit part number which is really a composite of several identifiers: base + change number + color number + location on vehicle + etc.

GM and others have a part number too, but it refers only to the base. Separate additional values are required which include: change number, color number, location on vehicle, etc.

Both of these are related in that they are used to specify THE part, but they are NOT semantically equal. They do not provide the same level of specificity. Therefore, although they are used for the same base purpose, they cannot be used interchangeably and therefore, they are not the same.

4.2.6 Condition 6:

FORM = YES

FIT = YES

FUNCTION = NO

This case happens primarily when multiple business processes are involved. Consider a scenario where a Doctor is treating Patients versus a scenario of a business process where a Clinic is communicating its Assets – its staff. In both cases the form and fit are the same, but the function is different. It is unclear what structural implications this case has.

4.2.7 Condition 7:

FORM = NO

FIT = YES

FUNCTION = YES

In this case there is a difference in form, as is the case with Person Name versus Organization Name. Both cases are serving the function to specify the Party. Last Name does not equal Organization Name, because they don't deliver the same level of precision. In order to achieve the same level of "Fit", it is Organization Name = Last Name + Middle Name + First Name. Fit ensures semantic equality.

4.3 Preliminary Block Structures

Applying these rules and the desire to illustrate the concepts presented in Section 3 has lead to an initial set of Block constructs that are at a level where we are accustomed to operating in the EDI world. The usage-independent nature of Blocks makes them inherently cross industry.

Blocks contain two types of information, Identity and Characteristics. Identity information is used to specify the unique, instance identity of the Block. The content is dependent on the type of Block. This will be examined in more detail in a subsequent section. Characteristic information is descriptive information, which is typically in one of two forms, pick-list or value plus unit of measure. Examples include: length, width, height, weight, eye color, temperature, etc.

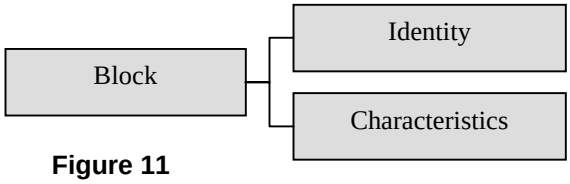


Figure 11

4.4 Party Blocks

The party answers a single *who* question. Parties in a business process and message can be individuals or organizations, or combinations of the two. In some cases the parties are also actors. For example, many purchasing applications need only buyer and seller organizations as actors, optionally identifying contact persons. With other processes, the party becomes the subject of the message, e.g. health care, education, or law enforcement. In these latter cases, the data represented in the process and subsequent messages become more detailed. The detail manifests itself in one of two ways, first with characteristic details [height, weight, eye color, etc.] conveyed at the Block level and secondly, details that need to be associated with the party but are not intrinsic to the party. For example, other parties, events, locations, etc. might need to be associated with a base party block in order to construct a complex structure. This is done in an abstract manner with Assemblies and a context specific manner with Modules. The key point is that these complex needs, beyond those of Characteristics, are accomplished with other blocks.

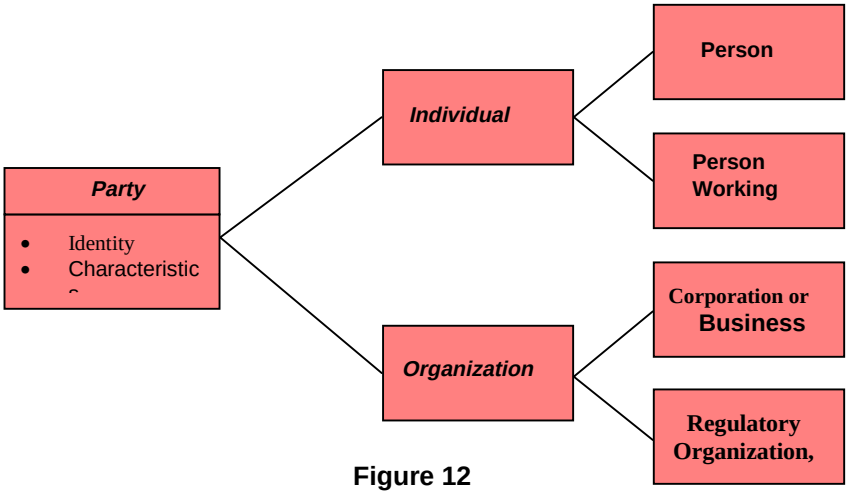


Figure 12

This approach allows Blocks to focus on what is directly attributable to a Block, usage independence.

The fundamental difference between Role Player and Subject parties is that Role Players tend to have Identification information, but not to have Characteristics. Therefore, any Party can be a Role Player.

Based on the set of structure rules detailed in Section 4.2, the top-level breakdown for Party is depicted in Figure 12. Differences in the Identity and Characteristics ranges specifically prescribe this breakdown. Identity information for an Organization includes a Name and an Organization ID number. However, there is a fundamental difference between Corporations/ Businesses and Regulatory Organizations, leading to a further breakdown subordinate to that of Organization. The Name probably doesn't vary, but the organization might have a number of ID numbers depending on context. However, they are all ID numbers that are suitable and appropriate for the identification of an organization.

Individuals, having First, Middle, Last Names as part of their Identity, clearly are different from Organizations. Further, as Individuals we are managed and served within our environments. Between 9:00 a.m. and 5:00 p.m., Parties take on an alter ego by assuming roles, such as Employees or Students. This calls for additional identity information: titles, status, etc. If this is the case, there are a couple of individual Blocks, that of Person and Person Working.

4.5 Resource

Economic resources answer the *what* question in a business document. As Figure 13 shows, resources break down into products and financial instruments. Products are the goods and services of value generated by companies for their customers, while the financial instruments – various forms of cash or credit – are the means by which the customers pay for those goods and services.

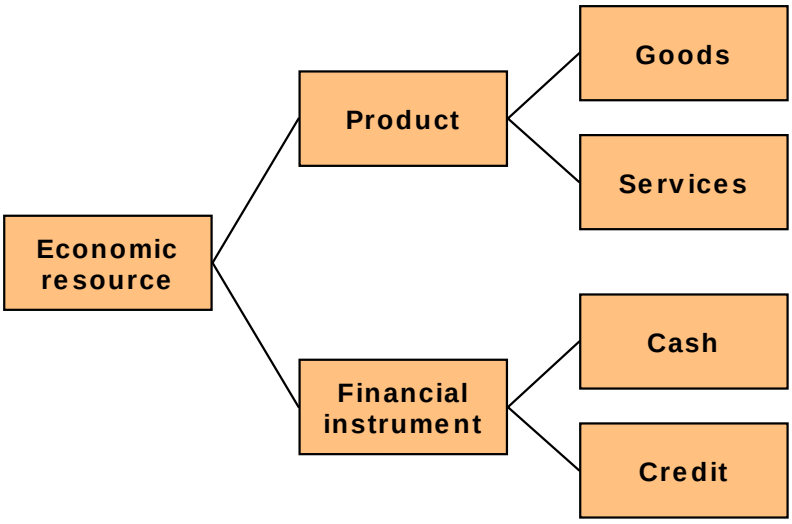


Figure 13

4.6 Events

Events answer the *when* question in business documents and are easily spotted with the tell-tale date and time details. As shown in Figure 14, preliminary thoughts are that there are two primary types of Events, basic events and experiences. Basic events include the Event Identity [which includes a Date/Time]. Basic Event examples include Birth, Incorporation, Shipment, events which are immutable – they happened. Experiences cover the type of specialized Event that are mutable and tend to have durations (certificates, level of attainment, status), and time periods such as in licensing. Further definition is still needed for capturing histories, such as audit trails or shipping/receiving histories.

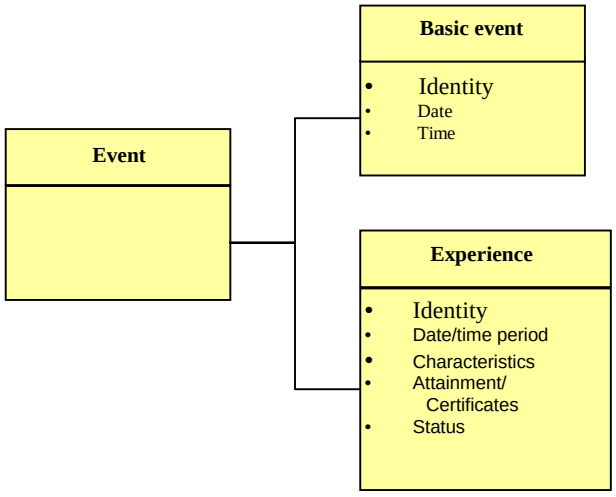


Figure 14

4.7 Location

Location represents the answer to the *where* question in a business process (Figure 15), and are either physical or electronic, but each provides as part of its identity a precise and unique address. Physical locations can be represented either in geography by latitude and longitude readings, or by postal and delivery addresses. Electronic addresses in order to be unique often need to follow standard schemes, such as Uniform Resource Indicators or ITU international telephone number conventions.

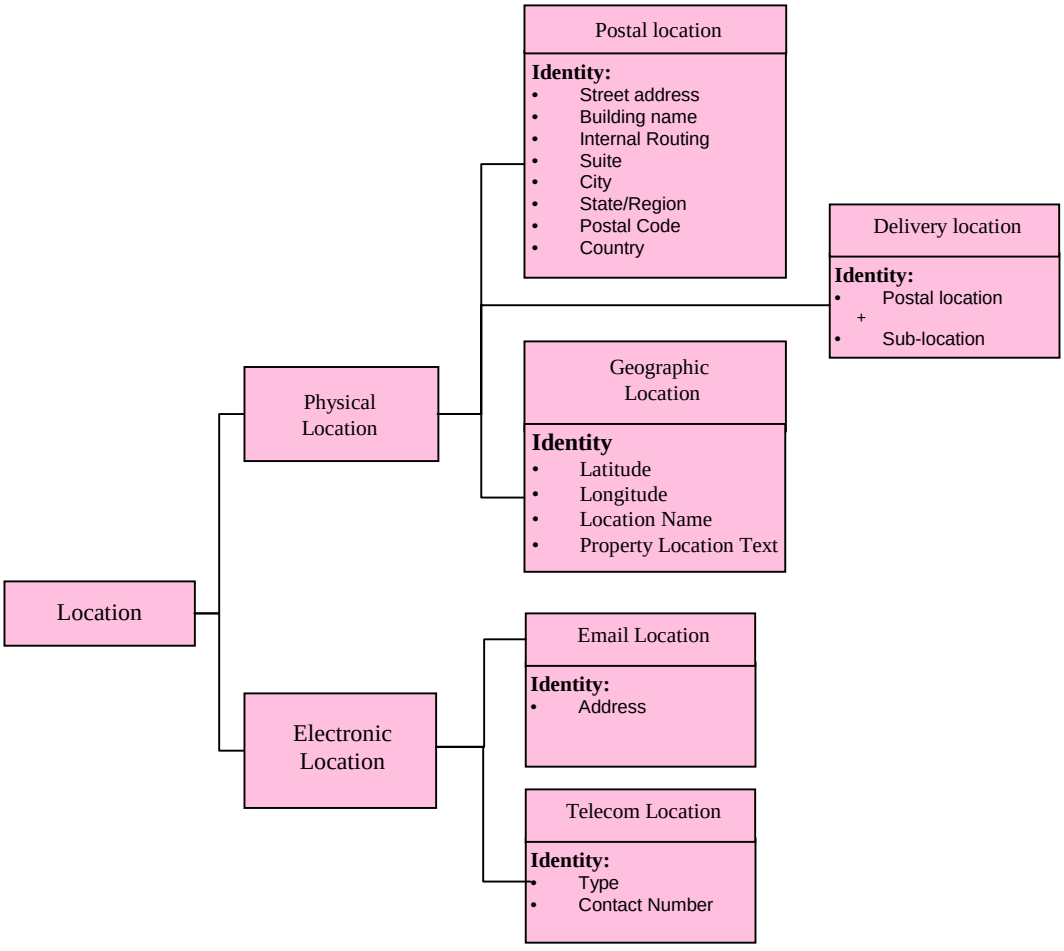


Figure 15

5. OTHER DESIGN ISSUES

This section presents a number of other miscellaneous design issues that are not directly addressed in the architecture. In some cases conclusions are presented. Where no clear consensus was achieved, only discussion is presented.

5.1 What Constitutes a "Bullet" Document?

Concept Defined

A document sent from one person or organization to one or more persons or organizations containing a single instance of a primary subject including supporting details or data.

Discussion

A "single primary subject" does not imply there can only be one line item in a single business document.

The current X12 TS 837 for health care claims allows information for more than one patient to be submitted in a single transmission or a single instance of a transaction set. Applying our definition of a document would require six documents for six patients, a document for each, that could contain multiple supporting details for that patient.

5.2 Default Override

Default and override are two related concepts, discussed here together because of their dependencies. The foundation for the discussion are the concepts embodied in current X12 EDI semantics, and formally expressed in X12.59 "Semantics in EDI".

Concept Defined

In order to specify a delivery of a line item, you must say what it is and to where it will be delivered. If XML maps those two things at each line item, it is simply syntax conversion. Using "default" requires (1) sorting capability, and (2) knowledge of doing comparisons to determine if the detail matches the default. If XML were to require that advanced processing capability and knowledge, simple off-the-shelf tools will not handle it, resulting in a situation that precludes bringing on board the SMEs.

Depending on your specific concerns this might be thought of as "duplication of data problem", "default and overriding data", or perhaps "table 1 & table 2 semantics".

Discussion

X12 Practice:

1) The X12 "Semantics in EDI" paper states the premise like this (paraphrasing here in a semi-code-like fashion)

1-A) If some data XXX appears in both table-1 and table-2, The data XXX is considered default values for all iterations table-2.

1-B) If the data YYY appears in an iteration of table-2, The data YYY overrides the earlier data XXX.

2) Many X12 docs have multiple sections that carry "structurally-like" but "semantically-different" data. What makes this worse is that in situations where the two "hunks" of data might be the same, a need (perceived or otherwise) for data compression leads to gyrations in either the message construction (read: weird loop or HL) or usage (read: gotta explain it in the IG). All so you don't have to send/specify the values twice.

Problems:

This is understood well by the X12 community, perhaps too well. This "fact" of the overriding defaults is not consistently pointed out in our semantic notes for particular Transaction Sets, and thus often must appear in implementation guides. And if the references to it are not called out well, a recipient can misinterpret the intent of the sender.

A related issue revolves around duplication of structure (and data values) in messages (and their instances). We have discussed this as a "multiple" roles issue. For instance, in a health care claim there is always a subscriber and a patient. Groups of segments are provided in the 837 for both, and the HIPAA guides (and other IGs) describe what values to send when both are the same person.

Straw Man Proposal:

Introduce specific "semantic attributes" to positively indicate in the instance data stream the situations/conditions described above. These attributes might only appear on modules or blocks. Caution is advised however, since using them in a finer-grain manner may introduce as many problems they solve.

The example below uses arbitrary names so as not to influence the final name selection.

Examples:

1) A typical "table1 is default"-"table-2 overrides" example

1-A) To indicate that something is a "default" we have an attribute for modules as in:

<ShipTo gork="default"> --ship data-- </ShipTo>

1-B) Later in a "table 2 iteration" (not limited to this, but to keep discussion simple) we have additional overriding shipping info:

```
<LineItem>
    <ShipTo gork="override"> --ship data-- </ShipTo>
    --line item data--
</LineItem>
```

2) A simple "Same As" or "Also Is" example,

2-A) A module of "subscriber" info stating it is also the "patient"

<Subscriber woof="Patient"> --party info-- </Subscriber>

-or-

2-B) A module of "subscriber" followed by a module of "patient"

<Subscriber> --party info-- </Subscriber>

<Patient woof="subscriber"> --info?-- </Patient>

3) Complex or "deep hierarchy" document. An attribute might be required to ensure linking the right "pairs" default/override or same-as/also-is modules. This proposal is for a "serialization" mechanism.

3-A) Variation of 1-A/1-B

```
<ShipTo  gork="default"  blat="001">  --ship data-- </ShipTo>
...
<LineItem>
    <ShipTo  gork="override" blat="001">  --ship data-- </ShipTo>
    --line item data--
</LineItem>
```

3-B Variation of 2-B

```
<Subscriber blat="001">  --party info-- </Subscriber>
<Patient  woof="subscriber" blat="001">  --info?--  </Patient>
```

Conclusions

By expressly stating the individual semantics being expressed in the instance document, we are able to avoid "implicit" relationships that now appear irregularly in semantic notes and IGs. The use of attributes here is appropriate, as they convey "semantic relationships" in a way that is outside of the "data content". I am unsure at this point if the two concepts (Default/Override in example 1 and SameAs/AlsoIs in example 2) should use the same attribute ("gork" & "woof" in the examples) in practice.

However, some open issues remain:

- It has yet to be determined whether or not every piece of information which forms a default has a default attribute designation.
- This may also have an affect on mandatory versus optional designations. For example, say that the information required in a certain part of the message, say the ship to in an order detail line, is mandatory in a semantic sense. But, if a default block ship to block is used in the header, then the information in the detail lines is optional. So, this makes the default ship to block in the header mandatory. However, it is possible to construct messages where there is no ship to in the header but each line item has a different ship to address.
- In cases like a health care claim where the Subscriber is the Patient, then you shouldn't have to supply some patient details, but otherwise these are mandatory. This could perhaps be supported with mutually exclusive sections, one for when the subscriber is the patient, and one for when the subscriber is not.

