Ontologizing EDI: First Steps and Initial Experience

Douglas Foxvog, Christoph Bussler
Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway
Doug.Foxvog@DERI.org, Chris.Bussler@DERI.org

Abstract

Electronic Data Interchange (EDI) standards for transmission of business messages were promulgated in the 1970s. As standards became more complex to provide additional message types and variants, programmers and businesses started calling for simpler, semantically-enabled, systems; some even predicting traditional EDI's imminent demise. Numerous projects to create successor systems using XML were unable to stop traditional EDI's growth. DERI is taking a different approach: ontologizing EDI to allow the creation of semantically enabled messages, while being backward compatible with traditional systems. We argue that ontologizing syntax first enables automatic calculation of the intersection of message formats necessary for initiating new EDI partnerships as well as provides the groundwork for ontologizing message meanings.

1. Introduction

With the emergence of computer networks it quickly became apparent to the business community that business relevant events like purchase orders, invoices, and payment advices can be sent electronically. This tremendously increases speed and accuracy over traditional paper-based means. The increase in throughput was highly beneficial to those businesses that have to process a high volume of business messages.

As a consequence electronic document standards first appeared in the 1970s with SWIFT [7] for electronic banking introduced in 1973, ANSI ASC X12 [16] for general business transactions in 1979, and EDIFACT [4] in 1988. These standards define the structure of business messages and the promise is that if businesses comply with those, the transmission becomes efficient. To large extent this is true. Once the transmission systems are set up, and the messages are created and consumed properly, the transmission is efficient.

However, as messages have been designed only syntactically, the set-up time is quite involved as it often takes developers a long time to agree on the precise meaning of the message content. A classic example is the shipping date. Does the shipping date refer to the date the supplier transfers the goods to the shipper or the date when the transport vehicle departs the premises?

A semantic definition would have precisely defined this and the ambiguity would not have come up. However, the standards did – and still do not – specify these semantic definitions.

Our goal is precisely this. We are taking version 5010 of X12 and describing it semantically (the process is termed 'ontologizing'). The benefit is apparent: once a standard is semantically defined, the use of it becomes easy as the interpretation of messages is clearly and formally defined.

In this paper we describe our first steps taken and first experiences. The work is not yet complete; however, initial results are ready to be shared.

Section 2 provides background of EDI itself. Section 3 discusses advantages and disadvantages of EDI while Section 4 describes alternative standards designed to overcome disadvantages of EDI. Section 5 explains the ontologizing process and its results in more detail. Section 6 discusses related work; and Section 7 shows our future plans.

2. Background of EDI

An electronic data interchange (EDI) system was set up in the 1970's for the exchange of documents in standardized electronic form between computer application programs in different organizations. Hundreds of standard message types were defined having specified formats. Each standard message type is based on a data model for a single type of inter-organizational transaction.

EDI specifies the formats of these messages in terms of message subcomponents, which in turn have their formats specified. The most basic message components are data elements, which can be filled with elements of specified code sets, numbers, dates of specified formats, or in some

instances free text. Components may be mandatory or optional, be allowed to repeat a maximal number of times, and – for data elements – be expressed with a minimum and maximum number of characters. Inter-component restrictions are also specified.

There are two major EDI standards, EDIFACT [4], defined as an open standard by the United Nations, and X12 [16], primarily used in the United States. Each of these standards comes out with versions several times per year – normally with incremental additions or deletions to the permitted formats.

X12 calls message types "Transaction Sets" which are composed of strings and loops of "Data Segments" in a specified format. Each Data Segment, similarly, has a specified format of "Data Elements". Some Data Elements are themselves "Composite" having a formatted sequence of simple Data Elements. EDIFACT uses a similar, but not identical, structure with differently named message component types.

```
ST*270*0001>
BHT*0016*00>
HL*1**21*1>
NM1*85*2******C2*987654>
HL*2*1*22>
TRN*1*REF1234>
NM1*QC*1*Bussler*Chris**Dr**21*IE87654321>
EQ*01>
SE*9*0001>
```

Figure 1: Example Transaction Set in X12 Syntax

A simple X12 Transaction Set is shown in Figure 1. This is an example of Eligibility, Coverage or Benefit Inquiry (Transaction Set 270). Each line is a Data Segment which starts with the code for that Data Segment followed by its Data Elements separated by formatting characters. Separators demark skipped optional Data Elements if later Data Elements are included.

The standards are broad enough to encode variations in message requirements used by different industries. Industry bodies define subsets of and modifications to the EDI standard for messages which they use. Individual companies add further restrictions to the EDI messages which they send and accept. These modifications are applied in order to adjust the standard definition to specific needs of the companies.

