

XML schema design for business-to-business e-commerce

Arofan T. Gregory

Commerce One, Mountain View, USA
arofan.gregory@commerceone.com

Abstract:

No abstract was provided for this paper.

Introduction

This talk addresses design considerations for maximizing the utility of XML schema languages in creating documents for business-to-business e-commerce applications, from the perspective of a developer designing an e-commerce system or a manager responsible for system implementation. In discussing the usefulness of XML Schemas, it touches on many related technologies, but restricts itself to the technological enabling of e-commerce, avoiding discussion of the many legal and business issues as having no direct bearing on schema design.

As never before, the emerging trends in business-to-business e-commerce are demanding greater levels of interoperability. This is very much a continuation of the original vision of SGML: standardization enabling the use of complex information across platforms and applications. The radical redefinition of the domain across which information may be reused has raised the stakes, and places much higher demands on the technology.

Of all current standards, XML Schema is the most important in living up to this challenge. The power of XML is also its greatest weakness: on the one hand, we can describe any data that our Internet applications care about, to whatever level of detail is needed. On the other hand, we all need to agree on what that data is. XML alone is not enough.

XML and EDI

The e-commerce world is clearly divided into two camps: EDI and XML. The interplay between these camps is interesting to watch - members of each have gone and flirted with the enemy, generally producing interesting (but not extremely useful) results. Both groups are faced with the same challenges and the same frustrations. They have different strengths, and different weaknesses. This section attempts to summarize how to combine the best of each, and how XML Schema gives us the ability to do this.

The EDI heritage

EDI - whether X12, UN/EDIFACT, or any other flavor - was an early technology that helped many large companies realize a tremendous vision, generally at great expense. In this way, it is extremely similar to SGML. The vision of electronic purchasing is very seductive: it increases efficiency, saves money, and offers capabilities that were not available to people working with traditional systems. EDI - used in its restrictive sense, because in truth we are all involved in "electronic document interchange" regardless of our preferred syntax - was the only way to realize this vision for many years.

The good

EDI is most notable in two ways: implementors from the EDI world have ten years and more of experience wrestling with the problems that have gone mainstream with the popularity of Internet-based B2B e-commerce. We are foolish to ignore this experience, whether in terms of the technology problems and issues that exist in this space, or in the ideas and lessons learned in the standardization process. EDI messages today provide us with the best standard descriptions of practically useful semantics for e-commerce.

It's not as simple as copying EDI messages in XML syntax, however - although many seem to feel that this approach is all we need. If you take any EDI "standard" message set - be it X12, EANCOM, UN/EDIFACT, OBI, or any other - you will find that many of the tags are not useful, or are not actually used in real-world implementations. A big part of leveraging the EDI experience lies in knowing which standard EDI constructs were successful and useful, and which ones were anomalous.

One good place to look for this kind of information is in the efforts to create common subsets of the EDI standards - the most prominent of these efforts is SIMPL-EDI, which recommends a set of "core" EDI constructs to be used for the standard procurement scenario. There are other, similar efforts elsewhere in the EDI world.

Another positive lesson to be gathered from the EDI world is the value of well-done documentation. As compared to SGML and XML DTDs, EDI implementation guidelines are generally superior. (As an example, look at the OBI specifications for the purchase order message, available for download on their website.)

The bad

EDI is not **only** a good thing, however. It is also a really bad thing. The more negative aspects of EDI languages include tag bloat and syntax overloading. Because implementation guidelines are not formally explicit, there is a high degree of latitude in how any given segment is used. Sometimes, developers simply use things for non-standard purposes. While it is true that a finite set of tags can describe all useful data structures, this approach generally results in a very large tag set that is difficult to learn and utilize. EDI has embraced this approach, and it has caused many problems with non-standard implementations.

EDI provides us with examples of messages that simultaneously provide multiple ways to encode a single bit of information, implemented such that some of the standard constructs are overloaded. This sort of thing is the direct result of the "kitchen sink" approach to building standards. As people in the EDI standards world are fond of saying: "everybody uses just 20% - the problem is that none of them use the **same** 20%."

Ultimately, many of these problems stem from having a syntax that does not have a mechanism for formal validation. There is no easy way to determine conformance, resulting in very high integration costs, tag bloat, and overloading.

The ugly

Consider the following illustrations, and ask yourself: "What am I looking at?"

