ERP Interfaces for Enterprise Networks

An XML approach

Szilveszter Drozdik
drozdik@sztaki.hu

Abstract: Although Enterprise Networks may be formed for different purposes, the need of integration always pops up. Integration requires interfaces, clear and consistent constructions of the communication. ***FLUENT*** *(Flow Oriented Logistics Upgrade for Enterprise Networks*) is a joint European project (Esprit IV) in which – among others - *ERP* (*Enterprise Resource Planning*) interfaces are defined and implemented for systems, like SAP and Baan. ERP systems provide proprietary interfaces, so FLUENT founded a common core interface specification easy to fit for any ERP interface. FLUENT ERP interface architecture defines XML schemas and transformations to convert a proprietary ERP data set to the format of another one and vice versa. This architecture is based on standardised XML solutions so exploits the advantages of the emerging XML trend

Key words: Enterprise Network Integration, ERP, XML, EDI, FLUENT

1. INTRODUCTION

FLUENT [FLU] is an international, multi-sectorial attempt to develop methods and tools for managing complex logistic flows, occurring in a distributed manufacturing network with multiple plants and co-operating firms. This kind of organisation has been attracting great interest from the industry community world-wide, under the impulse of:

- *Emerging trends in logistics management*: Virtual/extended enterprise paradigms and Integrated Supply-Chain Models
- *Evolving market conditions*: Decentralisation of facilities, Strategic alliances with suppliers and Complex distribution networks

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35492-7_50

G. L. Kovács et al. (eds.), *Digital Enterprise Challenges*

- *Enabling technologies available on the IT market*:Networking and Workflow solutions, Standards for ERP integration and Supply Chain Planning solutions.

Appealed by this scenario, companies of every size strive to integrate their logistics processes, but a coordinated IT solution is not yet available. So, traditional logistics functions like sales and purchase are left alone to face problems far beyond their intended role.

2. ERP INTEGRATION

The main purpose of the FLUENT ERP interface software is to achieve a seamless integration between the supply logistic chain that will be managed by FLUENT and the ERP product.

Supply chain management does not mean exposing ERP transactions via an Internet interface. FLUENT aim is to implement new functions supporting network-level processes, complementary to the standard functionality of ERP at the local level.

Theoretically, FLUENT allows integration with any ERP system, including legacy systems. ERP products of SAP, Baan/Singular, Diapason, system were considered and an EDI interface were planned to support EDI-linked business.

Hence, the integration software is based on the principle that no local function or ERP transaction shall be replicated in the new system. Hence, only data really necessary to node interactions in the supply network will be exposed at the network level.

This requirement can appear straightforward, but it is very common to find "web-transposed ERP functions" when looking at supply chain solutions on the market. It is thus necessary to mark clearly the difference between network transactions, to be supported by FLUENT, and local transactions supported by the ERP:

- Network transactions are those requiring the co-operation between nodes in order to take decisions, and for FLUENT the only data to be exposed at network level are those supporting these transactions. For example, a network process is the planning and negotiation of a supply flow through a link between two nodes. A data item relevant to this network-level transaction is the price applying for the transfer of product.
- Local transactions are those performed within the scope of a single node, and not requiring interactions at the supply network level. For example, local transactions are those of creation, definition and approval of a price list for a supplier or class of suppliers. The "price" data is obviously the

same as in the previous case, and a proper connection will be needed to update the price in the "Link" FLUENT object on the basis of listings changes. What is not required is to expose the "Listing" document (with all its approval, accounting and other details) at the network level. This would mean translating ERP functions into the new system.

Connections between two systems are realised through asynchronous alignment of data and events:

- Each system (FLUENT or the ERP) imports from/exports to the other system according to its own schedule. No synchronised update or common schedule between the two systems is required.
- Both systems are able to operate in their respective domains even in absence of up-to-date data from the other system. This means, for example, that FLUENT uses last availability data published from the ERP, while "real" availability can have changed in the meantime in the ERP domain.
- From a technological point of view, alignment of data can take place indifferently via LAN, WAN, or any other file transfer mechanism.