3. Advantages and disadvantages of EDI

Hundreds of thousands of companies use either ANSI X12 or EDIFACT or both of these systems [21]. They find EDI advantageous because it obviates a lot of paperwork and speeds up transactions, with computer programs processing documents received from business partners and generating such documents for transmission. EDI documents provide a legal record of business communications [25]. In addition to the expensive value-added networks (VANs) which were required in the past, EDI users are now able to transmit their data encrypted over the Internet at the far lower Internet connection costs. EDIINT [3] standards for email ("AS1"), HTTP/HTTPS ("AS2"), and ftp ("AS3"), which are implemented by many vendors, enable this.

However, the high start-up cost for adding EDI to a business system and the significant additional costs for initiating new relationships has kept many small and mid-sized companies from adopting EDI messaging [2,5,7,12].

The X12 standard is so large and general – in order to cover all possible message types used by any company or industry – that different industries and corporations have defined different subsets of the standard to cover those message types and variations of message formats which they use. The use of different standards is so prevalent that when two corporations wish to start up EDI communications they must negotiate a technical agreement to define exactly what subset of EDI they will use [14]. Such an agreement is called an implementation guideline in the industry.

These EDI variants define some optional EDI components as mandatory and others as forbidden, specify additional inter-component restrictions, identify a subset of codes within used code sets that will be accepted and used, may add additional codes, and restrict the Transaction Sets that will be used.

These issues are so complex that outsourcing of EDI set-up is being used by companies to reduce the time and expense of initiating an EDI relationship [2].

The lack of semantic rigor in the meanings of various components of EDI messages can become an issue [13]. If business partners do not formally agree upon mutually acceptable definitions, problems may arise later. For example, they must mutually decide if a "shipping date" is the date a product leaves the supplier's warehouse, the date that it is loaded onto the transport vehicle, the date that it leaves the supplier's premises, or the date on which it is placed on an international transport vehicle (e.g. ship or plane).

A non-semantically-enabled EDI is unable to interface with Semantic Web Services [23]. Semantic Web Service mediators [17] would not be able to generate EDI messages from semantic requests as long as the semantics of EDI messages is undefined.

Before reporting on the ontologizing of EDI in Section 5, we discuss "competitive" standards proposals to introduce other attempts to overcome EDI-specific disadvantages.

4. Intended successor systems

Beginning in the 1990s, perceived problems with EDI led to the development of systems intended as state-of-the-art replacements for the perceived antiquated ANSI X12 and EDIFACT systems. None of these systems was designed to be backward compatible with either X12 or EDIFACT.

Few of the new systems are semantically described. They mainly introduce a more open (object-oriented) syntax and therefore they have many of the same disadvantages as the traditional EDI standards.

Although a few of these systems are now being regularly used by major corporations, the number of companies using traditional EDI continues to grow [5].

The number of these standards and companies promoting them also continues to grow. Almost all of them are based on XML tagged files being transmitted over the Internet. A few of the more widely known systems are individually discussed in the remainder of this section.

4.1 ebXML

The ebXML standard was sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [27] and the Organization for the Advancement of Structured Information Standards (OASIS) [20] in 2001 as a modular suite of XML specifications designed to enable enterprises of any size or location to conduct business over the Internet [15]. Being based on XML, formats are a lot freer than in X12, with message components being named instead of having their identities derived from their position in a message.

Each company using ebXML publishes a "Collaboration Protocol Profile" (CPP) [19] which specifies the message types and message components which it uses. The intersection of two companies' CPPs can be used as a "Collaboration Protocol Agreement" (CPA) [19], or as the basis for a negotiated CPA, which specifies the message protocol between the two businesses.

The ebXML standard does not specify XML tags for message and message-part names. This allows parties to use the terminology inherent in their existing applications, but allows different parties to use different names for the same message and message-part types. They encourage the standardization of existing XML efforts but do not define standards that participants must use.

In this sense ebXML was intended to provide the infrastructure for message exchange without defining the message standard to be used by business partners themselves. ebXML was subsumed by UBL.

4.2 Universal Business Language (UBL)

Universal Business Language (UBL) is an effort started in 2003 to unify the plethora of XML standards for business documents [6]. UBL 1.0 was officially declared an OASIS Standard in November of 2004 with seven message types. An additional 14 message types are planned for UBL 2.0.

UBL incorporates ebXML. It includes a syntax-neutral model as well as an XML-specific syntax. Customization to use UBL with ANSI X12, EDIFACT, RosettaNet, etc. is mentioned as a possibility for the future.

4.3 RosettaNet

RosettaNet [9] is probably the most successful EDI "successor" system, being used by hundreds of companies in the information technology and electronic components industries. It provides standard "Partner Interface Processes" (PIPs) as templates for XML-based dialogs of series of messages between business partners.

RosettaNet not only defines message formats, but also has a master dictionary of terms for products, partners, and business transactions.

RosettaNet suffers from high start-up costs as does traditional EDI.

