Another Approach to Standardising EDI

Almost twenty years have elapsed since the initial steps were taken, but Electronic Data Interchange (EDI) still remains the elitist preserve of a few of the larger companies. Following current statistics, 35'000 companies in the U.S.A., 15'000 in the U.K., 5'500 in Australia together with the rest of the world make up a total of only 70'000 companies using EDI [1]. Compared to the number of businesses in the world, this rate of progress can only be described as very poor. The following article gives a summarization of a new approach to EDI standardisation which will lead to a faster dissemination of EDI [2].

In building our solution, we first analysed the problems that exist with the current standardisation processes of X12 and EDIFACT. Then we went back to first principles to see what we would have done if we were starting from scratch. Next we polished our crystal ball to try and predict where we would get to if we could eradicate all the current EDI problems. Finally we devised a scheme to allow us to move out of the current rut and toward our vision of the future in a series of small steps that can co-exist side by side with the current way of using EDI.

** by Ken Steel*
University of Melbourne

Current Problems

The problems in the current standardisation approach to EDI can be summarized in eight points:

1. It takes far too long to navigate the standardisation process.
2. There are at least two standards organisations involved (X12/EDIFACT).
3. When standards are updated, users tend not to move to the new version but stay with the one they are using multiplying the number of 'standards' in use.
4. In building the generic structure, there should be a rule base to guide the use of conditional fields. This is not done, so each industry has a working party to produce Implementation Guidelines which 'interprets' in its own way how to implement the 'standard'.
5. The result is a myriad of implementations of versions of EDIFACT and X12 messages. As time goes on this will get even worse to the point where one has to ask if it can still be called a standard.
6. The concept of standard segment contents means that by the time the actual interchange is implemented, there are only one or two data elements actually used in each segment. This can result in the overhead of the segment tag plus the delimiters marking unused data elements being greater than the character count of the actual data. This increases network costs and reduces the general efficiency of the data transfer and processing. Because segments are built in such a random way, there has been no attempt to place data elements within their relational context. This makes complex functionality difficult to unscramble.
7. With the advent of the concept of public data bases, the whole EDIFACT and X12 concept falls down completely. Unfortunately, the reason for this needs a lot of explanation and is outside the scope of this short summary. To handle this deficit, a new way of standardising EDI structures is needed, so why persist with the current one that only works some of the time?
8. The current standardisation is very cumbersome when applied in an Open-EDI environment where prior arrangements between trading partners do not occur. Open-EDI is needed for ad-hoc and short-term relationships which the cost of formal EDI trading agreements renders non-viable [3]. Current methods are not suited to interactive EDI.

An Alternative Solution

The principles of the new approach to EDI standardisation can be summarized in seven points:

1. Rather than standardise the structure of the interchange, allow the actual structure of the interchange to flex infinitely and devise an electronic method of communicating the actual interchange structure being used to the other party. In other words, design a standard EDIFACT 'meta-message' which can define the structure of any message used to carry the data for the transaction being processed. This has been done and the Interchange Structure Definition (ICSDEF) message [4] is currently being assessed by the local TAG.
2. The only data that needs to be interchanged are those data elements needed by the responding application. It is the responding - not the initiating - application that determines the data needed to process the transaction.
3. The concept of standardised segments can be eliminated and even though the familiar 3-character tag is used (to retain EDIFACT syntax), any suitable characters can be chosen for the segment tag (except for around 20 reserved for service segments) and any data elements *needed* between a specific set of trading partners are used. This eliminates most 'null' elements and the overhead of the delimiters marking the missing data.
4. The ICSDEF meta-message is sent in front of the message containing the transaction data the first time an interchange takes place between a specific set of trading partners and whenever the structure of the data interchange is modified. The receiving partner stores this ICSDEF in a directory for use in decoding any future messages for that type of transaction from that trading partner in the future.
5. The basis of the data element identification is not numbers, but meaningful names. The problem of relating names in different languages is handled in a neat way by a 'standard language label' which cannot be described here.
6. The names used are standardised by using a standard Data Dictionary for each transaction [5]. This is built electronically from EDIFACT tred, trcd, trcl and the appropriate EDIFACT message. The Data Dictionary is primarily intended for use across a machine-to-machine interface and also carries the semantic and granularity information. With further research it may be possible to build a standard Data Dictionary to cover an entire business/information system scenario, but small steps usually succeed where grandiose strategies fail. Here we need to collaborate with the BSR project to avoid re-inventing the wheel.
7. Whenever new functionality needs to be added to a transaction which involves additional data elements, the new data elements are added to all the involved trading partners' dictionaries and to the interchange structure via an ICSDEF message. This is also the reason why numbers cannot be used to identify data elements.

The overall objective in building up the approach in this way is to enable it to operate without human intervention in directly integrated knowledge-based applications a few stages further down the track.The standardisation is achieved in two parts: the Data Dictionary and the ICSDEF which constitutes a standard way of defining the transaction's context and the data structure of the interchange.

Implementation of the New Approach

A new breed of 'translator', we call it an EDI Server, is produced which loads an

ICSDEF message and uses it to decode/encode the incoming/outgoing transaction data. Many non-programming people fail to appreciate the radical difference from a conventional 'smart' translator when first presented with this concept. How successful I will be in getting the distinction across in this short summary is problematical.