20000305:102'DTM+158:20000305:102'DTM+159:20000722:102'NAD+SU+9876543
NY'NAD+MI+88835::92'GIS+37'NAD+ST+72681::92'LIN+++93235494:IN'PIA+1+0
04'REFF+ON:XXX00004'QTY+79:6660:EA'DTM+51:19991225:102'DTM+52:20000304
91225:102'DTM+11:20000302:102'SCC+1++W:16'QTY+1:960:EA'DTM+158:200003
20000313:102'SCC+4++W:16'QTY+1:900:EA'DTM+158:20000320:102'QTY+1:900:
:1080:EA'DTM+158:20000403:102'QTY+1:1080:EA'DTM+158:20000410:102'QTY+
'QTY+1:630:EA'DTM+158:20000424:102'QTY+1:990:EA'DTM+158:20000501:102'
:102'QTY+1:810:EA'DTM+158:20000515:102'QTY+1:810:EA'DTM+158:20000522:
0529:102'QTY+1:810:EA'DTM+158:20000605:102'QTY+1:630:EA'DTM+158:20000
20000619:102'QTY+1:810:EA'DTM+158:20000626:102'QTY+1:810:EA'DTM+158:2
158:20000710:102'QTY+1:766:EA'DTM+158:20000717:102'SCC+2'QTY+3:12610:
:20000416:102'SCC+3'QTY+3:17485:EA'DTM+51:19991225:102'DTM+52:2000052
0:EA'UNT+73+770001'UNZ+1+77'UNB+UNOA:2+BFT:ZZ+CAI:ZZ+000305:2338+78++

```

<!-- InvoiceParties contains names and address of parties and -->
<!-- their functions -->
<InvoiceParties>
  <Buyer>
    <NameAddress>
      <Name1>Ralph`s Automotive Parts</Name1>
      <Address1>10 Main St.</Address1>
      <City>Boulder Creek</City>
      <StateOrProvince>California</StateOrProvince>
      <PostalCode>96005</PostalCode>
      <Country>US</Country>
    </NameAddress>
  </Buyer>
  <Supplier>
    <NameAddress>
      <Name1>ABC Wholesale</Name1>
      <Address1>1222 Industrial Park way</Address1>
      <City>South San Francisco</City>
      <StateOrProvince>California</StateOrProvince>
      <PostalCode>96045</PostalCode>
      <Country>US</Country>
    </NameAddress>
  </Supplier>
</InvoiceParties>

```

(I don't know about you, but I don't ever want to hear my five-year-old daughter ask, "Mommy, why did Daddy go blind?")

What does XML bring to the party?

As demonstrated above, XML represents a relief from some of the more negative aspects of EDI. It has many well-known minor benefits: it is cross-platform, it understands Unicode natively, it is human-readable, etc. These are not the specific items that set it apart from EDI in the most important way, however.

Easily processable syntax

The fact that XML is a formal meta-language that is susceptible to standard machine validation is a major differentiator. The existence of XML parsers means that applications don't have to do their own structural validation. This alone reduces the cost of integration by minimizing the amount of code that needs to be written by developers.

Further, it is very easy to render XML documents as DOM trees, event streams, or other programmatic models. Many of these are supported in standard development packages, further reducing the cost of building applications. Ultimately, this stems from having a syntax that is designed to carry a wealth of metadata, and to make it available to processing systems.

Tools

XML - as opposed to SGML - increasingly has good tools support. Browsers, development tools, code libraries, distributed programming objects like COM and javabeans, repositories, databases, editing tools - the list of XML-enabled software packages is long and growing. These are not all specialized tools for working with just XML - in many cases, traditional enterprise tools have expanded to support XML. Not only will this reduce the cost of software, but also the cost of integration and system maintenance - XML programming skills are becoming increasingly more common.

"Webbiness"

Another - albeit less measurable - advantage of XML is cultural. Because XML is a web technology, it encourages experimentation and the "running code" ethic. Thinking tends to be more revolutionary, and developers at all levels tend to be self-empowered. EDI, as a result of its long history, tends to be a more conservative type of high-tech culture. In some ways, the "webbiness" of XML enables us to address old problems with fresh energy and enthusiasm.

Why XML is not an unmixed blessing

Just as with EDI, XML is not an unmixed blessing. It has been misused by those in the EDI world as well as those from the XML camp.

Ignorant application

Something that struck many people in the EDI world as an obvious way of doing "XML EDI" was to take the EDI syntax and describe it with tags. Thus, you could have an element like `<SEGMENT CODE="17485" />`. There is a real problem with this approach: it allows you to pass EDI messages back and forth over the wire, but does not allow you to leverage any of the structural validation that is an inherent part of an XML implementation. (You will notice that most EDI users are already capable of sending EDI messages over the wire to those who can understand them.) The point of XML is that it makes business semantics easy to read, easy to process, and easy to validate. This approach to XML-EDI does none of these things.

