2012 IEEE 14th International Conference on Commerce and Enterprise Computing
DOI 10.1109/CEC.2012.13, © 2012 IEEE

From Encoded EDIFACT Messages to Business Concepts Using Semantic Annotations

Robert Engel, Christian Pichler, Marco Zapletal, Worarat Krathu and Hannes Werthner
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{engel, pichler, marco, worarat, werthner}@ec.tuwien.ac.at

***Abstract*—Standardized business documents are a prerequisite for Electronic Data Interchange (EDI). One predominant standards family are the Electronic Data Interchange For Administration, Commerce and Transport (EDIFACT) formats created and maintained by the United Nations Centre for Trade Facilitation and Electronic Business. However, EDIFACT formats are currently specified in a textual manner where semantic relationships between data elements are neither formally nor explicitly described. While these relationships are usually easy to identify for humans, this information is hidden to machines. This hampers semantically accurate interpretation of EDIFACT messages. In addressing this shortcoming, we present an approach for conceptualizing EDIFACT messages in ontologies, the *EDIonto approach*. Thereby, we provide means for modeling and storing both structural information acquired from the official EDIFACT standards as well as semantic information about the relationships between data elements. Furthermore, we describe means for storing concrete EDIFACT messages in knowledge bases according to the developed ontologies. We evaluated our approach by comparing it to state-of-the-art tools for interpreting EDIFACT messages. Contrary to the other tools, the approach presented in this paper allows for the automated and semantically accurate processing of generic EDIFACT messages.**

I. Introduction

Electronic Data Interchange (EDI) [11], [4], [15] is used for information exchange in electronic business transactions between collaborating organizations. One prerequisite for successful implementation of EDI systems are standardized formats which are typically defined for a particular domain or industry through Standard Development Organizations (SDOs) such as the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). For instance, UN/CEFACT became originally famous for defining as well as maintaining the United Nations Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT) formats[18], [1]. Today, still many inter-organizational systems operate on the basis of EDIFACT message interchanges. According to a survey [19] conducted in 2007, an estimated 85%-90% of the total volume of electronic B2B transactions were carried out using traditional EDI standards at this time. Due to their wide industry adoption and long-standing establishment, EDIFACT standards will presumably continue to play an important role in Business-to-Business (B2B) interaction for years to come [19], [12].

However, data elements transmitted in EDIFACT messages do not always have invariant semantics. Their semantics may be determined by the values of other data elements, so-called *qualifiers*. Hence, for the correct interpretation of a data element one has to take into account possible semantic relationships with other data elements. While these relationships are usually easy to identify from the EDIFACT standards for humans, this information is neither contained formally nor explicitly in the standards. As a consequence, it is a-priori not accessible to machines. Past and current implementations of EDI systems are usually specifically programmed (i.e., hard-coded) to handle selected semantic relationships correctly. Moreover, many times these systems are specifically designed to implement exclusive bi- or multilateral agreements between trading partners concerning the "allowed" usage of data elements, so-called *Message Implementation Guidelines* (MIGs). However, best to our knowledge based upon the current state-of-the-art there is no generic mechanism for the automated and semantically accurate interpretation of *arbitrary* EDIFACT messages due to a lacking formalization of semantic relationships between data elements in the EDIFACT standards.

To address this shortcoming, we present an approach for building a conceptual understanding of EDIFACT messages using semantic technologies, the *EDIonto approach*. Thereby, we provide means for defining the above mentioned semantic relationships between data elements in a formal way. The explicit modeling of such relationships facilitates automatic and semantically accurate processing of EDIFACT messages by software applications.

We expect that the availability of such a machine-processable semantic understanding of the content of messages facilitates the construction as well as enhances the power of applications in which business data from EDIFACT messages needs to be further processed. Such applications on top of EDI systems may include tools for debugging, validation, visualization and mapping of messages as well as enhanced analyses of business data that is contained in messages. For example, the EDIonto approach will serve as a conceptual and technical foundation for subsequent research on business performance analyses as outlined in [13] and for redundancy analysis in inter-organizational business processes as outlined in [6].

The remainder of this paper is organized as follows. Section II describes the pitfalls of interpreting EDIFACT messages semantically accurately, therefore motivating the need for a conceptual understanding of such messages. In Section III, the current state-of-the-art is discussed. In Section IV, we describe our proposed approach for conceptualizing EDIFACT messages. In Section V, we evaluate our approach by comparing it to current state-of-the-art tools for interpreting and processing EDIFACT messages. Finally, in Section VI a conclusion and an outlook on future work are given.

II. Motivation: Interpretation of EDIFACT Messages

In EDIFACT messages, data elements that are syntactically equivalent (i.e., they occur at the same position in the same segment) may have varying semantics depending on the values of *qualifying data elements* at other syntactical positions in the message. Consider the following example of a *DATE/TIME/PERIOD* (DTM) segment instance, assuming that it was created based on version 97A of the official UN/EDIFACT directories[1]:

```
DTM+137:20110712:102'
```

To find out the meaning of this encoded segment instance, it is first necessary to interpret the EDIFACT standards. The segment directory[2] specifies that *DTM* denotes a segment that is related to dates, times or periods of time. Furthermore, it defines that the segment contains a composite data field (i.e., a data field that consists of several components) with a specific identifier (*C507*). Secondly, consulting the composite data field specifications[3] for this specific field (C507)[4] reveals that it consists of the following components: a (mandatory) *function qualifier*, an (optional) *text* and an (optional) *format code*. Thirdly, the function qualifier and the format code fields contain values that represent codes further defined in code lists of the standard. For example, the function code 137, as it appears in the example, is defined in the corresponding code list[5] as *"Document/message date/time"* with an additional explanatory text *"(...) Date/time when a document/message is issued (...)."*. The format code 102 resolves to a date format pattern of *"CCYYMMDD"*, where *"(...) C = Century ; Y = Year ; M = Month ; D = Day."*[6]. The information gathered so far from the standards is visualized in Fig. 1.