Firstly, the EDI Server is usually an object module that is actually linked in with the application program. It is accessed from the application via a service interface so that instead of reading and writing from disk files, read and write calls to the EDI Server are made. (There's a bit more to it, but this is the basic principle). There's also the inter-process message approach to accessing the EDI Server from the application, but in the interest of brevity, I won't dwell on that. Secondly, the EDI Server can be operated in any of batched, conversational or interactive EDI modes. Thirdly, the service interfaces are capable of being rigorously standardised, whereas the 'flat file format' associated with translators, if standardised, suffers from the same problems that EDIFACT and X12 standards do now. In performing the inter-structure mapping, the ICSDEF describing the EDI file is used to decode the incoming EDI message into data elements. The ICSDEF describing the data structure required by the application program is used to assemble the data elements into the format for handing on to the application program. (The reverse is used for outgoing messages).

EDI-enabling Existing Applications

Having sat through software producers' working parties and listened to their problems handling current EDI methods, it became apparent that the move from the EDI message to the application program was too hard for several reasons. Software producers wanted to EDI-enable their software without having to master EDIFACT and X12 complexity. A lot of the translators produced/used a fixed flat file format which required the software producers to add yet another conversion step to reconstitute the flat file into another flat file suitable for the application program. Those translators that allowed flexible mapping to the flat file were perceived to be too complex to handle. The multi-conversion process is too complicated for the level of user who buys shrink-wrapped packages and the support effort jumped up to non-viable levels. Translators add a lot of cost to the total software bill - often more than the package.

Using this approach, EDI-enabling is a breeze (comparatively). Assuming the application is already capable of reading/writing input/output from a file instead of from a VDU, all that has to be done is: Change the reads/writes to subroutine calls and build an ICSDEF to describe the structure of the data needed.

Advantages

This whole process can be done in just a few minutes per module, and there is a single step process from EDI message to application which makes it suitable for the shrink-wrap market. By building a set of ICSDEFs to describe each implementation of each version of each standard being used in a particular area, the new approach can be introduced unilaterally by a trading partner at either end, without affecting the other trading partner's ability to use conventional translators or 'Old-EDI'. Other advantages of this new approach when fully implemented are:

1. It's very suitable for accessing public data bases (the explanation is a bit involved for this short summary).
2. It can be used with 'Old-EDI', Open-EDI, Conversational-EDI or Interactive-EDI with equal ease.
3. It carries the infrastructure to work into knowledge-based applications.
4. It eliminates the current EDIFACT/X12 standards process and its delays.
5. It almost eliminates the need for working parties. The need for implementation guidelines disappears.
6. The structuring of segments means a large reduction in the number of segments in a message and a large reduction in segment overheads. This equates to lower network costs and higher processing efficiency. The programming task is a lot simpler.
7. EDIFACT/X12 messages as such are eliminated and only one 'translation' is needed. The 'flat file' out of one application is transmitted and the EDI Server at the other end maps this straight into the application. The ICSDEF describing the initiating applications file - the one transmitted - and the ICSDEF describing the structure of the data needed by the responding application are the two used by the EDI Server. The ISCDEF at both ends are produced by the respective application programmers - eliminating the need for any standards organisations to become involved in the middle.
8. EDI goes firmly 'under the hood' and becomes invisible to the user.
9. Users and their Working Parties (if needed) now concentrate on the functionality of the applications and the semantics involved with the Data Dictionary.

The Future

This approach to EDI is only a transitory stage and we have already placed it in the 'obsolete' file for Research. The next stage is to turn the EDI spotlights right off and to focus on models of the business processes (and others) - the basis of Open-EDI. By defining standard objects of functionality, ICSDEF can be replaced by FUNDEF which is a very short message of a few bytes per application which carries functionality definitions for the whole business/information system, not just one transaction. Now the trick is to resolve functionality differences between interoperating applications, and we're already well along this path with our research. And it's all done by the computers, no humans in sight.

The next step is to replace conventional application programming with knowledge-based software engineering. Here the line managers can operate the rule base controlling the applications to make the computer do precisely what they want it to do. When market or business conditions change, the applications can be changed in minutes without having to alter any computer programs. ■

References

[1] *Baker, M.:* Personal Communication, EDICA, April, 1994.

[2] *Steel, K.:* Matching Functionality of Interoperating Applications: Another Approach to EDI Standardisation, Working Paper ISO/IEC JTC/SC30 IT11/7/94-108, Melbourne 1994.

[3] *Cromack, G.:* The Open-EDI Initiative and Its Place in IT11/7, Working Paper ISO/IEC JTC1/WG3 IT117/94-101, January 1994.

[4] EDIFACT message ICSDEF currently has status 0 (release 1).

[5] *Steel, K.:* Construction of a Standard Data Dictionary for Use in an Open-EDI Environment, Working Paper ISO/IEC JTC1/WG3 IT1 17/94-xxx, September 1994.

[6] *Steel, K.:* The Definition of Interchange Structures in an Open-EDI Environment, Working Paper ISO/IEC JTC1/WG3 IT1 17/94-109, September 1994.

** Ken Steel is Manager of the ICARIS Research project at the University of Melbourne in the Department of Computer Science. The aim of ICARIS is to enable intelligent business systems which essentially run themselves, to enable large centralised databases to be split into departmental databases and synchronised, and to automate Electronic Commerce. Standards Australia is the local arm of ISO/IEC. The author is member of the Technical Committee involved in Open-EDI (IT11/7).*