To their credit, both X12 and UN/CEFACT have independently realized this. This phenomenon takes place because it allows EDI users to say that they are "doing XML," without having to put any additional effort into doing it **right**. What is sacrificed in doing this is any meaningful benefit that would result from using the new technology.

Arrogant newcomers

The arrogance with which some members of the XML community approach the very real challenges of business-to-business e-commerce is truly astounding. In truth, this is more indicative of naivete than it is of general personality defects - XML does possess many technological advantages in solving the same problems as EDI, but that does not mean that the problems themselves are simple. A corollary to the kind of arrogance that is sometimes seen when XML developers do "EDI" is taking a completely US-centric view of the problem. International trade has many complexities that are not found in domestic e-commerce in the US.

"Messages" aren't (only) documents!

Another negative aspect of XML technology is that it comes to us from an information-publishing tradition, with its own set of inherent assumptions about what aspects of functionality are important. This is most evident when we consider how XML DTDs are used to validate documents: they can validate the structural aspects of documents with great power, telling us where the rules have been violated, and how. What they fail to do usefully is datatyping: there is one basic "string" datatype used for leaf-node element content, and a set of strange and not terribly useful datatypes for attribute values (loose numeric types, strings without whitespace, etc.)

The key thing to realize here is that EDI messages are "documents," in one sense, but they are also something more - they are "messages" which require very strong data validation to be useful in an e-commerce setting. One can look at e-commerce as a set of relational systems talking to each other over the wire with documents/messages. It becomes easy to see why having strong data validation - the kind of validation that evolved in the EDI world around codelists, for example - is so important. XML wasn't built to do this kind of validation.

Schema as the answer

XML DTDs are simply not enough. However, XML schemas allow us to solve many of the technical challenges presented by e-commerce requirements, and to leverage the best of EDI technology.

Strong datatyping

Schema languages contain native datatypes that align with the standard datatypes of some programming languages: strong numeric types like signed integers, floats, doubles, etc.; enumerated lists; date, time, and datetime; and, of course, strings. XML Schema languages such as SOX, XDR, and the W3C Recommendation for XSDL further allow us to describe custom datatypes that are even more useful for particular applications. It becomes possible to capture code-lists as enumerations, to limit the length of string types to fit into typical database fields, and to describe numeric types very exactly.

Manageability and componentization

Further, some schema languages (SOX and XSDL, for example, and to some extent XDR) have inherent capabilities for reusing components by doing the kind of importing and referencing that are found in programming languages such as Java. It becomes possible to take another person's schemas - assuming they are publically available in the correct schema language - and to incorporate them into schemas that you are creating.

This is not a problem-free area, largely because of a lack of a standard scoping mechanism. There are some solutions here, however: the ability to use hierarchically structured namespaces in SOX, for example, allows tight control of element re-use, something not possible with the standard XML namespaces and DTDs. XSDL makes provision for even better scope control, and this could potentially become a standard mechanism for component reuse.

Component reuse is very important when we are seeking to produce interoperability, simply by helping us to guarantee leaf-level correspondence across business documents. (This point will be addressed in more detail below).

Extension/refinement: less is more (more or less...)

Another aspect of XML schema that helps us solve the tough problems of e-commerce is the ability to not only reuse another person's data models, but to "refine" them. In the best case, this mechanism gives us access to the chain of inheritance of a given data structure, in the same way that class-based object-oriented programming languages do. You can take another person's data type or element structure, and "subclass" it by adding (and, in the case of XSDL, subtracting) pieces that you need specifically. At runtime, processing applications can identify what the differences are between the parent class and its subclasses, enabling polymorphic processing (that is, using a child in place of the parent, where the parent was required) and default processing (handling just those bits of the data that you understand, instead of choking because some of the data is not what was expected).

The application of this capability from a design perspective is that we can build minimized components in anticipation that they will be extended to fit particular requirements. Less becomes more.

For supporting Global interoperability, the ability to extend, reuse, rename, and refine other people's components is a major enabling technology. While the capability alone is not sufficient - we must have well-documented (and preferably automated) methodologies for doing this - it does give us the raw power to solve many difficult problems in accommodating required variation within a class of business documents.

Data and documents

Schema - when properly employed - also makes it fairly easy to translate document structures (the business "message") into normalized relational structures. This is one of the places where XDR has been effectively used. If we remember that e-commerce can be seen as a system in which relational systems trade information by transmitting documents, the "graphing" capabilities of XML schema languages can provide us with lots of power. DTDs couldn't do this effectively - especially since the strongest datatyping (such as it was) existed in attribute values, and the simplest way to handle graphing requirements was to use element structures. DTDs just weren't designed to support this requirement.

How do you work this thing?