5.3 Two Roles for Same Instance Information: Explicit vs Referential Content

Concept Defined

Many business documents have data structures that repeat. Sometimes the identical data structures can contain identical content as well. Examples include *ship to/bill to*, *subscriber/patient*, *manufacturer/vendor*, etc. For these cases it's quite reasonable to consider whether specifying a way to eliminate repeating data (*referential content, implied content, or inferred content*) is better than just repeating the data (*explicit content*) where applicable. The following example illustrates an instance of this situation.

Example

Explicit Content

```
<HealthCareClaim>
    <Subscriber>
        <IdentificationCode>1</IdentificationCode>
        <Name>Santa Clause</Name>
        <Address>North Pole</Address>
        <WorkPhone>555-555-9627</WorkPhone>
    </Subscriber>
    <Patient>
        <IdentificationCode>1</IdentificationCode>
        <Name>Santa Clause</Name>
        <Address>North Pole</Address>
        <WorkPhone>555-555-9627</WorkPhone>
        <EmergencyContact>Mrs. Clause</EmergencyContact>
        <EmergencyPhone>555-555-9628</EmergencyPhone>
    </Patient>
    <ReasonForVisit>Chimney Smoke Inhalation</ReasonForVisit>
    <Total>73.48</Total>
</HealthCareClaim>
```

Referenced Content

```
<HealthCareClaim>
    <Subscriber>
        <IdentificationCode>1</IdentificationCode>
        <Name>Santa Clause</Name>
        <Address>North Pole</Address>
        <WorkPhone>555-555-9627</WorkPhone>
    </Subscriber>
    <Patient>
        <PatientSameAsSubscriber>true</PatientSameAsSubscriber>
        <EmergencyContact>Mrs. Clause</EmergencyContact>
        <EmergencyPhone>555-555-9628</EmergencyPhone>
    </Patient>
    <ReasonForVisit>Chimney Smoke Inhalation</ReasonForVisit>
    <Total>73.48</Total>
</HealthCareClaim>
```

Discussion

Arguments for the Referential Approach

1. Smaller XML instance documents
   a. Requires less bandwidth
   b. Requires less storage space
2. Consistent with a referential approach to data structures that some developers are comfortable with.

Arguments for the Explicit Approach

1. Easier to express as an XML schema design rule.
2. Easier to apply as an XML schema design rule. Schema standard working groups will set standards faster and be more confident in their decisions.

3. The data structure requirements of a business document can be expressed exclusively in the associated XML schema. Additional documentation is required for the referential approach.
4. Instance documents are clearer (arguably).
5. Easier for companies to implement.
   a. Slightly lower learning curve.
   b. Lower development, integration, and testing costs.
6. Lower costs to bring new trading partners on-line.

Notes

1. Some have suggested that their on-line purchase experiences validate the referential approach. Many B2C e-commerce sites (like Amazon.com) require bill-to and ship-to information. These sites often require that the user enter bill-to information and allow the user to simply click on a “Same as Bill-To” check box rather than enter duplicate information in the Bill-To fields (if applicable). This case really doesn’t apply to the rule under discussion since the driving factor for the user interface design (web page) is user convenience which does not necessarily suggest a corresponding data structure on the web server.
2. If one takes the referential approach, would the reference be *required* if the data matches? In terms of our example, if the subscriber data and reference data match, *must* the patient be referenced? Is it acceptable for the patient data to be explicitly expressed (i.e., duplicated)?
3. Are there cases where one would need to *know* that the patient *is* the subscriber?
4. This rule is related to but independent of the use of identification codes. For example, the XML schema may require subscriber and patient identification codes and not require any of the demographic information. Considerations about this type of data structure are not affected by the rule under discussion.
5. People’s time is money.
6. Delayed ROI is money.

6. METADATA AND DEFINITIONS

6.1 General

The metadata described here is the visible face of the DISA database for maintaining ASC X12's XML standards. Other metadata required for the maintenance and integrity of the database itself are not described.

The general view of the metadata is seven distinct dictionaries. One for each major architectural construct: Document, Template, Module, Assembly, Block, Component, and Primitive.

For all dictionaries, a primary key name will be maintained for each entry. This name will be unique across all dictionaries.

Additional names will be maintained for each syntax expression supported, initially this includes only the XML syntax. The uniqueness requirements for these additional name lists is an Open Issue.

Unless otherwise stated all MetaData items are mandatory, and must be included in the MetaData descriptions. This does NOT imply that all items in a list are mandatory in usage.

Unless otherwise stated all metadata items are text-string, with the general exceptions noted here

- **RequirementsFlag** values are one of the following:
  - **M** Manditory
  - **O** Optional
- **MinOccurs** values are integer numerics equal or greater than zero.
- **MaxOccurs** values either an indication as unbounded, or a integer numeric greater than zero.

It is the intent that the X12-XML MetaData itself will be made available in XML syntax, the precise format of an XML syntax to carry the MetaData is an Open Issue.

The database will also make available information on relationships between items in different dictionaries. This capability will allow for listing all references to individual items in one dictionary by items in other dictionaries. (e.g. all Templates with a particular Module in any TemplateSlot, or all Modules containing a particular Block)

6.2 Document

Definition: A distinct message description, reflecting a specific Business Process being used in a particular context.

Use: Expresses a single message format, reflecting the needs of a specific set of business contexts.

Properties:

- Represents an exchange of data that fulfills a single purpose in a business process
- Completely specific semantics.
- Basis for production of an individual XML schema

Composed of:

- A reference to a specific Template
- A set of specific Module references, made from the choices available in the Template
- A set of context references that drove the Module choices

Relates to/similar to (other specifications):

- Implementations of ASC-X12 Transaction Sets, or UN/EDIFACT Messages.
- Complete XML message specifications from other communities

Document

- **DocumentName**

  A descriptive name for the Document, for consumption my humans, used as the primary key for **Documents** in the DISA database. Maintained as unique by the standards development process.

- **DocumentXmlName**

  A meaningful name for the **Document**, in upper camel case form, suitable for use as the root XML element name in a message using a template. May be, but not required to be, identical with the DocumentName.

- **DocumentTemplate**

  A TemplateName, of the Template used for the Document.

- **DocumentModuleList**
  - An ordered list of **DocumentModuleListEntry**
  - This list is in the same order as, and has an entry for each entry in, the **TemplateSlotList** of the Template specified by **DocumentTemplate.**
- **DetailMaxOccurs**
  - The Maximum number of times the detail area can repeat.
  - The value here must be equal to the corresponding value in the specified **Template**, or a “hardening” of it (e.g. Un-Bounded in the **Template**, and 5 here).
- **ResponsibleSubCommittee**

  Designator for the ASC-X12 Sub-Committee with primary responsibility for maintenance.

DocumentModuleListEntry

- **DocumentModuleXmlName**
  - A meaningful name for the **Module** (when used here), in upper camel case form, suitable for use as a XML element name in a message.
  - This is used to disambiguate the situation where a single defined **Module** is used for two purposes in a single **Document**
- **ContextCategoryValueList**
  - Un-Ordered List of **ContextCategoryValuePair**
- **RequirementsFlag**
  - The value here must be equal to the corresponding value in the specified **Template**, or a “hardening” of it (e.g. Optional in the Template, and Mandatory here).
  - The value must also be compatible with the **MinOccurs** and **MaxOccurs** values in this **DocumentModuleListEntry**

- **ModuleMinOccurs**
  - The value here must be equal to the corresponding value in the specified **Template**, or a “hardening” of it (e.g. 1 in the **Template**, and 2 here).
  - The value must also be compatible with the **RequirementsFlag** and **ModuleMaxOccurs** values in this **DocumentModuleListEntry**
- **ModuleMaxOccurs**
- The value here must be equal to the corresponding value in the specified Template, or a “hardening” of it (e.g. Un-Bounded in the Template, and 7 here).
- The value must also be compatible with the RequirementsFlag and ModuleMinOccurs values in this DocumentModuleListEntry

ContextCategoryValuePair

- ContextCategory
- ContextCategoryValue

6.3 Template

Definition: A document "skeleton" fulfilling a single purpose in a particular business process

Use: Is the basis for defining document schemas for multiple business contexts.

Properties:

- Represents an exchange of data that fulfills a single purpose in a business process
- Somewhat abstract semantics (not entirely neutral semantically, but not fully specified by a specific context either)

Composed of:

- An ordered list of "slots" representing places in a completed schema that would be filled by a module.
- Each slot fufills a function in the Business Process being served.
- A single set of adjacent slots can be designated as a “Detail”, to be repeated as a unit in a Document constructed from a Template.

Relates to/similar to (other specifications):

- Can be identified with ebXML/CEFACT business process modeling.

Template

- **TemplateName**

  A descriptive name for the **Template**, for consumption my humans, used as the primary key for **Templates** in the DISA database. Maintained as unique by the standards development process.

- **TemplateXmlName**

  A meaningful name for the **Template**, in upper camel case form, suitable for use as the root XML element name in a message using a **Template**. May be, but not required to be, identical with the TemplateName.

- **TemplateFamily**

  Identification of the Business Process to which this **Template** applies.

- **BusinessProcess**

  Identification of the Business Process to which this **Template** applies.

- **BusinessProcessFamily**

  Identification of the family of Business Processes to which this **Template** applies.

- **BusinessProcessSubFamily**

  Further identification within the specific **BusinessProcessFamily** to which this Template applies.

- **TemplateDescription**
  - text-paragraph

  Describes general business purpose filled by a message using a **Template**. May also describe the buisness purpose fufilled by sending/receiving the message, or the circumstances surrounding the generation of the message.

- **TriggeringEventDescription**
  - Text-paragraph

  This describes the event in the business process being served that triggers the need to generate a message using the Template. This description may also include the range of expected responses to receipt of the generated message.

- **DetailMaxOccurs**

  The Maximum number of times the **TemplateSlots designated as** detail can repeat as a group.

- **TemplateSlotList**
  - Ordered list of **TemplateSlotListEntry**
  - Matched usage/requirement list
  - This list must contain at least one entry, in almost all cases will contain several entries
  - This list is the main purpose of the Template.
  - Each entry in the list must serve a different and distinct functional purpose in the template.

- **ResponsibleSubCommittee**

  Designator for the ASC-X12 Subcommittee with primary responsibility for maintenance.

TriggeringEventDescription

- Text-paragraph
- Describes the event in the business process being served that triggers the need to generate a message using the template. This description may also include the range of expected responses to receipt of the generated message.

TemplateSlotListEntry

- **TemplateSlotName**
  - Unique in parent **TemplateSlotList**
- **TemplateSlotPurpose**
  - Text-paragraph
  - Description of purpose served by **Modules** filling slot
- **TemplateSlotDetailFlag**

- **TemplateSlotModuleList**
  - Un-Ordered list of **TemplateSlotModuleListEntry**

TemplateSlotModuleListEntry

- **ModuleName**
- **ModuleSlotXmlNname**
  - A meaningful name for the **Module** (when used here), in upper camel case form, suitable for use as a XML element name in a message.
  - This is used to disambiguate the situation where a single defined **Module** is used for two purposes in a single **Template**
- **ContextCategoryValueList**
- **DetailFlag**
- **RequirementsFlag**

  The value must be compatiable with the **MinOccurs** and **MaxOccurs** values in this **TemplateSlotModuleListEntry**
- **ModuleMinOccurs**

  The value must be compatiable with the **RequirementsFlag** and **ModuleMaxOccurs** values in this **TemplateSlotModuleListEntry**
- **ModuleMaxOccurs**

  The value must be compatiable with the **RequirementsFlag** and **ModuleNinOccurs** values in this **TemplateSlotModuleListEntry**

ContextCategoryValueList

- **Un-Ordered List of ContextCategoryValuePair**

ConextCategoryValuePair

- **ContextCategory**
- **ContextCategoryValue**

6.4 Module

Definition: A set of related data that serves a specific purpose in a business document (Template).

Use: Fills a slot in a Template.

Properties:

- Answers a particular Semantic Question within the Business Process (e.g. Who/What/When/Where/Why)
- Re-usable within Templates
- May have the same contents as other modules.
- Unique Semantic Identity
- Semantic uniqueness
- Context specific semantics (concrete as opposed to abstract).

Composed of either:

- One-or-more Blocks and/or Assemblies arranged as a list
- Two-or-more Blocks and/or Assemblies arranged as a hierarchy

Relates to/similar to (other specifications):

- ebXML CEFACT Aggregate Business Information Entity
- Loops in ASC-X12 Documents, identified by code list values in the first segment of the loop

Module

- **ModuleName**

  A descriptive name for the **Module**, for consumption my humans, used as the primary key for **Modules** in the DISA database. Maintained as unique by the standards development process.

- **ModuleXmlName**
  A meaningful name for the **Module**, in upper camel case form, suitable for use as a XML element name in a message. May be, but not required to be, identical with the **ModuleName**.

- **ModuleDescription**
  - Text-paragraph

- **ModuleNode**

- **ResponsibleSubCommittee**

  Designator for the ASC-X12 Subcommittee with primary responsibility for maintenance.

ModuleNode

- **ModuleNodeName**

  A descriptive name for the **ModuleNode**, for consumption my humans. Maintained as unique, within this **Module** and its contents, by the standards development process.

- **ModuleNodeXmlName**

  A meaningful name for the **ModuleNode**, in upper camel case form, suitable for use as a XML element name in a message. May be, but not required to be, identical with the **ModuleNodeName**.

- **AssemblyName-or-BlockName-or-ModuleNodeList**

- **RequirementsFlag**
  The value must be compatible with the **MinOccurs** and **MaxOccurs** values in this **ModuleNode**

- **MinOccurs**

- **MaxOccurs**

ModuleNodeList

- **ModuleNodeListName**

- **ModuleNodeListXmlName**

- **Ordered list of ModuleNode**

6.5 Assembly

Definition : A group of related nouns (person/place/thing/event/purpose).

Use: For conveniently re-using related groups of blocks.

Properties:

- Re-usable within Modules
- Has a unique set of Blocks/Assemblies, though it may share Blocks with other similar Assemblies.
- May consist of a set of Blocks that is a subset of the Blocks of contained in another Assembly
- Unique Semantic Identity
- Semantic uniqueness
- Abstract semantics (context independent)
- May be similar to other Assemblies

Composed of either:

- Two-or-more Blocks
- One-or-more Blocks and one-or-more other Assemblies

Relates to/similar to (other specifications):

- ASC X12 segment groups (though segment groups are not named or stored as such in the X12 dictionary)
- EbXML/CEFACT Aggregate Core Components

Assembly

- **AssemblyName**

  A descriptive name for the **Assembly**, for consumption my humans, used as the primary key for **Assemblies** in the DISA database. Maintained as unique by the standards development process.

- **AssemblyXmlName**

  A meaningful name for the **Assembly**, in upper camel case form, suitable for use as a XML element name in a message. May be, but not required to be, identical with the **AssemblyName**.

- **AssemblyList**
- Ordered list of **AssemblyListEntry**

AssemblyListEntry

- **BlockName-or-AssemblyName**
- **RequirementsFlag**

  The value must be compatiable with the **MinOccurs** and **MaxOccurs** values in this **AssemblyListEntry**

- **MinOccurs**

  The value must also be compatiable with the **RequirementsFlag** and **MaxOccurs** values in this **AssemblyListEntry**

- **MaxOccurs**

  The value must also be compatiable with the **RequirementsFlag** and **MinOccurs** values in this **AssemblyListEntry**

6.6 Block

Definition: Completely (for intended business use) describes a single noun - person, place, thing, event, or purpose.

Use: Used for describing a single person, place, thing, event, or purpose

Properties:

- Describes something that can be named by a single noun
- Concise
- Must have identity information
- May have characteristics information
- Unique Semantic Identity
- Semantic uniqueness
- Abstract semantics (context independent)
- Re-usable within assemblies or modules
- May be similar to other blocks, as nouns are similar to other nouns.
- A block has a unique set of components, though it may share components with other similar blocks.
- A block may consist of a set of components that is a subset of the components of contained in another block.

Composed of:

- Two-or-more Components

Relates to/similar to (other specifications):

- X12 segments, partial or complete
- EbXML/CEFACT Aggregate Core Components

Block

- **BlockName**

  A descriptive name for the **Block**, for consumption my humans, used as the primary key for **Blocks** in the DISA database. Maintained as unique by the standards development process.

- **BlockXMLname**

  A meaningful name for the **Block**, in upper camel case form, suitable for use as a XML element name in a message. May be, but not required to be, identical with the **BlockName**.

- **BlockType**
- One of Person/Place/Thing/Event
- **ComponentList**
- Ordered list of **ComponentListEntry**

ComponentListEntry

- **ComponentName**
- **IdentificationCharectaristicFlag**
- **RequirementsFlag**

6.7 Component

Definition: A single semantic unit of information.

Use: Identification or a characterization within block

Properties:

- Unique Semantic Identity
- Semantic uniqueness
- Abstract semantics (context independent)
- Re-usable within blocks
- Has a unique set of primitives

Composed of either

- One value with a specified datatype representation
- Two values, each with a datatype representation. The first value is qualified by the second (e.g. a currency amount and a currency type, or a weight and a unit of measure).

