Ontologizing EDI Semantics

Doug Foxvog[1] and Christoph Bussler[2]

[1] Digital Enterprise Research Institute, Galway, Ireland
[2] Cisco Systems, Inc., San Jose, CA, USA
[1] doug.foxvog@deri.org, [2] cbussler@cisco.com

Abstract. Electronic Data Interchange systems are used to transmit business messages in a compact form, with the meaning of message components positionally determined. We propose translating EDI messages into "semantic" forms in which the statements encoded in the compact forms are explicitly expressed, allowing reasoning based on such messages by any program using the semantic language, allowing their generation and use by Semantic Web services through the use of mediators. We present a system for semantically encoding EDI messages through the use of semantic templates.

Keywords: Ontology, EDI, semantic encoding, semantic mediator, semantic template, WSML, Semantic Web Services.

1 Introduction

From the point where communication between two or more computer systems was required, there was a need to establish precise definitions of the communication. This included not only precisely defining message exchange patterns, but message content, so that the sender and receiver both understand the information content of the message in the same way.

The traditional approach was for the sender and receiver to agree on message syntax so that each could parse the message successfully. Soon it became clear that agreement on message content definitions is as important as agreement on syntax.

A sender of business messages populates them with business content which the receiver must interpret. The burden of interpretation lies on the receiver; and if it has many partners, there can be quite an effort to interpret the many incoming messages. Misinterpretation with resultant negative economic impact is a likely consequence.

A more accurate approach is to describe message structure and content independent from application context. Once an Electronic Data Interchange (EDI) system is semantically defined, senders and receivers can construct and interpret messages uniformly so that misinterpretation does not occur [1]. Communication should then work flawlessly for their mutual benefit.

This approach is followed in this work. The ANSI X12 EDI standard is used since it has been used world-wide for over 30 years. There is significant experience with the use of X12 and the meaning of its messages. The recent surge in works on semantics makes EDI a perfect area for combining these worlds to make communication precise.

J.F. Roddick et al. (Eds.): ER Workshops 2006, LNCS 4231, pp. 301 – 311, 2006.

1.1 Ontologizing EDI

Ontologizing EDI semantics involves the creation of a system that, given an EDI message, enables the meanings of each encoded statement to be generated in a semantic language. This can be done using templates for expressing the meanings of each of the subcomponents of a message and the relationships among them.

The meanings of the simplest data elements may be simple data values, individuals, classes, or relations that may hold among such values. This meaning may either be looked up in a code set or be a known or new instance of the type specified in the template. If the meaning does not come from a code set, the relationship between the message text and the represented thing needs to be defined, e.g. the name of a ship, or a number of grams. The main referent of a more complex message component is taken to be its basic "meaning".

Templates for complex message components would normally specify the meanings for their slots and the relations that interrelate the meanings of various components. Either the relations which define the templates must take relations and classes (as well as individuals and simple data values) as arguments or such templates must be expressed as rules. The use of standard forms instead of rules to define templates limits the choice of ontology language since many ontology languages do not permit classes or relations to be used as arguments to relations. The desire to use ternary relation further restricts the choice of language for expressing the templates.

Approach. EDI systems have message and subcomponent types. ANSI X12 [2] has over 300 message types (called "transaction sets"), while EDIFACT [3] has around 200. Message subcomponent types number in the thousands. Because of the large size of these EDI systems, ontologizing their semantics is a massive task, but can show benefits long before being completed.

Since few companies need to transmit student records, US Customs documents, and voter registration documents – all X12 message types – the semantics for these would reasonably be defined in different ontologies. On this basis, templates for the most common messages can be created first, with sets of semantically-related messages added in turn. The same data elements and data segments recur in unrelated message types. The semantics of such components should be defined in separate ontologies referenced by the message type ontologies. Knowledge bases which define named terms (e.g. airports) mentioned in messages should be imported along with the ontologies for the associated message type whenever the template is used.

1.2 Sample EDI Message

The Terminal Information transaction set (#319) is one of the simplest defined in the X12 standard. It contains only four data segments (including mandatory transaction set header and trailer), 32 data elements, and no inner loops.