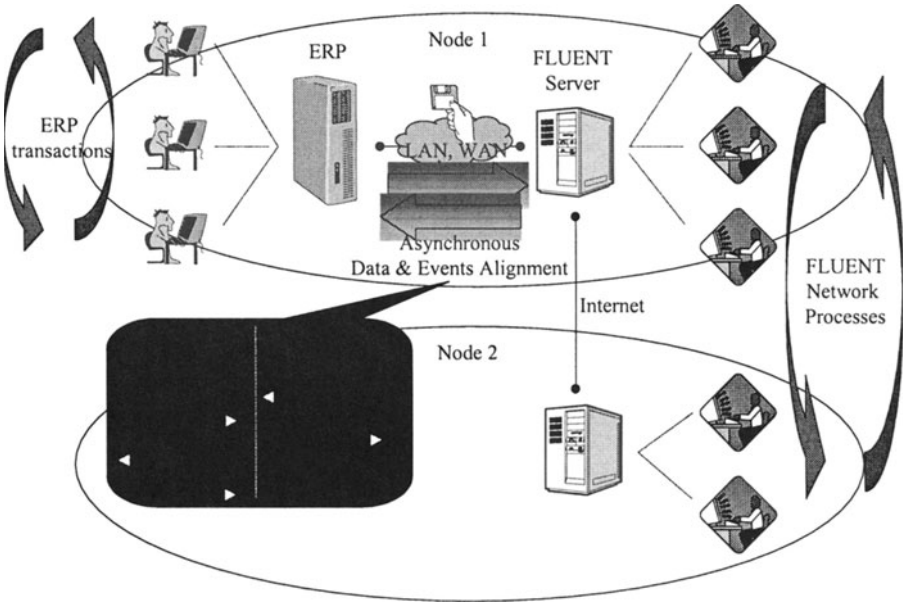

Figure 1. ERP Integration Model

To achieve this seamless integration a long-term strategic approach was chosen satisfying these goals:

- Usage of common available technology.
- Usage of a common language for exchanging data.
- Usage of simple and standard formats for data exchange.

The answer to these requirements is represented by XML.

3. XML IN A NUTSHELL

Here in this paper there is no place to introduce XML [XML], we just hope the reader has already met it. For a newcomer, very briefly, **XML** (*Extensible Markup Language*) is a widely adopted standard, is available on a large number of platforms, is pretty simple and straightforward and, because is based upon exchange of formatted ASCII files, can be easily implemented also if the platform upon which the ERP product is based doesn't support it. All is needed is simply to write, read and transfer well formed text files. In some features:

- XML is a way of adding intelligence to your documents. It lets you identify each element using meaningful tags and it lets you add information ("meta-data") about each element.
- XML is very much a part of the future of Web, and part of the future for all electronic information.
- XML is syntax for marking up data and it works with many other technologies to display and process information. It looks and feels very much like HTML.

In the following we apply some XML companion standards like XML Schema [XSCH] and Extensible Stylesheet Language Transformations [XSLT]. For further reading, see the references.

4. FLUENT ERP-XML INTERFACE

For understanding the interface, let we consider a typical communication scenario. The flow of communication will proceed according to the following steps, shown in Figure 2 (the described scenario is export of data from ERP to FLUENT, but the same considerations apply also to the reverse case):