Using solely the information from the standards as outlined above does not tell us explicitly about the relationship between the function and format qualifiers and the actual date text in between of them. Only the *additional* - and actually *assumed* - knowledge that the function qualifier and the format code relate to the date text field allows to determine the intended meaning of the segment. Hence, with this knowledge we can conclude that this segment instance conveys the information that the enclosing message was issued on July 12, 2011.

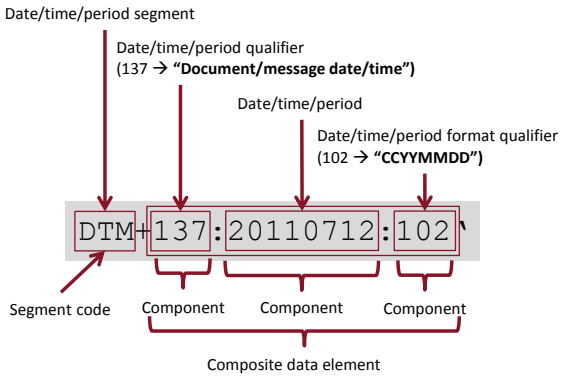

Fig. 1. Example of a DTM segment instance and corresponding information gathered from the EDIFACT standards

As another example, consider the following instance of a *NAME AND ADRESS* (NAD) segment instance, assuming that it was created based on version 97A of the official UN/EDIFACT directories:

```
NAD+ST+Vienna UT'
```

This segment instance consists of two data elements[7]: (i) *Party qualifier* (here: "ST") and (ii) *Party identification details* (here: "Vienna UT"). The party qualifier is coded and resolves to *"Ship to"*[8]. In this case, to humans it is clear from the context that the *party qualifier* qualifies the *party identification details* field. Contrary to the previously described example of a DTM segment instance where a component of a composite field qualifies another component of the same field, here a non-composite field qualifies another composite field. Knowing about this relationship between these data elements, we can conclude that this segment instance conveys the information that something should be shipped to "Vienna UT".

Qualification Relationships. The above described examples illustrate that the EDIFACT standards do nowhere explicitly nor formally specify *which* field is actually being qualified by function and format qualifiers. This applies not only to the above discussed examples, but is the general case for the EDIFACT standards. While to humans such information can be usually concluded from the context, it is currently not automatically recognizable by a machine. Hence, knowledge about these *qualification relationships*, as we call them, has to be provided in addition to the information gathered from the EDIFACT standards in order to allow machines to interpret messages semantically accurately. With the EDIonto approach, we account for this shortcoming and provide means for modeling these qualification relationships explicitly, as described in Section IV.

[1] http://www.unece.org/trade/untdid/d97a/ (all URLs in this paper were last visited on May 10, 2012)
[2] http://www.unece.org/trade/untdid/d97a/trsd/trsdi1.htm
[3] http://www.unece.org/trade/untdid/d97a/trcd/trcdi1.htm
[4] http://www.unece.org/trade/untdid/d97a/trcd/trcdc507.htm
[5] http://www.unece.org/trade/untdid/d97a/uncl/uncl2005.htm
[6] http://www.unece.org/trade/untdid/d97a/uncl/uncl2379.htm
[7] http://www.unece.org/trade/untdid/d97a/trsd/trsdnad.htm
[8] http://www.unece.org/trade/untdid/d97a/uncl/uncl3035.htm

III. State of the Art

The need for building ontologies for automating EDI has been observed in various research works such as [7], [16], [14], [3]. In particular, in [14], the authors recognize the problem that standards such as EDIFACT or ANSI X12 are defined in English prose and are thus unavailable for machine processing. The works presented in [7], [16], [3] are targeted towards overcoming interoperability issues through utilizing ontologies and semantic technologies.

Further existing approaches in the context of ontologizing EDI standards include works conducted in the course of the TripCom project[9]. The underlying vision of the TripCom project is enabling persistent asynchronous communication for web services [8] by creating an ontological infrastructure for business processes and business data. Therefore, one aim of the project was to define ontologies for EDI in terms of both syntax and semantics for overcoming heterogeneity problems. In [9], [10], the authors present an approach for ontologizing EDI based on semantic templates. Thereby, the authors utilize manually defined templates serving as a basis for deriving syntax and semantics from EDI standard specifications. One of the major challenges faced when ontologizing EDIFACT is extracting semantics which are defined through textual descriptions as part of the EDIFACT standard specifications. In the TripCom project [9], [10] the mechanism for dealing with textually defined semantics is unclear, whereas in the EDIonto approach the problem of implicit semantic relationships is explicitly addressed in the developed ontologies. From a technical perspective, the TripCom project utilizes the Web Service Modeling Language (WSML) [20], whereas the EDIonto approach builds upon the Web Ontology Language (OWL) [21].

The problem of processing qualified data elements semantically accurately is also relevant when mapping EDI standards to other data structures (e.g., data formats used in in-house applications) and has been addressed in this domain to a certain extent. State-of-the-art mapping tools usually allow for the definition of mapping rules for inbound EDIFACT messages that look for specific values in qualifying fields so that corresponding qualified fields can be mapped to a respective data element in the destination schema (see, for instance, [2]). Contrary to the EDIonto approach, this does not allow for the generic interpretation of qualification relationships since only the interpretation of qualified data elements that are paired with manually defined specific codes in qualifying elements is possible. While more complex rules may be technically possible in such environments, this usually requires programming efforts, i.e., the *hard-coding* of the qualification relationships. Furthermore, the definition of selected pairs of qualifying and qualified elements is usually associated only to a specific mapping definition and has to be repeated for new mappings.

