An Extensible Reference Model for Data Integration

G.N. Benadjaoud and B.T. David
Ecole Centrale de Lyon, Laboratoire GRACIMP, Dépt. MIS
B.P. 163, 69131 ECULLY CEDEX, FRANCE, Tél : (33) 72.18.64.42
Fax : (33) 78.33.16.15, email : benadjao@cc.ec-lyon.fr, david@cc.ec-lyon.fr

Abstract
The DEE (Data Exchange Environment) is a platform for the integration of isolated applications. It co-ordinates and controls the data exchanges between the different applications. The DEE uses an Extensible Reference Model (ERM) that gives a global and unifying view of the manipulated data. The MO-MER Model is an object oriented model, facilitating the extendibility of the ERM, because its Meta-level allows the addition and the modification of the manipulated concepts. The extendibility of the ERM is ensured by schema evolution methods that go with the MO-MER model. These methods allow the update of the ERM during its life cycle.

Keywords
CIM (Computer Integrated Manufacturing), integration, databases, data exchanges, data modelling, Object Oriented Model.

1 INTRODUCTION

Data-processing in manufacturing systems is used as an assistance tool for the elaboration of the products along the whole production process. Among these tools we will quote : Computer-Aided Design (CAD), Computer Aided Process Planning (CAPP), Computer-Aided Manufacturing (CAM) and Computer Aided Quality (QAO) systems. Due to the fact that these application software packages could be conceived and developed independently, they often remain disparate and isolated.

The data integration of isolated applications usualy employed is to let applications share and have access to the common data. Two approaches mainly used are either based on the application interfacing and the use of standard format or based on the federated databases.

Interfacing applications is to create a data translator between two applications that have to exchange data. To ease this interfacing and to reduce the number of translators, standard formats have been defined, like IGES (Initial Graphics Exchange Specification), SET (Standard d'Echange et de Transfert) (Wix. 1986, Scholz-Reiter. 1992) and STEP (Scholz-Reiter. 1992, Brun. 1992). They harmonise the data expression between applications. To exchange data using standard formats, the data are translated from the source application format to the standard format and from the standard format to the target application format. So, for each application two translators are needed : one for the data transfer from the application to the standard format, another one for the data transfer from the standard format to the application.

2 OUR APPROACH

We fixed as an objective the study of possibilities of integration of the isolated applications. Our aims were :

- to ensure a global data perception by conceiving a data referential. Its purpose is to express semantic links that exist between the manipulated information. It serves as an informational integration support. It is a general data model able to be enriched by time. This is why we call it Extensible Reference Model (ERM).
- to allow a reliable flow of all kinds of data between applications that can be geographically distanced and to use different data managers.
- to allow also a progressive data migration from applications to the ERM.
- to preserve existent applications that are valuable acquisitions of the companies.

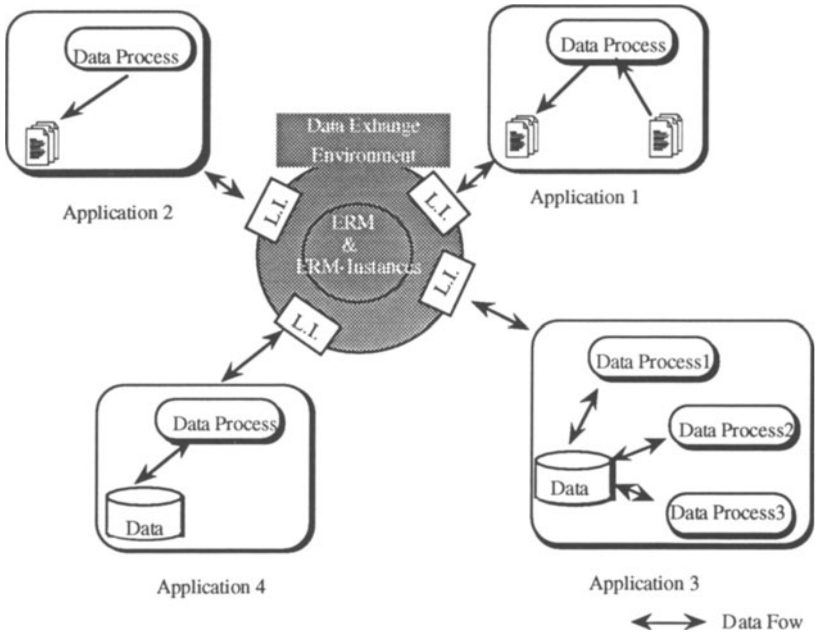


Figure 1 The Data Exchange Environment

The solution that we present is an environment for data exchanges called DEE (Data Exchange Environment). It is a platform for the integration of isolated applications that allows a co-ordination and a control of data exchanges between the different applications. The DEE has an Extensible Reference Model (ERM) that allows it to have a global and unifying view of the data manipulated by connected applications and also allows us to solve the modelling and the perception conflicts.