The title of this section is taken from a song describing the plight of those overwhelmed by any type of technology - whether primeval biological technology or the kind of meta-data-laden, heavily abstracted technology we are dealing with here - without having recourse to the reference manual as an option.

How do you work this thing? / That's what I wanna find out. / Everything it connects with / Only leaves a kink in the spout. (Robyn Hitchcock)

In many ways, this describes the internal dialogue of the typical would-be XML Schema developer. I hope to point out many of the issues that should be taken into consideration, and to suggest a basic process for writing schemas for e-commerce, as distinct from traditional document analysis activities for producing DTDs.

The goal of business libraries: interoperability

It is always a good idea to remember what your end goal is, and when you are creating XML Schemas to describe business documents, the basic requirement is simple: interoperability. Even when business documents are intended to be used in a point-to-point scenario (and I believe that this kind of implementation will become less and less common as the e-commerce revolution moves forward), you are still designing documents that hopefully will be usable by trading partners you do not do business with today.

In scenarios where there is some kind of business portal or trading community, the requirements for interoperability rise rapidly. The whole point of having portal is being able to easily communicate with many other members of the community, without having to do a one-off integration with their business systems.

Business-to-business vs. application-to-application

One of the issues here is the debate between business-to-business approaches to e-commerce interoperability, and application-to-application approaches to the same problems. What lies at the heart of this debate is a question of what it is that we are encoding in our documents - which set of metadata is most important.

Business-to-business applications focus on business semantics, using an approach that is as old as SGML - "If you tell me what it is, I can figure out what to do with it." Because this is a highly abstracted approach to using document interfaces for e-commerce systems, it is a tougher problem space than that addressed by application-to-application approaches. These - by interoperating at the level of the processes performed by typical business systems - pass programmatic interfaces in document form, as opposed to semantic ones.

There is a problem with this approach, however, that becomes worse over time. We live in an age when the capabilities of the most common processing applications grow rapidly, and the predominant systems today may not be in use at all in two years. By hard-coding the application interfaces of our systems into our business documents, we are virtually guaranteeing a maintenance nightmare for ourselves. It is easier in the short term, but more expensive in the long term.

Vertical orientation vs. horizontal orientation

Another way to look at interoperability issues is to examine the kinds of standard business languages that have been proposed to date (and the list is long, and growing). In both the EDI world and the XML world, there has been a phenomenon in which, to render the problem of defining useful semantics tractable, a vertical industry has described business data with the semantics that are commonly used within that vertical. This was a very common - and generally successful approach - used in the SGML world for technical documentation: DocBook, CALS, and PCIS are all examples of the successful use of this approach.

The problem of interoperability is greater in the e-commerce space, however, and for a very simple reason. While many businesses within an industry use a single set of semantics to refer to the technical information describing their products, buyers in the e-commerce world inevitably want to purchase goods from other industry verticals. Some very successful industry-specific initiatives such as IOTP and RosettaNet have a good deal of "horizontal" value, but because they have defined semantics in terms of a vertical industry, they have a hard time translating into other vertical industry sectors.

A suggested process for creating your business documents

What follows is a brief description of the basic process that I would recommend for those looking to create business documents. I recently saw a presentation in which the speaker claimed that there was no documented methodology for writing business schemas. This is both true and false: while there is not a specific literature around the creation of XML schemas for e-commerce, there is a parallel literature that can be of great value, coming from the SGML methodologies for performing document analysis and writing DTDs. This section is presented in reference to the "known" document analysis approach to building these types of applications. For those looking for a single source of information, I would recommend the book "Developing SGML DTDs" (Maler, Eve; El Andaloussi, Jeanne. Developing SGML DTDs: From Text to

Model to Markup. Englewood Cliffs, NJ: PTR Prentice Hall, 1996. Extent: 560 pages. ISBN: 0-13-309881-8.)

Process outline

The basic steps for creating document structures such as schemas and DTDs are very similar to those of any kind of software application or systems development. This is just a quick outline - more detail is given below about particular points of interest.

