A Meta Message Approach for Electronic Data Interchange (EDI)

C. Huemer*, G. Quirchmayr*, A M. Tjoa+
*Institute of Applied Computer Science and Information Systems, University of Vienna,
e-mail: {ch,gq}@ifs.univie.ac.at
+Institute of Software Technology, Vienna University of Technology,
e-mail: tjoa@ifs.tuwien.ac.at

Abstract. Still most of the current information systems are intra-organizational. Since products, services and production processes have become more information intensive, there is an increased need to transfer these information between organizations. Electronic Data Interchange (EDI) standards have been proposed to exchange these information between two independently operating information systems. All these standards have in common that they standardize business documents—which represent the relevant information—on instance type level. Since a data interchange between business partners is always based on a small subset of the standards, it is necessary to agree upon this subset. In our paper we present a concept for interchanging this agreement via EDI itself. The presented concept leads to a standardization on the meta level.

1 Introduction

Many years ago the industry discovered the great benefits of electronic information transfer. Various exchange formats, or so-called EDI standards, have been developed. Most of them were proprietary standards or at least restricted to either a certain branch (ODETTE) or a certain region (ANSI X.12). In 1987 the ISO published the syntax of UN/EDIFACT as an international and intersectorial valid standard [2,14]. As soon as the first directory of EDIFACT came out, companies started to build and sell their own EDI translation software and to use EDI messages for orders and invoices. Over the years, it became clear that it was rather difficult for a company to get started with EDI. A case study involving about 60 European organizations showed that most of these organizations were not able to derive the expected benefits from EDI [3]. So the implementation of EDI was slower than originally expected.

Although it is often stated that EDI is 80% business and only 20% technique [4], we feel that the technical aspects of EDI are quite underestimated. It is our opinion that the higher the complexity of EDI techniques the harder their integration into the business. A more efficient method of interchanging business data would therefore reduce the costs of implementation and only the benefit/cost ratio will be relevant when deciding whether or not to participate in EDI. The fact that 90% of the Fortune 1000 enterprises have invested in EDI, but less than 1% of the small and medium enterprises are involved in EDI, indicates that the current method of exchanging business data is not mature. Consequently, there is a growing need for new methods which will allow small and medium enterprises to participate in EDI.

It is easy to detect, that EDIFACT—as opposed to its intention—is not an international and branch-independent standard. International and intersectorial from a implementation point of view would mean that any message created by the sending application in the standard conform format will be automatically processable by the receiving application. This would require the following two conditions: First, both information systems must have the same understanding of the interchanged data. Second, the receiving application must be able to process any data that might be included in a standard message. Unfortunately, none of these conditions are fulfilled.

Business partners willing to exchange data electronically in a structured format have

first to agree on the actual data they want to interchange. The format of these data is mainly determined by the semantics the involved information systems are able to process. Hence, business partners have to sit down, discuss how they are going to interchange files and implement these specifications within specific translation software. Consequently, a detailed functional agreement is needed for each business relationship. It follows that business partners—although using the international and intersectorial EDIFACT standard—in fact, use a corresponding proprietary standard for each business relationship. In this paper we present a concept for exchanging the functional interchange agreements between the business partners via EDI itself. Other concepts which address the problem of functional agreements include Open-edi [10], Business System Interoperation (BSI) [13], and Object Oriented edi [1].

2 Shortcomings of the Current EDI Standards

Although the advantages of EDI are well known, most of the SMEs are not able to participate in EDI. Legal aspects and security problems are some of the reasons. But there is also a substantial number of shortcomings in the information technology aspect of the current standards. The following problems are encountered [7,11,12]:

- Resulting structures are too complex and consequently too hard to read and to navigate.
- Multiple standards and different versions of each standard are in use.
- Semantics are not part of the EDI standard.
- Semantic interpretation of the standard is included in implementation conventions, which are different for each industry sector and/or geographical region.
- A detailed interchange agreement is necessary to establish an EDI relationship to a trading partner.
- Overhead in network costs and reduced processing efficiency due to segment tags and delimiters marking unused data.
- Standards are published only in English.
- Business Processes are not considered by the standards. Consequently, integrating EDI into the business processes of an organization is too hard to perform, especially for small and medium size enterprises.
- The current translation software is too inflexible. The process of retrieving EDI messages from a mailbox, translating it into a flat file, convert the flat file into a database import format and import this into the database of the business application is much too complex.
- A change request to an EDI standard is much too time-consuming due to the bureaucracy of the standard organizations.