[9] http://tripcom.org/ontologies

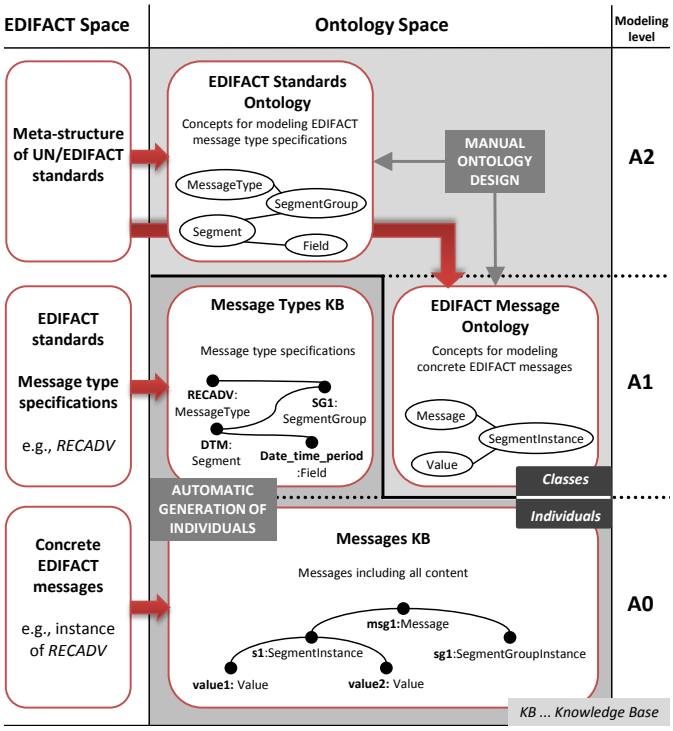

Fig. 2. Overview of the approach for ontologizing EDIFACT

IV. Conceptualizing EDIFACT

As demonstrated in Section II, successful interpretation of EDIFACT messages requires at least the following bodies of knowledge as an input (cf. Fig. 2, *EDIFACT Space*): (i) the messages themselves, (ii) the EDIFACT standards (i.e., message type specifications) and (iii) abstract knowledge on how to read messages and standards (i.e., the meta-structure of the standards). In the EDIonto approach, these bodies of knowledge are modeled in ontologies and corresponding knowledge bases (cf. Fig. 2, *Ontology Space*), as described in the following.

A. Architecture

The EDIonto approach comprises four main building blocks as shown in Fig. 2 (*Ontology Space*). The meta-structure of the EDIFACT standards with regard to message type specifications is modeled in the *EDIFACT Standards Ontology*. The meta-structure with regard to the generic structure of EDIFACT messages (i.e., regardless of specific message types) is modeled in the *EDIFACT Message Ontology*. These ontologies are created manually. Message type specifications and concrete messages are stored in the *Message Types KB*[10] and in the *Messages KB*, respectively. The individuals in these knowledge bases are created automatically. Message type specifications are parsed from the official UN/EDIFACT directories[11] using a custom parser, as described in detail in Section IV-D. Analogously, concrete EDIFACT messages are parsed and

[10] Here, KB is used as an acronym for *Knowledge Base*.