4.4 Context Inspired Component Architecture

Context Inspired Component Architecture (CICA) [1] is a syntax-neutral architecture for XML-based business messaging introduced as a new standard by the ASC X12 Committee – the committee which created and maintains the traditional X12 EDI standard. They see a problem with a proliferation of redundant XML-defined message types and want to ground them as much as possible on a standard industry-neutral set of XML concepts.

This is a very important development as the major US EDI standards organization starts providing EDI semantics and moves away from syntactical message requirements. However, the semantics is only at the broadest level of major message types and most common message components (such as person and address).

4.5 Open-edi Reference Model

The Open-edi reference model introduced standard business scenarios in 1997 [14]. It separated a "business operational view" (BOV) from a "functional service view" (FSV) of business messages and information exchange. The businesses deal with the BOV, which can be supported by a changing FSV, which handles technical connections, message protocols, and specific functional

capabilities at a lower level.

Open-edi was not an implementation specification, but merely specified a framework for coordinating the integration of existing standards and the development of new standards. Its concepts have been incorporated into later standards.

4.6 UMM/UML

UML is a modeling language which is used to create an object-oriented approach to EDI based upon the Open-edi principles [13]. Not only are individual message types modeled, but so is the structure of message-passing interactions using such messages. A single EDI message type was modeled as a proof-of-concept in 2001.

4.7 OAGIS

The Open Applications Group's Integration Specification (OAGIS) addresses the issue of certain industries creating their own messaging standards that may conflict with those of different industries [24]. They encourage industries to create specifications for industry-specific messaging needs, while OAGIS provides a general basis of message types.

4.8 eCO Framework

The 1999 eCo Interoperability Framework Specification "builds a bridge between disparate, proprietary electronic commerce solutions" [30]. The eCo Interoperability Framework was announced as providing "a single common protocol through which eCommerce systems can describe themselves, their services and their interoperability requirements."

However, the e-commerce systems to be interrelated were expected to be described in terms of XML, with no provision for a "bridge" being built to either ASC X12 or EDIFACT.

4.9 Commerce XML (cXML)

Commerce XML (cXML) presents a document type definition (DTD) for a few standard business message types in XML [29]. Although over 40 companies, including major corporations, collaborated on this standard it did not reach critical mass.

4.10 Common Business Library (xCBL)

The Common Business Library (xCBL) was an open XML specification of a set of XML business document types and their components [32], sponsored by Commerce One. In 2003 it was aligned with UBL and released as XML-S instead of XML DTDs. Its website is now defunct.

4.11 Summary

In summary, a quite impressive array of XML-based standard proposals has been worked on, coincidentally emerging with the emergence of XML itself. While some of these attempts failed, others are becoming adopted by industry, such as the case of RosettaNet.

It remains to be seen if these efforts really are able to replace X12, EDIFACT, or other long established fixed syntax standards like SWIFT.

One common theme, however, is that all these attempts are XML based, and thus are primarily syntax-based approaches. The semantics of the messages is rarely addressed and hence many of the disadvantages that can be found in EDI will prevail. Another observation is that some of these attempts failed or merged with other ongoing ones, clearly indicating that a consolidation is taking place.

Our approach is different in nature. Instead of promoting XML-based standards, we promote semantics-based standards and our first step in this direction is ontologizing ANSI X12 EDI.

5. Ontologizing EDI

The initial idea was to simply ontologize ANSI X12 EDI. This was a rather naïve idea as it quickly became clear that the syntax of X12 itself has not been completely formalized, let along the semantics. Many syntactical restrictions are defined in English prose thereby disallowing formal checking of the consistency of message syntax. The X12 Transaction Set 868, "Electronic Form Structure", does, however, provide a method for electronic description of most of the format. Its intended usage is not well defined in the standard.

Ontologizing EDI involves two distinct steps: ontologizing the syntax and the semantics. We report our initial experience with ontologizing X12 syntax below.

5.1 Syntactic vs. semantic ontologizing

Ontologizing EDI involves ontologizing both the format (syntax) and the semantics of the messages and their components. The discussion below refers specifically to ontologizing the ASC X12 standard, but most of the points apply to EDIFACT as well.

To ontologize X12 EDI syntax, we first define and encode a vocabulary for specifying the formats of Transaction Sets, Data Segment groups, Data Segments,

Composite Data Elements, simple Data Elements, Data Element codes, and Code Sets. Through the use of this vocabulary, the syntax of each of the components is defined.

Ontologizing these formats allows the formats to be described in a machine-readable form. Inter-element restrictions can be encoded in a formal way instead of via English prose. Subsets of the standard can be expressed in such a way that intersections and conflicts between them can be simply calculated.

Ontologizing the syntax provides a basis for ontologizing the semantics of messages and for converting to other messaging systems using different syntax.