3 The Meta Message Approach in Detail

The proposed method is based on an EDIFACT meta message, which allows the transmission of the format of the messages carrying the business data. Candidates for meta messages are the *Directory Definition Message* (DIRDEF), the *EDI Implementation Guideline Definition Message* (IMPDEF), and the *Functional Agreement Definition Message* (FAGDEF).

DIRDEF is a message developed by the UN. It allows the transmission of an EDIFACT Directory set or parts thereof in EDIFACT syntax [15]. IMPDEF is used to put the contents of a Message Implementation Guideline (MIG) into an EDIFACT message [5]. Nevertheless, both DIRDEF and IMPDEF are in some respect not optimal for our purpose. Therefore, we combine the best features of DIRDEF and IMPDEF and extend

them by further concepts to create a new meta message—FAGDEF—which is best suited for our approach.
However, it is a global goal that our approach is independent of the specific format of a meta message. This means that we provide a generic approach which can easily be adapted to a specific kind of meta message. Consequently, we achieve a great flexibility in the sense that it is possible to accommodate the approach to future versions of existing meta messages as well as to new types of meta messages.
The message independent approach can be described as follows: The core component of our method comprises a tool for building functional agreements which we call *Functional Agreement Designer.* This tool allows the design of a meta message. Furthermore, mapping tables to the internal storage format of directories and functional agreements can be specified according to this design. Hence, each kind of meta message can be imported to derive functional agreements with the *Functional Agreement Designer.* These functional agreements can be based on subsets of existing messages and on wild subsets which manipulate the messages in a non standard conform way. Furthermore, functional agreement specifications can also incorporate concepts not covered by the standard, like fixed and optional components. For functional agreements including additional concepts which are disregarded by the standard we use the term ‘exchange agreement’. It is obvious that also subsets of already defined exchange agreements can be established. Accordingly, the *Functional Agreement Designer* enables the adoption of a message design to the real business needs of the user's company. The self-created functional agreements can be translated into a meta message of any included meta message format. These meta messages can be transmitted electronically to the partner company. Furthermore, the message definition in the meta message format should be a valid input format to the EDI translation software. The translation software is used to map an instantiated EDI message carrying business data to the input format of the business application. If the business partner also uses an EDI translation software that is able to accept the meta message as input format, he will be able to handle messages in the format created by the initiating company. This means a consequent extension of the basic idea of EDI, because the functional agreement on the interchange structure between two companies will be based on EDI [8].

3.1 Functional Agreement Definition Message

As mentioned above, both DIRDEF and IMPDEF are not optimal for exchanging functional agreements in EDIFACT environments. DIRDEF allows to specify component usage just for one step down the component hierarchy. This means that it is possible to cite which segments are used in a message. But it is not possible to designate the usage of data elements within a certain segment of a message. Consequently, all segments of the same type are structured equally regardless of their position in a message. This might be sufficient for the standard specification, but is not adequate for the specification of exchange agreements between business partners. The same problem applies for codes assigned to a coded simple data element. DIRDEF does not provide a possibility to specify that in a certain position within a certain message one subset of codes is allowed and at a different position within a different message (or even in the same message) a different subset of codes is applicable.
In contrast, IMPDEF provides the concept of multi level component definitions. But unfortunately, this concept is equivocally implemented in IMPDEF. This is due to the fact that the position specific component specification is made totally independent to the general subdirectory definitions.
In addition to these problems the specification on the usage of a component is too gen-

eral in the standard specification. The standard covers only two kinds of requirement designators to indicate that a component must be used or can be used in an interchange. But these are not sufficient to describe in detail the requirements of a component. On the one hand a component might be specified in the standard as conditional, but is required by the receiving application. This problem could be solved with DIRDEF and IMPDEF. But on the other hand more specific requirement designators are needed to denote the usage of components. A complete segment usage can be described by combinations of the following designators:

- Mandatory: The component must always be specified
- Conditional: The component could be specified. No specification means a null value for the corresponding field.
- Optional: The component could be specified. No specification means that a default value is applied for the corresponding data field
- Default: A default value for an optional data field.
- Fixed: A fix value specified which can not be updated in an interchange