A sample message instantiating this transaction set is shown in Figure 1. The first segment is a header segment. The second is a Beginning Segment for Cargo Terminal Information. The third is a Cargo Detail segment. The fourth segment (fifth line) is a trailer segment. This message will be explained and used as an example below.

```
ST*319*000123>
BA2*LYKL*7*7A*Dubai Ports LTD*FedEx*01*Pier 1*T*New Orleans>
CD1*BXGL*012345**BOLNUM123456*LT****TerminalLoc4321*1*BBL94*6
9***T*BR*1.02*********A*LYKL*SO*Shipment02345*YD*doug foxvog>
SE*4*000123>
```

Fig. 1. Sample Instantiation of X12 Terminal Information Transaction Set

1.3 Document Layout

The rest of this document lays out what is necessary so that a semantic system could take in an EDI message and produce a set of statements expressing the encoded information in a logical language. Section 2 describes a design and ontology for semantic templates for information structures. Section 3 presents illustrative template encodings of the message type illustrated in Figure 1. Section 4 describes the tasks of the mediator in producing the semantic translation given an input EDI message and knowledge bases of semantic templates and presents logical statements generated for the above message using these templates. Section 5 presents related work. Section 6 summarizes the work that has already been done and outlines the work to follow.

2 Templates

A template for an information structure provides all the information needed to generate a sentence in the logical language which is implicit in an instantiation of that structure. A complex structure will have several templates for different implicit sentences. Each template specifies the predicate and arguments for a sentence. Each value may be derived from the instantiated information structure or be explicitly fixed in the template.

We have created a set of relations and classes for expressing such templates as described below and encoded them in WSML-Flight and CycL. Their definitions can be found at http://www.wsmo.org/TR/d27/v0.2/ontologies .

CycL allows for consistency checking and other reasoning during the creation of the ontologies. WSML was selected because it is a Semantic Web language which permits ternary relations and allows for classes and relations as arguments. An OWL encoding is not provided – that would require decomposing ternary relations into multiple binary properties, making template encoding more complex.

2.1 Template Classes

The classes to be defined for encoding the meaning of EDI messages are templates, code sets, formats, and positions in formatted structures. Terminology used for specifying EDI syntax described in [4] is used. Classes defined for templates include:

`CodeSet`: A mapping between short text strings and a set of things.

`ComponentTemplate`: A structure for determining a statement encoded in an instantiation of an Information Structure.

ComponentTemplate-Matching: A `ComponentTemplate` in which the generated statement must already exist in the data base. This can be used to bind a term being generated to an existing element. If all terms are already bound, it is a request to verify that the statement is known.
FormulaArgPosition: A position in a formula or formatted structure.

2.2 Template Relations and Function

The basic template relations are:

componentOf[Sub]Type indicates that an instantiation of the specified info structure means an instance/subclass of the specified class.
subcomponentOf[Sub]Type indicates that the filler for the `N`th position of an instantiation of the specified info structure means an instance/subclass of the specified class. This may be a sub-concept of the class to which the filler structure is restricted.
hasSameMeaningAs indicates that the filler for the first specified position in the structure has the same "meaning" as the filler for the second. The same term will be used for each in the derived sentences. This is a ternary relation.
functionalPredicateEncodes indicates that the "meaning" of a filler for an instantiation of the specified structure is the unique thing related to the filler by the specified predicate, e.g., `passportNumber`. This is a ternary functional predicate.
directlyEncoded indicates that the "meaning" of an instantiation of the `N`th slot in an info structure is the same as that of whatever fills it. This is the normal case.
[subcomponent]usesCodeSet indicates that the specified code set maps the filler for a specified [slot of an] info structure to its "meaning". This is not a functional predicate – sometimes codes may come from any one of several code sets.
templateForComponent indicates one of possibly several templates for a logical sentence encoded by instantiations of a given information structure.
templateRelation[-Encoded] indicates the relation for a given template. The relation may either be specified or encoded at a specified position of the info structure. These are functional predicates.
templateArg<N>[**-Encoded/Value]** (for `N` 1, 2, or 3) indicates that the `N`th argument for a given template is the specified value (or encoded or present but not encoded) at the specified position of the referenced info. Functional predicates.