[11] http://www.unece.org/trade/untdid/directories.htm

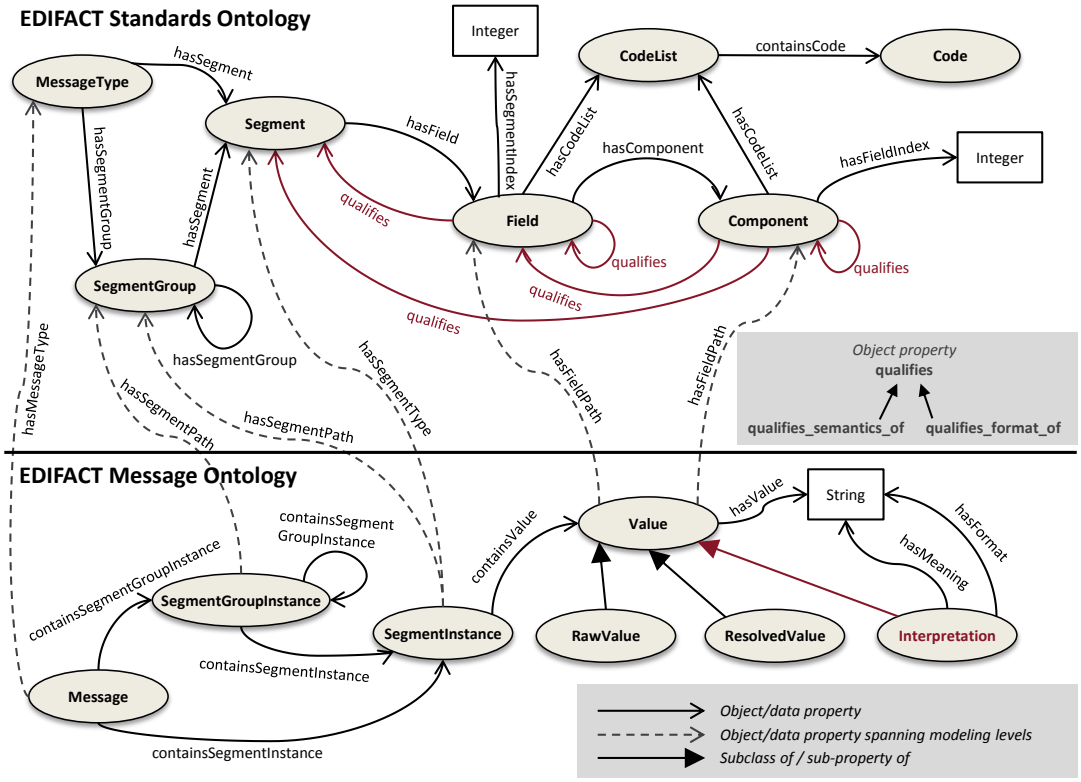

Fig. 3. EDIFACT Standards Ontology and EDIFACT Message Ontology schemas

integrated into the Messages KB using a custom parser that utilizes the knowledge about message type specifications in the Message Types KB, as described in detail in Section IV-E.

The aforementioned building blocks stand in a hierarchical relationship that can be described using a set of modeling levels inspired by the Meta Object Facility (MOF) [17] defined by the Object Management Group (OMG)[12]. The basic idea is that each modeling level contains instantiations of the entities on the next higher modeling level. Thereby, the first modeling level of our architecture, A0, represents concrete objects from the real world. The second level, A1, represents actual models abstracting from the real world. Based on A1, on the A2 level meta-models are specified that the models on the A1 level must correspond to.

Fig. 2 shows the main building blocks of the EDIonto approach along with the corresponding modeling levels they reside on. Arguably, concrete EDIFACT messages represent real-world objects. Hence, the Messages KB resides on the A0 modeling level. Concrete messages are instantiations of specific message types and must adhere to corresponding message type specifications. In other words, the corresponding message type specifications are models of the messages. Hence, the Message Types KB resides on the A1 modeling level. At the same time, the EDIFACT Message Ontology provides a generic model for the structure of concrete messages and thus resides on the A1 level, too. Finally, the EDIFACT Standards Ontology models the meta-structure of message type specifications. Hence, it is a model of a model (i.e., a meta-model) and resides on the A2 level of the EDIonto architecture.

[12] http://www.omg.org/news/about

B. EDIFACT Standards Ontology

The EDIFACT Standards Ontology is shown in the upper part of Fig. 3. Resembling the meta-structure of the EDIFACT standards, the defined concepts include message types (class `MessageType`), segment groups (`SegmentGroup`), segments (`Segment`), (possibly composite) data fields (`Field`), components of composite fields (`Component`), code lists (`CodeList`) and codes contained in code lists (`Code`). Individuals of these concepts can be connected and put in hierarchical structure in the same way as they are organized in the EDIFACT standards using a number of object properties. As shown in Fig. 3, these object properties include `hasSegmentGroup`, `hasSegment`, `hasField`, `hasComponent`, `hasCodeList` and `hasCode`. In order to keep track of the order of elements, additional data properties `hasSegmentIndex` and `hasFieldIndex` are defined to specify the position of the data elements in their corresponding containing elements.

Qualification Relationships. In addition to the above described concepts for modeling the EDIFACT standards, we introduce object properties `qualifies_semantics_of` and `qualifies_format_of` for modeling qualification relationships between entities. The former expresses that a specific entity qualifies the *semantics* of another entity. The latter