To preserve the existent, the connection of applications to the DEE is a weak coupling ; it is done by using local interfaces (L.I.) (Benadjaoud 1996). These allow applications to communicate with the DEE and they allow the DEE to overcome problems of differences of DBMS (at the data model level and data manipulation language level). They have the following functions:

- the data transfer from the application to the DEE and vice versa,
- the elimination of data representation and data access differences that may exist between applications and the DEE,
- the encapsulation of the application data access by harmonising accesses to data. Thus they ensure the independence of the DEE towards the connected applications.

In this paper we will focus our presentation on the ERM and the MO-MER model. The details on the architecture of the DEE can be found in (Benadjaoud. 1996).

3 THE ERM

The ERM is a unifying schema of information processed by the different applications. It eliminates conflicts of perception that may exist in the real entity as represented by different applications.

To allow the ERM to represent complex entities, to support different viewpoints of a same entity, to express and to maintain the coherence of static and dynamics data, we have developed an object oriented model, called MO-MER that serves as a canonical representation model of data. Thus, applications manipulating complex entities, such as CAD systems and CAM systems, can be connected to the DEE as well as other future applications that will use object oriented DBMS.

The ERM is a reference for all applications connected to the DEE. It offers a global view on shared objects and relationships between them. It is also a reference for the DEE so that the latter could undertake its data distribution functions.

Adding to the global schema of shared data that we call shared schema, the ERM contains for each application connected to the DEE an export schema that represents the data that the application is ready to share with others. These export schemas define the contribution of applications to the federation. Correspondences between export schemas and the shared schema link each entity in the shared schema to its correspondent in export schema. To each correspondence are associated one or several predicates that define conditions to be satisfied to make this correspondence valid.

The design of a ERM is made in an incremental manner ; we create from existent applications (and specific models) a part of the ERM that will then be enriched as a new application will be connected to the DEE.

The ERM offers a set of access functions allowing the DEE to obtain information concerning the shared schema and correspondence between entities. These functions allow, for example, to know the attributes of an object, the correspondence between two objects, the type of an attribute, etc..

4 THE ERM-INSTANCES

The ERM-Instance allows us to gather a determined set of shared data. Data contained in the ERM-Instance can represent:

- data of the new applications connected to the DEE,
- data of applications that are not well organised (the case of applications using files), and/or
- the most shared data.

In the first case, the new applications are the ones developed specifically for the company that may use the ERM- Instance as a data manager. It means that the new applications non-specific to the company, that possess their own data manger, will continue to use their data managers and will be connected to the DEE through local interfaces.

In the second case, ERM-Instance allows applications to put their data together, a semantics modelling of their data with the expression of links between data. Thus, the storage of data of the same project in a unique storage system, for example, is made possible. This will facilitate, thereafter, the re-use of data, but a re-engineering of data is necessary ; create the conceptual schema from the inputs and the outputs of each application. The transfer of this data project from the application to the ERM-Instance is made, gradually, by using the DEE functionnalities.

The DEE, in this case, can be seen as a tool for a progressive migration of non structured data to a central database (ERM-Instance) by undertaking transfers of data from applications to the ERM-Instance.

In the third case the ERM-Instance groups objects that are the most required by applications. These objects constitute a database that groups global characteristics of used objects . It allows:

- the limitting of the data redundancy and the data incoherence,
- the saving of time for the users in the preparation and the transfer of data.

5 THE MO-MER MODEL

The MO-MER Model is an object oriented model, facilitating the extendibility of the ERM, because its Meta-level allows the addition and the modification of the manipulated concepts. The extendibility of the ERM is ensured by schema evolution methods that go with the MO-MER model. These methods allow the update of the ERM during its life cycle.

To take into account the future DBMS that could be used by new applications to connect to the DEE, we have made ensured that the basic concepts manipulated in MO-MER converge to those described in Object Database Standard developed by ODMG (Atrood. 1993).