1. **Define System-Level Requirements:** This includes looking at what your users need, what kinds of software systems are involved, and what kind of information you are designing for. In the case of traditional DTD document analysis, the scope of the exercise was generally the enterprise, or at most a particular industry. In the case of e-commerce schemas, the trading community is your scope. It is much larger, has more users, and generally presents you with a larger problem space. The end result of this activity is a set of requirements that will point out what kinds of documents you need to create, and what your design principles will need to be. In the case of e-commerce, it is important to focus on the business process that you are enabling, because you are not looking at single document types, but groups of documents that make up the exchanges in dialogs. Users are not only (or primarily) human: they tend to be computers performing transactions. However, humans are important, too - they are systems developers and the operations people who have to maintain your applications and resolve errors.
2. **Create a Design:** This activity involves looking at the requirements and figuring out how you will meet them. **Before you start designing, you will need to do your homework!** This means doing research into what standards work is available, and it means examining other efforts at solving the same problems, both in the XML world and in the EDI world. When doing XML schema development for e-commerce, you never need to start from scratch, and you are a fool if you do. The last thing the world needs is another one-off XML business library. The end result of this activity is a thorough written record of the design, including commentary on what sources were used and what was determined about them. This document should be created in reference to the requirements and design principles coming from step 1.
3. **Developing the Schemas:** Code. This involves choosing a schema language (or several) sufficient to meet your needs, and producing a first draft of the documents themselves. If possible, it is a good idea to do a reference implementation so that the schemas can be tested. Also, getting a wide body of reviewers is always a good idea - in this case, that means users from each class identified, and preferably getting trading partners you did not interview in your requirement-gathering stage to comment on whether they could use what you have produced. What comes next is a standard cycle of testing, debugging, and testing again.
4. **Document the Schemas:** Most schema languages have a facility for putting commenting inside the code. This is useful, but not enough. Produce thorough and professional documentation, based on your design documents and the records of your development cycle.
5. **Publish your Work:** This is best done through the public repositories that are being established: XML.org, BizTalk.org, etc., but will depend on your particular scope of effort. Your particular trading community may have its own points of distribution.

A couple of minor tips

In my experience, it is a very good idea not to use mixed content in XML business documents. Some schema languages - such as SOX - don't even allow it. Whitespace handling in XML is a topic of much debate, but it can be fairly neatly avoided by simply not allowing mixed content in your documents.

Don't be afraid to "plagiarize." When defining element structures, data types, codelists, etc. it is a really good idea to literally use what others have produced, particularly if there are relevant standards for what you are doing. In the US, there is a legal argument that claims that "business forms" such as XML schemas are not even intellectual property. Most people and organizations who publish XML schemas and implementation guidelines are happy to see others build on what they have done.

Design for extension and refinement. If you correctly analyze your data structures in terms of the type relationships, you will establish a set of archtypal components that can then be usefully subclassed. This approach will maximize the flexibility of your data structures, and can be seen at work in many places - look at xCBL, ebXML, and even the modelling work done in the UN/EDIFACT working groups.

Existing resources - using EDI

There are many EDI sources that are extremely useful when creating business schemas. Chief among these are the standards bodies or similar initiatives (X12, UN/CEFACT, EAN, OBI, RosettaNet). Typically, you can get implementation guidelines that will give you fully documented transaction sets consisting of the documents needed to carry out a particular exchange. In some cases there are also good process models.

It is strongly suggested that, when looking at any given EDI transaction set, that you have recourse to someone who has worked with that particular transaction set before, and understands which segments are used in reality, and which ones are hold-overs still included only for legacy support, or are simply not used.

The EDI messages represent the single best source of information today regarding the semantics of e-commerce messages. There are many forward-thinking movements within the EDI world that are worth paying attention to: the BSR, object-oriented EDI, interactive EDI, and others.

Initiatives, standards, and libraries - using XML

What follows is a list of some of the better resources that exist inside the XML world. In some cases, these initiatives span both the EDI and the XML technology spheres. Each has a brief description, but is worth taking a close look at.

1. **CommerceNet eCo Initiative:** CommerceNet is a non-profit consortium of companies concerned with e-commerce in the US. They have published a set of architecture specifications, along with some recommendations about building business semantics, their eCo project. These serve as the one of the foundations for the work currently going on in the ebXML initiative.
2. **ebXML:** This is a joint project of OASIS and UN/CEFACT, covering architectural and semantic harmonization between EDI and the XML worlds. It is attempting to model data structures in a syntax-neutral way, guaranteeing that round-trip transformations are possible among all sets of business semantics. This work is described in greater detail below.