A second ontological phase involves specifying the semantics of each Transaction Set, Data Segment group, Data Segment, Composite Data Element, simple Data Element, and code in each code set of a chosen X12 subset. Each Transaction Set and Data Segment group can be mapped into a set of statement templates. Each Data Segment and Composite Data Element will map into a statement template or relation. Each Data Element and code in each code set will be defined as a corresponding class, relation, or individual.

The semantics of Data Segments relate the meanings of their component Data Elements. Those of Data Segment loops and Transaction Sets relate the meanings of Data Segments and smaller loops which they include. Multiple ontologies can be used so that a process need only include those ontologies which it needs.

5.2 Advantages of ontologizing EDI

It is not necessary to ontologize all of EDI before benefits accrue. Ontologizing the syntax eases the specification of EDI subsets, enabling companies to publish the subset of EDI that they use, which would allow automation of the often time-consuming process of negotiation with a new partner to select a mutually-acceptable EDI subset for document transmission.

With the addition of ontologized semantics, semantically-enabled systems (such as those on the Semantic Web) would be able to reason with the content of received EDI messages and to create their own EDI messages.

Automated conversion between messages written in different systems (such as X12 and EDIFACT) could be enabled with ontologized semantics – and may be possible in limited cases after only ontologizing the syntax. Such message conversion would not be limited to the basic EDI systems, but could include the various XML-based systems as well, once the meanings of their XML tags are ontologized.

5.3 Issues with ontologizing EDI

There are significant problems inherent in ontologizing the semantics of X12, although far fewer accrue with ontologizing the syntax. The daunting task of ontologizing EDI semantics has apparently dissuaded many from attempting the task, but the inherent value in doing the syntax first seems not to have been considered. Once the syntax is ontologized (a far simpler task), the semantics can be incrementally added, with a resulting incremental advantage.

5.3.1. Issues with ontologizing EDI semantics. The first issue with ontologizing EDI semantics is that there are massive numbers of terms to ontologize. There are 36,730 data element codes in the 5010 version of X12, whose meanings must be arranged in complex directed acyclic graphs (DAGs) of classes, since many are subclasses of several others, however not all of them represent classes). A few thousand of these codes have duplicate names, but each such pairing needs to be examined to determine whether both codes with the same name actually have the same meaning.

The meanings of 1410 simple Data Elements, whose values may be numbers, dates, members of one of the code sets, or possibly free text, must be ontologized. The meaning of a Data Element in a message is affected by which of 34 Composite Data Element types and 1019 Data Segment types it occurs within. Even though an instantiation of a type of Data Segment normally is a complete statement, its meaning depends upon the context in which it appears within a message (for which there are 317 types of Transaction Sets).

However, this recounting of the number of terms to ontologize is not complete, since the standard also includes 650 external code sets that are not described in the standard. A large number of these are not public and would need to be purchased and their associated licensing issues dealt with. We do not yet know if the average size of these code sets is 20, 50, 200, or more codes. It is possible that the task of ontologizing these code sets would be greater than that for the rest of the X12 standard.

Another issue is that EDI terms are not formally defined. Their meanings must be deduced from their English names. Different companies and industries may use the same Data Segment or Data Element, e.g., “shipping date with different meanings.

```
instance X12_TS_997 memberOf x12:TransactionSet     ; define Transaction Set "997"
         hasFormatDescriptor hasValue X12_TS_997FD
instance X12_TS_997FD memberOf x12:FormatDescriptor ; define its Format Descriptor
         hasFormat hasValue X12_TS_997F
         formats1stComponent hasValue X12_ST_DS
         formats2ndComponent hasValue X12_AK1_DS
         formats3rdComponent hasValue X12_AK2_LOOP
         formats4thComponent hasValue X12_AK9_DS
         formats5thComponent hasValue X12_SE_DS
instance X12_TS_997F memberOf x12:Format
         formatFor1stComponent hasValue fc:X12_FC_M1
         formatFor2ndComponent hasValue fc:X12_FC_M1
         formatFor3rdComponent hasValue fc:X12_FC_O1
         formatFor4thComponent hasValue fc:X12_FC_M1
         formatFor5thComponent hasValue fc:X12_FC_M1

; define Data Segment "AK1" and its Format Descriptor and relate them
instance X12_AK1_DS memberOf x12:DataSegment
         hasFormatDescriptor hasValue X12_AK1_DSFD
instance X12_AK1_DSFD memberOf x12:FormatDescriptor
         hasFormat hasValue X12_AK1_DSF
         formats1stComponent hasValue X12_DE_479
         formats2ndComponent hasValue X12_DE_28
         formats3rdComponent hasValue X12_DE_480

instance X12_DE_479 memberOf x12:DataElement      ; define Data Element "479" & its format
         hasDataElementFormat hasValue IDString
         minLength hasValue 2
         maxLength hasValue 2

instance X12_FC_M1 memberOf x12:X12FormatCode      ; define one of the format codes
         optionality hasValue x12:Mandatory
         maxRepeats hasValue 1

instance X12_CUR_DS memberOf x12:X12DataSegment    ; define inter-element restrictions
         formatRestriction hasValue fr:X12_FR_C1110
         formatRestriction hasValue fr:X12_FR_L101112
; define details of first of these format restrictions.
instance X12_FR_C1110  memberOf x12:X12FormatRestriction
         restrictionType hasValue X12_FR_Conditioinal
         restrictionLine1 hasValue 11
         restrictionLine2 hasValue 10

instance X12_TS_855 memberOf x12:X12TransactionSet ; define Segment-Element restriction
         formatRestriction hasValue fr:X12_FR_TS855a
instance X12_FR_TS855a  memberOf x12:X12SegmentElementRestriction
         hasRestriction hasValue X12_FR_ExclusiveUse
         segmentNumber hasValue 17
         elementNumber hasValue 1
```