Furthermore, the combination of requirement designator and maximum number of occurrences specified for a component is inadequate. In the EDIFACT standard definition only one designator and one number of occurrences can be specified. For example

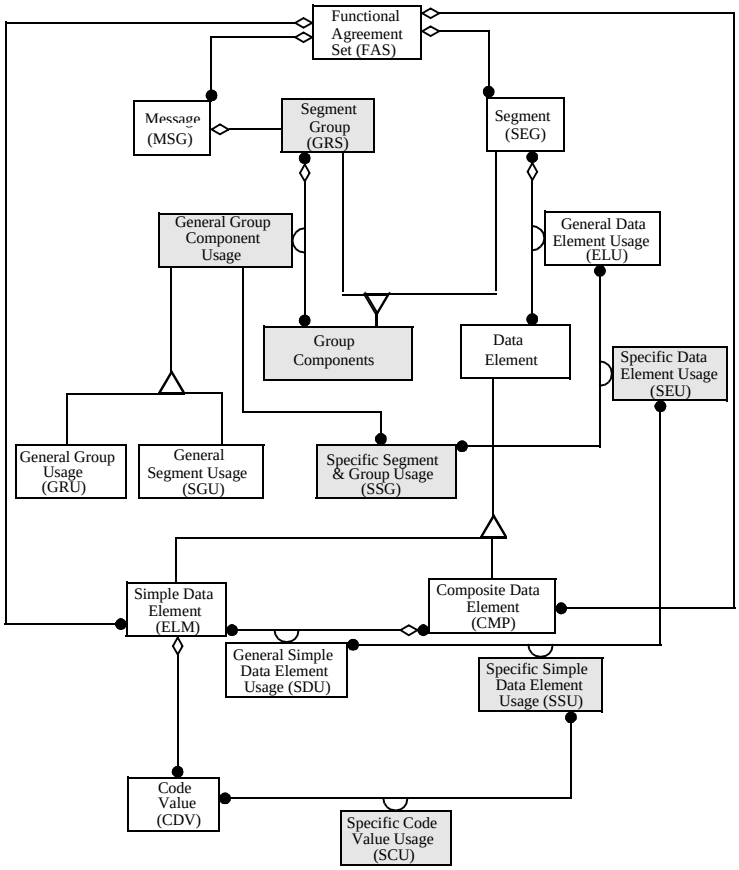

Fig. 1. OMT-Model of the FAGDEF Message

`M 9` means that the component must be specified at least once, but can be specified up to 9 times. It is not possible to specify who many times a component must be included and how many times it could be stated. The combination of the various above mentioned requirement designators and a number of occurrences specified for each designator would be more appropriate for the specification of data fields to be interchanged.
To overcome these problems, we define a new EDIFACT message which exactly meets the requirements of our objective. According to the overall goal we call this message *Functional Agreement Definition Message* (FAGDEF). A complete technical description of FAGDEF in style of an EDIFACT boiler plate is given in [9].
The basic technology used in FAGDEF is explained by means of the OMT diagram in Figure 1. The classes in the white boxes and their interrelationships of Figure 1 are derived from DIRDEF and IMPDEF. The class represented in grey boxes and their relationships to other classes are the result of two new main concepts in FAGDEF.
The first new FAGDEF specific concept is the explicit representation of segment groups. This concept is necessary, because of different treatment of the same group for fixed, optional, mandatory, and conditional usage. Accordingly, we establish the class *Segment Group* as first level component for message structures. Conceptually, the described message structure is identical to the structure of one fictitious segment group (Segment Group 0). This fictitious segment group covers all segments and segment groups of the first message level. Therefore we have a 1-to-n link between *Message* and *Segment Group*. Segment groups are composed of segments and further subgroups. Therefore, there is a n-to-m relationship between *Segment Group* and *Group Components*, which is a generalization of *Segment Group* and *Segment*. The general usage of a group component within a segment group is described by *General Group Component Usage*, which is a link attribute to the n-to-m relationship between *Segment Group* and *Group Components*. Depending on whether the component is a segment or a segment group, the specializations *General Segment Usage* or *General Group Usage* are responsible for the link specification.
The second main concept is that of multi level component definitions, which is also used in IMPDEF. In FAGDEF it was our goal to implement this feature unequivocally. As a consequence we have tried to keep a consistent relationship between the position specific multi level usage and the corresponding general component usage. This means, for example, that the usage of data elements in a segment at a specific position within a message should be specified in context with the general description of the data element usage in the corresponding segment. Conceptually this means that in IMPDEF the specific component usage is a relationship between the specific component usage of the above level and the component elements. By way of contrast we have implemented this feature as relationship between the specific component usage of the above level and the general component usage of the level in question. For this purpose we explicitly distinguish between FAGDEF meta segments which describe the general component usage and meta segments which describe the specific component usage.
At the first level of component usage we are faced with multiple specific derivations of the general group component usage. Therefore, there is a 1-to-n relationship between *General Group Component Usage* and *Specific Segment & Group Usage*. On the following levels we keep the relationship between the specific usage of the above level and the general usage of the same level. Thus, *Specific Element Usage* is a link attribute to the n-to-n relationship between *Segment Usage* and *General Element Usage*, and *Specific Simple Data Element Usage* is a link attribute to the n-to-m relationship between this *Specific Element Usage* and *General Simple Data Element Usage*. Finally, the specification of a subset of valid codes in *Specific Code Value Usage* is a link