Relates to/similar to (other specifications):

- EbXML UN/CEFACT basic core component
- ASC X12 Data Element

Component

- **ComponentName**

  A descriptive name for the Component, for consumption my humans, used as the primary key for Components in the DISA database. Maintained as unique by the standards development process.

- **ComponentXMLname**

  A meaningful name for the Component, in upper camel case form, suitable for use as a XML element name in a message. May be, but not required to be, identical with the **ComponentName**.

- **ComponentContent**

ComponentContent

- **SingleComponentContent-or-PairedComponentContent**

SingleComponentContent

- **ComponentRepresentationType**
- **ComponentMinLength**
- **ComponentMaxLength**

PairedComponentContent

- **First SingleComponentContent**
- **Second SingleComponentContent**
- **Second Component RequirementsFlag**

ComponentRepresentationType

6.8 Primitive

Definition: A Primitive is a unique semantic entity, having a unique semantic identifier.

Use: Identify like entities.

Properties:

- Unlike a Component, a Primitive may be identified indirectly through a code list value of a Component.
- Each such Component may identify the Primitive using a different code list value.
- While both Components and Primitives represent unique semantic entities, a Primitive conveys only its identity, whereas a Component conveys both an identity and a value."

For example, "Federal Tax Identification Number" is a unique semantic identity. It might be used as a Component, whose value represents an individual social security number (E.g., X12 DE 325 Tax Identification Number {NOTE: DE 325 is not an exact equivalent of Federal Tax Identification Number. The example is intended to show the possible use of FTIN as a Component).

Or it may be used as a Primitive associated with a Component such as 'Reference Identifier', and represented by an associated code list value of 'TJ' (E.g., see X12 DE 128, 'Reference Identifier Qualifier). And the primitive might also occur in another Component, such as 'Tax Identifier' with an associated code list value of '01'.

Primitive

- **PrimitiveName**

  A descriptive name for the **Primitive**, for consumption my humans, used as the primary key for **Primitives** in the DISA database. Maintained as unique by the standards development process.

- **PrimitiveDescription**
  - text-paragraph

  This is further descriptive information about the **Primitive**. This may include, its purpose, typical uses, and common synonyms.

6.9 User View of the Secretariat Database

This section describes a user-view of the “Database” maintained by DISA for ASC-X12’s XML activities. No inference on actual structure of data or the tools used to provide the “Database”

The notion of user-view describes both the data that must be supported, but also capabilities needed to make best use of the information. In particular, two fundamental needs must be met:

1) Providing open-access to Standards in XML produced by ASC-X12

2) Facilitating the standards development and maintenance activities.

At the highest level the database has Dictionaries and Lists. Dictionaries hold the descriptions of the X12-XML standards. Lists aid users in locating particular standards, for example by similarity in form or purpose.

There are 7 dictionaries, each matching a distinct semantic granularity:

- DocumentDictionary
- TemplateDictionary
- ModuleDictionary
- AssemblyDictionary
- BlockDictionary
- ComponentDictionary
- PrimitiveDictionary

Lists

- SimilarTemplateList

  A list maintained of Templates that fulfill similar business purposes. This list is maintained for comparison during maintenance of existing Templates or development of new Templates. This list is also useful for discovering appropriate Templates to serve a particular business purpose.

7. XML SYNTAX DESIGN

7.1 General

7.1.1 Scope and purpose

This section addresses XML syntax design issues that are common to both the design of XML messages (instance documents) and schemas describing those messages.

7.1.2 Versioning

This report anticipates that ANSI ASC X12 will continue with its current policy of one major and two minor releases each year, and that the whole of the XML syntax standard would be reissued at each release.

The preliminary recommendation for a mechanism to handle versioning is:

- A unique root namespace for each version. An example for version 5010 might be urn:schemas.x12.org/005010/
- Schemas for each release would be accessible on the World Wide Web via URLs that correspond to the namespace of the release.

7.1.3 Internationalization Features

Since the scope of X12's XML standards is primarily the United States, this report does not recommend extensive features to support internationalization. XML 1.0, in Unicode, supports all major national character sets that are likely to be needed. This report recommends using "Oxford English" spellings for names.

7.1.4 Software Processing Considerations

This report proposes taking a fairly neutral position on software processing considerations due to the rapidly evolving nature of XML software. However, there are a few considerations to be noted in this area:

- Related to the discussion of "what constitutes a document", this report recommends that instance documents (supported by the schemas that define them) be kept to a "reasonable" size since many XML parsing APIs load the entire document into memory.
- For ease in processing, this report recommends a maximum level of nesting of elements. We anticipate that this level will be in the neighborhood of ten levels of nesting below the root element of a document.

7.1.5 General Naming Conventions

This report recommends the use of so-called "Oxford English" in the spelling of names. It also recommends, per the ebXML specifications, upper camel case for elements and lower camel case for attributes. Other aspects of naming conventions as specified in the ebXML Technical Architecture V1.0.4, Section 4.3, are also recommended.

7.2 Messages

7.2.1 Scope and Purpose

This section addresses XML syntax design issues relevant to the design of XML messages (instance documents).

7.2.2 Naming Conventions

Naming conventions are not addressed in this version of the report.

7.2.3 Absence of Data and Related Considerations

Absence of data - If an element or attribute does not occur in an instance document, no semantics shall be interpreted from it, i.e. no default values shall be assumed. Nothing can be inferred other than that the creator of the document did not include the element or attribute in the document.

Spaces - Spaces sent as values for string type elements or attributes shall be interpreted as spaces. Leading and trailing spaces should be removed, but are assumed to be significant if they appear. The default whiteSpace facet of XML Schemas, that of preserving white space, is to be used.

Zeroes - A zero appearing in a numeric type element or attribute shall be interpreted as a zero value.

Nullability - In certain cases, it may be desirable to convey that an element has no value (a null value) rather than indicating that it has a value of spaces or that it is not present in a document. In these cases, the originator of the instance document should convey explicitly that an element is null using the null type (e.g. xsi:null="true"), rather than using zero, spaces, or an empty element.

7.2.4 Comments

This report recommends against inserting comments in instance documents on the grounds that the X12 standards are designed for computer-to-computer processing without human intervention.

7.2.5 Elements vs. Attributes

Description: Often it is possible to model a data item as a child element or an attribute.

Benefits of Using Elements

- They are more extensible because attributes can later be added to them without affecting a processing application.
- They can contain other elements. For example, if you want to express a textual description using XHTML tags, this is not possible if description is an attribute.
- They can be repeated. An element may only appear once now, but later you may wish to extend it to appear multiple times. (NOTE: an element can be “bounded” for finite instances of the element or can be “unbounded”)
- You have more control over the rules of their appearance. For example, you can say that a product can either have a number or a productCode child. This is not possible for attributes.
- Their order is significant if specified as part of a sequence, while the order of attributes is not. Obviously, this is only an advantage if you care about the order. (NOTE: cardinality can be captured through the “sequence” grouping of elements)

- When the values are lengthy, elements tend to be more readable than attributes.

Disadvantages of Using Elements

- Elements require start and end tags, so are therefore more verbose. (NOTE: not all elements require a start and end tag – elements can be declared in a single-line i.e.
    - <xs:element name="x12document" type="x12documenttype"/>

Benefits of Using Attributes

- They are less verbose (NOTE: depending on naming convention attributes should not be verbose in schemas, "attribute" names following a naming convention that removes any reference to the localized element it described as this is unnecessary and repetitive. If applying the attribute from within an AttributeGroup then the contextual value of the attribute name should be contained within the attributename..
- Attributes can be added to the instance by specifying default values. Elements cannot (they must appear to receive a default value)
- Attributes are atomic and cannot be extended and its existence should serve to remove any and all possible ambiguity of the element it describes. They are "adjectives" to the element "noun".

Disadvantages of Using Attributes

- Attributes may not be extended by adding children, whereas a complex element may be extended by adding additional child elements or attributes.
- If attributes are to be used in addition to elements for conveying business data, rules are required for specifying when a specific data item shall be an element or an attribute.

Recommendation: Use elements for data that will be produced or consumed by a business application, and attributes for metadata.

7.2.6 Namespaces

Namespaces are more of a concern in designing schemas and are discussed in greater length in that section. In regard to instance documents, this report recommends:

- Minimal namespace prefixes be required in instance documents. Ideally, only the root element, if even that, would require a namespace prefix.
- Explicit namespace references shall not be used at the element or attribute level below the root document element.

7.2.7 Communication Integrity - Envelope, Security, and Related Information

These issues are beyond the scope of this report since it deals primarily with representing business semantics in XML syntax and not with broader implementation issues. This report supports the direction of X12 in recommending use of ebXML specifications wherever appropriate.

7.2.8 Processing Instructions

Description: Processing instructions can be used to pass information to the processing application.

Benefits:

Risks: Processing instructions usually contain information that should normally be included in the document as XML.

Recommendation: Do not use processing instructions in either the schema document or the instance.

7.3 Schema

7.3.1 Scope and Purpose

This section addresses XML syntax design issues that are relevant to the design of schemas describing XML messages

7.3.2 Schema Considerations for Namespaces, Nullability and Related Issues

String type - An empty string type element or attribute satisfies mandatory constraints in XML schema (elements with minOccurs of 1 or mandatory attributes). Therefore, elements or attributes with a type of string that is defined as mandatory shall be defined with a minimum length requirement of 1. An open issue in this report is whether or not to require a pattern of at least one non-space character for such required elements or attributes. To satisfy the requirement for a string element or attribute, XML schema considers any Unicode character to be valid. One space in a string element or attribute is considered valid.

Nullability - An element shall not be marked as nullable if it is mandatory, i.e., minOccurs is one. Conversely any element defined with minOccurs of zero shall be nullable.

7.3.3 Content Models

- **Use of Mixed Content**

  Description: Elements with mixed content are allowed to have both child elements and textual content.

  Benefits: Mixed content is useful for textual descriptions, which may or may not contain markup to indicate emphasis, formatting, etc.

  Risks: The textual content of mixed elements cannot be validated or constrained to any particular data type.

  Recommendation: Do not allow mixed types since they are inappropriate for usage in documents designed solely for data exchange.

- **Wildcards**

  Description: XML Schema allows wildcards to be specified in content models (using <any>) and attribute declarations (using <anyAttribute>).

  Benefits: Wildcards allow a content model (or attribute list) to be highly flexible, making them more extensible.

  Risks: Wildcards can sometimes allow invalid data (e.g., a product with two sizes when only one is allowed), so they should generally be used only for elements in other namespaces.

  Recommendation: Disallow use of wildcards.

- **Abstract Types**

  Description: Abstract types allow use of complex types in such a way that a single element name can be used to represent various types in an XML document instance. Abstract types are complex types that act as “templates” that cannot be directly used in an XML document instance. In order to use an abstract type, a derived type must be used to represent the abstract type in an XML document instance.

  Benefits:

  - Extensibility - other schemas can use the abstract type as the basis for derived types.
  - Abstract types provide a mechanism for enforcing "at least one of" business constraints as a requirement for a person to have at least one identifier present, but either name or an ID number might be acceptable. By requiring an abstract element in the schema and having two concrete elements that could satisfy it, this functionality is supported.

  Risks: It is possible that a processing application (such as a data translation product) may not be able to easily handle this technique. That is, a processing application may be need to be configured to recognize an element named *EmployeeAddress* as always having a single, static type *(such as UnitedStatesAddressType)* rather than a type that can vary depending on the XML document instance.

  Recommendation: Abstract types should generally not be used because they contribute to a degree of uncertainty about what an XML document instance will look like, i.e., they contribute to randomness. They may be used in specific circumstances where an "at least one of" constraint is required.

- **Use of Groups**

  Description: XML Schema allows fragments of content models to be named and referenced from multiple complex types. It is also possible to create attribute groups that can be reused in multiple complex types.

  Benefits: Use of groups promotes reuse.

  Risks: Occasionally, too much reuse can complicate maintenance. In addition, the functionality offered by groups is very similar to that offered by types. The unnecessary use of too many schema features when only a few features would be sufficient can hinder understandability.

  Recommendation: This report makes a preliminary recommendation to avoid use of groups and instead try to use types as much as possible.

  (NOTE: In an effort to achieve both reusability and interoperability, the declaration of groups should serve this purpose. Through Schema design it is possible to combine both the localized features and global constructs using “complexType” and “simpleType” components.

- **Substitution Groups**

  Description: XML Schema allows for elements to substitute for other elements by defining substitution groups. An element can be declared to be a substitute for another element, the "head" element, allowing the new element to appear anywhere the head element may appear.

  Benefits: Substitution groups result in flexible, extensible types.

  They can simplify content models, by specifying only the "head" element in the content model and using substitution to allow all the possibilities.

Risks: Excessive flexibility. Another schema author can significantly alter a type by declaring substitution elements.

Recommendation: Prohibit substitution groups.

- **Group Redefinition**

Description: XML Schema allows a schema author to redefine the types or groups of another schema document.

Benefits: Redefinition is useful for making small changes to an existing schema document.

Risks: Because the redefined components replace the original components, they can have adverse effects on other components defined in the original schema document.

Redefinition is underspecified in the XML Schema recommendation, and it is likely that different processors treat redefinitions slightly differently.

Recommendation: Do not use redefinition.

7.3.4 Types

- **Anonymous vs. Named Types**

Description: XML Schema allows for types (simple and complex) to be named (and defined globally) or anonymous (and defined locally).

Benefits of Named Types

- Named types may be defined once and used many times. This encourages reuse and consistency, simplifies maintenance, and reduces the size of schemas.
- Named types can also make the schema document more readable, when the type definitions are complex.
- Named types can be redefined and have other types derived from them. This increases their flexibility and extensibility.

Benefits of Anonymous Types

- They are slightly less verbose.
- They can be more readable when they are relatively simple. It is sometimes desirable to have the definition of the type right there with the element or attribute declaration.

Recommendation: Always use named types.

- **Built-In Simple Types**

Description: XML Schema has 44 built-in data types, covering numbers, strings, dates and times, XML 1.0 types such as NMTOKENS and ID, boolean, anyURI, and other common types. These types have specific lexical formats, e.g., a date must be CCYY-MM-DD.

Benefits

- Using the built-in types increases interoperability with other XML applications.
- Values of built-in types are automatically validated by the processor, e.g., a date cannot be April 31.

Risks: The built-in types may not have the lexical formats that you have traditionally used.

Recommendation: Use only XML Schema built-in data types. Further, we shall use a subset of the full types, with that subset to be defined in development of X12's XML equivalent of X12.6.

- **Type Redefinition**

  Description: XML Schema allows a schema author to redefine the types or groups of another schema document.

  Benefits: Redefinition is useful for making small changes to an existing schema document.

  Risks: Because the redefined components replace the original components, they can have adverse effects on other components defined in the original schema document.

  Redefinition is underspecified in the XML Schema recommendation, and it is likely that different processors treat redefinitions slightly differently.

  Recommendation: Do not use redefinition.

- **Type Derivation**

  Description: XML Schema allows a type to be derived from another type (its base type), either by extension or restriction. Extension adds attributes, and adds elements to the end of the content model of the base type. Restriction limits a base type to a more restrictive set of valid values.

  Benefits: Restriction allows more refined data types to be created which allows stricter validation in specific cases.

  Extension allows the base type to be used with additional extensions, which encourages reuse.

  Risks: Derived types can be used for type substitution (see "Type Substitution"). If type substitution is not to be allowed, the base complex type should have the block attribute specified.

  Recommendation: Allow type derivation.

- **Type Substitution**

  Description: Type substitution allows for the use of derived types in an instance document. If an element is declared to be of a base type, the element may appear in the instance having any type that is derived from the base type. To do this, it must use the xsi:type attribute to identify the derived type to which it conforms.

  Benefits: Type substitution allows an element to have one of several types in an instance document. For example, a generic address type can be created, with extensions for specific countries, e.g. UKAddressType, USAddressType, etc. The address element can then appear in the instance using whichever of these types is appropriate.

  Risks:

  - Can lead to problems in processing by applications when a type specified in an instance document overrides the type specified in a schema.
  - If you do not intend to allow flexibility of the type of an element, you should not allow type substitution.

  Recommendation: Disallow type substitution.

7.3.5 Local vs. Global Declarations

Description: Elements and attributes can be either declared globally or locally. Globally declared elements and attributes appear at the top level of the schema (with xsd:schema as their parent). Locally declared elements and attributes are declared entirely within a complex type.

Benefits of Global Declarations

- They can be reused in many complex types.
- A globally declared element can be the root element of the instance document for validation purposes (a locally declared element cannot.)
- Global element declarations can participate in substitution groups; local element declarations cannot.

Benefits of Local Declarations

There can be many locally declared elements with the same name but different types and/or different default or fixed values. For example, it is possible to have a "title" element that is a child of "person", which has the valid values "Mr.", "Mrs." and "Ms.". Another element named "title" that is a child of "book" can have free-form text. Because global element declarations are unique by name, there can only be one globally declared element named "title".

Recommendation: Declare elements and attributes locally, except for the root element.

7.3.6 Use of Default/Fixed Values

Description: XML Schema allows fixed or default values to be specified for elements and attributes.

Benefits: Additional information can be added to the instance without requiring the instance author to specify it.

Risks: When a schema is not present, the default or fixed value cannot be filled in.

Recommendation: Disallow use of default and fixed values.

NOTE: There are cases where the use of default values has ”value”. In the event X12 wants to reconsider this recommendation, this section from the primer provides a good explanation.