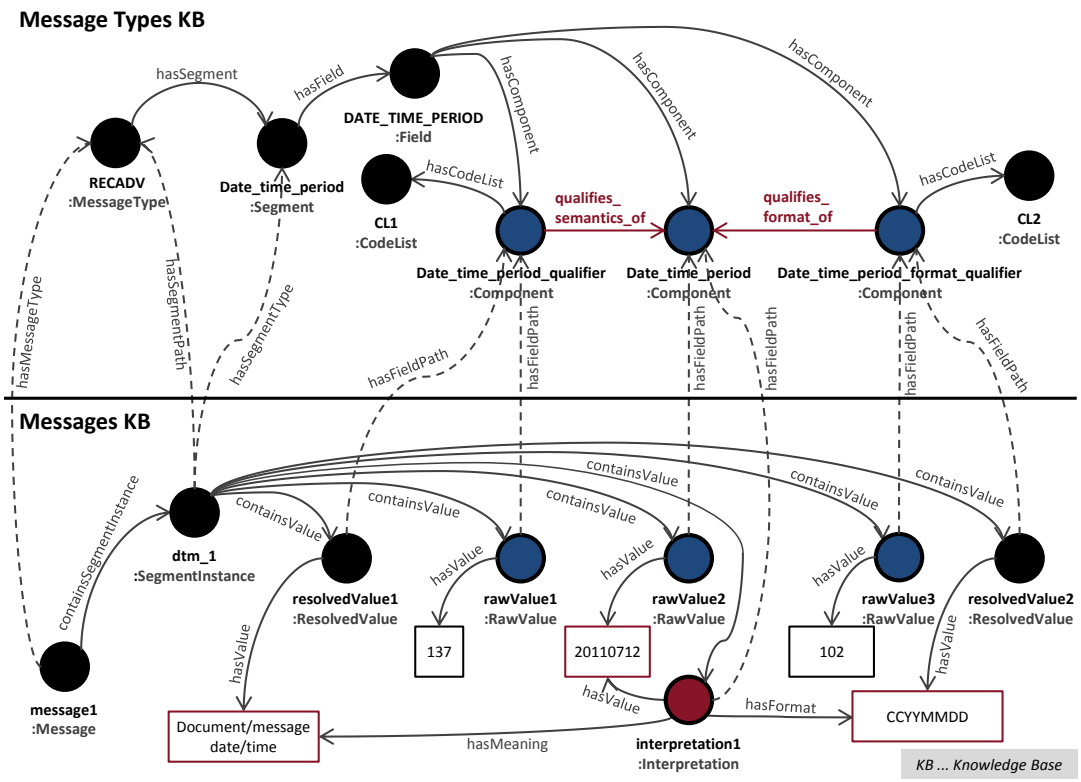

Fig. 4. Sample knowledge bases (excerpt): For the Message Types KB, a simplified example of a concrete instance of the DTM (*Date/time/period*) segment with associated fields and components is given. Furthermore, an exemplary individual of `MessageType`, RECADV (*Receiving Advice*), containing the DTM segment at its top level (i.e., not nested in segment groups), is shown. Note the qualification relationships between the `Component` individuals. For the Messages KB, a simplified example of a concrete RECADV message with a single segment instance of the DTM segment and corresponding values is shown. Furthermore, an example for an *interpretation* of a date value resulting from the evaluation of the above mentioned qualification relationships is shown at the bottom. Some elements have been excluded from the figure for comprehensibility.

expresses that a specific entity contains information about the (syntactic) *format* of another entity. Both of these object properties belong to a common object property super type `qualifies`. Due to the nature of the EDIFACT standards, there is a limited number of possibilities of what types of entities can qualify which other types of entities. The possible combinations are shown by the `qualifies` relationships between `Component`, `Field` and `Segment` in Fig. 3.

C. EDIFACT Message Ontology

Individual EDIFACT messages and their content are modeled generically (i.e., independently of specific message types) by the EDIFACT Message Ontology (cf. lower part of Fig. 3). The concepts include messages (`Message`), segment group instances (`SegmentGroupInstance`), segment instances (`SegmentInstance`) and concrete values (`Value`). Values are further subclassed for raw values (`RawValue`), resolved values (`ResolvedValue`) and interpretations (`Interpretation`). Segment group instances, segment instances and concrete values are linked to the concepts from the message structure specifications through the `hasSegmentPath` and `hasFieldPath` object properties, respectively. In Section IV-E, the usage of these concepts is described.

D. Message Types Knowledge Base

Concrete specifications of individual EDIFACT message types are encoded as individuals of the concepts in the EDIFACT Standards Ontology in the Message Types KB. These individuals are created automatically, i.e., the individual message type definitions of the official UN/EDIFACT directories are parsed and for each EDIFACT message type (e.g., ORDERS, RECADV etc.) an individual of class `MessageType` is created. For each type of segment defined in the standard, an individual of `Segment` is created. Analogously, individuals of the respective concepts are created for segment groups, individual fields, components of (composite) fields, code lists and codes that are defined in the standards. A shortened example of an instantiation of the DTM (*DATE/TIME/PERIOD*) segment type along with an exemplary instantiation of a message type (RECADV - *Receiving Advice*) is visualized in the upper part of Fig. 4 (Message Types KB).

As visible in Fig. 4, when parsing and storing the message type specifications from the EDIFACT standards into the ontology the names of fields, segments etc. are slightly changed so that the individuals in the ontology have self-speaking names. This is done by replacing all characters of labels in the standards that are not alphanumeric by an underscore and

also converting some of the characters to lowercase. For example, the segment named "DATE/TIME/PERIOD" becomes an individual named "Date_time_period" in the ontology.

Qualification Relationships. In addition to the message type specifications, in the Message Types KB the qualification relationships between individual fields, components and segments are explicitly specified. After the message type specifications have been integrated into the ontology, these qualification relationships are added manually. Consider the example of the DTM segment instance introduced in Section II. Using the object properties `qualifies_semantics_of` and `qualifies_format_of`, it can be explicitly specified that the *function qualifier* qualifies the *semantics* of the *text*, and the *format qualifier* qualifies the *format* of it (cf. upper part of Fig. 4, Message Types KB).

E. Messages Knowledge Base

Concrete EDIFACT messages and their contained business information are stored as individuals of the EDIFACT Message Ontology in the Messages KB. For every message that should be integrated into the ontology, a new individual of class `Message` is created. For each segment in the message, a new individual of `SegmentInstance` is created and linked to the corresponding `Message` individual by means of an object property `containsSegmentInstance`. For example, a concrete instantiation of a DTM segment in a *RECADV* (Receiving Advice) message becomes an individual of `SegmentInstance` linked to a `Message` individual as shown in Fig. 4 (Messages KB). Concrete values in the messages (i.e., everything that is not a segment identifier nor a syntactic delimiter) become individuals of class `RawValue` and are linked to the corresponding `SegmentInstance` by means of an object property `containsValue`. Values which are known from the EDIFACT standards to represent codes of some code list are *resolved*, i.e., the code is looked up in the code list and the corresponding meaning becomes a new individual of class `ResolvedValue`. For example, when a *function qualifier* of the DTM segment in a message conforming to version 97A of the EDIFACT standards contains the value "137", a new individual of class `ResolvedValue` is created that has a data property assertion `hasValue = "Document/message date/time"`, reflecting the resolved meaning of this code.

Qualification Relationships and Interpretations. By utilizing the knowledge about qualification relationships in the Message Types KB, *interpretations*, as we call them, can be automatically derived. We define an interpretation as a concrete value from an EDIFACT message that is further enriched with information about its semantics and format. Consider the example of the DTM segment instance introduced in Section II. When integrating this segment instance into the Messages KB, an individual of class `Interpretation` with the data property assertions `hasValue = "20110712"`, `hasMeaning = "Document/message date/time"` and `hasFormat = "CCYYMMDD"` (cf. lower part of Fig. 4) is created.