Figure 2: Formatting statements in WSML

5.3.2. Issues with ontologizing EDI syntax. One issue that arises with merely ontologizing X12 syntax is that there are numerous standard X12 subsets ("guidelines") used by different industries. Unless EDI programmers in each industry encoded their own subsets, ontologizers of the original system must determine each of these local standards. Discovering and obtaining these might be more difficult than it first appears.

Another issue that may arise is that messages stored and transmitted in ontologized format would be significantly longer in bytes than current X12 messages. If message length is a concern, compression could help or the human-readable/machine understandable format could be generated only when necessary and messages could be stored and transmitted in their current formats.

A third issue that may arise is a possible difficulty in convincing companies which currently use traditional EDI to switch to a form that could later provide semantics but does not yet do so. The benefits of automatic negotiation for setting up new B2B relationships could very well sway those companies which are looking for more business partners. Unlike other proposed successor systems, an ontologized EDI would be backward compatible with existing EDI relationships, reducing what is probably the source of the greatest resistance to moving beyond EDI.

```
<x12:TransactionSet rdf:ID="X12_TS_997">
    <x12:hasFormatDescriptor rdf:resource="#X12_TS_997FD">
</x12:TransactionSet>
<x12:x12Format rdf:ID="X12_TS_997F" >
         <x12:formatFor1stComponent rdf:resource="&fc;X12_FC_M1" />
         <x12:formatFor2ndComponent rdf:resource="&fc;X12_FC_M1" />
         <x12:formatFor3rdComponent rdf:resource="&fc;X12_FC_O1" />
         <x12:formatFor4thComponent rdf:resource="&fc;X12_FC_M1" />
         <x12:formatFor5thComponent rdf:resource="&fc;X12_FC_M1" />
</x12:x12Format>
<x12:FormatDescriptor rdf:ID="X12_TS_997FD">
    <x12:hasFormat rdf:about="#X12_TS_997F"  />
    <x12:formats1stComponent rdf:resource="#X12_ST_DS" />
    <x12:formats2ndComponent rdf:resource="#X12_AK1_DS" />
    <x12:formats3rdComponent rdf:resource="#X12_AK2_LOOP" />
    <x12:formats4thComponent rdf:resource="#X12_AK9_DS" />
    <x12:formats5thComponent rdf:resource="#X12_SE_DS" />
</x12:FormatDescriptor>

<x12:DataSegment rdf:ID="X12_AK1_DS">
    <x12:hasFormatDescriptor rdf:resource="X12_AK1_DSFD">
</x12:DataSegment>
<x12:FormatDescriptor rdf:ID="X12_AK1_DSFD">
    <x12:hasFormat rdf:resource="#X12_AK1_DSF" />
    <x12:formats1stComponent rdf:resource="#X12_DE_479" />
    <x12:formats2ndComponent rdf:resource="#X12_DE_28" />
    <x12:formats3rdComponent rdf:resource="#X12_DE_480" />
</x12:FormatDescriptor>

<x12:DataElement rdf:ID="X12_DE_479">
    <owl:comment> Functional Identifier Code </owl:comment>
    <x12:minDELength 2 />    <x12:maxDELength 2 />
    <hasDataElementFormat rdf:resource="x12;IDString" />
</x12:DataElement>

<x12:x12FormatRestriction rdf:resource="&ds;X12_CUR_DS" rdf:resource="#X12_CUR_DSFR" />
<x12:FormatRestriction rdf:ID="X12_CUR_DSFR">
       <form:restrictionType rdf:resource="&x12;X12_FR_Condtional" />
       <form:restrictionLine1 11^^xsd:Integer />
       <form:restrictionLine2 10^^xsd:Integer />
</x12:FormatRestriction>

<x12:x12FormatRestriction rdf:resource="&ds;X12_TS_855" rdf:resource="#X12_TS_855a_FR" />
<x12:FormatRestriction rdf:ID="X12_TS_855a_FR">
       <form:restrictionType rdf:resource="&x12;X12_FR_ExclusiveUse" />
       <form:restrictionLine1 17^^xsd:Integer />
       <form:restrictionLine2 1^^xsd:Integer />
</x12:FormatRestriction>
```