Default values of both attributes and elements are declared using the default attribute, although this attribute has a slightly different consequence in each case. When an attribute is declared with a default value, the value of the attribute is whatever value appears as the attribute's value in an instance document; if the attribute does not appear in the instance document, the schema processor provides the attribute with a value equal to that of the default attribute. Note that default values for attributes only make sense if the attributes themselves are optional, and so it is an error to specify both a default value and anything other than a value of optional for use.

The schema processor treats defaulted elements slightly differently. When an element is declared with a default value, the value of the element is whatever value appears as the element's content in the instance document; if the element appears without any content, the schema processor provides the element with a value equal to that of the default attribute. However, if the element does not appear in the instance document, the schema processor does not provide the element at all. In summary, the differences between element and attribute defaults can be stated as: Default attribute values apply when attributes are missing, and default element values apply when elements are empty.

7.3.7 Keys and Uniqueness

Description: Sometimes it is desirable to associate information within an XML document with other information in the document when those items of information may or may not already be implicitly related by being siblings under the same parent element. This can be done strictly at the level of business semantics by defining elements or attributes to link the information items through a common reference. Schema provides several mechanisms to do this and enforce the validity of such links at the XML parser level.

ID/IDREF

This concept originated with DTDs, and is also used in XML Schema. In this technique, an ID value is used by an XML processor to associate information within an XML document. This allows information to be separated within an XML document, yet still be associated during processing. A parser can verify that there is a corresponding ID value in an XML document instance for a given IDREF value

Benefits of ID/IDREF Technique:

- It allows information in an XML document instance to be linked during processing by a processing application
- It ensures validation of the associations by an XML processor — i.e. that there is a corresponding ID value for an IDREF value — without defining extra processing (i.e., it is “built in” to an XML processor).

Risks:

- It does not allow links between entities in an XML document instance to be recognized by an XML processor
- An ID value must be unique within an XML document. This means that in the above example, there could never be the same ID value for a customer and an invoice. This requirement is not realistic, as the ID values for two different entities may not only be of the same structure but may also have the same values in certain cases.
- An ID value must begin with a letter and cannot contain whitespace or non-alphanumeric characters (except for underscore).

KEY/KEYREF

This concept originated with XML Schema. Unlike the ID/IDREF technique, this technique allows links between entities in an XML document instance to be recognized by an XML processor. It also allows ID values to be repeated within XML documents without yielding an error from an XML processor (as with the uniqueness technique, discussed below). Additionally, it adds the requirement that the element or attribute specified in the field element of a constraint declaration must always appear in an XML document instance.

Benefits of KEY/KEYREF Technique:

- It allows links between entities in an XML document instance to be recognized by a schema processor
- It allows ID values to be repeated within XML documents without yielding an error from a schema processor
- ID values do not have the format constraints that were imposed in the ID/IDREF technique; that is, an ID value may be of any datatype

Risks of KEY/KEYREF Technique:

- Constraint declaration names must be unique within an XML document instance, regardless of namespace - this applies for externally referenced schemas as well.
- A schema processor may not detect an incorrect XPath expression in either the selector or field element of the constraint declaration. This can cause the constraint to not be enforced, resulting in potential violations of the key constraint.

XLink/XPointer

This technique utilizes two relatively new XML concepts to link entities within XML document instances. It allows links to be specified either within an XML the same document instance as the entities being linked (through use of a “simple” link or “extended” link), or outside of it in a different XML document instance (through use of an “extended” link). Extended links can be very useful in cases where an XML document instance cannot be updated; they also allow linking information to be centralized in one place if required.

Benefits of XLink/XPointer Technique:

- It allows links between entities in an XML document instance to be recognized by a schema processor (although the schema processor must be XLink- and XPointer-aware)
- The use of XLink constructs allow the links to be given special handling in an XLink-aware processor. For instance, additional XLink constructs may be used to allow links to be highlighted for selection
- Extended links can be specified either in the same XML document instance as the entities that they link or outside of it in a different XML document instance

Risks of XLink/XPointer Technique:

This technique has several disadvantages:

- An ID value must be unique within an XML document. For more information, see ID/IDREF section above.
- An ID value must begin with a letter and cannot contain whitespace or non-alphanumeric characters (except for underscore). For more information, see ID/IDREF section above.
- Since the XLink and XPointer standards are both very new (XLink became a W3C Recommendation in June 2001 and XPointer is currently a Candidate Recommendation), there is currently very little XML processor support for them
- Use of extended links requires a fair amount of additional information to be specified for each entity that is being linked; e.g., xlink:locator elements, role attributes, xlink:arc elements, etc.

Recommendation

The recommendation for the appropriate syntax technique must be consistent with the functionality defined by the architecture, and this aspect of the architecture is not as yet fully defined. It is discussed somewhat in Section 6.3, but not sufficiently for purposes of specifying a final syntax recommendation. However, the preliminary recommendations are:

- The KEY/KEYREF technique should be used to enforce links between entities in an XML document instance.
- The uniqueness technique should be used to enforce uniqueness when the element or attribute specified in the field element is not mandatory. The KEY technique (without KEYREF) should be used to enforce uniqueness when the element or attribute specified in the field element is mandatory.

Extreme caution should be applied in each of the above techniques to ensure that the XPath expression that is specified is correct, so that the uniqueness constraint can be properly enforced.

It is also recommended that the following situation never be allowed:

- Uniqueness must be enforced AND
- Links are required AND
- The element or attribute specified in the field element is not mandatory

There is no technique that is available to handle the above situation, because in the KEY/KEYREF technique the element or attribute specified in the field element must appear in the XML instance document. For this reason, it is recommended that in all cases where links are required, the element or attribute specified in the field element be declared as mandatory.

Special attention should also be given to the fact that constraint declaration names must be unique within an XML document instance.

This report recommends against use of XPointer and XLink since they are still relatively immature.

7.3.8 Annotations and Notations

- **Annotations**

  Description: XML Schema allows schema components to be annotated using the <annotation> element. The annotation element can contain one or more <documentation> or <appinfo> elements that can themselves have any attributes and contain any text or child elements.

  Benefits:

  - An annotation adds descriptive information that makes a schema component easier to understand.
  - Structured annotations are machine- as well as human-readable, allowing them to be used by applications or to generate specification guides.

  Risks: Excessively large or repetitive annotations actually decrease the readability of the schema document, and slow down validation.

  Recommendation: Use annotations for all type definitions, and define a standard format and structure for those annotations that is consistent with the metadata defined in section 7. Do not use XML comments in schemas.

- **Notations**

  Description: Notations can be used to specify the type of a file (for example, a graphics image) that is related to an XML document via an external entity.

  Benefits: Notations can be useful for identifying the type of a file.

  Risks:

  - There is no standardized way to process notations.
  - Generally, notations are unnecessary because the processing application already understands the type of a related file.

  Recommendation: Do not use notations.

- **Documentation**

Description: W3C Schema introduces a standard *<documentation>* element that can be used to enclose comments. The DTD-style comment technique is also supported in W3C Schema. The <documentation> element can have two attributes:

- A "source" attribute that contains a URL to a file containing supplemental information
- An xml:lang attribute that specifies the language in which the documentation is written.

Benefits of Using <documentation> Element: Use of the <documentation> element to add comments to a schema rather than the DTD-based approach is advantageous because it allows the comments to be processed by a processing application or program such as a stylesheet. Once this is done, there is no limit to what can be done with the extracted comments.

Risks of Using <documentation> Element:

There are no risks to using this technique.

Recommendation: The *<documentation>* element SHOULD be used for comments. The DTD-based comment technique SHOULD NOT be used.

7.3.9 Processing Instructions from Schema Level <APPINFO>

Description: The <appinfo> element is the XML Schema equivalent of the processing instruction. Like processing instructions, the <appinfo> element offers a place in which to provide additional information that can be passed to a processing application by an XML parser.

Benefits of <appinfo> Element

The <appinfo> element can be very useful for passing processing commands or other types of supplemental information to a processing application.

Risks of Using <appinfo> Element

The use of the <appinfo> element is considered highly risky at this time, due to the immaturity of XML schema processors available. There is no guarantee that a given XML schema processor will properly pass the processing instructions to an application, or, if it does, that an application will be able to accept them or handle them properly.

Recommendation

The <appinfo> element MUST NOT be used.

7.3.10 Length

In X12 syntax standards the typical pattern regarding data maintenance for the length of data elements is that they have consistently gotten longer. XML schema does not require a maximum length. This report recommends not using fixed or maximum length except in the case of coded values, where appropriate.

7.3.11 Namespaces

XML schemas allow for instance documents that have zero, one or many namespaces. The namespace of an instance document is specified as a "target namespace" of the schema document.

Benefits of Using No Namespace

- It is simpler: there are fewer design decisions to be made, and instance documents are more readable.
- Allows for use of "chameleon" design. In other words, when a schema that has no targetNamespace is included in another schema, the components within the included schema taken on the same namespace as the including schema - therefore, they are "chameleons".

Disadvantages of Using No Namespace

- Most XML processors cache schema components for validation by namespaces. If no namespace is used, there will be no caching. Processing is therefore much less efficient without namespaces.
- Most current XML schema designers are using namespaces, so not using them will go against convention and may likely cause several complications.
- More work is required to avoid result name collision, i.e. if there is an element in the included schema that has the same name as an element in the including schema, an error will result.

Benefits of Using One Namespace

- The vocabulary of an instance document is immediately recognizable.
- One namespace declaration does not significantly complicate an instance document.

Disadvantages Using One Namespace

- The size of a single namespace for the whole of X12's XML implementation may be rather large, even when a particular instance document uses a limited number of components from the namespace. Processing efficiency is reduced if a single, large namespace is used.

Benefits of Using Multiple Namespaces

- Namespaces can be used to categorize components.
- Helps to avoid name collision.
- It is easy to distinguish "core components" from extensions.

Disadvantages of Using Multiple Namespaces

- Multiple namespaces lead to a more complex design.

Recommendation

The preliminary recommendation is to use a tiered, hierarchical approach to namespaces. One core namespace could include components to all functional X12 subcommittees. Each functional subcommittee (or other logical grouping) could have a unique namespace that imports the common namespace. All instance document schemas related to the subcommittee (or other logical grouping) could use that subcommittee namespace. Each instance document schema could declare its own unique target namespace.

1997

8. SUMMARY OF PROPOSED DESIGN RULES

As noted earlier, this report envisions that a new ANSI ASC X12 standard would be created based on the work of this Reference Model. That standard would in essence be the XML equivalent of the current X12 syntax X12.6 standard. Just as the X12.6 standard deals with general metadata and definitions as well as syntax issues, the new XML based standard would do the same. Similarly, as there are X12 Design Rules and Guidelines that enforce X12.6, there would be Design Rules and Guidelines that enforce the new XML syntax work. Like X12.6, they would deal with metadata and definitions. However, due to the manner in which this report proposes that the development process be conducted, they would not deal with syntax issues. This section presents an outline of what the design rules might incorporate and how they might be used.

This report envisions that the primary work of X12's Technical Assessment Subcommittee and the industry oriented subcommittees would be to define standards for XML data content and semantics according to the architectural framework described in this Reference Model. These standards would be stored in a database maintained by X12's secretariat in much the same way that the current X12 standards are stored in DISA's database. What is different in the XML environment is that XML syntax output, in the form of W3C XML Schemas, would be created from that database. The database contains the "source normative form" of the standard, while eventually automated procedures would create the target or implementation form of the standard. This has no direct equivalent in the X12 syntax environment.

In addition, most aspects of syntax that are defined in X12.6 are described by recommendations such as XML and XML Schema that are developed by the World Wide Web Consortium. The XML equivalent of X12.6 would describe how the database content would be represented in these XML syntaxes based on the recommendations of the XML syntax design presented in Section 7. In short, it would specify the target implementation form of the standard. Since this aspect of the standard, from an X12 perspective, would be applicable to producing database output, it would not be enforced by TAS and there would be no need for corresponding design rules. The standard itself would provide sufficient guidance to the secretariat to develop automated procedures to produce the XML schemas. Organizations that wished to implement their XML syntax components following X12's design could also use the X12.6 equivalent as a specification, but again there would be no need for formal design rules since there would be no enforcement activity.

Therefore, the design rules would deal with metadata and definitions, enforcing the corresponding elements from the XML version of X12.6. They would be based on the high level architecture described in Section 3 and the metadata and definitions described in Section 6. This set of rules would be used by X12's Technical Assessment Subcommittee as it reviews new and modified XML standards. It would incorporate rules such as:

A template slot must specify a MinOccurs value

A module may not contain another module

A block must contain at least one identity component

A component may not contain another component

This report envisions that such design rules would be developed in conjunction with the XML equivalent of X12.6).

(To clarify one minor point of potential confusion, the XML syntax aspect of the XML equivalent of X12.6 would not, strictly speaking, constitute "production rules". Production rules are generally applicable when there are defined source and target syntaxes, and rules are defined to describe the transformation from one syntax to another. In the case of this report, the source form is expressed in logical form only without a specific implementation syntax. Therefore production rules are not applicable.)

9. CONTROL STRUCTURES

9.1 External Control Structures

Control information (analogous to the information provided in the ISA and GS segments) may be included in envelope structures outside an ASC X12 XML document (the root element of an XML instance document as defined in this specification). This is information that is independent of the type of business document contained.

Preliminary requirements for information in the external control structure:

- Organizational routing (e.g., VAN ID)
- Internal routing
- Unique ID (detection of missing/duplicate documents)
- Information required for document processing

Examples of such information include:

- Sender/Receiver ID (e.g., ISA sender/receiver IDs)
- Internal routing (e.g., GS sender/receiver/application IDs)
- External routing IDs (e.g., e-mail address of sender/recipient)
- Control/sequence numbers (e.g., ISA/GS/ST control number)
- Date/time

9.2 Document Control Structure

ASC X12 XML documents have a document control structure, analogous to X12 ST/SE segments. It shall be a well-formed XML document with a single root element.

Preliminary requirements for information in the document control structure:

- Identification of the message type of the document
- Demarcation of the beginning and ending of the document (e.g., in a data stream)
- Internal routing (if not provided by external control structure)

9.3 Internal Control Structures

At the beginning of the ASC X12 XML document, related business relationship information is given. This contains the type of information found in the X12 BEG-like segments. This may be implemented using the first slot of the template or at the level of the root element (e.g., as attributes). Application control and document identification information is included at this level. Examples include:

- Unique business document ID information (e.g., date, time, instance number)
- Related business relationship information (e.g., purchase order number or contract number related to an invoice or advance ship notice transaction)

ANNEX A: DEFINITIONS

Abstract types	Allow use of complex types in such a way that a single element name can be used to represent various types in an XML document instance
Annotation	Information for human and/or mechanical consumers. The interpretation of such information is not defined in the XML Schema specifications. The annotation element can contain one or more <documentation> or <appinfo> elements.
AnyAttribute	XML Schema allows wildcards to be specified in attribute declarations
AnyElement	XML Schema allows wildcards to be specified in content models (using <any>)
Attribute	A name=”value” field within an XML element, providing information associated with that XML element
Attribute Declaration	An attribute declaration is an association between a name and a simple type definition, together with occurrence information and (optionally) a default value. The association is either global, or local to its containing complex type definition. Attribute declarations contribute to validation as part of complex type definition validation, when their occurrence, defaults and type components are checked against an attribute information item with a matching name and namespace
Attribute Group	A set of attribute declarations, enabling re-use of the same set in several complex type definitions
Attribute Group Definition	An attribute group definition is an association between a name and a set of attribute declarations, enabling re-use of the same set in several complex type definitions
Built-in Datatypes	**Built-in** datatypes are those which are defined either in the XML Schema specification (as primitive types) or in this specification, and can be either primitive or derived
Character set	The encoding method for the data values of the document, based on Unicode format.
Complex Type	An XML element type that allows nested elements in their content and may carry attributes
Complex Type Definition	A complex type definition is a set of attribute declarations and a content type, applicable to the attributes and children of an element information item respectively. The content type may require the children to contain neither element nor character information items (that is, to be empty), to be a string that belongs to a particular simple type or to contain a sequence of element information items that conforms to a particular model group, with or without character information items as well.
Complex type extension	Extension adds attributes, and adds elements to the end of the content model of the base type.
Complex type restriction	Restriction limits a base type to a more restrictive set of valid values.
Core Component	Refers to common elements that are generally useful, reusable in different contexts.
Datatype	A **datatype** is a 3-tuple, consisting of a) a set of distinct values, called its value space, b) a set of lexical representations, called its lexical space, and c) a set of facets that characterize properties of the value space, individual values or lexical items.