V. Evaluation

We evaluated the EDIonto approach qualitatively by comparing it to two predominant state-of-the-art tools for interpreting EDIFACT messages. These tools include the proprietary software tool *GEFEG.FX*[13] (version 6.1 SP1), and the open-source framework *Smooks* (version 1.4) in a particular way of utilization, namely for transforming UN/EDIFACT messages to XML documents[14].

GEFEG.FX is an industry state-of-the-art tool for documenting EDI messages as well as inspecting and debugging them. Furthermore, it supports the development of mappings between different business document formats. Smooks is the leading open-source framework for converting EDIFACT messages to XML documents where some semantic annotations are added to data elements and values in the resulting XML message representations according to information gathered from the EDIFACT standards.

A. Evaluation Framework

For the comparison we focused on the main objective of the EDIonto approach: Making EDIFACT messages semantically accurately interpretable for both humans and machines. In order to interpret a message correctly, (i) structure, (ii) contained values and (iii) relationships between values need to be known. Hence, we checked the message representations of the EDIonto approach, Smooks and GEFEG.FX with respect to the following properties: (i) explicitness of message structure, (ii) availability of values and (iii) derivability of qualified semantics. In the following, these properties are described in detail.

Explicitness of Message Structure. When interpreting EDIFACT messages, the syntactical position of a data element is of substantial importance for its correct interpretation. This can be further broken down into two aspects: (i) the *logical* and (ii) the *realized* position of a data element.

The logical position of a data element denotes the containing segment group with regard to the hierarchical structure of the corresponding message type as defined in the EDIFACT standards. For example, a *Receiving Advice* message (RECADV) may contain various *line items* (LIN segments) along with corresponding *quantities* (QTY segments). In order to know about the overall meaning of the LIN and QTY segments, it is, however, necessary to know in which *logical* segment group of RECADV messages they have occurred. Assuming they have occurred in so-called Segment Group 25, they belong to a *"[...] group of segments providing details of the product or service received."*[15].

The realized position of a data element denotes its alignment with a specific instance of a segment group in a concrete message. For example, in order to relate individual quantity figures from QTY segments in a RECADV message to their respective line items in LIN segments, it is necessary to be

[13] http://www.gefeg.com/en/gefeg.fx/fx_functions.htm
[14] http://www.smooks.org/mediawiki/?title=UN/EDIFACT
[15] http://www.unece.org/trade/untdid/d97a/trmd/recadv_d.htm

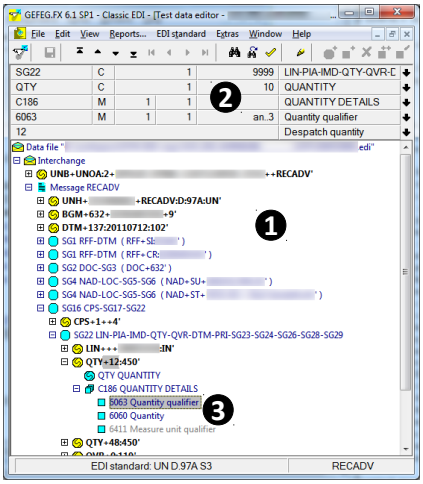

Fig. 5. Modified screenshot of a *Receiving Advice* (RECADV) message being displayed in GEFEG.FX. Some values have been blurred for the sake of confidentiality.

Fig. 6. Screenshot of an XML representation of a *Receiving Advice* (RECADV) message generated using the Smooks framework. Some tags are collapsed as indicated by the plus signs in front of them.

aware which of them occur pairwise in the same segment instances. If a QTY and a LIN segment occur in the same segment group instance, they are in the same *realized* position and can be therefore considered related.

Hence, we evaluate message representations with regard to the visibility and explicitness of both the logical and realized structure of EDIFACT messages.

Availability of Values. Obviously, correct interpretation of EDIFACT messages requires the availability of the contained values in the message representation used for interpretation. However, as illustrated in Section II, some data elements in EDIFACT messages are coded. The contained codes can be resolved to their actual meanings if looked up in the corresponding code lists. This requires that the corresponding code list is known and available to the interpreting human or machine. For many coded data elements, the corresponding code lists are included directly in the UN/EDIFACT directories. For these data elements, the codes can be looked up automatically by software.

Hence, we evaluate message representations with respect to (i) whether the values from actual messages are available and (ii) whether codes in coded data elements are automatically resolved, i.e., whether the resolved meaning of the codes is explicitly available in cases where automatic look-up is possible.

Derivability of Qualified Semantics. As illustrated in Section II, semantically accurate interpretation of EDIFACT messages requires awareness of the qualification relationships between data elements. Hence, we evaluate existing tools and the EDIonto approach with regard to whether those qualification relationships can be concluded from the corresponding message representations. Note that this requirement of *"can be concluded"* has different implications for humans and machines: while to humans qualification relationships are usually identifiable from the context, machines need explicit and formal representation of this information. For this reason, we break this dimension down into two sub-dimensions: derivability of qualified semantics (a) by humans and (b) by machines.

B. Experimental Setup

We set up the EDIFACT Standards Ontology (cf. Section IV-B) and the EDIFACT Message Ontology (cf. Section IV-C) in OWL [21] using the Protégé Ontology Editor[16]. Then, we implemented Java libraries for parsing the textual specifications of the UN/EDIFACT directories and populated the Message Types KB (cf. Section IV-D) with corresponding individuals using the Apache Jena[17] framework (cf. Section IV-D). Subsequently, qualification relationships have been added for exemplary segment types, such as DTM (*Date/time/period*), RFF (*Reference*) and QTY (*Quantity*), using Protégé. Finally, we implemented a library that uses the structural information from the Message Types KB to accurately parse concrete EDIFACT messages and integrate them into the Messages KB (cf. Section IV-E). We used this library to generate a corresponding Messages KB from a real-world EDIFACT data set from an automotive supplier company[18]. Then we inspected the resulting ontologies with the Protégé Ontology Editor; simultaneously, the same raw EDIFACT messages were transformed to XML using the Smooks framework and, in parallel, imported into the GEFEG.FX tool in order to perform the comparison.