One function is necessary to denote positions in a formatted structure, whether in the referenced structure, or nested in subcomponents or supercomponents of the structure.

FormulaArgPositionFn. The arguments are numbers which indicate the subcomponent number within the indicated structure. A zero indicates the next higher level. Thus, `FormulaArgPositionFn(3,2)` is the second component of the third component, and `FormulaArg-PositionFn(0)` is the structure itself, not a subcomponent. The enclosing structure is `FormulaArg-PositionFn(0,0)` and *its* third component is `FormulaArgPositionFn(0,0,3)`.

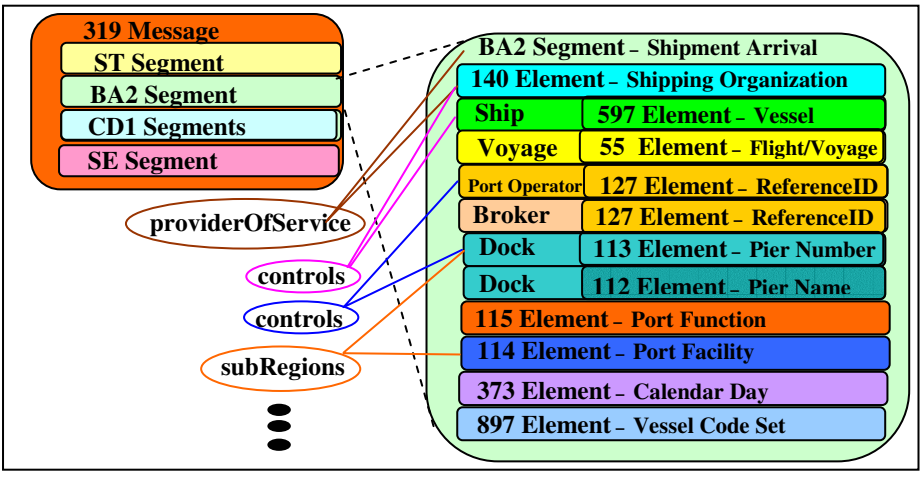

Fig. 2. Format of Transaction Set 319

3 Example Encoding of EDI Message Template

Example encodings of templates for an X12 transaction set are presented below, including representative templates for the transaction set and some of its subcomponents. The use of the templates is explained in Section 4. Statements generated by the templates from a sample message of Figure 1 are presented in Chapter 5. The complete templates (in WSML) of all subcomponents of this message type can be found at http://www.wsmo.org/TR/d27/v0.2/ontologies .

```
instance X12-TS-319 componentOfType ShipmentArrival
instance X12-BA2-DS componentOfType ShipmentArrival
instance X12-CD1-DS componentOfType Shipment
relationInstance hasSameMeaningAs(X12-TS-319,
         FormulaArgPositionFn(0),FormulaArgPositionFn(2))
instance X12-TST-319-2 memberOf ComponentTemplate
         templateForComponent hasValue X12-TS-319
         templateRelation hasValue transportees
         templateArg1-Encoded hasValue FormulaArgPositionFn(0)
         templateArg2-Encoded hasValue FormulaArgPositionFn(3)
```

Fig. 3. Sample Templates for Transaction Set 319

Figure 2 shows the format of the Terminal Information transaction set (#319). `ST` is the Transaction Set Header data segment; `BA2` is the Beginning Segment for Cargo Terminal Information; `CD1` is the Cargo Detail data segment; and `SE` is the Transaction Set Trailer data segment. All data segments are mandatory, but the `CD1` segment may occur multiple times, while each other segment appears only once. The `ST` and `SE` segments are members of every transaction set, providing only meta-information about the message. A few relations implicit in the message are shown.