5.1 Presentation of MO-MER

```
Class WoodScrew
   properties of the class
         SuperClass: Screws;
         keys: SerialNumber;
         Constraint: Diameter /Length < 1;
   properties of instances
         SerialNumber: Integer;
         Head: (Round, Flat);
         Diameter: Real;
         Length: Real;
         Matter: String;
   operations on instances
   Price (): - > Real /* returns the Price of live it that is calculated in function of
                        the diameter, the used matter and the length) */
```

Figure 2 The structure of a class

The MO-MER object model is presented as follows:

- The basic concept of the modelling is the object. Objects are grouped in classes. All objects of a same class have the same behaviour and the same characteristics.
- The behaviour of an object is defined by a set of methods described in the class.
- The state of an object is defined by values of its characteristics. These characteristics can be attributes of the object or relationships with other objects.

The class is the conceptual entity that describes the object, it groups a set of objects (instances of the class) that share common characteristics (described by attributes) and have the same behaviour (described by operations). The example in figure 2 is a description of the class WoodScrew. The class WoodScrew represents the set of objects having a Diameter field that is an integer, a Head field, a Length field and a Matter field.

The relationship between an object obj and its class C is a "IS_A" relationship, and obj is an instance of C. For example the screw whose value is the tuple (1234, Round, 2,5, 10, steel) is an instance of the class WoodScrew.

From a class, it is possible to define other classes (sub-classes) more specific (specialised) completing characteristics of their mother class using the inheritance mechanism offered by the MO-MER model. The class WoodScrew, for example, as defined previously, is a sub-class of the class Screw. It inherits the attributes Diameter and Length.

A specialisation can be multiple due to the multiple inheritance that allows a class to have several direct super-classes, and to inherit the union of attributes and methods of its super-classes.

The MO-MER offers constructors such as List, Bag and Set that allow an attribute not only to have mono-valued values but also multi-valued values.

- The Bag is a collection of objects that authorises the duplication of objects.
- The Set is a collection of objects that does not authorise the duplication of objects.
- The List is a collection of objects that authorises the duplication of objects and institutes an order relationship between objects.

Two classes are related if the domain values of an attribute of one class is the set of objects of the other class. For example, to express the relationship MilledBy that connects a WoodScrew to its milling machine, we add to the class WoodScrew an attribute called MilledBy whose domain values is the class milling machine. To express the same relationship in the opposite way, we add to the class Milling Machine the attribute Mills whose domain values is the class WoodScrew. The constructor Set, in this case, has allowed us to express that n WoodScrews are connected to a Milling Machine. Thus relationships [1,1], [1, n] and [n, n] can be expressed by the constructors Bag, Set and List.

Two objects are related if one of them is a value of an attribute of the other. For example, to express the relationship MilledBy that connects a wood screw to a milling machine, we set the attribute MilledBy by the identifier of the object milling machine, by which the screw has been manufactured.
(obj4, (SerialNumber: 021425, Head: Round, Diameter: 03, Length: 8.5, MilledBy: obj8))
(obj8, (ToolNumber: 14, MillingTime 14.03)).

The MO-MER distinguishes two types of relationship : exclusive relationships and non exclusive relationships.

```
Class WoodScrew
properties of the class
        SuperClass: Screws;
        key: SerialNumber;
        constraint: Diameter / Length < 1;
properties of instances
        SerialNumber: Integer;
        Head: (Round, Flat);
        Diameter: Real;
        MilledBy: MillingMachine;
        Length: Real;
        Matter: String;
operations on instances
        Price (): - > Real;
```

```
Class MillingMachine
properties of the class
        SuperClass: Tool;
        key: ToolNumber;
        properties of instances
        ToolNumber: Integer;
        Mills: Set <WoodScrew>;
        MillingTime: Real;
operations on instances
        NumberScrewByDay (): - > Integer;
```

Figure 3 A relationship between two classes

Exclusive relationships: a relationship between two objects is exclusive if the two objects cannot be connected by the same relationship to other objects. This type of relationship allows us to model the relationship composed-component that connects a complex object to its components.