Figure 3: Formatting statements in OWL–

5.4 Steps for ontologizing syntax

The process of ontologizing EDI syntax is quite straight-forward. First, the vocabulary for specifying the Format of the EDI components and inter-element restrictions is specified. For X12, these components include Transaction Sets, Data Segment Groups, Data Segments, Composite Data Elements, simple Data Elements, and Code Sets. The relations include has-Format, hasDataElementFormat, hasCode, hasFormat-Restriction, and hasTSFormatRestriction.

The formats and restrictions of the selected EDI messages and their components are specified using this vocabulary. More complex structures are assigned a format and a list of components. In languages that only have binary relations, separate classes are created for the relation instances and each argument position becomes an attribute of that relation.

Each simple Data Element has its minimum length, maximum length, and basic format (integer, string, etc.) specified. Each Data Segment and Complex Data Element has its list of component Data Elements defined along with its format. The format for a Data Segment specifies the optionality and maximum number of repeats of each of its component Data Elements in sequence. Each Transaction Set has its list of component Data Segments (and loops thereof) defined along with its

```
(isa X12-TS-997 X12TransactionSet)                ; Define the format of a Transaction Set
(comment X12-TS-997 "Functional Acknowledgement Transaction Set")
(hasX12Format X12-TS-997 X12-FunctionalAcknowledgement-TSF
      X12-ST-DS X12-AK1-DS FunctionalAcknowledgement-Loop-AK2 X12-AK9-DS X12-SE-DS)

(isa X12-FunctionalAcknowledgement-TSF X12Format) ; Define a Transaction Set
(x12FormatIs X12-FunctionalAcknowledgement-TSF (X12FormatCodeFn 2 Mandatory 1)
     (X12FormatCodeFn 1 Optional 1) (X12FormatCodeFn 2 Mandatory 1))

(isa X12-AK1-DS X12DataSegment)                   ; Define a Data Segment
(hasX12Format X12-AK1-DS
       (X12FormatFn (X12FormatCodeFn 2 Mandatory 1) (X12FormatCodeFn 1 Optional 1))
       X12-DE-479 X12-DE-28 X12-DE-480)

(isa X12-DE-479 X12DataElement)      ; Define a Data Element and specify its components
(comment X12-DE-479 "Functional Identifier Code")
(hasX12DataElementFormat X12-DE-479 IDString 2 2)

;  If the 11th Element in the CUR Data Segment is filled, the 10th must be.
(x12FormatRestriction X12-CUR-DS X12_FR_Conditional 11 10)
;  If the 10th Element in the CUR Data Segment is filled either the 11th or the 12th must be
(x12FormatRestriction X12-CUR-DS X12_FR_ListConditional 10 11 12)

(x12TSFormatRestriction-ForbiddenElement X12-TS-855 17 1) ; Restrictions on TS and DS loop
(x12TSFormatRestriction-ElementFilledBy PurchaseOrderAcknowledgement-Loop-PO1 31 1 1)
(x12TSFormatRestriction-MandatoryElement PurchaseOrderAcknowledgement-Loop-PO1 1 2)
```

Figure 4: Formatting statements in CycL

format. The format for Transaction Sets (and internal loops) specifies the optionality and maximum number of repeats of each of its component Data Segments/loops. Finally, the syntactical restrictions are encoded.

X12 is provided in HTML files, which allows that, through the use of shell scripts and editor scripts, the ontologized format of most components may be batch produced. Transaction Sets, however, are quite complex, often containing nested loops, so their formats need to be manually crafted.

5.5 Steps for ontologizing EDI semantics

Ontologizing EDI semantics is a massive task, but can show benefits long before being completed.

Because EDI is designed to be able to express all kinds of business messages for all existing industries, it is possible and useful to create separate ontologies for different Transaction Sets or groups of Transaction Sets, as well as associated Data Segments, Data Elements, and Code Sets.

For example, few companies would need to transmit student records, patient records, and US Customs documents, as well as voter registration documents – all of which are defined in the X12 standard.

The ontologist must be aware that the same Data Elements and/or Data Segments may occur in several unrelated Transaction Sets. In such cases, the semantics of such components should be defined in separate ontologies that can be referenced by or imported to the appropriate Transaction Set group ontologies.

Once the Transaction Set classes are defined, an initial group of Transaction Sets is selected and ontologies for those Transaction Sets and their components are created.