A 319 message is about a shipment arrival. The BA2 segment is about the same shipment arrival. CD1 segments are about different shipments which arrive in the same ship. Figure 3 shows how this is encoded using the classes and relations described above. For clarity, namespace references are removed. The mediator generates a `transportees` statement using this template. This is the most common template pattern – relating one attribute with its two arguments. Additional templates relate items specified in the BA2 segment to items specified in the CD1 segments.

```
relationInstance subcomponentOfType (X12-BA2-DS, 2, Ship)
   "  subcomponentOfType (X12-BA2-DS, 1, ShippingOrganization)
   "  subcomponentOfType (X12-BA2-DS, 6, Dock)
   "  subcomponentOfType (X12-BA2-DS, 7, Dock)  . . .
   "  hasSameMeaningAs (X12-BA2-DS, FormulaArgPositionFn(6),
                                    FormulaArgPositionFn(7))
```

Fig. 4. Types of Subcomponents of BA2 Data Segment

A BA2 segment starts with seven mandatory data elements followed by four optional ones. It starts with codes for the shipping organization and for identifying the ship itself. Each component enters into at least one template for a sentence explaining its meaning within a 319 transaction set.

Although the data elements may store codes or names, their "meaning" is taken to be the thing that the code or the name represents. For example, data element 140 is called "Standard Carrier Alpha Code", but its "meaning" is the cargo carrier, not the four-character SCAC assigned to that carrier. The BA2 segment gives two identifiers for the pier at which the ship docks: a number and a name. The `hasSame-MeaningAs` relation ensures the same term will be generated from the templates by the mediator for the two slots. Types associated with message components are encoded as in Figure 4; component formats as described in [4].

```
instance X12-DE-140
         usesCodeSet hasValue SCACCodeSet
         componentOfType hasValue ShippingOrganization
instance X12-DE-597
         componentOfType hasValue TransportationDevice
relationInstance subcomponentOfType (X12-BA2-DS, 2, Ship)
instance X12-DET-597 memberOf ComponentTemplate
         templateForComponent hasValue X12-DE-597
         templateRelation hasValue identificationStrings
```

Fig. 5. Example Template for Data Element

The meaning of the instantiations of data elements may either be determined from code sets or obtained through templates. For example, the first data element (140) of the BA2 data segment (597) is encoded using the Standard Carrier Alpha Code, while the second data element may be identified by name or other ID. Figure 5 shows the representations for these two cases. Note that Data Element 597 represents a transport

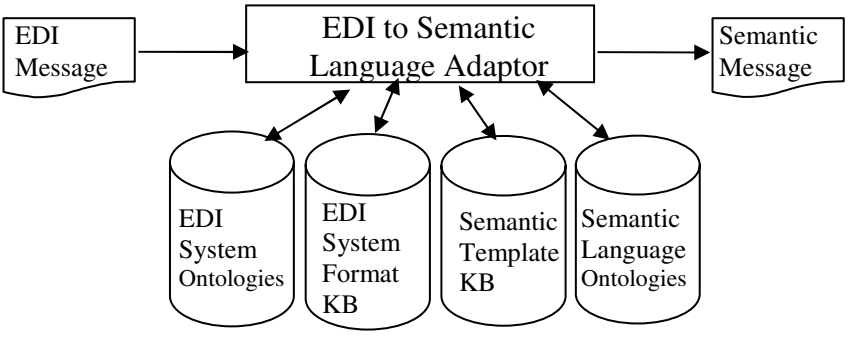

Fig. 6. Adaptor Data Sources

device, but is restricted to a ship in the context of BA2. The templates for the rest of Transaction Set 319 are presented at http://www.wsmo.org/TR/d27/v0.2/ontologies .

4 Generation of Logical Statements from EDI Messages

The task of a Semantic Mediator is to convert a message between two formats or ontologies, at least one of the forms being semantic [5]. Such a mediator uses a pre-established mapping between terms and structures in the two systems and may convert between languages if necessary. To generate logical statements from EDI messages, a mediator is needed that uses the template rules and the formats to convert individual messages into a set of logical statements (in WSML in our case).

The task of the X12 to WSML mediator is to separate the components of the X12 message programmatically; using the syntax defined in WSML for EDI components to determine which semantic templates to use. The mediator matches the templates against the extracted message components to generate the WSML statements which express the meaning of the X12 message. See Figure 6.

The mediator in the other direction, given a WSML message, uses the same templates to generate an X12 message, using the syntax described for the message type and its subcomponents. This mediator may have to add information required by mandatory message elements from data existing in the knowledge base (e.g., a party's contact information) or which it calculates (e.g. the number of data segments being transmitted) if that information had not been included in the input WSML message.