Non exclusive relationships : a relationship between two objects is non-exclusive if the two objects can be connected by the same relationship to other objects.
To illustrate these two types of relationship, let us take the following three classes :

```
Class Car ....                              Class Engine
        EquippedBy: Exclusive engine;       .....
        ManufacturingSupervisor: Person;    Class Person
                                            .....
```

The relation EquippedBy that connects a car to the engine with which its equipped, is an exclusive relationship, because an instance of the class Engine M0 can be connected only to one instance Car V0. An other instance Car V1 can not have the instance M0 as a value of EquippedBy.

The relationship ManufacturingSupervisor that connects a car to the person who has supervised its manufacture, is a non-exclusive relationship, because a same person, Jean Claude, can supervise the Car V0 and the Car V1.

A conceptual schema in the MO-MER Model is expressed by:

- the specification of the set of classes that compose it,
- a lattice of specialisation whose arcs correspond to the relation super-class / sub-class,
- a graph of relationship whose arcs correspond to relation reference to.

The example illustrated by the figure 4 represents the specialisation lattice of the class screwers whose sub-classes are the class Screw and the class Nail. The root of this lattice is the class Object.

Relationships [1, 1] between two objects are represented by a simple arrow and relationships [1, N] are represented by double arrows. Relationships [N, M] are represented by double arrows on each extremity of the line. The example illustrated by the figure 5 is a graphical representation of the example illustrated by the figure 3 where a milling machine is in relation with several Screws.

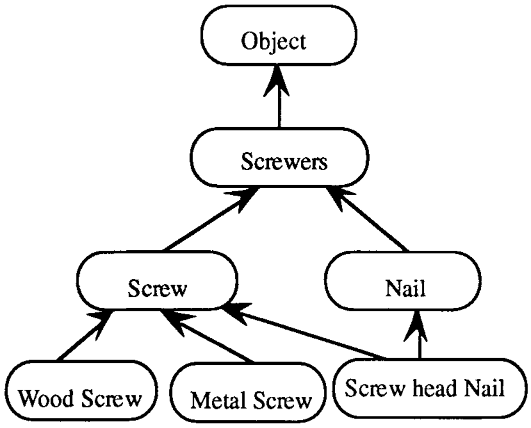


Figure 4 A lattice of inheritance

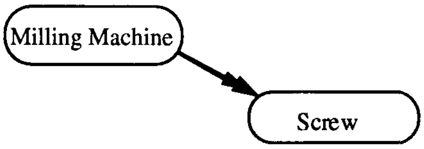


Figure 5 A graph of relationships

5.2 The meta-circularity of MO-MER

The model is based on a uniform architecture of three levels: instance/class /meta-class, where all is object and the meta-classes are real classes in the sense that they can have instances and sub-classes. This architecture facilitates, on the one hand, the evolution of the ERM which is important for the openness of the DEE to new applications, and on the other hand the extendibility of the basic classes used in the ERM in the case where new concepts (new semantic relationships between objects, new types, etc.) are used by new applications.

The MO-MER is a reflexive model. It allows the ERM to manage information concerning itself, and this offers several advantages. The first one is what we call the uniformity of representation; the primitives of the model are used to manage all information including the meta-information. The second is access and manipulation uniformity; the extraction of data that is managed by the same access primitives of the model whatever the type of the information or its state.

5.3 Correspondence

The ERM is constituted of a shared schema representing all objects manipulated by applications and a set of export schemas. Each export schema represents manipulated and exported data of an application.

A correspondence links a class of an export schema to its equivalent in the shared schema. For a given class in the shared schema, the correspondence enumerates attributes of this class that can be found in a given export schema. It plays the role of attribute filter.

To avoid the redundancy of representation of the same class in the shared schema and in the export schemas, classes of export schemas are not represented as such. An export schema is represented by a set of correspondences. Each of these correspondences is characterised by the class of the shared schema to which it is linked (associated class), a list of attributes of this class exported by the export schema and the predicate that has to be verified to validate this correspondence.