attribute to the n-to-m relationship between *Specific Simple Data Element Usage* and *Code Value*.

3.2 Software for the Meta Message Approach

This subsection covers a description of the *Functional Agreement Designer* which is a software tool to support our Meta Message Approach. Furthermore, we present the necessary interactions between the *Functional Agreement Designer* and the translation software to be able to exchange messages on the basis of self-designed functional agreements. The whole software needed to perform application-to-application data exchange is depicted in Figure 2.

The *Functional Agreement Designer* is composed of two core components, the *Meta Message Mapper* and the *Browser & Editor*. Both components are based on the *Directory Definition & Functional Agreement Set Database* which integrates the different components of the *Functional Agreement Designer*. The database of the *Functional Agreement Designer* contains information on meta message designs (*Meta Message Design Database*), on structures of EDIFACT standard directories, of subsets thereof, and of exchange agreements (*Directory & Agreement Structures Database*). Furthermore, it covers information on the mapping between the meta messages and the EDIFACT or functional agreement structures (*Mapping Table Definitions Database*). Furthermore, there must be a connection between the *Functional Agreement Designer* and the communication interface to be able to receive and send meta messages.

In order to browse through an EDIFACT standard directory or through an exchange agreement, it first has to be included into *Directory Definition & Functional Agreement Set Database*. This task is performed by the *Meta Message Mapper*. For this purpose the *Meta Message Mapper* must be aware of the design of the meta message. This design specification might be received by another meta message which is already known by the *Meta Message Mapper*. But it can also be designed from scratch with the *Meta Message Designer*, which is part of the *Meta Message Mapper*. The design of all included meta messages is stored in the *Meta Message Design Database*. Note that the functionality of the *Meta Message Designer* is identical to that of the *Browser & Editor*.