Default attribute values	Data values that imply a default value if they do not explicitly appear in the XML instance document
Derived Data Types	**Derived** datatypes are those that are defined in terms of other datatypes. A datatype is said to be derived by **restriction** from another datatype when values for zero or more constraining facets are specified that serve to constrain its value space and/or its lexical space to a subset of those of its base type. Every datatype that is derived by **restriction** is defined in terms of an existing datatype, referred to as its **base type**. **base type**s can be either primitive or derived
Element	A fundamental unit of XML information, which has an element name, optional attributes, optional data value, and an associated type definition. Elements may be nested, one inside another.
Element Declaration	An element declaration is an association of a name with a type definition, either simple or complex, an (optional) default value and a (possibly empty) set of identity-constraint definitions.
Facet	A **facet** is a single defining aspect of a value space. Generally speaking, each facet characterizes a value space along independent axes or dimensions
Fixed attribute values	An attribute value that always has the same value
Globally defined attributes	Attribute definitions that are defined at the highest level in the XML Schema document, so that the definitions can be reused.
Globally defined elements	Element definitions that are defined at the highest level in the XML Schema document, so that the definitions can be reused.
Groups	XML Schema allows fragments of content models to be named and referenced from multiple complex types.
Lexical Space	A **lexical space** is the set of valid *literals* for a datatype
Locally defined attributes	Attributes that are not globally defined, and therefore the definition can not be referenced (reused) in other contexts.
Locally defined elements	Elements that are not globally defined, and therefore the definition can not be referenced (reused) in other contexts.
Mixed Content	A combination of child elements and character data nested within an element
Named Types	Named types may be defined once and used many times.
Namespaces	An XML namespace is a collection of names identified by a URI reference, which are used in XML documents as element types and attribute names
Notations	Can be used to specify the type of a file (for example, a graphics image) that is related to an XML document via an external entity.
Primitive Data Types	A Primitive is a unique semantic entity, having a unique semantic identifier. **Primitive** datatypes are those that are not defined in terms of other datatypes; they exist *ab initio*
Processing instructions	Can be used to pass information to the processing application.
Simple Type	Simple types cannot have element content and cannot carry attributes
Simple Type Definition	A simple type definition is a set of constraints on strings and information about the values they encode, applicable to the normalized value of an attribute information item or of an element information item with no element children. Informally, it applies to the values of attributes and the text-only content of elements
Substitution groups	An element can be declared to be a substitute for another element, the "head" element, allowing the new element to appear anywhere the head element may appear.

Target namespace	The namespace of an instance document.
Type Derivation	XML Schema allows a type to be derived from another type (its base type), either by extension or restriction.
Type Redefinition	XML Schema allows a schema author to redefine the types or groups of another schema document.
Type Substitution	Allows a base type to be substituted by any derived type
Union types	The union operation is supported by XML Schema for element typoes. For example, a code list may be defined as the union of two other code lists.
Uniqueness constraint	Schema provides several mechanisms to enforce uniqueness of elements or keys in an XML instance document.
User-derived Datatypes	**User-derived** datatypes are those derived datatypes that are defined by individual schema designers
Value Space	A **value space** is the set of values for a given datatype. Each value in the **value space** of a datatype is denoted by one or more literals in its lexical space.
Wildcard	A wildcard is a special kind of particle that matches element and attribute information items dependent on their namespace name, independently of their local names
XML Schema	An XML document that defines the allowable content of a class of XML documents. A class of documents refers to all possible permutations of structure in documents that will still confirm to the rules of the schema

2084

ANNEX B: EXAMPLES FROM FINANCE INVOICE PILOT

CICA – Flexible Modular Approach to XML Message Design Invoice Example

Background

There are many design objectives for this architecture, but there are three at the highest-level.

- Implementable "bullet" messages. Demanded is that the message represent the complete semantic picture of what is required to participate in this business process, without supplementary semantic qualification. Further, this requirement is for both semantically complete and concise messages.

- Cross industry interoperability. This requirement demands a solution that provides a mechanism for supporting the needs of multiple industries within the same overall framework. In other words, supporting the needs for communities which are made up of more than a single industry or a number of sub-industries, in such a manner as to bring stability to the common information, and seamless support for managing the difference required by industry/product.

- Autonomy. Many industries want to achieve interoperability, but do not want to sacrifice their ability to provide timely solutions – autonomy. This solution must find a way to support the spirit of the standard, while still enabling various industries the latitude to include 'proprietary' components. This needs to be supported in a manner that enables autonomy without sacrificing cross industry interoperability.

- Attractiveness. The sum total of the solution must provide definite benefits to the community in achieving their overall objectives. In other words, the approach has to represent a faster, cheaper and better way than building from the ground up. This requirement places value on ease of use, minimization of entry barriers, and a simple reuse philosophy.

Setting requirements at this level ensures solutions for SMEs.

Example background

Invoicing was deliberately selected for this example, because of its wide appeal and inherent cross industry nature. Financial institutions have a lot to say about certain details, such as the tally of dollars and payment details. Detail about the product provided or services rendered are specified by the various trade groups. At every level, Invoicing is cross industry.

Invoicing, from a business document design perspective, has two primary sources of complexity – two conditions which make designing a business document complex.

- Business Process. Invoicing is done within more than one Business Process, with different triggering events. One example is the Event based process, where an Invoice is triggered based on a business action, such as the shipment of goods or the delivery of services. Another example is Statement based, an Invoice triggered by a time interval, like utility bills & credit card statements. In these two examples, the overall high level organization of the information is very different.

- Product. Invoices have two primary subjects, the dollars changing hands and the documentation of the goods/services justifying the dollars. There is significant variation in how to represent goods ordered from a catalog versus visiting nursing care provided to a patient. This architecture must provide mechanisms for enabling this natural variation in content, while still enabling the cross industry capabilities.

Based on Business Process, a number of Invoices have been identified, as described in figure 1. Each row in the table represent a different business process need for an Invoice, identified in the first column, followed by a column describing the Goods oriented use of the Template, followed by the Service oriented use of the Invoice.

Template	Goods Oriented	Service Oriented
Delivery Based	Invoice generated upon single shipment	Invoice generated upon service performed
Event Based	Invoice generated upon single shipment to multiple locations	Invoice generated for multiple services at conclusion of event
Time Based	Invoice generated for consolidated shipments	Invoice generated for service on timed basis (ie. Monthly)
	Invoice generated on timed basis for period shipments	
Balance Forward	Invoice generated for period shipments with account balance	Invoice generated on timed basis with account balance

Figure 1a

Example Overview

The Example Invoice has systematically been decomposed into the requisite component parts. In general terms, as shown in Figure 1b, the basic layers start with the Template and the Slots [shown at the top], then decomposes to a set of Modules [Middle Layer], and ending with a set of Blocks [bottom layer]. Block are one of four types, depicted with subdivisions within the lower layer, events, parties, locations and resources.

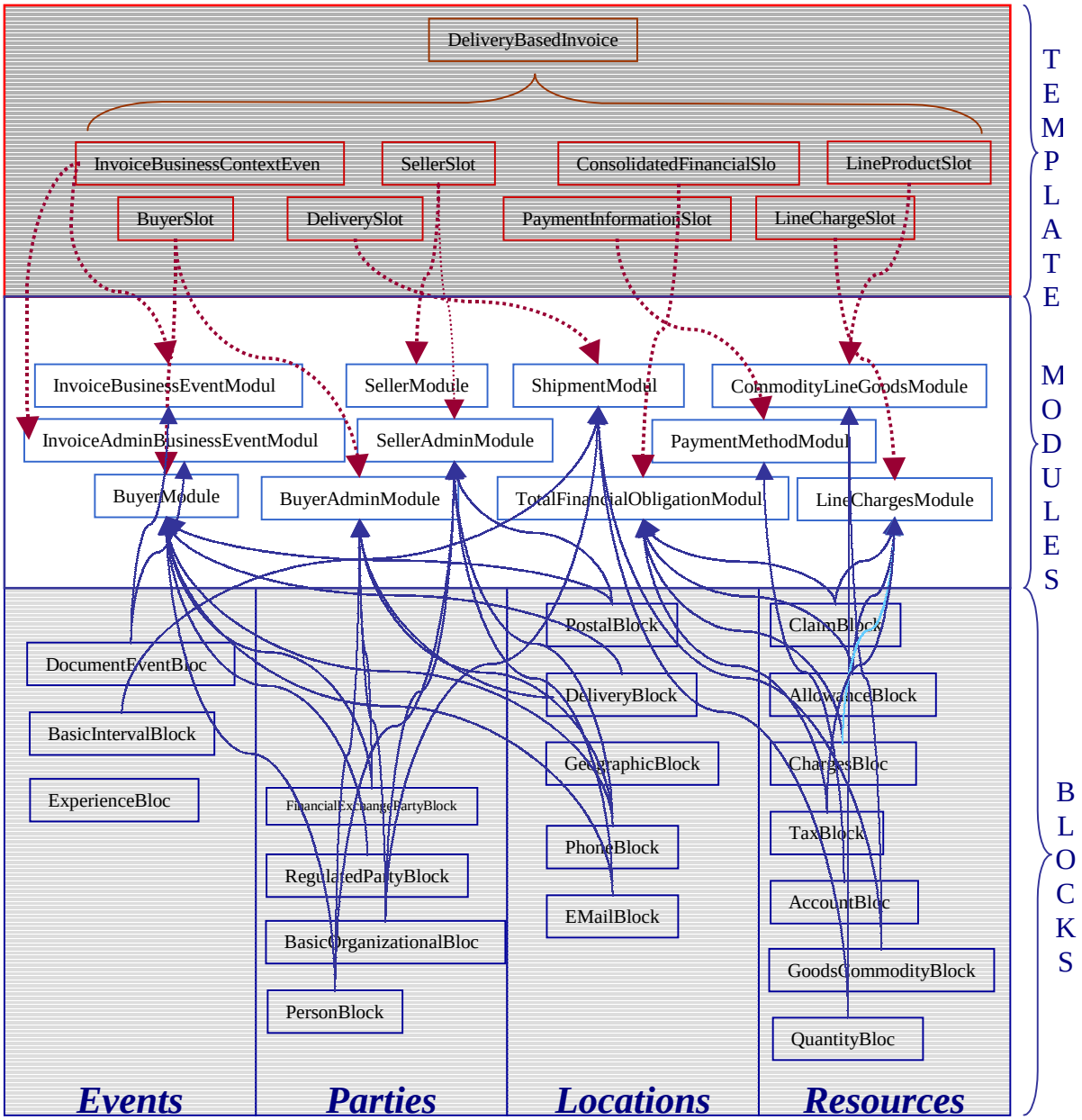


Figure 1b

In this DeliveryBasedInvoice example, there are a total of eight (8) slots, one of the Slots is the BuyerSlot shown in Figure 1b at the lower left, within the Template layer.

Each Slot within the Template will have one or more Modules. The BuyerSlot in our example has a BuyerModule and a BuyerAdminModule, shown in the lower left corner of the middle layer and connected with the dashed red lines. These link between the Slots and the Modules is conditional, Context identifiers determine when to use each Module.

A Module is constructed from reusable components, Assemblies and Blocks. While Modules are made from Assemblies and Blocks, but the characteristics of Modules is dramatically different. Modules are Context specific in that Modules are at a level where they can be mapped into an application system. Examples of Modules include: Buyer, Seller, Patient, Student, etc. In contrast, Assemblies and Blocks are neutral constructs, composed based on the need for the form – a Party with a First, Middle and Last Name. Assemblies are reusable groupings of Blocks, so in practice you can use Blocks or Assemblies. This Figure 1b illustration shows the use of Blocks in building Modules. The BuyerModule is a complicated Module, and is constructed with a number of Blocks, in some cases using the same Block more than once for different purposes. This is shown in Figure 1b with the solid arrows, pointing upward from the Block layer to the Module layer.

Example Specifics

In compliance with the CICA architecture, the Business Process determines how many Invoices. This example focuses on a single Invoicing example, the case where for each Action against an Order, an Invoice is generated – Delivery Based in Figure 1a. This Invoice example covers the case where an Order initiates the process, and for each Shipment of Goods or Delivery of Services made against the Order, an Invoice is generated.

Enough detail is provided to illustrate the Architecture, although the contents are scaled back for purposes of the example.

Template

The center of the architecture is the Template, and is the first of three constructs that are semantically significant to the architecture itself. The Template is Business Process specific, in that the Business Process determines its high-level composition and use. Therefore, much of the Template metadata is designed to capture the business circumstances where this Template is used, and how this template relates to other templates [the top half of figure 2]. Documenting the Business Process circumstances where this template is used to enable locating, differentiating and facilitating proper Template use.

The second half of figure 2 specifies the Slots. Slots identify the Template composition at the high level, in industry neutral or abstract terms. Templates, like the name implies, provide a high level guide to the contents without directly containing the contents. The Slots are determined based on the business process, and roughly answer the 'who', 'what', 'when', 'where', and 'why' questions about the business exchange. The 'who' is primarily about the participants in the business exchange, which might be the participants in the document exchange or might be more inclusive. The 'what' specifies the subject('s) of the message. The 'when' specifies the event('s), past, present or future. The 'where' specifics pertinent location('s). In general, the 'why' is specified by the document itself.

The level of the Slots, and the separation between the Template and the document contents, together are a critical design component of the CICA architecture. This design becomes the foundation for enabling both modular reuse & modular substitution.

In this DeliveryBasedInvoice example, Figure 2, the metadata specifically collects:

- Template Name, the name used to refer to this specific Template
- Template Family, the name for the class of Template, sometimes thought of as the abstract superclass
- Business Process specifies the specific business process within which this specific Template appears
- Business Process Family is the name for the set of peer business processes, sometimes referred to as the abstract superclass for the Business Process
- Business Process Sub-Family is a subdivision of the Business Process Family, within which this Template Family appears.
- Triggering Event Description is a description of the business condition, which uniquely distinguishes use of this Template over others of the same Template Family.
- Responsible Subcommittee is the organization within X12 responsible for this Template.

The Slots, are specified using the following:

- Area is used to differentiate between Header information that applies to the entire document, versus Detail information, which specifies the subject of the document, versus potential Summary information.
- Template Slot is the actual SlotSlot Description is free form text describing the purpose of the Slot, the information the Slot is designed to represent
- Slot Description is free form text describing the purpose of the Slot, the information the Slot is designed to represent

2214
2215

TemplateName	DeliveryBasedInvoice
TemplateFamily	Invoice
BusinessProcess	DeliveryBasedPayment
BusinessProcessFamily	Payment
BusinessProcessSubFamily	Invoicing
TemplateDescription	Invoice generated upon delivery of service or shipment of goods
TriggeringEventDescription	Each single shipment or delivery of service
Responsible Subcommittee	F

2216

Area	TemplateSlot	Slot Description	Req't	Min	Max
H	InvoiceBusinessContextEventSlot	Specifies the Business Environment of the document	M	1	1
H	BuyerSlot	The Buying Party	M	1	1
H	SellerSlot	The Selling Party	M	1	1
H	DeliverySlot	The Event detailing the execution of the reqd product	M	1	1
H	ConsolidatedFinancialSlot	Total Financial Obligation	M	1	1
H	PaymentInformationSlot	Payment Method	O	1	1
D	LineProductSlot	Specifies the Product [Goods or Service]	O	1	1
D	LineChargeSlot	Charges associated with single line	O		

Figure 2

2217 **Slot = InvoiceBusinessContextEventSlot**

2218 This Slot contains Event references, which includes specifying the unique reference
2219 information for this specific Event, Invoice, in addition of previous Events relevant to this
2220 business process. Modules defined for use in this Slot will specify these details.

2221 **Modules**

2222 A Module fits in a Template Slot, and therefore its contents must be compliant with the stated
2223 purpose of the Slot. Multiple Modules may be developed to fit into the same Slot, provided
2224 that the Module is not 'the same' as another Module, and that the Module fulfills the function
2225 specified by the Slot in the Template. Determining which Module to use in a Slot is based on
2226 Business Context.

2227 In this example, there are two Modules specified for this slot, shown in Figures 3 & 4. The
2228 first Module is the default Module, and is specifies Business level details about the
2229 document, and in the case of Invoice that equates to the unique reference information for this
2230 event, this Invoice, and previous events, the Purchase Order.

2231 The second Module developed for this slot contains an additional event, the Contract, and
2232 this Module is used in cases involving the US Federal Government, where there is a
2233 requirement for additional administrative information.

InvoiceBusinessEventModule	Assy/Block	A/B	Req't	Min	Max
InvoiceEvent	DocumentEvent	B:20	M	1	1
POEvent	DocumentEvent	B:20	M	1	1

2234

Figure 3

2235
2236
2237

InvoiceAdminBusinessEvent Module	Assy/Block	A/B	Req't	Min	Max
InvoiceEvent	DocumentEvent	B:20	M	1	1
POEvent	DocumentEvent	B:20	M	1	1
ContractEvent	DocumentEvent	B:20	M	1	1

Figure 4

Slot = Buyer

This Slot specifies a primary Party to this exchange. At the business process level, the Buyer plays a number of sub-roles. The primary function, Buyer, 'implies' a number of responsibilities – receiving the Product [goods or service], making payment, and the initiator. In practice, each of these sub-functions can have different Parties, Locations, and Contact points. But, these are all associated with the 'Buyer'.

Modules

The first Module Specified is for the default Buyer, which includes the basic sub-roles described above. In Figure 5, the BuyerModule is specified, with a Usage Name [first column], the Assembly or Block name used [column 2], designator [A=Assembly, B=Block, plus unit number in column 3], followed by requirement, minimum and maximum use.

BuyerModule	Assy/Block Name	A/B	Req't	Min	Max
Buyer	FinancialPartyAssy	A:20	M	1	1
BuyerContact	ContactAssy	A:10	O	1	1
ShipTo	DeliveryPartyAssy	A:21	O	1	1
ShipToContact	ContactAssy	A:10	O	1	1
BillTo	ActorPartyAssy	A:22	O	1	1
BillToContact	ContactAssy	A:10	O	1	1

Figure 5

The second Buyer, is like the first Buyer, only it contains an additional sub-role, the recipient of the Invoice, and requisite Contact. The Module is used in administrative intensive environments, such as when dealing with the US Federal Government. In this business context, the bar is raised in terms of required information, and pieces of information considered optional in other environments are now considered Mandatory.