Classes or Individuals are created for each Data Element Code that is applicable for the chosen group of Transaction Sets. Classes or Relations are created for each applicable Data Element. Relations or rules (sentence templates) are created for each Data Segment. Rules are created for each Data Segment group, including Transaction Sets.

This process may proceed bottom-up (from Data Element Codes to Transaction Sets), top-down (starting with Transaction Sets), or in a mixed fashion. Care must be taken when creating the terms for the codes that codes from different Code Sets may have the same meaning but different names and that codes with the same name from different code sets may have different meanings.

The classes created from the Data Codes and Data Elements must be carefully arranged hierarchically into directed acyclic subconcept/superconcept graphs. The created individuals must be assigned to their most specific classes. Auxiliary classes, being a generalization of a group of represented classes, may well be suggested even though they are not explicitly encoded in the standard. Such classes should be created.

5.6 Current status

We are in the midst of specifying ANSI X12 EDI format in multiple ontological languages. We encode the formats initially in one language and convert the encodings to other languages mostly through editor scripts. The X12 format source documents are provided as HTML files, but in most cases (except for Transaction Sets) sets of editor scripts can be used to translate the HTML versions into an ontology language.

We have created the vocabulary for specifying X12 formats and inter-element restrictions in WSML-core [10] (see Figure 2), OWL– [11] (OWL minus, RDF syntax: see Figure 3), CycL [8] (see Figure 4), TRIPLE [26], and FLORA-2 [28].

Each example shows the definition of a Transaction Set, its format in terms of included Data Segments, the definition and format of one of its Data Segments in terms of Data Elements, and the definition and format of one of those Data Elements. Format restrictions on a more complex Data Segment and Transaction Set are also provided.

Using this vocabulary, we encoded, as of January, 2005, the formats of all 1410 Data Elements, all 34 Composite Data Elements, all 1019 Data Segments, and an initial 65 most broadly used of the 317 Transaction Sets defined in the 5010 version of the X12 standard. All 1565 inter-Data Element restrictions defined in syntactic notes for Data Segments and Composite Data Elements have been encoded. The encoded Transaction Sets had twenty syntactic notes restricting Data Elements on included Data Segments, all of which were encoded in the ontology languages. We specified all 36,730 valid codes which were associated with the included Code Sets. We have included the codes for only for two (countries and currencies) of the 650 external Code Sets.

No effort has yet gone into systematically ontologizing the semantics of the ANSI X12 terms. However, the countries and currencies encoded from the external code sets have been identified as such and the associations between currencies and their respective countries were ontologized in passing.

6. Related Work

There seem to be no reported projects involved in ontologizing any major portion of ANSI X12 syntax or semantics.

In the process of creating XML versions of EDI, several companies, for example Altova [33], have generated XML tags for X12 and/or EDIFACT and tools for converting between traditional EDI messages and XML files. The intent of such projects is not to describe the syntax or semantics, but merely to convert between syntaxes.

The Implementation Guideline Markup Language (igML) [31] was created as an XML standard for defining X12 variants. The igML Working Group no longer has a web presence.

Other projects have ontologized large existing taxonomies or other large sets of computerized terms. The Cyc project has ontologized NAICS industry taxonomy, the UNSPSC product ontology, FIPS 10-4 geographical terms, among other term sets, integrating them into existing graphs of existing classes [22]. The DARPA Agent Markup Language (DAML) Program has also ontologized UNSPSC [18], but has mapped the narrower term relationships between levels (e.g. between "class" and "commodity") as subclass relations (which are not always valid) and has not added in missing subclass relations.

7. Future Work

We intend to encode the formats of the entire ANSI X12 Transaction Sets used in the information science and electronics devices fields first. Once the formats of these Transaction Sets are specified, we intend to start a parallel effort ontologizing their meanings, along with those of their component Data Segments, Data Elements, and codes in their associated Code Sets following the method described in Section 5.5, while continuing to encode the formats of the remaining X12 Transaction Sets.

Along with this effort, we will investigate ontologizing the syntax and semantics of the various XML-based systems, such as RosettaNet and UBL, so as to enable translations between them and the traditional EDI systems.

8. Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, and SWWS; by Science Foundation Ireland under the DERI Lion and M3PE project.

9. References

[1] Accredited Standards Committee X12. "The Future of Standards Development: Context Inspired Component Architecture – Using CICA Architecture for Building XML Messages", *http://www.disa.org/x12org/MEETINGS/x12trimt/cica.cfm*, October 2004.

[2] ADX Corporation, "Distribution Giant Hughes Supply Selects ADX to Electronically Connect Entire Supply Base",

http://www.adx.com/news/index.php?z=pressrelease&ID=135, Newark, CA, USA, 2002.

[3] A. Adshead and D. Thomas, "Web EDI forces suppliers to change tactics", *Computer Weekly*, July 2003.