5.4 An example of a modelling in MO-MER

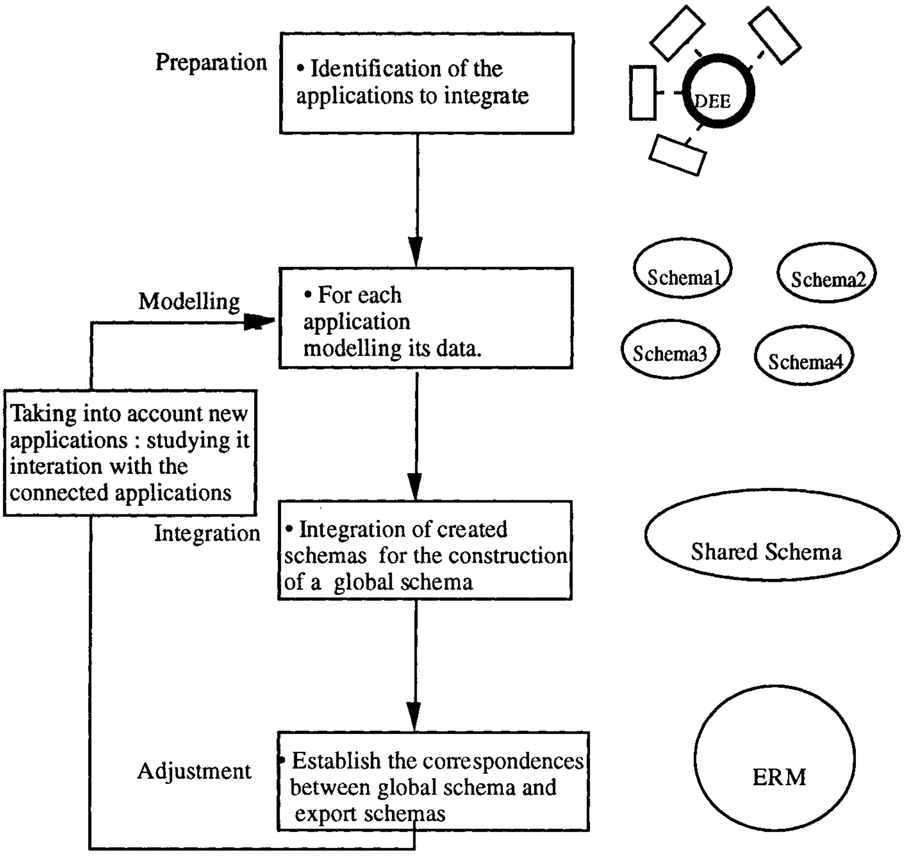
Figure 6 illustrates an example of an ERM. The instruction Create ERM announces the creation of a ERM whose name is given. The following described classes constitute the shared schema of the ERM. The instruction Create export schema announces the creation of an export schema whose name is given. The set of the followed described correspondences constitute the export schema .

```
/* ERM for car design */
Create ERM CarDesign ;
Class Car
{Class Properties
        Superclass : Object;
        Key : No_Ref;
        Contraint : No_Ref>99999 et No_Ref<=999999
 Instance Proprieties
        No_Ref : Integer;
        EngineCharacteristics  : Engine;
        BodyCharacteristics: CarBody;
        OptionCaractéristics  : Option ; }

Class Engine
{ Class Properties
        Superclass : Object;
        Key : No_RefEngine
 Instance Proprieties
        No_RefEngine: Integer;
        Engine_Type : String;
        Power : Real;
        Acceleration: Real; }

Class CarBody
{ Class Properties
        Superclass : Object;
        Key : No_RefBody;
 Instance Proprieties
        No_RefBody : Integer;
        Body_Type : String; }

Create exporte Schema SchemaBody;

Correspondence C1
{ AssociatedClass : Car;
ListeOfAttribut : No_Ref,BodyCharacteristics, Option;Characteristics
 AssociatedPedicat : C1.No_Ref=Car.No_Ref }

Correspondence C2
{ AssociatedClass : CarBody;
ListeOfAttribut :No_RefBody;, Body_Type;
AssociatedPedicat : C2.No_RefBody = CarBody.No_RefBody ; }

Create exporte Schema SchemaEngine;

Correspondance C3
{ AssociatedClass : Car;
ListeOfAttribut :No_Ref , EngineCharacteristics
AssociatedPedicat : C3.No_Ref  = Car.No_Ref+5401 ; }

Correspondance C4
{ AssociatedClass : Engine;
ListeOfAttribut : No_RefEngine, Engine_Type, Power ;
 AssociatedPedicat : C4. No_RefEngine= Engine.No_RefEngine; }
```

Figure 6 An ERM specification using MO-MER

6 A METHOD FOR THE ERM DESIGN

The ERM design method is divided into three processes : preparation, modelling and adjustment.