3. **OBI:** The "Open Buying on the Internet" group has done a lot of good work in describing some common processes and implementation guidelines to help standardize how e-commerce is done today. They have published some XML DTDs, but these are currently being re-visited, as they did not (in this author's opinion) stand up to the otherwise high-quality work of this group. There is a supply-side emphasis here, and the involvement of a portal in their processes has not been included, but this is an excellent place to go to get a good idea of what exists.
4. **RosettaNet:** Originally intended to standardize the high-tech supply chain for semi-conductors, computer parts, etc., RosettaNet has produced some truly excellent work, especially in the area of process description (the "PIPs"). This is another group that has published both implementation guidelines and XML DTDs, and again, the DTDs do not represent the best of their kind, making no attempt to leverage the many benefits of XML Schema. This is a good place to go for descriptions of data sets, however, and it may well produce something more interesting in the XML line now that the W3C has produced a candidate recommendation for XML Schemas.
5. **OAG:** The Open Applications Group has done some very good work, mostly in the application-to-application space, but with some cross-over into the arena of business semantics. Again, they first published a set of DTDs that did not leverage much of what is possible with XML, but they are another good source to consider, and are another initiative to watch. They are not necessarily something to copy - see my comments regarding application-to-application messages exchanges above - but they should not be ignored, either.
6. **BizTalk:** Microsoft's BizTalk initiative has many parts: it is a repository for distributing other people's XML business schemas in XDR (and deserves a browse, just to see what's there) and it also represents a standard architecture for building e-commerce servers within (and between) enterprises. The BizTalk messaging specification is an XML "envelope" for enclosing any business documents in a way that is compatible with their architecture. Although BizTalk does not support portal-based exchanges today, that will probably be included in future versions. This is something that should be examined by anyone doing research, as it covers most of the major issues surrounding (but not including) the actual creation of a set of XML messages for e-commerce.
7. **BASDA:** BASDA is a European consortium of bankers that has produced some very good schemas for some of the basic message types in the e-commerce space (Purchase Orders, for example). They do not leverage all of the cutting-edge capabilities of extensions and refinement, nor do they illustrate how component re-use can be managed, but their schemas are very worth consideration as thoughtful (and complete) examples of their type.
8. **xCBL:** xCBL, the "XML Common Business Library," is an attempt to produce an XML version EDI semantics in such a way that they leverage the best features of XML schemas. Based on the SOX (Schemas for Object-Oriented XML) schema language, but distributed in DTD and XDR form as well, xCBL is the best existing example of how the refinement features of XML schemas can be used to build a component library. The SOX parser given away as the core of Commerce One's XML Developer's Kit has some powerful features such as support for polymorphism and an ability to handle an extension syntax not too different from that appearing in the W3C XML Schema candidate recommendation. Although xCBL was created and is maintained by Commerce One, it is intended to be useful for any e-commerce applications based on XML. It is capable of being used with business communities that exist in portals, and does not assume that trading will be point-to-point. Although Commerce One's applications are based on the SOX version of xCBL, it is not restricted in intent to their products, and has been used in implementations based entirely on other software packages.
9. **CXML:** CXML is Ariba's proprietary data formats encoded as XML DTDS. Despite the existence of CXML.org, this is nothing more than the data structures used inside Ariba's applications, and it's

creators have admitted as much. It is useful as an example of the data sets needed to support real-world trading, however, within the processes supported by the application from which it derives. One thing to note, however, is that it does not provide a clear separation between "messaging" information and the business document data, which is a problem that should not be duplicated.

Process and context, messaging, security, and architecture

There are a whole set of "architectural" aspects to e-commerce systems that cannot safely be ignored when designing schemas to describe the messages exchanged by applications. This section gives a brief summary of the major areas from this perspective.

Process and context

The EDI standards world has developed a very intriguing notion of "context," which is the set of descriptors that define a single point within the entire realm of e-commerce exchanges. "Context" involves not only the specific business process, but also such factors as the industry which that process is serving, what regional or international aspects exist, and so on.

The idea is that "context" can provide a useful way of characterizing the applicability of a particular message within the entire e-commerce space. In reality, one Purchase Order cannot be used by everyone, because there are different demands on the data that must be transmitted within different contexts to support the same basic business process. (The joke goes, "One size does **not** fit all, and size matters!")

There are also a number of interested groups that have attempted to define standard business processes and sub-processes, most often using UML as a modelling technique, and some XML-based syntax for describing the process flow. (This is very important work, and the best example is XMI.) ebXML is using both context and processable XML "choreography" descriptions to help define the way in which data components may be used and extended, and is further attempting to define reusable sub-processes that will themselves function as components for describing business processes.

When designing documents meant to maximize interoperability, one must consider the effects of different business processes and contexts. This will influence the way in which data elements are extended and refined, particularly, and may well impact how a componentization scheme is organized.

XML messaging

XML messaging is an interesting topic, but is not one that we will cover here, other than to point out one important rule: **Do not mix "messaging" information with business data.** What this means is that you should let things like guaranteed delivery, transmission errors, addressing across multiple protocols, and routing information live at the level of the "envelope" that carries your message, whether that is a BizTalk wrapper, a MIME envelope, or anything else. Don't require a delivery application to read your business document, or you will find yourself severely limited as to how your documents can be used.