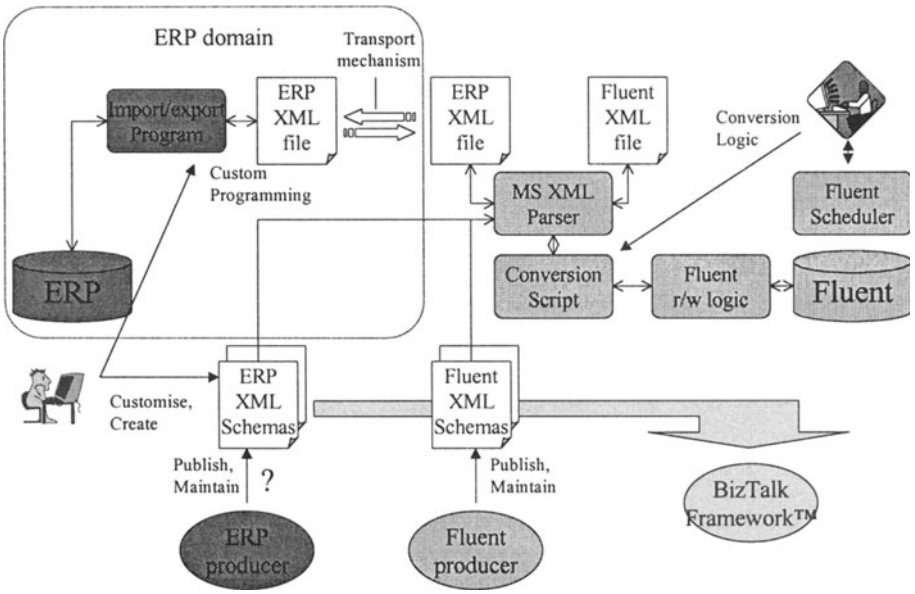

Figure 2. Communication Scenario

1. The ERP system will describe an XML Schema of the data it will make available for FLUENT, each entity that will be transferred to FLUENT will need to have a schema defined. Based upon this schema the ERP system will generate the XML files upon some arranged scheduling (ie: each time a new file is needed, each night, each week), the XML schema will be also made available on the FLUENT machine. This schema will be referred as "Starting schema" in the following.
2. For each entity that FLUENT will need to process a schema will be defined by the FLUENT Consortium. This schema will be referred as the "FLUENT schema" in the rest of the paper.
3. The Starting schema and the FLUENT schema can be the same or can be different, it depends on the ERP system. This choice (having two possible schemas) was made because there is a increasing interest in XML by the ERP producer and so there can be a situation where an existing schema can be already available for the ERP product the customer has available, in this case there is no need to create a new schema but the existing one can be used.
4. Recently Microsoft has been supporting a common framework for these operations named "BizTalk" that is already endorsed by the major ERP producer (SAP, Baan, PeopleSoft and JD Edwards just to name few). FLUENT and is also part of this initiative and so all the schema will be defined using the BizTalk format.

5. The XML files will be transferred on the FLUENT machine using some kind of common transfer method (e.g.: FTP, Samba, CFS or some other kind of file transfer mechanism), at this point the ERP system has finished its work. As far as the ERP is concerned, this approach seems to be the less demanding in terms of adoption of new technologies and changes to the local environment. The EDP personnel has to know:

- How to create and manage an XML Schema.
- How to write XML text files according to this schema.
- How to transfer files to the FLUENT machine.

The last two of the operations above would be needed even if XML schemas and XML files were not involved in the process.

5. AN INTERFACE EXAMINATION METHOD

Our examination on the interfacing issues lays on some obvious considerations. Let us have two systems (A and B) and two transformations (T_{AB} and T_{BA}) to exchange some structured data (x) between them (Figure 3).

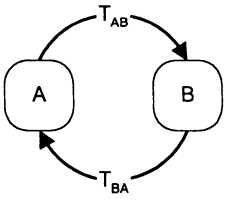

Figure 3. Simple Transformations

Let we define a test transformation:

$$T = T_{BA} \circ T_{AB}, \text{ so}$$

$$T(x) = (T_{BA} \circ T_{AB})(x) = T_{BA}(T_{AB}(x))$$

The test transformation simply transfers data from A to B, and backward. We would like to get back the same data we sent, but our wishes rarely come true. Repeating T again and again on the result of the previous T we can make a sequence of x_i :

$$x_{i+1} = T(x_i)$$

And if T satisfies some criteria it should do, the limit value of the sequence signed χ must exist:

$$\lim_{i \to \infty} x_i = \chi$$

Obviously χ is the fix point of T as

$$T(\chi) = \chi$$

Here comes the sense of the thing as here we get back the same data we sent. Unfortunately T is not supposed to behave so nice in case of arbitrary data. χ is the only one T likes, and χ may be completely useless for us. In order to be able to gauge the usefulness of χ or any other x_i, we should introduce an evaluating function.

The evaluating function E(x) measures the efficiency of the utilisation of the structure, that is E compares the actual data to the ideal amount of data the structure could carry. Function E produces a value between 0 and 1 where 0 means data good-for-nothing and 1 means data the structure is completely utilised by. The construction of the value is hidden and does not matter. Now we may examine the amount of the information χ contains.

In the best case $E(\chi) = 1$. It means T is perfect, consequently T_{AB} and T_{BA} are both perfect, so the interface works fine in both direction and we may use it as it is.

If $E(\chi) = 0$, the interface worth nothing. It does not mean that T_{AB} and T_{BA} are both inaccurate, both of them may be good enough and the interface can work quite fine without the loopback.

In any other case, we may consider whether χ provides data reasonably enough or not. Anyway $E(\chi) < 1$, there must be some interface insufficiency or excessive requirements occur. If χ is good enough, we should compare x_0 to χ and decide to eliminate data loosing substructures, because those needlessly engage resources.