Variations of these mediators could handle other EDI formats such as EDIFACT or different semantic languages such as OWL. Features needed by the mediator for expressing the templates (ternary relations, functions, and relations and classes as arguments) are not needed in either the input or output languages.

5 Generation of Logical Statements from X12 Messages

A portion of the intended output of the conversion of the sample X12 message shown in Figure 1 using the WSML templates is presented below. Although semantic output

is far less compact than the X12 format, it allows for reasoning about the message content. The mediator creates new terms from the names of the instantiated concepts.

The BA2 segment of Figure 1 encodes the following: the shipper for this shipment arrival has SCAC code LYKL (which turns out to be Lykes Lines Ltd). Ship no. 7 is the carrier on voyage no. 7A, which ends with this arrival. Dubai Ports Ltd. is the port operator and FedEx is the broker for the shipment. Ship no. 7's arrival occurs at Import Pier 1 (numbered "01") at the port of New Orleans. The arrival of this ship is part of transshipment – the port is not the final destination of its shipment lots.

```
instance ShipmentArrival_1 memberOf ShipmentArrival
         providerOfService hasValue LykesLinesLimited
         transshipmentPort hasValue PortOfNewOrleans
instance Dock_1 memberOf Dock
         identificationStrings hasValue "01"
         placeName-ShortForm hasValue "Import Pier 1"
```

Fig. 7. Encoded Meaning of BA2 Segment

Figure 7 presents the encoding of part of this information, which the mediator should generate from the templates. The memberOf statements are generated based on the subcomponentOfType assertions of Figure 4. A template relating the zeroth and first argument of the BA2 segment generates the providerOfService statement. Similar templates are responsible for the other statements. Statements are similarly generated from the CD1 segment.

```
instance X12TS_1 memberOf X12TransactionSetInstantiation
         instantiationOfAIS hasValue X12_TS_319
         identificationStrings hasValue "000123"
relationInstance verify(segmentCount, X12TS_1, 4)
  "      verify(identificationStrings, X12TS_1, "000123")
```

Fig. 8. Encoded Meaning of ST Segment

The Transaction Set Header of Figure 1 indicates that the message is an instantiation of a 319 transaction set with ID string "000123". The trailer templates generate two statements that should be verified by the system: the number of segments included in the message and its control number. No new information is provided by the trailer. Figure 8 shows the WSML code produced by the templates for these data segments.

```
instance ShipmentArrival_1 memberOf ShipmentArrival
         transportees hasValue Shipment_1
instance FedEx   controls hasValue Shipment_1
```

Fig. 9. Encoded Meanings at the Message Level

Figure 9 lists the meaning of several statements generated at the message level. The shipment defined in the CD1 segment is transported in the arrival defined in the BA2 segment and the broker specified in the BA2 segment controls the shipment defined in the CD1 segment. With multiple CD1 segments, similar statements would be generated using the same BA2 data. If OWL-like blank nodes were used to generate the terms referred to in each segment, the generation of such statements linking terms defined in different segments would be more difficult.

6 Related Work

Beginning in the 1990s, problems with EDI led to the development of systems intended as state-of-the-art replacements for the apparently antiquated X12 and EDIFACT systems. The new systems were not designed to be backward compatible with X12 or EDIFACT to allow translation of such messages into the new forms.

Few of the newer systems are semantically described. They mainly introduce a more open syntax (XML) and transmission medium (the Internet) and thus maintain those disadvantages of traditional EDI which result from semantic opacity. For an overview of some of the major systems, see [4].

Some private companies market proprietary software that produce and accept business messages in both traditional EDI and more recent XML-based formats [6, 7, 8]. However, these systems appear not to be based on ontologies.

The ASC X12 Committee saw a problem with a proliferation of redundant XML-defined message systems and wanted to ground the new systems as much as possible on a standard industry-neutral set of XML concepts. They created a Context Inspired Component Architecture (CICA) [9, 10] as a syntax-neutral architecture for XML-based business messaging. They have started converting traditional X12 message types into the new architecture, releasing their first five X12 XML message schemas in December 2005. This is an important development as the major US EDI standards organization starts providing EDI semantics and moves away from syntactically rigid 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). A semantic description of most of the content of business messages is not provided.