Preparation
• Identification of the applications to integrate
DEE
Modelling
• For each application modelling its data.
Schema1
Schema2
Schema3
Schema4
Taking into account new applications : studying it interation with the connected applications
Integration
• Integration of created schemas for the construction of a global schema
Shared Schema
Adjustment
• Establish the correspondences between global schema and export schemas
ERM

Figure 7 An incremental method for the ERM design

6.1 The Preparation

The preparation is the study of data exchanges between the different applications. It is a step of skimming through the manufacturing system applications in order to detect:

- the functional interaction between the applications,
- the needs in data exchanges,
- the shared data,
- operated data exchanges.

The different tasks to perform are:

- to establish a sketch of the data schema of each application. This sketch has to be sufficiently detailed in such a way that the data manipulated by the application could be noticed.

- to study the intersection of applications data in order to detect shared data.
- to study the manually or automatically undertaken data exchanges.

6.2 The modelling

The modelling process of an application data in this method is divided into three steps:

- the definition of the classes: the number of classes used in a manufacturing system is generally very high. It is necessary to reduce this set of classes by referring to the area in which the application data evolve. It is also necessary to define for each class the attributes that characterise it by consulting the data manipulated by the application.

- the definition of the hierarchy of classes : it is to define the inheritance lattice of classes, i.e. to define the relationship classes/sub-classes. This relationship is identified by using the generalisation or the specialisation technique. The generalisation technique consists in considering each defined class as a sub-class and seeking the potential super-classes that verify the inheritance. The specialisation technique consists in considering each defined class as a super-class and seeking the potential sub-classes that verify the inheritance.

- the definition of the relationships between classes: the relationships between objects can be deduced only if the real world modelled is well understood, because data as given in files or on the classical database do not carry enough information to able the reconstruction of relationships.

6.3 The integration of data schemas of applications

The shared schema is obtained by integrating the applications data schemas. The integration of schemas is a technique used in database to construct a global conceptual schema from the merging of the different data schemas. The objective of the integration is to detect conflicts between the different schemas and to construct a conceptual schema without any redundancy. This objective is generally reached by defining a strategy of schema integration, by constructing the associate global schema and by verifying the validity of the obtained schema .

Many studies have been carried out to solve the problem of schema integration. Since 1977, many integration methods and many comparative studies have been published (Batini. 1986, Geller. 1992, Hayne. 1990, Lim. 1994, Spaccapietra. 1994, Suzuki. 1995). Each integration method has its own steps. Globally, these steps can be considered as a combination of the following steps:

- pre-integration: an analysis of schemas is undertaken to choose the strategy of integration to be adopted. Two types of integration strategies exist: binary or n-ry ones. Binary strategies consider the integration of schemas two by two, n-ry strategies consider the integration of any number of schemas simultaneously.

- comparison of schemas: schemas are compared to determine relationships between concepts and to detect conflicts that may exist.

- conflict resolution: once conflicts are detected, this phase creates compatible and therefore "integrable" schemas. Each conflict detected during the phase of comparison is eliminated by applying necessary transformations on schemas.

- merging of schemas: this phase constructs the global schema. It consists of a simple superposition of schemas if all conflicts have been resolved.

- the result verification : it allows us to test the quality of the global schema according to the completeness, the correction, the comprehension and the "minimality" criteria. The completeness guarantees that during the integration no information has been lost. The correction verifies that structures created by transformations in the global schema are valid according to the data rules of the model. The comprehension guarantees that defined structures are sufficiently explicit to be understood by users. Finally, the "minimality" guarantees that the global schema is exempt from redundancy.

6.4 The adjustment

In this step we complete the design of the ERM:

- by defining and by establishing correspondence between export schemas and the shared schema of an ERM,
- by detailing links between representations of same data in the different applications and in the shared schema .

This last point can be made by drawing up a table whose rows represent the exchanged data and columns indicate its representation in each application that hold it and in the shared schema. The figure 9 illustrates an example of this table.

Data	Repr. in the s/ERM		Representation in A			Representation in B		
	Name	Type	Name	Type	Conversion fct	Name	Type	Conversion fct
Temperature	T	Real	Temp	Real	Temp=T*1.8+32	Tps	Real	____
Volume	Vol	Real	VG	Real	____	VoluGaz	Real	____
Pression	Press	Real	Pression	Real	____	Pr	Integer	Pr = int(pressi)

Figure 9 Data exchanged between the application A and the application B

At the end, we allocate to the modelled ERM to DEE and we parameter this last by:
- defining the initial data flow between the different applications,
- establishing the authorised user list and access rights of each.