A good approach is to define an X data structure first as necessary and sufficient for our systems, then we design and interface of T_{AB} and T_{BA} where

$$X = \chi$$

6. INTERMEDIATED TRANSFORMATIONS BY FLUENT ERP INTERFACES

Our case study begins with three systems (Figure 4). Let system A the FLUENT one, system C is the ERP one and system B is the intermediate one.

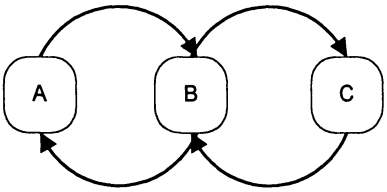

Figure 4. Intermediated Transformations

Why intermediation necessary? In case of n systems without an intermediating one there are (n^2- n) transformations. The intermediating solution requires (2 * n) transformations only. The role of the intermediate system is to provide a common base for any system to join and participate in the communication.

Unfortunately ERP systems do not really support a common way to exchange information between them. In most case an ERP product declares some interface for the world outside its own way. Recently XML became an interface language of the communication and ERP systems are announced to support XML interfaces. It does not mean a common interface with anybody using XML, since the XML Schemas ERP systems use are different ones. XML just makes the interfacing easier by a standardised data format. FLUENT recognised and exploited the advantages of the XML-based interfacing and declared its own XML interface.

In a typical case system S specifies an XML subsystem S' and export/import transformations where $E_{SS'}(\chi) \approx 1$. System A (FLUENT) and C (an ERP) do provide such A' and C' XML spaces and transformations. The only thing to do is to implement transformations between A' and C' (Figure 5).

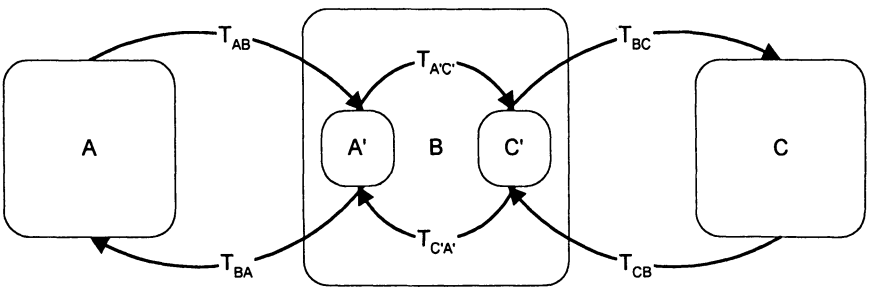

Figure 5. Intermediation Detailed

The solution of Figure 5 should not be called really intermediated as system B is phantom one. System B is a collection of XML Schemas,

transformations and tools supporting the interface between A and C. Furthermore B is unclosed since when a new ERP system (D) joins the network B should be expanded with D' and corresponding transformations to A'.

Although there is no real ERP intermediating specification, FLUENT itself could be seen as a kind of ERP integration solution since ERPs could be connected to each other through FLUENT (Figure 6).

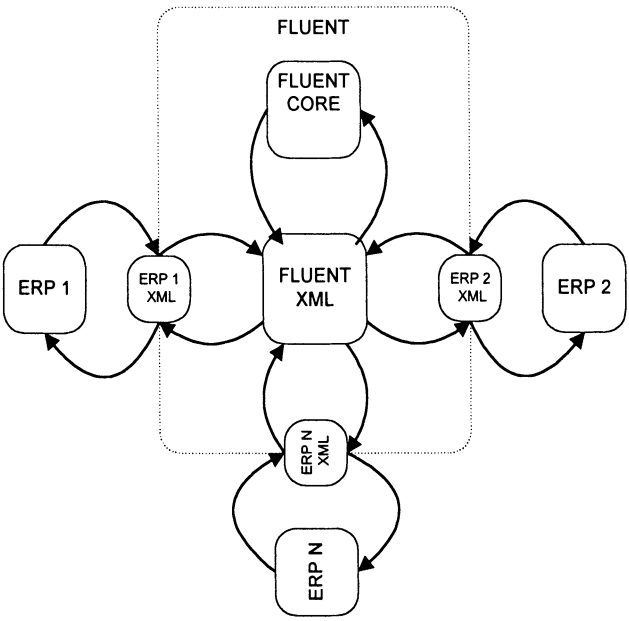

Figure 6. Intermediation by FLUENT XML

Our considerations on interface examination can be extended for the case of using FLUENT to integrate different ERP systems. As any combination of transformation routes may occur, we should test chains of transformations. According to a well-known saying a chain is as strong as the weakest link does. Consequently the chain can be tested link by link. A link test produces χ and $E(\chi)$ values for a link. We also can produce a value for the chain as the minimal value of the links:

$$\text{Min}\,(E_i(\chi_i))$$

This way we can simply point out that one transformation loosing data might infect the whole system. If there is a chain going through each ERP system and through FLUENT between ERPs, the chain uses each of the transformations at least once. Obviously, if a transformation loose data somewhere, there is no way to reproduce it, so the following links get less information.

7. FLUENT SPECIFIC INTERFACING PROBLEMS

Although FLUENT does not specify requirements for the ERP integration, the nature of the integration assumes some features the ERP should satisfy.

First we consider the main problematic point of the integration. As FLUENT itself works with Business Objects (BO), the work around is BO centric. Each import operation results one or more BOs and each export operation is fed from BOs. FLUENT objects represent Product, Aggregate BOM, Available Inventory, Planned Inventory, Independent Demand Order, Materials Requirement, Shipment, Receipt, Supply Item, Supply, Forecast and Unit Of Measure descriptions.

The FLUENT ERP Interface software realises a given number of *Connections*. Each connection represents an automated process for importing, exporting or aligning data between the ERP and FLUENT Business Objects.

Import connections update FLUENT objects based on ERP data; *Export* connections update ERP tables based on FLUENT objects; *Align* connections operate in both directions, keeping data aligned with respect to latest changes in both systems.

In the case of an import operation, we should create FLUENT Business Objects from the ERP side. The import procedure should solve the following problems:

- Semantically mapping ERP information into BO attributes can be difficult or even impossible and require ERP queries.
- One result may be calculated from other ones, so a dependency tree grows up.
- Some BO attributes reference to other ones (in FLUENT syntax of course), so the procedure should check the existence and coherence of the references at FLUENT side.

FLUENT having recognised the problems above, classified BO attributes as:

- *Mandatory*: The field value is required by FLUENT in order to define a proper instance of the object (defining attribute).
- *Default available*: If the ERP has it, the field value is useful on the FLUENT side. Event if the field is not exported by the ERP, new instances can be defined since FLUENT can use a default value.
- *Optional*: The field is optional on the FLUENT side, and it will be instantiated only if the ERP has it.

Unfortunately even if we consider the mandatory attributes only (limiting quite much the usability of the interface), we could not ward off the great mass of the complications.

Export operations face similar difficulties, and in the align case, conflict may arise if a data element is updated in both systems. In such cases, as a general rule, the winner will be the latest update.

8. FLUENT - EDI INTERFACING PROBLEMS

Special cases of ERP integration when we are to build interface for another interface like **EDI**. EDI (Electronic Data Interchange) is a computer-to-computer transmission of (business) data in a standard format. Parties using EDI exchange messages. EDI allows specifying specialised message types (so called subsets) over the general syntax and semantics. Companies specified countless subsets for their own needs as ordering or offering products, notifying price changes and so one. In many cases, EDI is used to exchange ERP like information, so at this point we may introduce FLUENT as a solution of ERP integration.

When we build FLUENT Business Objects from EDI messages, we face some deeper difficulties as if we would work upon an ERP interface:

- There is no standardised subset each company use, so we should produce an interface for each subset.
- We just get messages often reflecting completely different approaches of the FLUENT use.
- There is no way to query the ERP for filling up a BO.
- When we export a FLUENT BO, we should build up messages filling up with EDI and subset specific data fields FLUENT does not provide.

Considering the problems above, we suggest specifying FLUENT compatible EDI subsets. Unfortunately there is no guarantee anybody accepts new subsets.

9. CONCLUSION

FLUENT provides software tools and techniques for the integration of distributed logistic and business flows.

The integration is based on standardised XML documents and transformations.

The integration can be difficult in case of special ERP systems. Although we may solve the integration is some way, we lose data during the interfacing procedures.

More problems arise when we integrate FLUENT with EDI.

10. REFERENCES

Table 1. References Table

Ref.	See Here
FLU	http://fluent.gformula.com
XML	http://www.w3.org/XML
XSCH	http://www.w3.org/XML/Schema
XSLT	http://www.w3.org/TR/xslt