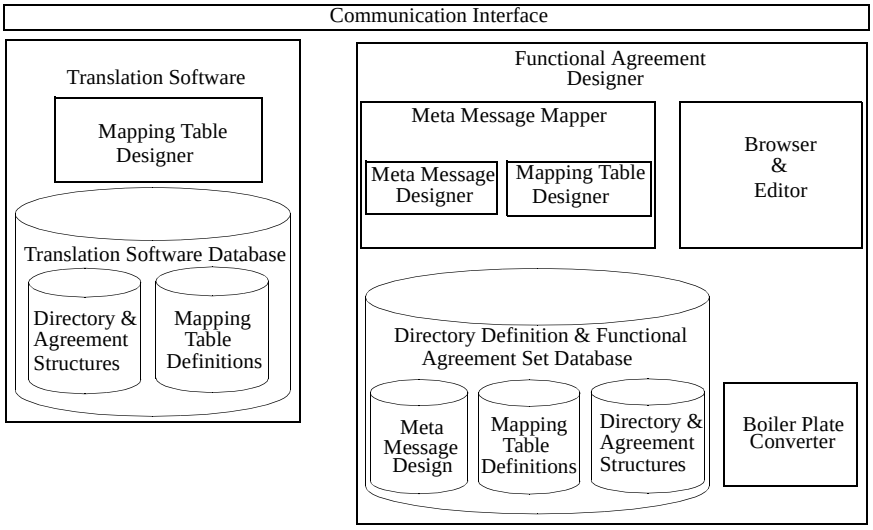

Fig. 2. Software for the Meta Message Approach

Afterwards mapping tables which describe how to transform ‘conventional’ messages (=the contents of a meta message) into the internal representation of the *Functional Agreement Designer* must be defined. This function—similar to that of specifying mapping tables in the translation software—is fulfilled by the *Mapping Table Designer.*
After specification of the mapping tables, meta messages of an included type might be received electronically via the communication interface. According to the type and version of a meta message the appropriate mapping table is automatically loaded and the contents of the meta message is transformed into the format of the *Directory & Agreement Structures Database*. Received meta messages might include descriptions of EDIFACT standard directories, (wild) subsets of the standard directories, implementation conventions, or exchange agreements created by a business partner.
All directories and their derivations included in the *Directory & Agreement Structures Database* can be accessed via the *Browser & Editor.* The *Browser* is designed to navigate through the EDIFACT standard directories, subsets and exchange agreements in a very flexible and clearly arranged manner. It allows access at each level of the EDIFACT hierarchy (messages, segments, composite data elements and single data elements) and offers links between these levels [6]. The *Browser* also serves as a starting point for the Editor. By using the *Browser & Editor* it is possible to create subsets of standard directories, completely new messages and exchange agreements [9]. The results are stored in the *Directory & Agreement Structures Database.*
Although a stand-alone *Functional Agreement Designer* might be useful for the documentation of functional agreement definitions, its full power will only be reached if the designed specifications can be transferred to the translation software. Consequently, it is necessary to convert the definitions stored in the internal database into meta messages. For this purpose mapping tables can be specified with the *Meta Message Designer* to export the (wild) subsets and functional agreements into a meta message format (which must be included in the *Meta Message Design Database*). Note, that the expressiveness of the meta message should be at least as powerful as the expressiveness of the output alternative of the *Browser & Editor,* because otherwise self-created specifications might be lost.
Another criterion for the selection of the meta message is the ability to import it into the translation software. At the moment commercially available translation software is not flexible enough to import directory specifications given in a meta message into the translation software database. But we expect them to be open at least for standardized meta messages in the foreseeable future. May be they will also be equipped with a tool similar to our *Meta Message Mapper* to except any meta message format. If the meta message created by the *Functional Agreement Designer* can be imported into the translation software, mapping tables for ‘conventional’ EDIFACT messages based on created functional agreements can be defined. Accordingly, EDIFACT messages whose format was created with the *Functional Agreement Designer* can be received via the communication interface, translated in compliance with the mapping tables into an interface file, which is finally imported with the converter software into the business application’s database. Vice versa, data exported from the business application’s database might be translated according to the mapping tables into an EDIFACT message based on a *Functional Agreement Designer* specification, and sent via the communication interface.

3.3 Exchanging functional agreements

In this subsection we describe our proposed scenario for exchanging functional agreements which are based on the business needs of the business partners via EDI and the

subsequent exchange of messages based on these agreements. The description of this scenario should be read in conjunction with Figure 3 which depicts the overall process. The first step for the initiating business partner is to include the EDIFACT standard directories into the *Browser & Editor*. Hopefully, in the near future this can be done by receiving on-line a meta message from an EDIFACT reference database (1a). At the moment it is only possible to import them from the directory descriptions provided in ASCII formats via the *Boiler Plate Converter* (1b).
Now the initiating business partner develops an EDIFACT conform or wild subset (2a) or an exchange agreement (2b) on the basis of a standard version. The resulting exchange specification will include only those components which will actually be exchanged in the business transaction. Via the documentation tool the initiating business partner can print a documentation of the self-created functional agreement, which can be used as implementation guideline (3a, 3b). The functional agreement definition can be exported from the *Functional Agreement Designer* into a meta message (4a, 4b). In order to ensure that the business partner has the same understanding of the functional agreement the meta message must be sent to the responding business partner.