Summing up, we compared the following three types of message representations of EDIFACT messages using the above described evaluation dimensions:

- *EDIonto approach:* Ontological message representation (cf. Fig. 4)
- *GEFEG.FX:* In-GUI message representation (cf. Fig. 5)
- *Smooks:* XML message representation (cf. Fig. 6)

[16] http://protege.stanford.edu/

[17] http://incubator.apache.org/jena/

[18] The same EDIFACT data set was also used for a case study on inter-organizational business process mining from EDI messages in [6] and is further described therein.

TABLE I
COMPARISON OF ONTOLOGIES GENERATED WITH THE EDIONTO APPROACH AND STATE-OF-THE-ART TOOLS FOR INTERPRETING EDIFACT MESSAGES WITH RESPECT TO ASPECTS OF INTERPRETABILITY OF CORRESPONDING MESSAGE REPRESENTATIONS

	Explicit message structure		Availability of values		Derivability of qualified semantics	
	Logical	Realized	Not coded	Coded	By humans	By machines
EDIonto ontologies	✓	✓	✓	✓	✓	✓
GEFEG.FX	✓	✓	✓	✓	✓	
Smooks	✓	✓	✓		✓	

C. Results

Table I shows a summary of the results. In the following, each of the evaluation criteria defined in Section V-A is discussed with respect to each of the examined types of message representations of the examined tools and the EDIonto approach, respectively.

Explicitness of Message Structure. In the ontological representation of the EDIonto approach, values (`Value`) are connected to fields and components (`Field`, `Component`) through `hasFieldPath` object property assertions. These fields and components are for their part connected to logical fields, segments, segment groups and message types via `hasComponent`, `hasField`, `hasSegment` and `hasSegmentGroup` object property assertions. The combination of these classes and object property assertions allow for the instant determination of the logical position of a value in an EDIFACT message. Moreover, the classes `SegmentGroupInstance` and `SegmentInstance` along with the object properties `containsSegmentInstance` and `containsValue` allow to assign values to the segment instances and segment group instances they occurred in. This accounts for the identifiability of the realized message structure.

In GEFEG.FX, the content of messages is shown in a tree that resembles the hierarchical structure of EDIFACT message type specifications (cf. ❶ in Fig. 5). When the user clicks on an element of the tree, further explanations from the EDIFACT standards are shown in the upper part of the window (cf. ❷ in Fig. 5). Therefore, the logical position of data elements is immediately visible to a user. Furthermore, instances of data elements are grouped in instances of segment groups. This allows a user to also identify the realized structure of the examined message.

Analogously to GEFEG.FX, the XML representation of EDIFACT messages generated by Smooks provides a hierarchical tree that allows to identify both the logical structure from the names of the XML tags as well as the realized structure by the nesting of concrete data elements in concrete instances of segment groups (cf. Fig. 6).

As a result, all of the examined types of message representations allow for the reconstruction or derivation of the both the logical and realized message structures.

Availability of Values. In the ontological representation of the EDIonto approach, all values from messages are represented as individuals of `Value`. Furthermore, coded values are identified as such by being related to a field or component that has a code list (`CodeList`) attached via the `hasCodeList` object property. During integration of concrete EDIFACT messages, codes are looked up and resolved, resulting in individuals of the `ResolvedValue` class (see Fig. 4 for examples). Therefore, both uncoded and coded values are available.

The GEFEG.FX tool is knowledgeable about whether fields or components are coded and also of the corresponding code lists. Codes are automatically resolved and presented to the user. An example can be seen in Fig. 5 at ❷, where the coded *quantity qualifier* having a value of "12" is resolved to its meaning: "Despatch Quantity".

In the examined version of Smooks, automatic code resolution is not performed. The code lists are also not included in the XML output which makes manual code resolution impossible unless additional data sources containing the code lists (e.g., the EDIFACT standards) are consulted.

Derivability of Qualified Semantics. In the ontological representation of the EDIonto approach, the qualified semantics of data elements are immediately visible through the derived interpretations (`Interpretation`). The *meanings* and *formats* of data elements are directly linked through assertions of the `hasMeaning` and `hasFormat` data properties. For example, consider the automatically derived interpretation of a value of a `Date_time_period` component as shown in the lower part of Fig. 4: The meaning (*"Document/message date/time"*) and format (*"CCYYMMDD"*) are directly attached to interpretation `interpretation1` having the qualified value *"20110712"*. Since this is an explicit and formal representation, the qualified semantics of data elements are identifiable by both humans and machines.

In the XML representations generated by Smooks and the in-GUI message visualizations of GEFEG.FX, qualifying fields and qualified fields are shown together with the corresponding labels from the EDIFACT standards. However, qualification relationships between them are not explicitly represented. For example, the QTY (*Quantity*) segment shown at ❸ in Fig. 5 and at ❸ in Fig. 6 shows a qualifier (*Quantity qualifier*) directly next to a value (*Quantity*), but between those fields there is no explicit link displayed. As a consequence, this generally allows humans to conclude that there is a qualification relationship in place; however, due to the lacking formal representation, machines generally cannot do so.