BuyerAdminModule	Assy/Block Name	A/B	Req't	Min	Max
Buyer	FinancialPartyAssy	A:20	M	1	1
BuyerContact	ContactAssy	A:10	O	1	1
ShipTo	DeliveryPartyAssy	A:21	O	1	1
ShipToContact	ContactAssy	A:10	O	1	1
BillTo	ActorPartyAssy	A:22	O	1	1
BillToContact	ContactAssy	A:10	O	1	1
RecieveInvoice	ActorPartyAssy	A:22	O	1	1
RecieveInvoiceContact	ContactAssy	A:10	O	1	1

Figure 6

Slot= Seller

The Seller Slot is the place where the Seller is specified.

Modules

SellerModule

The first, shown in Figure 7, is the Default Module and contains the basic information to support the basic business process. This Module specifies two Parties, the actual Seller, and the optional Seller Contact.

SellerModule	Assy/Block	A/B	Req't	Min	Max
Seller	ActorPartyAssy	A:22	M	1	1
SellerContact	ContactAssy	A:10	O	1	1

Figure 7

SellerAdminModule

The Administrative intensive Seller, Figure 8, contains an additional sub-role, the ship from, that is mandatory in the business exchange is with the US federal government.

SellerAdminModule	Assy/Block	A/B	Req't	Min	Max
Seller	ActorPartyAssy	A:22	M	1	1
SellerContact	ContactAssy	A:10	O	1	1
ShipFrom	DeliverParyAssy	A:21	M	1	1
ShipFromContact	ContactAssy	A:10	O	1	1

Figure 8

Slot = DeliverySlot

This slot specifies the Event when the shipment was made or the services were rendered.

Module

ShipmentModule

This Module specifically documents the shipment of goods, which is the primary topic area this example covers. It is likely that specifying the Event of Services Delivered will require a different Module.

ShipmentModule	Assy/Block	A/B	Req't	Min	Max
Shipment	GoodsCommodity	B:40	M	1	1
ShipmentQuantity	Quantity	B:41	O	1	1
Carrier	BasicOrganization	B:3	O	1	1
DeliveryEvent	IntervalEvent	B:30	O	1	1

Figure 9

Slot = ConsolidatedFinancial

This slot contains the consolidation of expenses associated with this invoice, including subtotals and totals. It is likely that only a few Modules will be defined for this Slot, in that its purpose is to provide a complete tally of changes, independent of the details of the product [goods or service].

Module

ConsolidatedFinancialModule

This example is focused on the simplest case of Goods, so this module is specific to the needs of Goods charges.

ConsolidatedFinancialMod	Assy/Block	A/B	Req't	Min	Max
ProductClaim	Claim	B:60	M	1	1
ProductAllowances	Allowances	B:61	O	1	1
ProductCharges	Charges	B:62	O	1	1
ProductTax	Tax	B:63	O	1	1
FreightClaim	Claim	B:60	O	1	1
FreightAllowances	Allowances	B:61	O	1	1
FreightCharges	Charges	B:62	O	1	1
FreightTax	Tax	B:63	O	1	1
TotalClaim	Claim	B:60	M	1	1
TotalAllowances	Allowances	B:61	O	1	1
TotalCharges	Charges	B:62	O	1	1
TotalTax	Tax	B:63	O	1	1

Figure 10

Slot = LineProductSlot

The Slots, LineProductSlot & LineChargeSlot , comprise the Detail Area of the template. Together, they represent a "line item".

Modules

CommodityLineGoodsModule specifies the details required by the LineProductSlot for the case where the product is a Goods type of product. It is anticipated that either for each business sector or other high level grouping of industries, there will need to be different Modules.

CommodityLineGoods	Assy/Block	A/B	Req't	Min	Max
OrderedGoods	GoodsCommodity	B:40	M	1	1
OrderQuantity	Quantity	B:41	M	1	1

Figure 11

Slot = LineChargeSlot

The LineChargeSlot is companion to the LineProductSlot, in that the two as a unit make up the secondary subject of the Invoice, also known as the Detail Area. The primary subject is the Invoice charges, but the supporting detail for those charges are specified in these two Slots.

Modules

LineChargesModule specifies the charges associated with the companion Modules occupying the LineProductSlot.

LineCharges	Assy/Block	A/B	Req't	Min	Max
ProductPricing	Claim	B:60	M	1	1
ProductAllowances	Allowances	B:61	O	1	1
ProductCharges	Charges	B:62	O	1	1
ProductTax	Tax	B:63	O	1	1

Figure 12

Assemblies

In the Invoice example, the most complex structuring case involves the Buyer Module. In that case, a number of types of groupings are required, for a few different purposes. The general cases where groupings are required are as follows:

1. To associate together a list of choices, which is the motivation behind the Elocation Assembly.
2. Grouping reusable units.
3. Groupings made for technical reasons, to avoid having the same construct used multiple times at the same level, with different semantic purposes
4. Groupings made for documentation clarity, using hierarchy to explicitly associate semantically related contents.

In this example, Figure 13, lists a set of groupings established in support of the Buyer Module.

In general, any number of aggregations can be created between the Module and the Block, for either technical or documentation purposes. This architecture supports this need two ways, creating reusable Assemblies or with hierarchy in the Module. This example uses Assemblies. Each of the five (5) Assemblies listed in Figure 13, aid in solving technical problems associated with constructing the Buyer Module.

The Elocation Assembly groups the Blocks into an Assembly for the purposes of Choice. The Choice is represented in the Req't column with the value of A for Any. Exclusive, or X, is also supported.

The other groupings are also for technical reasons, to ensure that the various Party/Location constructs are semantically specific.

FinancialPartyAssy:20	Assy/Block	A/B	Req't	Min	Max
FinancialExchangeParty	FinancialExchange	B:1	M	1	1
Address	PostalBlock	B:10	O	1	1
ElectronicContact	ELocationAssy	A:10	C	1	1

ELocationAssy:10	Assy/Block	A/B	Req't	Min	Max
BuyerPhone	PhoneBlock	12	A	1	1
BuyerFax	PhoneBlock	12	A	1	1
BuyerEmail	EMailBlock	14	A	1	1
CellPhone	PersonBlock	12	A	1	1
PagerNumber	PhoneBlock	12	A	1	1

DeliveryPartyAssy:21	Assy/Block	A/B	Req't	Min	Max
DeliveryParty	BasicOrganizationBlock	B:3	O	1	1
DeliveryAddress	DeliveryBlock	B:11	O	1	1

ActorPartyAssy:22	Assy/Block	A/B	Req't	Min	Max
Party	BasicOrganizationParty	B:3	O	1	1
Postal	PostalBlock	B:10	O	1	1
ElectronicContact	ELocationAssy	A:10	C	1	1

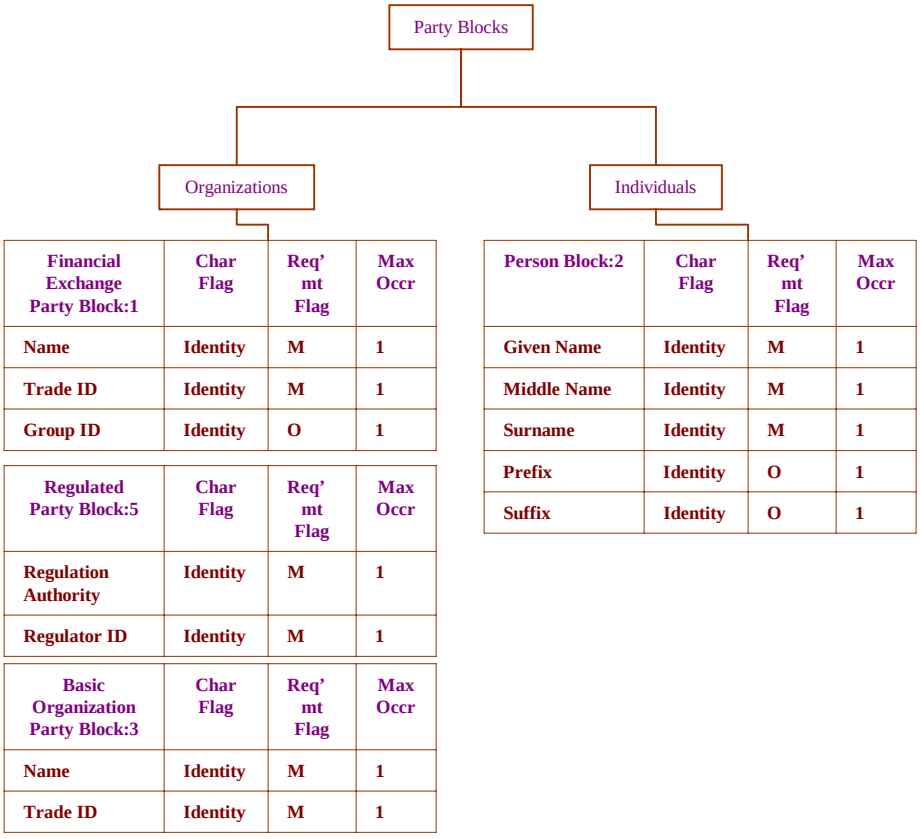
ContactAssy:23	Assy/Block	A/B	Req't	Min	Max
Contact	PersonBlock	B:2	O	1	1
EContact	ELocationAssy	A:10	M	1	1

Figure 13

Blocks

Party Blocks

Party Blocks, as illustrated in Figure 14, show how they are inter-related. This relationship as hierarchical, because the point is to create subdivision in them, so that a user can subset the dictionary, using high-level criteria, thus reducing the answer set into a manageable size list for final selection.



Financial Exchange Party Block:1	Char Flag	Req'mt Flag	Max Occr
Name	Identity	M	1
Trade ID	Identity	M	1
Group ID	Identity	O	1

Regulated Party Block:5	Char Flag	Req'mt Flag	Max Occr
Regulation Authority	Identity	M	1
Regulator ID	Identity	M	1

Basic Organization Party Block:3	Char Flag	Req'mt Flag	Max Occr
Name	Identity	M	1
Trade ID	Identity	M	1

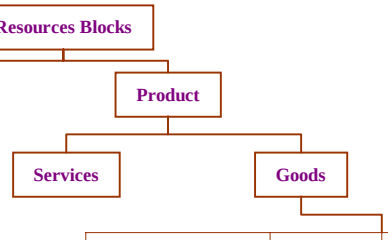
Person Block:2	Char Flag	Req'mt Flag	Max Occr
Given Name	Identity	M	1
Middle Name	Identity	M	1
Surname	Identity	M	1
Prefix	Identity	O	1
Suffix	Identity	O	1

Figure 14

Blocks Continued

Resources

Resource blocks include all of the things of Value, which are used in routine business transactions. Specifically, these include both the Products [Goods & Services] and the Money.



Claim Block:60	Char	Req' mt	Max
Gross Amount	Identity	M	1
Net Amount	Identity	M	1
Currency	Identity	O	1
Terms	Char	O	1

Allowance Block: 61	Char	Req' mt	Max
Type	Identity	M	1
Amount	Identity	M	1
Rate	Identity	O	1
Currency	Identity	O	1
Terms	Char	O	1

Charges Block: 62	Char	Req' mt	Max
Type	Identity	M	1
Amount	Identity	M	1
Rate	Identity	O	1
Currency	Identity	O	1

TAX Block: 63	Char	Req' mt	Max
Type	Identity	M	1
Amount	Identity	M	1
Rate	Identity	O	1
Currency	Identity	O	1

Account Block: 64	Char Flag	Req' mt Flag	Max Occr
Organization No.	Identity	M	1
Account	Identity	M	1
Type	Char	O	1

Goods Commodity Block:40	Char Flag	Req' mt Flag	Max Occr
Marking/Tracking #	Identity	M	1
Name	Identity	M	1
Measurements	Char	O	1
Commodity	Char	O	1
Safety Information	Char	O	1

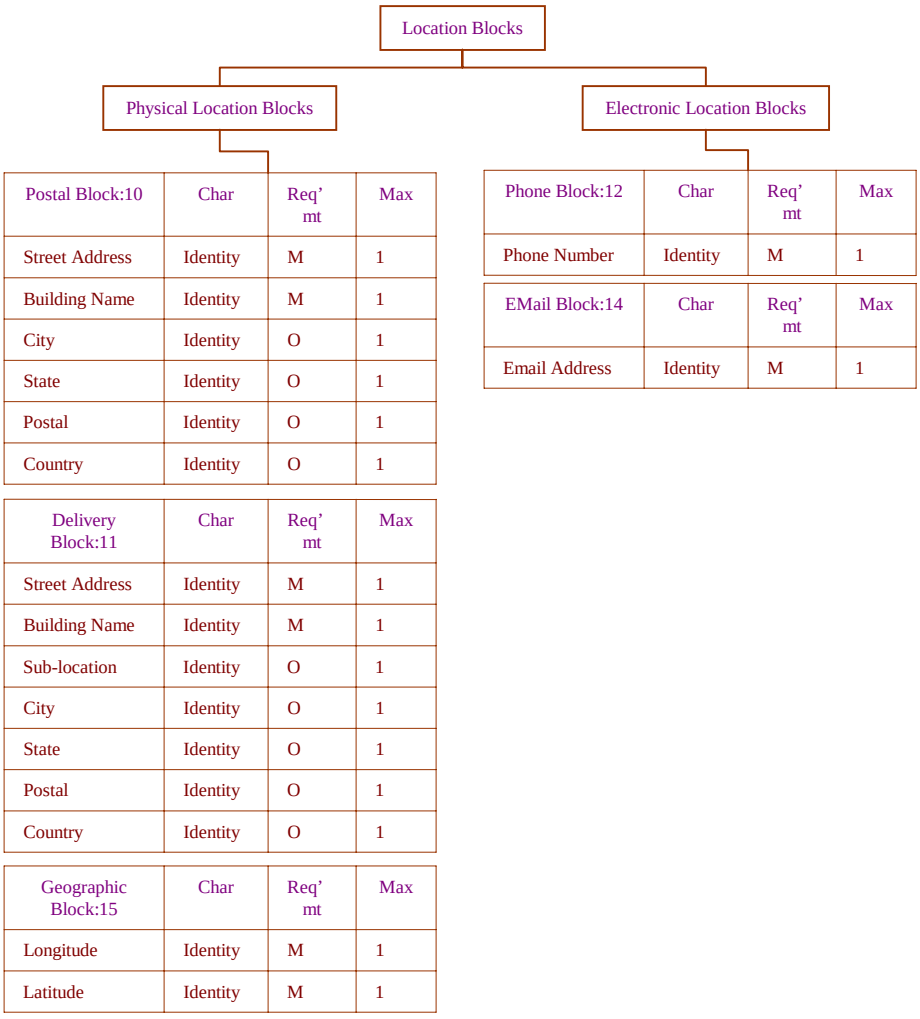
Quantity Block: 41	Char Flag	Req' mt Flag	Max Occr
TTL Required Quantity	Identity	M	1
Current Activity Quantity	Identity	M	1
Net Activity Quantity	Identity	O	1
Balance Quantity	Identity	O	1

Figure 15

Blocks Continued

Location

Location Blocks are used to specify each type of possible location, answering the question "Where". These are first, subdivided by physical versus logical or electronic Location.



Postal Block:10	Char	Req'mt	Max
Street Address	Identity	M	1
Building Name	Identity	M	1
City	Identity	O	1
State	Identity	O	1
Postal	Identity	O	1
Country	Identity	O	1

Delivery Block:11	Char	Req'mt	Max
Street Address	Identity	M	1
Building Name	Identity	M	1
Sub-location	Identity	O	1
City	Identity	O	1
State	Identity	O	1
Postal	Identity	O	1
Country	Identity	O	1

Geographic Block:15	Char	Req'mt	Max
Longitude	Identity	M	1
Latitude	Identity	M	1

Phone Block:12	Char	Req'mt	Max
Phone Number	Identity	M	1

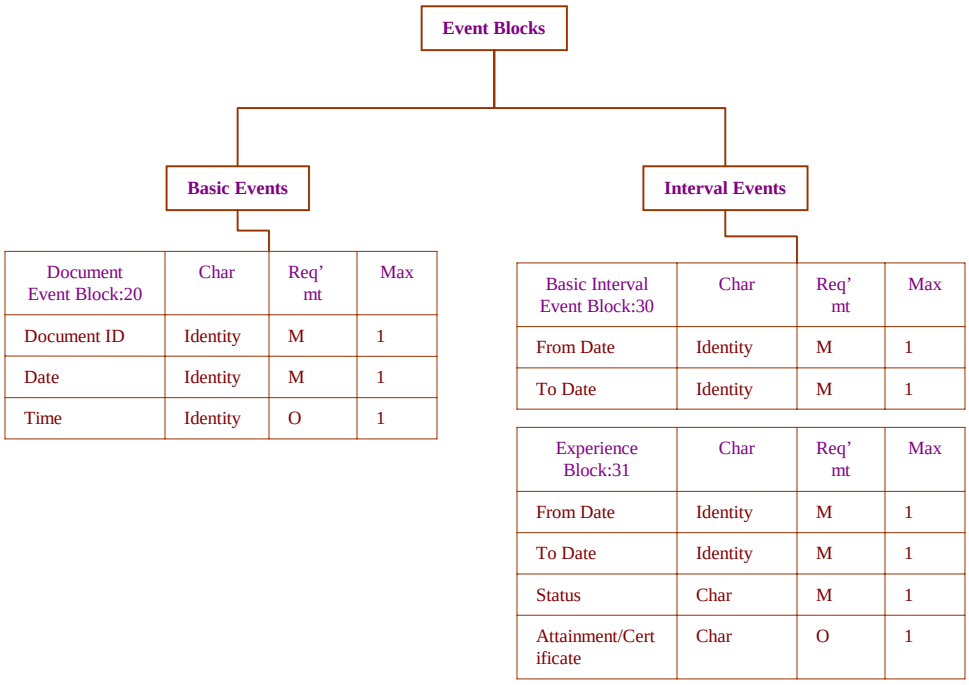
EMail Block:14	Char	Req'mt	Max
Email Address	Identity	M	1

Figure 16

Blocks Continued

Events

Event Blocks answer the When question. Events are first subdivided by whether they cover an interval or a basic event, shown in Figure 17, and attempts to add more detail than has been covered by the Invoice work for illustrative purposes.