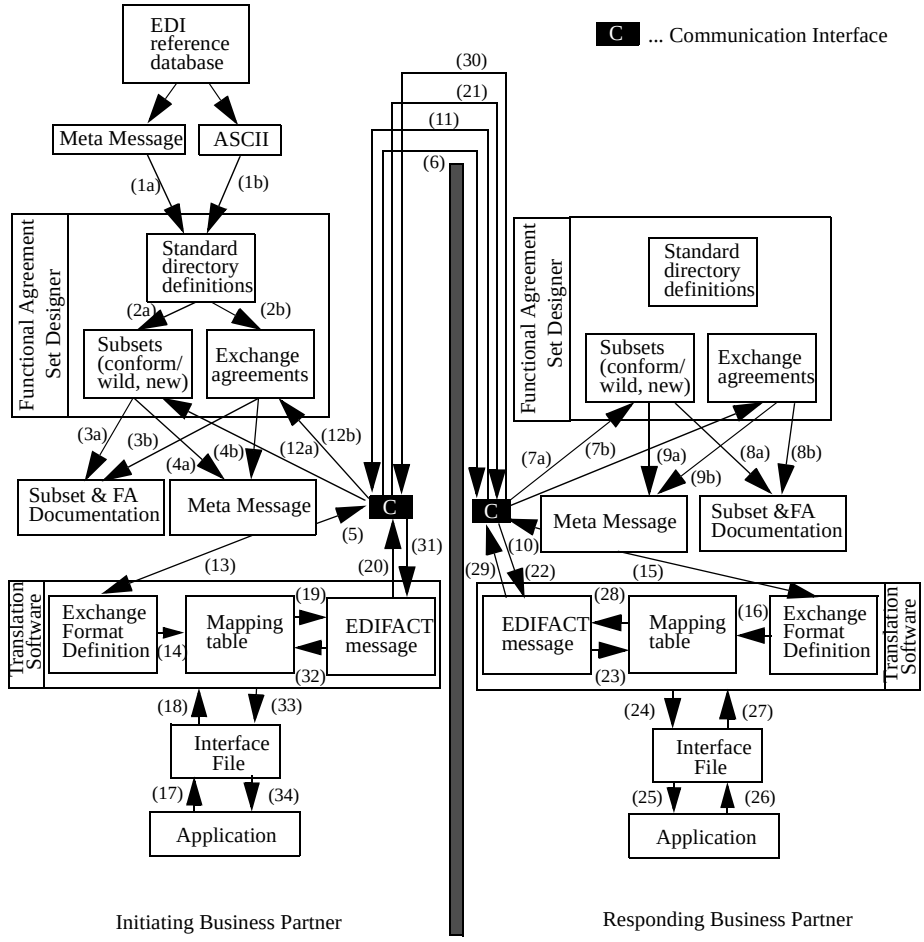

Fig. 3. Scenario for Exchanging Functional Agreements

Thus, the resulting meta message is passed to the communication interface (5), which is responsible for the transmission to the business partner (6).
The responding business partner receives the meta message via his communication interface and imports the functional agreement definition into his *Functional Agreement Designer* (7a, 7b). He is now able to verify the definition created by the initiating business partner via the *Browser & Editor.* If he detects any discrepancies, he can also adopt the functional agreement. Similarly to the originator, the responding business partner may print a documentation of the functional agreement (8a, 8b) and export the functional agreement definition into a meta message (9a, 9b).
If the responding business partner has made any changes to the functional agreement specification, he must inform the initiating business partner. Therefore, he passes the meta message including the new functional agreement specification to his communication interface (10) in order to transmit back to the originator (11). The initiating business partner receives the meta message via his communication interface and imports the adopted functional agreement specification into his *Functional Agreement Designer* (12a, 12b). Now, he can again verify and/or adopt the functional agreement specification. If he makes any improvements, he must again inform the responding business partner who can then react on the changes. Consequently, the processing steps 3 to 12 may repeat as along as both business partners agree on a common functional agreement specification. The agreement process will end when a business partner receives a meta message where no changes were made.
As soon as there is an agreement on the exchange format, both business partners will provide the functional agreement specification to their translation software (13, 15). Note that for this purpose the translation software must accept the type of meta message as valid input format. When this occasion arises both business partners are in the position to design their mapping tables (14, 16).
Now, business transactions based on this functional agreement specification might be processed. A business scenario based on the defined functional agreement which is presented in the steps 17 to 25 will be similar to a common EDIFACT scenario. The responding business scenario is depicted in the steps 26 to 34.