7 CONCLUSION

DEE makes the applications in the manufacturing systems integrated applications with preserving them. This is a great advantage cause the existing applications are valuable acquisitions of the companies. The second advantage of the DEE is its openness in integrating new application, cause the applications are connected to the DEE through local interfaces that encpasulate the access to the applications data.

The MO-MER model has two advantages. The first one its similarity with the ODMG model that is considered as the future object oriented model standard. This will facilitate the integration of new applications using object oriented data models. The second advantage is the reflection of the MO-MER that has facilitated the communication between the DEE and the ERM. This one offers information on itself and on the data it manages (ERM-Instances).

The design of the ERM can take a long time. The ERM design method presented here has been oriented in a way to allow a progressive design of the ERM.

8 REFERENCES

Ahmed, R.; Smedt, P.D.; Du, W. ; Kent, W. ; Ketbachi, M.; Litwin, W. A.; Rafii, A. and Shan, M.C. (1991) The pegasus heterogeneous multidatbase system, IEEE Computer, Vol. 24, No 12, December 1991, p.17-p.27.

Andersson, M.; Dupont, Y.; Spaccapietra, S.; Yetongnon, K.; Tresh, M. and Ye, H. (1993), The FEMUS Approach in Building a Federated Multilingual Database System, 3rd International Workshop on Research Issues on Data Engigneering : Interoperability in Multidatabase Systems (RIDE-IMS'93), Vienna, Austria. April 19-20, 1993.

Atrood, T ; Duhl, J ; Ferran, G; Loomis, M ; andWade, D (1993) The Object Database Satandard: ODMG-93, Ed. R. G. G. Catell, 1993.

BATINI, C ; LENZERINI, M ; NAVATHE, S.B. (1986) A Comparative Analysis of Methodologies for Database Schema Integration, ACM Computer Surveys. Vol 18, No 8, 1986.

Benadjaoud, G.N. ; David, B.T. ; (1996) Data Integration : A Federated Approach with Data Exchanges, 2nd IEEE/ECLA/IFIP International Conference on Architectures and Design Methods for Balanced Automation Systems, Lisbon, Portugal, 17-19 June1996, ed. Chapman and Hall.

Brun, J.M (1992) IPDES final report : Common Data Model, Esprit Project 2590, june 1992.

Chung, C. W.(1990), DATAPLEX : An Access to Heterogeneous Distributed Database, Communication of the ACM, Vol. 33, No. 1, January 1990.

Clement, D.; Ganesh, M.; Hwang, S. Y.; Lim, E. P.; Mediratta, K.; Srivastava, J.; Stenoien, J. and Yang, H.R. (1994), Myriad : Design and Implementation of Federated Database Prototype, Proc of 10th IEEE Data Eng. Conf., 1994.

Geller, J. ; Perri, Y. ; Neuhold, E. andSheth, A. (1992) Structural Schema Integration with Full and Partial Correspodence using the Dual Model, Information Systems Vol. 17, No. 6, pp.443-464, November 1992.

Hayne, S. ; RAM, S. (1990) Multi-User View Integration System (MUVIS) : An Expert System for View Integration, Proceeding of Int. Data Engineering, Los Angeles, Californie, Février 1990.

Lim, E. ; Srivastava, J. ; Shekhar, S.(1994) Resolving Attribute Incompatibility in Database Integration : An Evidential Reasonning Approach, Proc. of 10th IEEE Data Eng. Conf. 1994.

Scholz-Reiter, B. (1992), CIM Interfaces, Concepts, standards and problems of interfaces in Computer-Integrated Manufacturing, ed. Chapman and Hall 1992.

Sheth, A.P. and Larson, J.A. (1990), Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol. 3, No. 22, p. September 1990.

Spaccapietra, S ; Parent, C. (1994) View integration : a step forward in solving strucutral conflits.", in IEEE Transaction on Data and Knwledge Engineering.

Suzuki, G. ; and Yamamuro, M. (1995) Schema Integration Methodology Including Structural Conflict Resolution and Checking Conceptual Similarity -Conceptual Graphs Approach, Proccedings of 6th Int. Hong Kong Database Woorkshop, Hong Kong, March 1995.

Wix, J. and McLelland, C. (1986), Data Exchange between Computer Systems in the Construction Industry, BSRIA 1986.