Document Event Block:20	Char	Req' mt	Max
Document ID	Identity	M	1
Date	Identity	M	1
Time	Identity	O	1

Basic Interval Event Block:30	Char	Req' mt	Max
From Date	Identity	M	1
To Date	Identity	M	1

Experience Block:31	Char	Req' mt	Max
From Date	Identity	M	1
To Date	Identity	M	1
Status	Char	M	1
Attainment/Certificate	Char	O	1

Figure 17

Components

Components are not detailed here, pending ongoing work within the Finance to associate the content requirements in this Invoice example to the content specified by Core Components.

Flexible Modular Architecture

The Template linked with Modules is the first step in enabling the flexible modular architecture. This establishes the backbone for modular substitution, which allows a new level of autonomy over previous 'kitchen sink' approaches. Kitchen sink approaches are motivated by the recognition that related contents are related, however, having only one way to unite them forces a unproductive combining effort, resulting in a single ambiguous unit – kitchen sink. CICA recognizes the need to relate, related constructs, and enables the need with layers of abstraction. At this level, the Slot is the abstraction layer, the Module is the specific. Multiple, implementable Modules logically fit into the same place in the template, the Slot. At the time the Modules are linked with Slot, the Context conditions where the use of the Module is determined are specified.

Document Name	DeliveryBasedGoodsInvoiceDocument
Template Name	DeliveryBasedInvoiceTemplate
TemplateFamily	Invoice
BusinessProcess	DeliveryBasedPayment
BusinessProcessFamily	Payment
BusinessProcessSubFamily	Invoicing
TemplateDescription	Invoice generated upon delivery of service or shipment of goods
TriggeringEventDescription	Each single shipment or delivery of service
Responsible Subcommittee	F

Area	TemplateSlot	Usage	Context C/V	Context C/V	Context C/V	Module	Req't	Min	Max
H	InvoiceBusinessContextEventSlot	D				InvoiceBusinessEventModule	M	1	1
H	InvoiceBusinessContextEventSlot	C	8:US Gov			InvoiceAdminBusinessEventModule	M	1	1
H	BuyerSlot	D				BuyerModule	M	1	1
H	BuyerSlot	C	8:US Gov			BuyerAdminModule	M	1	1
H	SellerSlot	D				SellerModule	M	1	1
H	SellerSlot	C	8:US Gov			SellerAdminModule	M	1	1
H	DeliverySlot	C	2:goods			ShipmentModule	M	1	1
H	ConsolidatedFinancialSlot	D				ConsolidatedFinancialModule	M	1	1
H	PaymentInformationSlot	D				PaymentModule	O	1	1
D	LineProductSlot	D	2:commodity			CommodityLineGoodsModule	O	1	1
D	LineChargeSlot	D	2:goods			LineChargesModule	O		

Figure 18

Documents

Documents are largely derived from the Module linked Template, where for each Slot, Context is specified, thus resulting in a “bullet” document. A semantically explicit and concise Document, for use within context. Figures 19 and 20, show Documents, derived from specifying Slot level Context, from the same Template.

Document Name	DeliveryBasedGoodsInvoiceDocument
Template Name	DeliveryBasedInvoiceTemplate
TemplateFamily	Invoice
BusinessProcess	DeliveryBasedPayment
BusinessProcessFamily	Payment
BusinessProcessSubFamily	Invoicing
TemplateDescription	Invoice generated upon delivery of service or shipment of goods
TriggeringEventDescription	Each single shipment or delivery of service
Responsible Subcommittee	F

Area	TemplateSlot	Usage	Context C/V	Context C/V	Context C/V	Module	Req’t	Min	Max
H	InvoiceBusinessContextEventSlot	D				InvoiceBusinessEventModule	M	1	1
H	BuyerSlot	D				BuyerModule	M	1	1
H	SellerSlot	D				SellerModule	M	1	1
H	DeliverySlot	C	2:goods			ShipmentModule	M	1	1
H	ConsolidatedFinancialSlot	D				ConsolidatedFinancialModule	M	1	1
H	PaymentInformationSlot	D				PaymentModule	O	1	1
D	LineProductSlot	D	2:commodity			CommodityLineGoodsModule	O	1	1
D	LineChargeSlot	D	2:goods			LineChargesModule	O		

Figure 19

2392

DocumentName	DeliveryBasedAdminInvoiceDocument
TemplateName	DeliveryBasedInvoiceTemplate
TemplateFamily	Invoice
BusinessProcess	DeliveryBasedPayment
BusinessProcessFamily	Payment
BusinessProcessSubFamily	Invoicing
TemplateDescription	Invoice generated upon delivery of service or shipment of goods
TriggeringEventDescription	Each single shipment or delivery of service
Responsible Subcommittee	F

Area	TemplateSlot	Usage	Context C/V	Context C/V	Context C/V	Module	Req't	Min	Max
H	InvoiceBusinessContextEventSlot	C	8:US Gov			InvoiceAdminBusinessEventModule	M	1	1
H	BuyerSlot	C	8:US Gov			BuyerAdminModule	M	1	1
H	SellerSlot	C	8:US Gov			SellerAdminModule	M	1	1
H	DeliverySlot	C	2:goods			ShipmentModule	M	1	1
H	ConsolidatedFinancialSlot	D				ConsolidatedFinancialModule	M	1	1
H	PaymentInformationSlot	D				PaymentModule	O	1	1
D	LineProductSlot	D	2:commodity			CommodityLineGoodsModule	O	1	1
D	LineChargeSlot	D	2:goods			LineChargesModule	O		

2393

Figure 20

2394

ANNEX C: CORE COMPONENTS CONTEXT CATEGORIES

In keeping with ASC X12's goal to align with the ebXML Core Component work, the following table and descriptive text are reproduced from Section 6.2.2 of the UN/CEFACT – ebXML Core Components Technical Specification, Part 1 (8 February 2002, Version 1.8). The UN/CEFACT – ebXML Core Components Technical Specification is copyrighted by UN/CEFACT and this excerpt is reproduced with that body's permission.

Note: The URL for the full document is
http://www.unece.org/cefact/ebxml/ebXML_CCTS_Part1_V1-8.pdf

A comprehensive list of values must be specified for each context category. The ebXML CC specification has identified one or more available sources for each category. X12 plans to identify an "X12 selection" for the context categories that have multiple resources.

6.2.2 Approved Context Categories

Table 6-4 contains the eight approved *Context Categorie*s.

[C32] When describing a specific *Business Contex*t, a set of values will be assigned to the business situation being formally described.

[C33] Applied *Business Context* will be from the list of approved context categories.

Table 6-4. Approved Context Categories

Business Process	The business process as described using the ebXML Catalogue of Common Business Processes as extended by the user.
Product Classification	Factors influencing semantics that are the result of the goods or services being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to materials)
Industry Classification	Semantic influences related to the industry or industries of the trading partners (e.g., product identification schemes used in different industries).
Geopolitical	Geographical factors that influence business semantics (e.g., the structure of an address).
Official Constraints	Legal and governmental influences on semantics (e.g. hazardous materials information required by law when shipping goods).
Business Process Role	The actors conducting a particular business process, as identified in the Catalogue of Common Business Processes.
Supporting Role	Semantic influences related to non-partner roles (e.g., data required by a third-party shipper in an order response going from seller to buyer.)
System Capabilities	This context category exists to capture the limitations of systems (e.g. an existing back office can only support an address in a certain form).

6.2.2.1 Business Process Context

In describing a business situation, generally the most important aspect of that situation is the business activity being conducted. *Business Process Context* provides a way to unambiguously identify the business activity. To ensure consistency with business process activities, it is important to use a common point of reference. The definitive point of reference for international standards is the UN/CEFACT *Catalogue of Common Business Processes*

[C34] Assigned *Business Process Contexts* shall be from the standard hierarchical classification: provided as part of the UN/CEFACT *Catalogue of Common Business Processes.*

[C35] *Business Process Context* values may be expressed as a single business process at any level, or may be expressed as a set of business processes at any level.

[C36] *Business Process Context* values may be taken from extensions to the business processes described in the *Catalogue of Common Business Processes* as provided for in that document.

[C37] When business process extensions are used, they shall include full information for each value sufficient to unambiguously identify which extension is providing the value used.

6.2.2.2 Product Classification Context

The Product Classification Context describes those aspects of a business situation related to the goods or services being exchanged by, or otherwise manipulated, or concerned, in the business process. Recognised code lists exist that provide authoritative sources of product classification contexts.

[C38] A single value or set of values may be used in a *Product Classification Context.*

[C39] If a hierarchical system of values is used for *Product Classification Contex*t, then these values may be at any level of the hierarchy.

[C40] If more than one classification system is being employed, an additional value specifying which classification scheme has supplied the values used shall be conveyed.

[C41] Product classification context code values shall be taken from recognised code lists to include:

- *Universal Standard Product and Service Specification* (UNSPSC)
  - Custodian: Electronic Commerce Code Management Association (ECCMA)
- *Standard International Trade Classification* (SITC Rev .3)
  - Custodian: United Nations Statistics Division (UNSD)
- *Harmonised Commodity Description and Coding System* (HS)
  - Custodian: World Trade Organization (WTO)
- *Classification Of the purposes of non Profit Institutions serving households* (COPI)
  - Custodian: UNSD (This provides a mapping between the first three.)

6.2.2.3 Industry Classification Context

The Industry Classification Context provides a description of the industry or sub-industry in which the business process takes place.

[C42] An *Industry Classification Context* may contain a single value or set of values at any appropriate level of the value hierarchy.

[C43] The *Industry Classification Context* value hierarchy must be identified.

[C44] *Industry Classification Context* code values shall be taken from recognised code lists to include:

- *International Standard Industrial Classification* (ISIC) -- Custodian: UNSD
- *Universal Standard Product and Service Specification* (UNSPSC) Top-level Segment [digits 1 and 2] used to define industry. --Custodian: ECCMA

[Note] There are many other industry classification schemes that may be used for *Industry Classification Contex*t.

6.2.2.4 Geopolitical Context

Geopolitical Contexts allow description of those aspects of the business context that are related to region, nationality, or geographically based cultural factors.

[C45] *Geopolitical Context* shall consist of appropriate continent, economic region, country, and region identifiers.

[C46] *Geopolitical Regional Classification* may associate one or more values with any business message or component. are related to region, nationality, or geographically based cultural factors. country, and region identifiers. any business message or component.

[C47] *Geopolitical Regional Classification* shall employ the following hierarchical structure:

Global
[Continent]
[Economic Region]
[Country] - ISO 3166.1
[Region] - ISO 3166.2

[C48] At any level of the *Geopolitical Regional Classification* hierarchy, a value may be a single value, a named aggregate, or cross-border value.

[C49] *Geopolitical Regional Classification* hierarchy values shall structured as follows:

- **Single Value:** A single value indicating a single continent, economic region, country, or region, depending on position within the hierarchy.
- **Named Aggregate:** A related group of values (which may themselves be single values, named aggregates, or cross-border pairs of values), which have been related and assigned a name. A named aggregate contains at least two values.
- **Cross-Border:** One or more pairs of values, designated *To*, *Fro*m, or *Bi- directional,* indicating the direction of cross-border context. Values may be named aggregates or single values.

[Example] The following example shows an extract of the basic, single-value hierarchy of recommended values, based on the common ISO 3166.1 *Country Codes.* (The value at the top of any hierarchy is always understood to be *Globa*l.)

Europe
Eastern Europe
AL – ALBANIA
AM – ARMENIA

[C50] Points in the *Geopolitical Regional Classification* hierarchy shall be specified by the use of the node value, or by the full or partial path.

[C51] The full path of the *Geopolitical Regional Classification* hierarchy must be used to understand the hierarchy when complex constructs are employed.

[C52] A single-point specification is understood to inherit all of the properties of the single-value hierarchy except where otherwise specified.

[C53] *Geopolitical Values* will be taken from ISO 3166.1 and 3166.2

6.2.2.5 Official Constraints Context

The Official Constraints Context category describes those aspects of the business situation that result from legal or regulatory requirements and similar official categories. This category contains two distinct parts:

- Regulatory and Legislative. These are normally unilateral in nature and include such things as customs.
- Conventions and Treaties. These are normally bi- or multilateral agreements and as such are different from regulatory and legislative constraints.

[C54] The *Official Constraints Context* will consist of at least two values:

- Identification of the legal or other classification used to identify the context values.
- Identification of the official constraint itself. These values may represent a hierarchical structure depending on the official constraints system being referenced.

Because there is no known global classification of all *Official Constraints Contexts* as used here, any implementation must provide a set of recognised official constraints classifications for use within the appropriate *Core Components* Registry implementation.

[C55] Individual *Core Component* implementations shall register used official constraint classification schemes with the appropriate supporting *Core Components* Registry implementation.

6.2.2.6 Business Process Role Context

The Business Process Role Context describes those aspects of a business situation that are specific to an actor or actors within the business process. Its values are taken from the set of Role values provided by the Catalogue of Common Business Processes. A Business Process Role Context is specified by using a value or set of values from this source.

[C56] *Business Process Role Context* values shall be taken from an approved list provided by the business process model library being employed.

[C57] The UN/CEFACT *Catalogue of Common Business Processes* shall be the definitive source of *Business Process Role Context* values for all UN/CEFACT *Business Information Entitie*s.

6.2.2.7 Supporting Role Context

The Supporting Role Context identifies those parties that are not active participants in the business process being conducted but who are interested in it. A Supporting Role Context is specified with a value or set of values from a standard classification.

[C58] *Supporting Role Context* values shall be taken from the UN/EDIFACT *Code List for DE 3035 Party Roles.*

[Note] Users are cautioned that duplication exists in the current version of the required code list. UN/CEFACT will review this code list to clarify duplicates and identify non- *Supporting Role Context* values.

6.2.2.8 System Capabilities Context

This category identifies a system, a class of systems or standard in the business situation. The System Capabilities Context requires a least one pair of values: an identification of the

classification scheme being used and a value from that scheme. A valid System Capabilities Context may include more than one such pair of values.

[C59] *Systems Capabilities Context* values shall consist of pairs of values. Each pair shall be comprised of an identification of the referenced classification scheme and the value(s) being employed.

[Note] There is no known classification of all types of information systems and standards. It is recommended that a mechanism for the registration of system and standard names be provided by the ebXML registry, as valid values for the *System Capabilities Context.*

2556

ANNEX D: BACKGROUND

1.0 Background

The Extensible Markup Language (XML) was developed by the World Wide Web Consortium (W3C), the de facto standards body for the Internet and the World Wide Web. The first working draft paper on the concept of XML was published 14 November 1996. The original goal was, "…to enable SGML to be served, received, and processed on the Web in the way that is now possible with HTML." A primary design consideration was to design XML, "…for ease of implementation, and for interoperability with both SGML and HTML." Much of the original concept was applied to using XML as a means for graphical communication. The idea of its use for conducting EDI was applied later when the first studies were done on this subject in late 1997. Early work on XML/EDI was conducted both jointly and independently by ANSI ASC X12, UN/CEFACT, CommerceNet, and the XML/EDI Group as well as other organizations. The goals of XML/EDI as defined by the XML/EDI Group are:

- To deliver unambiguous and durable business transactions via electronic means
- Utilize existing systems and processes
- Protect the investment in traditional EC/EDI
- Provide a migration path to next generation XML/EDI systems
- Use existing business processes as implemented
- Facilitate direct interoperation in an open environment

In November 1999, work began on the ebXML project, a joint UN/CEFACT and OASIS initiative, whose mission was to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure, and consistent manner by all parties. The project concluded in May 2001 and delivered a modular suite of specifications that enable enterprises to conduct business over the Internet. The specifications address the following areas:

- Messaging Services
- Registries and Repositories
- Collaborative Protocol Profile
- Implementation, Interoperability, and Conformance
- Core Components and Business Process Models

The ebXML specifications are currently being transitioned to UN/CEFACT and OASIS for the purpose of developing global electronic business standards.

X12 began work on XML/EDI in 1998 with the creation of an ad-hoc XML work group that transitioned to X12C/TG3. X12C/TG3 in conjunction with CommerceNet produced a paper entitled “Preliminary Findings and Recommendations on the representation of X12 Data Elements and Structures in XML”. In addition to this collaborative effort, X12C/TG3 produced a technical white paper providing additional information on using XML to represent business exchanges. In February 2000, the X12 Steering Committee chartered the X12 XML Task Group to develop recommendations for the Steering Committee in conjunction with the X12 subcommittees on XML. The resolutions approved by the Steering Committee in June/October 2000 were:

The ANSI ASC X12 Steering Committee fully supports the continuation of the mission, goals, and efforts of ebXML. ASC X12 will pursue its XML development efforts within the framework defined by ebXML.