VI. Conclusion

In this paper, we presented an approach for conceptualizing EDIFACT messages using semantic technologies, the *EDIonto approach*. Thereby, we tackled the problem of implicit semantic relationships between data elements in EDIFACT formats that hamper automated processing of EDIFACT messages in a semantically accurate way. Our proposed approach provides means for modeling these relationships explicitly in ontologies. This allows to automatically derive semantic accurate interpretations of business data contained in concrete messages. We evaluated the approach by comparing the ontological message representations provided by the EDIonto approach with alternative representations of messages used in state-of-the-art tools for EDIFACT. Compared to these other tools, the ontological message representation of the EDIonto approach is the only means for automatic and semantically correct interpretation of qualified data elements in EDIFACT messages. We expect that our approach facilitates use-cases where such messages need to be further processed by algorithms or presented to a user, such as in applications for debugging, validation, visualization and mapping of messages as well as analyses of business data that is contained in messages.

Our future research plans concentrate on an investigation of how redundantly transferred information can be detected in EDI-based inter-organizational business processes using the EDIonto approach, as motivated and outlined in [5]. Furthermore, the EDIonto approach will serve as a basis for subsequent research on business performance analyses based on EDI messages as outlined in [5], [13]. Moreover, we intend to further extend the EDIonto approach so that it can serve as a knowledge base for the automated validation of arbitrary EDIFACT messages.

Acknowledgment

This research has been conducted in the context of the *EDImine* project and has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT10-010.

References

[1] J. Berge. *The EDIFACT Standards*. Blackwell Publishers, Inc., 1994.

[2] J. Dawson and J. Wainwright. Processing EDI Code Pairs. In *Pro Mapping in BizTalk Server 2009*, pages 317–333. Apress, 2009.

[3] Y. Ding, D. Fensel, M. Klein, B. Omelayenko, and Ellen Schulten. The Role of Ontologies in eCommerce. In *Handbook on Ontologies*, pages 593–616. 2004.

[4] M.A. Emmelhainz. *EDI: A Total Management Guide*. John Wiley & Sons, Inc., 2nd edition, 1992.

[5] R. Engel, W. Krathu, M. Zapletal, C. Pichler, W.M.P. van der Aalst, and H. Werthner. Process Mining for Electronic Data Interchange. In *12th Int. Conf. on Electronic Commerce and Web Technologies (EC-Web 2011), Clermont-Ferrand, France, August 29 to September 2, 2011*, LNBIP 85, pp. 77-88. Springer, 2011.

[6] R. Engel, W.M.P. van der Aalst, M. Zapletal, C. Pichler, and H. Werthner. Mining Inter-organizational Business Process Models from EDI Messages: A Case Study from the Automotive Sector. In *24th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2012), Gdansk, Poland, June 25-29 2012*, LNCS 7328, pp.222-237. Springer, 2012.

[7] D. Fensel. The Role of Ontologies in Information Interchange. In *Proceedings of the 2nd International Scientific and Practical Conference on Programming (UkrPROG 2000)*. Kiev, Ukraine, 2000.

[8] D. Fensel. Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In *Proceedings of the IFIP Int. Conf. on Intelligence in Communication Systems (INTELLCOMM), Bangkok, Thailand, 23-26 November 2004*, LNCS 3283, pp. 43-53. Springer, 2004.

[9] D. Foxvog and C. Bussler. Ontologizing EDI: First Steps and Initial Experience. In *Data Engineering Issues in E-Commerce, 2005. Proceedings. International Workshop on*, pages 49 – 58, 2005.

[10] D. Foxvog and C. Bussler. Ontologizing EDI Semantics. In *Advances in Conceptual Modeling - Theory and Practice*, volume 4231 of *LNCS*, pages 301–311. Springer, 2006.

[11] N. Hill and D. Ferguson. Electronic Data Interchange: A Definition and Perspective. *EDI Forum: The Journal of Electronic Data Interchange*, 1:5–12, 1989.

[12] G.K. Janssens. Electronic Data Interchange: From its Birth to its New Role in Logistics Information Systems. *Int. Journal on Information Technologies and Security*, 3:45–56, 2011.

[13] W. Krathu, C. Pichler, M. Zapletal, and H. Werthner. Semantic Inter-organizational Performance Analysis using the Balanced Scorecard Methodology. In *35th Jubilee Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2012), Opatija/Abbazia, Croatia, May 21-25, 2012*, to appear in 2012.

[14] F. Lehmann. Machine-Negotiated, Ontology-Based EDI (Electronic Data Interchange). In *Proceedings of the Workshop at NIST on Electronic Commerce, Current Research Issues and Applications*, pages 27–45. Springer-Verlag, 1996.

[15] J.-M. Nurmilaakso. EDI, XML and e-business frameworks: A survey. *Computers in Industry*, 59(4):370–379, 2008.

[16] B. Omelayenko. Ontology Integration Tasks in Business-to-Business E-Commerce. In *Engineering of Intelligent Systems*, LNCS 2070, pp. 119-124. Springer, 2001.

[17] Object Management Group (OMG). Meta-Object Facility 2.0 (MOF). http://www.omg.org/cgi-bin/doc?ptc/03-10-04 (last visited April 27, 2012), 2004.

[18] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). United Nations Directories for Electronic Data Interchange. http://www.unece.org/trade/untdid/directories.htm (last visited July 4, 2012).

[19] K. Vollmer, M. Gilpin, and J. Stone. *B2B Integration Trends: Message Formats*. Forrester Research, 2007.

[20] W3C. Web Service Modeling Language (WSML). http://www.w3.org/Submission/WSML/ (last visited May 10, 2012), 2005.

[21] W3C. OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/ (last visited May 7, 2012), 2009.