The Universal Business Language (UBL) is an effort started in 2003 to unify the plethora of XML standards for business documents [11]. UBL 1.0 was officially declared an OASIS Standard in November of 2004 with eight message types. UBL has XML definitions for many information components (such as addresses and product codes) that are present in a variety of common business messages (such as invoices and shipping manifests). Use of UBL for messages between industries was envisioned in conjunction with industry-specific messages being used within an industry. However, mappings of message types and their components to definitions of equivalent terms used in XML-based EDI (RosettaNet, HL7, CDIX, etc.) are not provided. No translator is provided to map X12 or any other standard into UBL. Nor is a method provided for specifying the mapping between different languages.

In the 1990s, Cycorp took two large medical term taxonomies, SNOMED[1] and MeSH[2], and mapped them to an anatomical sub-ontology which it was developing [12]. Use of the taxonomies aided in the production of the ontology; however, it was highly enriched from the bare taxonomic forms of the sources and the majority of terms in the taxonomies was omitted as the scope of the target ontology was narrower. This project was not intended to create a semantic encoding of an existing system, but to extract a specific subset of information encoded in two systems to aid in developing an ontology for a field and to establish a mapping between the semantically defined terms and the pre-existing terms in the taxonomies.

7 Summary and Future Work

We have demonstrated that semantic templates can be manually created for traditional EDI systems that would enable a data mediator to produce a semantic description of a message encoded in that system. Such a mediator would use these templates and a semantic description of the EDI message format in the translation.

We plan to use these techniques to ontologize standard subsets of EDIFACT for the Sixth Framework Programme specific targeted research project TripCom[3] as a basis for business to business process integration. TripCom will allow business data to be represented as RDF triples for inter-business interactions.

Acknowledgments. This work is funded by the EU projects the DIP (FP6 - 507483) and TripCom (FP6 - 027324) and by Science Foundation Ireland under the DERI Líon project.

References

1. Bussler, C: B2B Integration, Springer-Verlag, Heidelberg (2003)
2. X12/DISA Information Manual, ASC X12S/92-707, Data Interchange Standards Association (DISA), Alexandria, VA (1992)
3. Berge, J.,: The EDIFACT Standards. NCC Blackwell: Manchester, England (1991)
4. Foxvog, D., Bussler, C.: "Ontologizing EDI: First Steps and Experiences"; Proceedings of the International Workshop on Data Engineering Issues in E-Commerce (DEEC 2005), Tokyo, Japan, 9 April 2005, 49-58
5. Mocan, A. and Cimpian, E.: Web Service Modeling Execution Environment - Conceptual Model (WSMX_O), DERI Technical Report D13.3 v0.3
6. Meehan, M.: Aerospace Group Backs New EDI-to-XML Bridge, Computer World, 12 October 2001
7. Extreme Translator, http://www.xtranslator.com
8. Mercator Is First Integration Solution Provider to Offer ACORD, XML, EDI, HIPAA, and SWIFT, Business Wire, 21 May 2002

[1] http://www.snomed.org/snomedct/
[2] http://www.nlm.nih.gov/mesh/meshhome.html
[3] http://tripcom.org/

9. Accredited Standards Committee X12 (2004). The Future of Standards Development: Context Inspired Component Architecture – Using CICA Architecture for Building XML Messages. Available from *http://www.disa.org/x12org/MEETINGS/x12trimt/cica.cfm* .
10. ASC X12 Releases First Set of W3C-Compliant XML Schemas Based on CICA, Insurance News Net, 15 December 2005. Available at http://www.insurancenewsnet.com/print.asp?a=featured_pr&id=55171
11. Crawford, M.: UBL, 2001. Available from *http://www.oasis-open.org/committees/download.php/4610/xml2003.pdf*
12. Lehmann, F. & Foxvog, D.: Putting Flesh on the Bones: Issues that Arise in Creating Anatomical Knowledge Bases with Rich Relational Structures, Knowledge Sharing across Biological and Medical Knowledge Based Systems, Papers from the 1998 Workshop: Technical Report WS-98-04, Fifteenth National Conference on Artificial Intelligence, Madison, WI, 26 July 1998, 41-50