- X12 will develop accredited, cross-industry, XML business standards. All XML business standards and associated schema development work will be done in collaboration with UN/CEFACT Work Groups and shall be based on the UN/CEFACT business process/core component work.
- The X12 Steering Committee will petition ANSI for official recognition as an ANSI accredited XML business standards body
- X12C will function as the X12 XML technical experts with respect to all internal and external XML technical specifications including the development of XML design rules in conjunction with X12J
- The X12 Steering Committee shall task DISA to begin working with X12X TG4 WG2 to market X12's role in developing ANSI accredited XML business standards.
- The X12 Steering Committee shall task the Process Improvement Group (PIG) to include the need to recognize the requirement for an accelerated process for XML standards development as part of their work plan

The X12 Steering Committee shall task the Process Improvement Group (PIG) to work with the Policies and Procedures Task Group (P&P) to provide expertise and assist the EWG/X12 on the Joint Development Task Group in the development of an aligned approval process that meets the needs of both organizations related to the development and maintenance of XML core components.

Every effort has been made to build on the experience and work done previously by ebXML, the UN/CEFACT Work Groups, CommerceNet, and ANSI ASC X12 in document definition methodologies and core components. The X12/XML design rules presented in this document are based on design decisions reached through a process of issue identification, presentation of examples, and evaluation of the pros and cons of each available action. They provide a set of syntax production rules that define the conversion of standardized, cross-industry business messages into XML documents.

2.0 Overview of ebXML Business Process and Core Components

The business process determines characteristics of the business document payload. For example, if the business process is Ordering then the order information must specify details about the order itself (payment, delivery, references to external business agreements, etc.). There are certain characteristics of the Order Document, which typically do not vary across industries, while other details (such as those required because of product type) will vary dramatically.

Business documents, by their very nature, communicate a semantically complete business thought: who, what, when, where and why. The what in electronic business terms is typically the product. It is widely recognized that products are goods or services. Goods are manufactured, shipped, stored, purchased, inspected, etc., by parties. Services are performed by parties, and may involve goods and/or parties. Parties can be either organizations or individuals, and can be associated with other parties and products. And these products have events associated with them, inspections, transportation, building, sale, etc.

This problem is addressed by a combination of structured information and the use of context. This structure uses a series of layers, designed to take into account commonality across industry business process. Further the structure is designed to support specialization based on the specific use of contexts. Context is the description of the environment within which use will occur. For example, if one was to say that "someone was pounding on my car with a hammer", the response is very different depending whether it is a repair shop or a neighbourhood youth. Context is what is used to direct interpretation.

A component is a ‘building block’ that contains pieces of business information, which go together because they are about a single concept. An example would be bank account identification, which consists of account number and account name.

Core components are components that appear in many different circumstances of business information and in many different areas of business. A core component is a common or “general” building block that basically can be used across several business sectors. It is therefore context free.

Re-use is the term given to the use of common core components when they are used for a specific business purpose. The purpose is defined by the combination of contexts in which that business purpose exists. Each context specific re-use of a common component is catalogued under a new business information name ‘that uses core component X’.

A domain component is specific to an individual industry area and is only used within that domain. It may be re-used by another domain if it is found to be appropriate and adequate for their use, and it then becomes a core or common component.

Components can be built together into aggregates.

As described above for components, aggregated components can be common components. These are generic and can be used across several business sectors. They can be re-used for a specific business purpose, defined by a combination of contexts. Each context specific re-use of a common aggregate component is catalogued under a new business information name ‘that uses core component X’.

There are also domain specific aggregated components.

Aggregates and components can be gathered into "document parts". These are useful assemblies which can individually satisfy a business process’s requirement for information, or which may be "sewn together" in a structured way to achieve the same. For example, the structured combination may be to satisfy a business process’s need for information presented in a particular way for efficiency of processing.

An individual document part and the "sewn together" parts, come at increasingly domain-specific and context-specific levels. They form documents or partial documents that satisfy a business process or a part of a business process.

Figure 21 illustrates how core components can be built into business documents by explicitly linking components with the ebXML Business Process Worksheets, and the underlying modelling approach. The top right-hand corner of the Figure comes from Figure 8.4-1 in the ebXML Business Process Overview document.

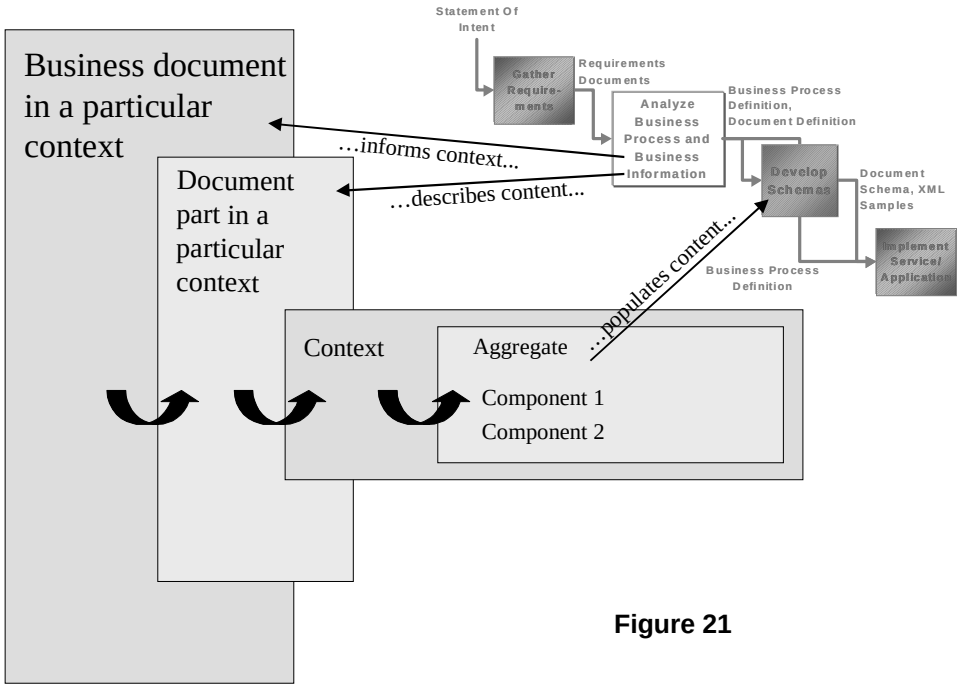


Figure 21

Note that in this instance document parts are pieces of business information required to satisfy a particular business process, from a specific contextual viewpoint.

3.0 Relationship to other XML Efforts

Since most other XML efforts lack an overriding semantic organization, many efforts have been directed to production of "bullet" messages. This effort is directly applicable by narrow definition of the business purpose underlying each Template. In particular, ebXML efforts have componentry definitions with instances that span several levels. The architecture proposed here provides a structured mechanism to impose a semantic discipline in this arena.

Several XML efforts have modeling as a primary tenet. Modeling may prove to be the best way to develop items at the top levels of this architecture (certainly Templates and Modules and possibly Assemblies). X12 feels that this architecture allows modeling to be used at high levels, where it is most effective.

ANNEX E: Framework Approaches for Implementing XML Syntax

This annex applies the general recommendations of the syntax section to the semantic architecture. It presents options for a framework for representing each type of semantic construct in XML instance documents and schemas. At the time of publication consensus had not been reached on all of the details, so in many cases two options are presented. Option A represents the majority position at the time of publication, while Option B represents the minority position.

1.0 Primitives

Primitives represent the leaves in a tree representation of an instance document.

Option A

Instance Representation - Primitives are represented as XML elements.

Schema Representation - Primitives are represented as simpleTypes, derived from base schema datatypes.

Option B

Instance Representation - The primitive representing the primary value of a component is represented as an element. Supplementary values are represented as attributes.

Schema Representation - Supplementary primitives are represented as simpleTypes, derived from base schema datatypes.

2.0 Components

Components represent the first level of inner nodes in a tree representation of an instance document. Components are represented as complexTypes.

Option A

Instance Representation - A component is represented as a parent element with named child elements, in sequence, of allowable primitive types.

Schema Representation - A component is a complexType with named child elements of types defined for primitives.

Option B

Instance Representation - A component is a complexType whose value is the primary value of the component. Supplementary primitives are represented as attributes.

Schema Representation - A component is a complexType with no element children, a type based on one of the defined primitive types, and a set of named attributes that are of the types defined for primitives.

3.0 Blocks

Blocks represent the second level of inner nodes in a tree representation of an instance document. Blocks are represented as complexTypes.

Instance Representation - A block is represented as a parent element with child elements, in sequence, of types allowable for components

Schema Representation - A block is a complexType with named child elements of types defined for components.

4.0 Assemblies

Assemblies represent the inner nodes in a tree representation of an instance document. They occur at all of the levels between components and modules, which are one level below the root element.

Option A

Instance Representation - An assembly is a parent element with child elements

Schema Representation - An assembly is a complexType with named child elements, in sequence, of types defined for blocks or assemblies.

Option B

Instance Representation - An assembly is not identifiable as such in an instance document.

Schema Representation - An assembly is a model group of elements, each of a type based on one of the types allowed for blocks or assemblies.

5.0 Modules

Modules represent the first level under the root node in a tree representation of an instance document.

Instance Representation - A module is represented as a parent element with child elements

Schema Representation - A module is a complexType with named child elements of types allowed for blocks or assemblies

6.0 Templates

Templates are logical entities with no direct XML representation. This is primarily due to the fact that they are skeleton documents only, and an XML syntax representation in schema would therefore be incomplete and not valid.

7.0 Documents

Documents correspond to XML instance documents.

Schema Representation - A document is represented by the root element (with children) of an XML schema. Each child is a locally named element specific to the document, of a defined module complexType.

8.0 Modular Organization and Namespace Architecture

The goals of this architecture are:

- As nearly as practical, a one-for-one syntax representation of items from the semantic architecture
- Enable re-use at the syntax level by organizations outside of X12

Major aspects of the architecture are:

- XML standards follow the one major, two minor release schedule of X12. All schemas have a root URN namespace, (and corresponding root http URL) of schemas.x12.org, followed by the major and minor release. eg. Root namespace urn:schemas.x12.org/005010, root URL http:schemas.x12.org/005010 for major release 005 and minor release 010 (majmin in the following examples).
- Items shared by two or more subcommittees have the root namespace URN and URL of schemas.x12.org/majmin/common
- Declarations for shared blocks, components, and assemblies - Are declared in the common target namespace. If the number is manageable to fit into one physical schema, it is common.xsd under majmin. If more than one, they are divided into logical groupings with separate schema files (combined with xsd:include) under the common directory
- Items unique to a subcommittee have the root namespace URN and URL of the subcommittee in the form of schemas.x12.org/majmin/SubcommitteeDesignator
- Declararations for modules (and any blocks, components, or assemblies unique to a subcommittee) - Are declared under the root subcommittee URN SubcommitteeDesignator/common, and schema file URL SubcommitteeDesignator/common.xsd. The X12 common namespace is imported into each subcommittee common schema.
- Document definitions - Are declared under the root subcommittee URN and URL, with target namespace and schema file specific to the document. The subcommittee name-space is imported into the schema file.
- All element and attribute forms are unqualified. Names from namespaces imported from another namespace into a schema document must have a namespace prefix, but local names of the schema target namespace don't need one. In instance documents only the root element requires a namespace prefix.

ANNEX F: Architectural Comparison with Other Initiatives

General Comparison of Approaches

X12

The primary contribution of the X12 CICA approach is to establish strong semantic granularity and abstractions in Component and Message design. The motivation behind the design is to develop an architectural approach that achieves three primary objectives, which on the surface appear as conflicting:

1. 'Bullet' documents, semantically explicit and concise documents fully reflective of the implementation.
2. Cross industry support, seamlessly supporting the commonality and differences required to enable cross industry messaging and application environments.
3. Minimal administration overhead, offering autonomy to industries needing to produce immediate solutions, interoperating through the same cross industry framework.

Core Components

In contrast to the X12 XML efforts, CEFACT's Core Components work has primarily focused on the physical details of specifying Core Components and the least common denominator compositional semantics, working primarily from a bottom up approach.

The context classification work, including the identification of Context categories and sources for specifying Context is very well developed. The CCT work and the naming conventions efforts are substantive and innovative work.

The design objectives utilize context, as an enabler to innovative new approaches to managing differing content needs. Further, this effort is committed to cross industry solutions.

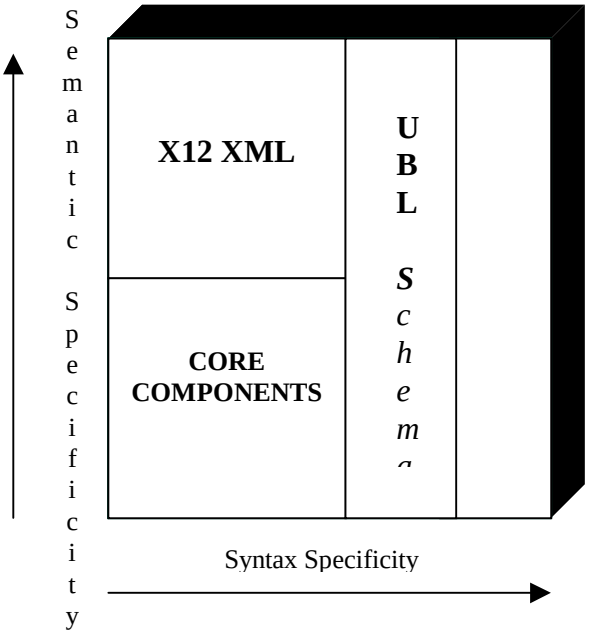


Diagram 1

UBL

UBL's stated approach is to start with Core Components and produce XML schema representations for the initial set of documents. This approach is designed to be a short-term effort.

Summary of General Comparisons

In general, while at the detail level, many parallels exist between the X12's CICA work, and that of the CEFACT Core Components, but conceptually, there is little overlap in subject areas covered. As illustrated in "Comparison 1", the X12 work has primarily focused on subjects, which the Core Components works has been silent on – semantic granularity and a strong document model. In contrast, the Core Components work has primarily focused on

the representation aspects. Finally, the UBL work is taking content results, and representing that output in schema.

Specific CICA to Core Component Comparisons

This diagram illustrates a rough comparison between the approach outlined in this document and the CEFACT Core Components technical specification (CCTS).

<table>
<tr><th>X12 XML Design</th><th>CEFACT CC Tech Spec.</th></tr>
<tr><td>Document</td><td rowspan="2"></td></tr>
<tr><td>Template</td></tr>
<tr><td>Module</td><td>Business Information Entity</td></tr>
<tr><td>Assembly</td><td rowspan="3">Core Component</td></tr>
<tr><td>Block</td></tr>
<tr><td>Component</td></tr>
</table>

Diagram 2

As shown in the diagram, there are generally direct relationships among the work efforts, by design. To try and keep things clear this document will call them all “Parts”.

Neutral Parts

Assemblies, Blocks, and Components are analogous to Core Components. Furthermore, Components are essentially Basic Core Components, while Blocks & Assemblies are essentially Aggregate Core Components.

The X12 XML design has imposed three levels of granularity where CCTS has one. This is a very conscious decision to allow rational rules, appropriate to the wide range in sizes, to be applied to each level.

These parts are neutral in both approaches, in that “context has not been applied”. Also in both approaches these parts are designed for reuse in a wide variety of messages.

The X12-Finance Invoice pilot XML project has demonstrated this equivalence and interchangeability. Approximately 70% of the Components needed were taken from ebXML financial Core Components. It is the intention of X12F, Finance, to submit the remaining 30% newly created Components to CEFACT for consideration as Core.

Context Applied Parts

Modules are analogous to Aggregate Business Information Entities. In both approaches these are constructed of the smaller “Neutral Parts”. Also in both approaches these are data structures of some size and importance.

These parts have “Context Applied” in both approaches. Both BIE’s and Modules are useful in particular business contexts. Both would be reusable only between business processes that share most-or-all contexts.

Again, X12-Finance will submit the modules developed in the pilot XML invoice work as BIE’s for consideration by CEFACT.

Messages

The X12 XML Design has explicit support for document assembly in the form of Documents, Templates, and Modules. Templates & Modules allow related messages to be directly constructed out of a common pool of parts. This approach allows similar messages to be identical in areas with identical needs, and to differ markedly where needed.

The X12 XML Design approach is also uniquely capable of supporting individual business needs by the use of proprietary “modules” in “standard” templates.

The CCTS, in its most recent draft out for review, is rather silent on how to create a “Message”. X12 Communications&Controls intends to propose that the Template and Document assembly notions be adopted in the CCTS.

XML Syntax & Schema

The X12 XML Design has details dedicated to both XML syntax and the production of schema for complete messages.

To the extent possible with mutually-evolving efforts, the X12 XML Design seeks to be compatible with the approaches being taken by the OASIS-UBL effort for schema.

- The CCTS is officially “Syntax-Neutral”; no mention is made of particular syntax details. While XML is the obvious syntax to express the CCTS work in, this is considered a separate area of work.

2891