4 Summary

The meta message approach is designed to overcome the problems of current standards. The basic idea is to reduce standardization to one type of standardized message, namely a meta message used to describe all other messages. Hence, an interchange agreement is based on a meta message and realized by EDI itself. When the structure of the business messages are not any more standardized, but agreed upon between business partners by a meta message, multiple versions of a message are no longer a problem. An exact message structure is defined for each business relationship. Therefore, change requests to the standardization bodies are not so important any more. They can simply be defined in the meta message.
Business processes are partially reflected in the meta message approach. The sender application program determines which data can be produced by the business process implemented. This specification is made available to business partners via the meta message. Business partners verify whether or not the implemented business processes in their information system can process these data. The process of interchanging meta data will continue until an appropriate interface between business processes is reached.
If a meta message will only cover those data types and codes included in a real interchange, the long and complex structure of EDIFACT is reduced to an absolute minimum. Therefore, there is no overhead in network costs and no reduced processing

efficiency due to segment tags and delimiters marking unused data. Nevertheless, the problem of dispersed semantics is not solved by the meta message approach.
The process of determining a business relation specific interchange format also ensures that the involved business partners will have a common understanding of the semantic interpretation of included data types. The semantics of data types can be documented in free text fields of the meta message. Semantics are part of the interchange agreement, but not provided in a processable format.
The meta message approach makes high demands on future translation software. Tomorrow's translation software must be able to accept the format of a meta message as valid input format. It must be more flexible by allowing mapping tables to be the direct interface between the EDI message and application data structures. However, multilinguality is not explicitly covered by the meta message approach, but could be easily incorporated [6].

References

[1] AC.1; The next Generation of UN/EDIFACT - An Open-edi Approach using IDEF Models & OOT; June 1997; http://www.premenos.com:80/klaus/ooedi/

[2] Berge J.; *The EDIFACT Standards*; NCC Blackwell Limited; Oxford; 1991

[3] Bielli P.; *Social and Economic Impacts of EDI: Executive Summary*; Department of Information Systems;
Bocconi University; Milano;

[4] Emmelhainz M.A.; *Electronic Data Interchange: A Total Management Guide*; Van Nostrand Reinhold; New York; 1990

[5] Hage C.; *EDI implementation guideline definition message (IMPDEF)*;
http://www.chage.com/chage/edi/impdef/impdef6b.txt

[6] Huemer C., Tjoa A M.; *A Multilingual Browser for the UN/EDIFACT Communication Standard*; Proceedings of the 21th Euromicro Conference; Como, Italy; 1995; pp 748 - 753

[7] Huemer C., Quirchmayr G.; *The Dilemma of EDI - Problem Analysis of Current Standards*; Proceedings of the 7th Information Resources Management Association (IRMA) International Conference; Washington; May 1996; pp 101 - 107

[8] Huemer C., Quirchmayr G., Tjoa A M.; *Modelling Functional Agreements in EDIFACT Environments*; Proceedings of the IFIP/ICCC International Conference on Information Network and Data Communication; Trondheim, Norway; June 1996; pp 87 - 100

[9] Huemer C., Electronic Data Interchange (EDI): Standards, Shortcomings, Solutions; Ph.D. Thesis; Institute of Applied Computer Science and Information Systems, University of Vienna; February 1997

[10] International Organization for Standardization; *The Open-edi Conceptual Model*, ISO/IEC JTC1/SWG-EDI Document N222; 1991;
http://www.premenos.com:80/klaus/open-edi/concept.html

[11] Steel K.; *Matching Functionality of Interoperating Applications: Another Approach to EDI Standardisation*; ISO/IEC JTC1/SC30 IT11/7/94-108; September 1994

[12] Steel K.; *The ICARIS Project*; ECA 95; University of Melbourne, Department of Computer Science; 1995

[13] Steel K.; *Business System Interoperation*; University of Melbourne; Department of Computer Science; July 1996; ftp://turiel.cs.mu.oz.au/pub/edi/bsiintro.doc

[14] United Nations, Economic and Social Council; *UN/EDIFACT Syntax Rules*; ISO 9735; International Organization for Standardization; Geneva; 1987

[15] United Nations, Economic and Social Council; *Directory Definition Message (TAG 52.315)*; Trade Data Interchange Protocols; Geneva; 1993