[4] Berge, J., *The EDIFACT Standards*. Manchester, England, NCC Blackwell, 1991.

[5] Business.GOV, "eCommerce FAQs", *http://www.business.gov/phases/growing/use_technology/ecommerce_faq-3.html*, November, 2004.

[6] M. Crawford, "UBL", *http://www.oasis-open.org/committees/download.php/4610/xml2003.pdf*, January 2001.

[7] A. Crede, "Electronic Commerce and the Banking Industry: The Requirement and Opportunities for New Payment Systems Using the Internet", *Journal of Computer-Mediated Communication*, Vol. 1, No. 3, December 1995.

[8] Cycorp, "The Syntax of CycL", *http://www.cyc.com/cycdoc/ref/cycl-syntax.html*, 2002.

[9] S. Damodaran, "B2B Integration over the Internet with XML: RosettaNet Successes and Challenges", *Proceedings of the Thirteenth World Wide Web Conference*, 2004, pp. 188-195.

[10] J. de Bruijn, "The WSML Family of Representation Languages", *Technical Report D16.1*, Digital Enterprise Research Institute, Innsbruck, Austria, 2004.

[11] J. de Bruijn and A. Polleres, "OWL–", *Technical Report D20.1*, Digital Enterprise Research Institute, Innsbruck, Austria, 2004.

[12] E. Grygo, "STC, Tibco boost transaction software with XML", *InfoWorld*, Volume 21, Issue 51 (December 18, 1999), page 12.

[13] C. Huemer, "Defining Electronic Data Interchange Transactions with UML", *Proceedings of HICSS-34*, Maui, January 2001, p. 7031.

[14] ISO, "Open-EDI Reference Model", *ISO/IEC JTC 1/SC30 ISO Standard 14662*, 1997.

[15] K. Kayl, "ebXML: The Key Components", *http://java.sun.com/features/2001/09/ebxmlkey.html*, January 2001.

[16] F. Lehmann, "Machine-negotiated Ontology-based EDI (Electronic Data Interchange)", *Electronic Commerce: Current Research Issues and Applications*, Springer, Berlin, 1996, pp. 27-45.

[17] H. Lin, T. Risch T. Katchaounov, "Adaptive Data Mediation Over XML Data", *Journal of Applied System Studies* (JASS), Cambridge International Science Publishing, 2001.

[18] D. L. McGuinness, "Ontology http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml", *http://www.iswc2003.semanticweb.org/ontologies/106*, Palo Alto, USA, 2001.

[19] OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee, "Collaboration-Protocol Profile and Agreement Specification", *http://www.ebxml.org/specs/ebcpp-2.0.pdf*, 2002

[20] Organization for the Advancement of Structured Information Standards, "OASIS", *http://www.oasis-open.org/*, 2004.

[21] M. Rawlings, "Update to Market Impact of the ebXML Infrastructure Specifications", *http://www.rawlinsecconsulting.com/ebXML/ebXML5a html*, 2001.

[22] S. Reed and D. Lenat, "Mapping Ontologies into Cyc", *Proceedings of the AAAI*, Edmonton, Canada, 2002.

[23] D. Roman, H. Laursen, and U. Keller, eds., "Web Service Modeling Ontology", WSMO Working Draft, 2004, www.wsmo.org/2004/d2/v1.0.

[24] M. Rowell, "OAGIS - A 'Canonical' Business Language: Providing both Vertical and Horizontal Requirements", *Open Applications Group, Incorporated Document Number 20020429,* 2002.

[25] Ruh, J., *The Internet and Business: A Lawyer's Guide to the Emerging Legal Issues*, Fairfax, VA, 1996.

[26] M. Sintek and S. Decker, "TRIPLE---A Query, Inference, and Transformation Language for the Semantic Web", *International Semantic Web Conference* (ISWC), Sardinia, June 2002.

[27] United Nations Centre for Trade Facilitation and Electronic Business, "United Nations Centre for Trade Facilitation and Electronic Business", *http://www.unece.org/cefact/*, 2004.

[28] G. Yang, et al., *Flora-2 User's Manual*, Version 0.92, Dept. of Computer Science, Stony Brook University, June 2003.

[29] __. "Commerce XML (cXML)", *http://www.oasis-open.org/cover/cxml.html*, 2002.

[30] __. "eCo Interoperability Framework Specification", *http://www.oasis-open.org/cover/ecoInteropSpec.html*, 1999.

[31]__, "Implementation Guideline Markup Language (igML)", *http://xml.coverpages.org/igml.html*, February, 2001.

[32] __. "XML Common Business Library (xCBL)", *http://www.oasis-open.org/cover/cbl.html*, 2001.

[33] __. "XML Global And Altova Reduce Data Integration Costs With The Launch Of The Enterprise-Ready XML Integration Workbench", *Market Wire*, May, 2002.