Digitally signed XML

Digital signatures require that, once signed, an XML file not be edited. Any editing will invalidate the signature. This has a tremendous impact on how you design your documents, and is one of the main drivers for the point above. When analyzing a business process flow to create the supporting messages, be very careful that you do not require a message to be edited after it has been created and signed.

Summary

This discussion of how to create business documents with XML schemas would not be complete without pointing out a simple, important fact: **You probably will not have to write your own business documents.** If you're smart, you will leverage all of the work that already exists in the various standards and examples discussed above, which means that it may be easier to simply tweak someone's existing product, and hopefully do it with their cooperation and within their customization methodology.

If any single effort currently deserves attention, it is the ebXML Harmonization effort. This will be further discussed in the next section, but is probably the best example to date of an attempt to truly reap the benefits of both EDI and XML.

Where no one's gone before

The use of XML schemas in business-to-business e-commerce is in its infancy. The simple fact that the W3C was late producing a draft recommendation for a standard XML schema language has done much to halt the use of schemas in e-commerce. This is rapidly changing, and the next section is an attempt to describe those aspects of this area that deserve notice, and that will potentially impact your decisions about schema design.

Interoperability and scalability: future requirements and capabilities

First, we must look at what the requirements are likely to be from the standpoint of scalability.

EDI was small

EDI was used by a relatively small number of organizations. Due to the prevalence of the internet inside the business world, and the growing availability of software packages that can be used by small- and medium-sized businesses to get into the game, we can anticipate a much larger number of trading partners than has heretofore been the case.

This is especially true because the initial cost to get into the e-commerce game is being greatly reduced by the web-based portal phenomenon. If you only need to integrate once to connect with a reasonable number of trading partners, then we will see that issues around scalability are directly more critical.

Dynamic trading

Another aspect of e-commerce in the future is that trading will become more dynamic. With architectures designed to provide rich metadata about the product and service offerings on the web, then it should become possible to negotiate contracts, and conduct transactions, in real time using computer programs designed to leverage this metadata.

The implications of this for schema design are many, but mostly focus around the assumptions a designer can make about how data must be described. It is no longer enough to determine the semantics used by two trading partners, and to encode that set of semantics into their business documents. The burden will be to design generally useful semantics that can be easily extended to account for nuances introduced by new trading partners. This is not an easy task! Polymorphism, of course, will be very helpful if we do our document design properly. by designing super-classes intended to be renamed for specific use.

Portal services vs. point-to-point

In the EDI world, almost all business services (as distinct from VAN-based services around actually transmitting the messages) were provided either by the buyer or the supplier. There is no concept of value-added "middle-man" services at the level of the trading community. The existence of portals is changing this. Portals can now serve as a place where value can be added in many different ways to support many different types of transactions, from translation of catalog content to payment processing to tracking of shipping to visibility up and down the supply chain.

Because this kind of services provision impacts the flow of a business process (its "choreography" in terms of message exchange) it directly impacts the data that is carried by each of the messages making up the overall process.

Negotiating business process and contracts

Further, in a dynamic trading world, it is possible that business process and contracts will be negotiated on the fly, according to metadata that trading partners supply about themselves. This places a huge premium on being able to handle different "contexts" with the same basic message, and to be able to vary the content of particular messages based on how the business-process exchange is defined at run time.

Maintenance of systems across the virtual marketplace

Virtual marketplaces existing at specific portals make the problem of software version-compatibility a nightmare. With enough metadata, and with usable default processing, componentization, and support for polymorphism, the issues of backward compatibility are greatly simplified. If your trading partner can describe which versions of which data objects they support, then all-at-once migration from one message format to another is not required for every member of the trading community.

One good example of how such a problem can be solved is to look at DCOM, and to extrapolate from there how the finite nature of the problem space can be used to build on this type of example.

Industry-vertical information demands

One aspect of industry-specific marketplaces will be having a mechanism to accommodate information sets specific to that industry. Again, this problem can be solved through good design, and the judicious application of extensions, default processing, and polymorphism. This is really an aspect of "context," and one that will consistently make demands on our "standard" message structures.

Global trading

One of the great promises of e-commerce portals is the existence of a global network of virtual marketplaces, capable of being interconnected. The ability to monitor the international supply chain could solve many very difficult problems that have remained intractable for many years. Assuming that it is possible to create such a network - and theoretically, at least, it is - there are a number of issues that will come into play.

Regional variation

Yet another aspect of "context," the differing trading practices, regulations, and conventions existing in different regions of the world will need to be accommodated, putting even more emphasis on the ability to default process and extend/refine data structures. This also has many architectural implications, but these are mostly not within the realm of business document design.

Customs, transport, and the supply chain

International trade has long been plagued by a set of very tough requirements, many the legacy of complex, paper-based systems. In order to determine the cost of a purchase, the cost of shipping must be determined. Transport - often multi-modal, and by definition crossing international boundaries - must be arranged, and letters of credit must be requested and obtained. Even without considering the various forms of insurance, repackaging, and the third-party arrangements through which much of this trade is accomplished, we can see the need for an inter-connected network of transactions, whose documents are integral to the processing of other transactions. (For example, without proof of various other transactions, many banks will not honor letters of credit.)

There are several impacts on business document design: the focus shifts from designing a single document, to identifying cross-document reuse of information, such that it can be managed across the entire interlocked set of transactions. A premium is placed on having a solid componentization scheme, and further, being able to easily place this information into persistent database stores that can help manage the entire flow, and serve as a source for the required paper documentation that enables shipments to pass through customs.

While this sounds daunting, the promise here is immense. If we look at the current inability to track the location of multimodal transport en route - mostly because different shippers use different tracking numbers - then we can see that there is a huge value to managing the supply chain. Annoying issues such as government-required statistics reporting become simple value-add services at the portal. The list goes on and on. Within the EDIFACT world, there is an awareness of many of these problems. What is lacking is any idea of how to implement a solution. With schema-based messages, we begin to see our way to how such a system could be built.

Internationalization and translation requirements

Translation of business information is not much needed, with the notable exception of catalog content. At the same time - and especially with international trade documents, which frequently contain several languages and currencies as different parts of the same document - these issues cannot safely be ignored. To give a brief list of examples of the effects of multi-language and international support requirements on business documents:

We must be able to distinguish between local dates and times, and dates and times relative to Greenwich Meridian, to accommodate trading partners in other time zones.

We must be able to handle multiple currencies, frequently within the same document (shipping costs, for example, are typically charged by the shipper's native currency, which may not match those of the buyer and/or supplier. Such currencies indicators must be attached to the data structures at the appropriate level of granularity.

We must be able to handle multiple languages within a single document, as the contractual clauses of international trade documents are often in both the seller's language and the buyers.

One thing to take into consideration is the extreme difficulty of combining good structural validation and multiple "parallel" language-versions of content in a single document. Add to this the fact that most tag-enabled translation programs are not set up to handle this kind of data structure, and you will quickly see that having parallel documents makes much more sense than having parallel elements in the same document. Note that this places a requirement to have enough data to be able to establish this association between documents.

The repository

As we get further into the possibilities moving ahead, we begin to find that repositories, supplying schemas and other information to processes at run time are a critical aspect of the portal-based systems. Rather than being mere distribution points, repositories will need to be well-designed to meet the needs of run-time systems. We may find that the standard registry/repository interfaces currently being proposed by XML.org and ebXML will be critical in enabling a global network of trading communities. Such functionality as default processing and polymorphism requires access to the extended schemas, and negotiated choreographies and business contract will also require trading-partner profile information that will only be encountered at run-time.

All of the significant architecture specifications - CommerceNet's eCo Architecture; ebXML, and BizTalk, to name three - understand the importance of having a functional repository to perform these kinds of tasks.

Summary

When HTML was first published, the hypertext experts of the SGML world claimed that it was insufficient to do "serious" hypertext. No doubt they were right, but the sheer demand for any amount of functionality in this area, and the simplicity of HTML, made it a phenomenon. Now, more "serious" applications of Web-based technology are driving the data formats used in more complicated directions. Undoubtedly, we will make many mistakes in the use of XML-based technology in the e-commerce arena, and hopefully learn from them. It is also true, however, that the use of XML schemas is a critical part of developing usable systems in this area, and today is a focus for the development of useful data structures for Internet-based trading.

Author

Arofan T. Gregory

Lead Scientist and Manager, XML Common Business Library

Commerce One

Document Engineering

Postal Address:

Mountain View

CA

USA

E-mail: arofan.gregory@commerceone.com

Arofan Gregory — Arofan Gregory currently maintains the "XML Common Business Library" for Commerce One. He worked for seven years in commercial book and journal publishing before becoming an SGML consultant at Passage Systems. After working as a consultant and manager of consulting there, he started his own consultancy, Aeon, also specializing in practical SGML implementation. Before coming to Commerce One, he worked for Documentum, Inc., as their Practice Leader for Technical Publications, with a focus on SGML and XML systems for repository-based Internet publishing. He has presented at major SGML and XML conferences, most recently at XML '99, and has served as an editor for the CommerceNet eCo Semantics Specification for XML business documents. He is an active participant in the ebXML harmonization initiative, and sits on MicroSoft's BizTalk Steering Committee.