International Journal of Soft Computing 2 (2): 225-235, 2007


# Data Integration Method Based on Mediator Approach and Using UML and XML Schema

[1]Younes Djaghloul, [1]Zizette Boufaïda, [2]Veronique Heiwy and [2]Charles-François Ducateau
[1]Laboratoire Lire, Université Mentouri, Constantine, Algeria
[2]CRIP5, Université René Descartes, Paris, France

**Abstract:** In this study, we propose a coherent method to solve the problem of data integration of heterogeneous data sources. The method uses UML extensions in order to facilitate the development of specific mediators. The latter, permit the conversion of queries and data according to semantic correspondence tree and the modification on databases. Our approach uses the linguistic mapping techniques with the XML language to ensure the conversion of data and queries. The mediators are implanted using distributed object and web service technology. The method is validated by using it to develop an E-Commerce solution.

**Key words:** Mediators, UML, data integration, XML schema, distributed objects

## INTRODUCTION

The data integration consists of providing an unified view of data that can be stored in different databases and with different models. This problem has been treated under different aspects. Garcia-Molina *et al.* (2002) discusses some of important characteristics of the data integration, such as: the difference in the type, in the name and the semantic.

Heterogeneity can be classified into two main classes. The structural ones mean that data is stored in different structures. The semantic heterogeneity considers the meaning of the contents in different sources.

The study done on this topic (Batini *et al.*, 1986; Sheth and Gala, 1989; Hyne and Ram, 1990; Spaceapietra and Parent, 1991; Tukwila,) is based on the principle of the use of a common model named canonical model. The role of this later is to represent the various visions of the users and to transform local schema into a canonical schema.

Otherwise, mediators are used in this context to achieve data integration without incorporating data sources. The latter remaining unchanged. The *Momis* project (Bergamaschi *et al.*, 2002) is an interesting example of integration approach made by mediators.

According to this analyzis, we can see that there is no methodological framework to answer to the problem of data integration. Our contribution aims at giving to the developers' tools for solving this problem. Concretely, it consists of an integration method based on the OMT process and on the UML diagrams, offering solutions to the integration problems resulting from several interoperability levels (diversity of platforms, structural differences, integration problems.). Using the UML standard for representing diagrams is fruitful as it increases the communication power between developer teams.

The integration approach supported by the proposed method is based on the mediation. In this context, conversion has a prime role. Each schema of the database is translated in order to have a unique global model considered in the method as an XML document with a special structure. Thus, we find the notion of semantic canonical model. The mediators will transform each global query on the canonical model into sub-queries that can be understood by the local systems. Then, the returned results are converted back to the global model. The mediators then analyze the results, combining values returned with the knowledge from the application domain. This combination gives a result closer to the reality and to the semantic of the results. The mediators created by the method are based on the distributed object and Web services technologies. Actually, The web services are considered as the best choice to implement application in the WEB. In this research, Web services are used as "window" for the distributed object, so it improves the interoperability of the whole system.

We distinguish the global mediators (query manager), which the role is to intercept the global query of the clients, to transmit them to the local mediators and to retrieve the results transmitted to them. These results are expressed in XML and use the same schema. The mediator merges the documents and returns the global document as the final result.

**Corresponding Author:** Younes Djaghloul, Laboratoire Lire, Université Mentouri, Constantine, Algeria

The local mediators are of two kinds: the formers named *DB2XML* have in charge the "Selection" queries. They have in charge the mapping of a global query into a local query using the "semantic correspondences tree", then to transform the data into an XML document, using a conversion algorithm and finally to convert this document into another one, using the global attributes. The second named *DBModifier* have in charge the modification operations on the database (add, delete, modify). They convert the global query into locals ones using the attributes correspondences trees.

In the following, we start by presenting a state of the art in this domain. The third section describes the main steps of the method proposed. In order to show the efficiency of our proposal, this method is then applied to an on-line computer material sale distributed on three sites. Each site has its own "proprietary" solution to manage the sales. The goal to be achieved is the integration of the data of the three sites according to the proposed approach.

## STATE OF THE ART

Numerous research activities have been conducted on data integration. This section presents some work related to this area.

Tow main approaches are used in the area of data integration. The structural one and the semantic one.

**Structural integration:** In this approach, structural heterogeneity means that different information systems store theirs data in different structures. The mapping between the elements is immediate and it is done according the type and the name of each element.

**Semantic integration:** This approach considers the contents of the information and its intended meaning. The main role of any approach of data integration is to define a semantic mapping between the different sources.

**Schema matching:** In (Rahm and Bernstein, 2001), one presents a good survey about the automatic schema matching. one can classify the mapping approach in these categories in the following:

**Instance vs. schema:** Mapping approaches can consider instance data (i.e., data contents) or only schema-level information.

**Element vs. structure:** Mapping can be performed for individual schema elements, such as attributes, or for combinations of elements, such as complex schema structures.

**Language vs. constraint:** The mapping can use a linguistic-based approach (e.g., based on names and textual descriptions of schema elements) or constraint-based approach (e.g., based on keys and relationships).

**Matching cardinality:** Each element of the resulting mapping may match one or more elements of one schema to one or more elements of the other, yielding four cases: 1:1, 1:n, n:1, n:m. In addition, there may be different match cardinalities at the instance level.

**Auxiliary information:** The mapping do not only rely on the input schemas , but also on auxiliary information, such as dictionaries, global schemas, previous matching decisions and user input.

In order to create the appropriate mapping between the different schemas, the linguistic approaches category is an interesting solution.

**Linguistic approaches:** In this category, one uses the name and the text to propose the semantic similarity between elements.

Two main approaches belong to this category, Name matching and Description Matching.

**Name matching:** Name-based matching matches schema elements with equal or similar names. Similarity of names can be defined and measured in various ways, including:

**Equality of names:** One supposes that names intends the same semantics.

- Equality of canonical name ,(e.g: Prd an Product).
- Equality of synonyms (paper and article).
- Equality of hypernyms (e.g., book *is-a* publication and article *is-a* publication imply book and article have height degree of similarity).
- User-provided name mapping. The user provides manually a mapping.

**Description matching:** In this case, one uses the natural language comment to express the intended semantic. According to theses comments, the integrator can propose an appropriate mapping between elements.

**Mediation approach:** In addition to the schema mapping techniques, the mediator paradigm constitute a serious approach to build a data integration system. The MOMIS project (Berganaschi *et al.*, 1999) provides a process to create special mediators and a global thesaurus. The latter contains a semantic description and the mapping between the different elements. In MOMIS, a global ontology (WordNet) is first used to propose a semi-automatic mapping and some proposed tools help the integrator to continue the integration task.

## THE PROPOSED METHODS

Our method is based on the OMT (*Object Modeling Technique*) method (Runbaugh *et al*., 1995) for the process and on the UML diagrams (Runbaugh *et al*., 1998) for the notation. OMT is a widely used method in application development and it covers an important part of the development process (Analysis, Conception, Implementation) and it can be used in different domains. In this study, the proposed method is inspired from OMT but several modifications are done in order to solve the specific problem in data integration.

UML is considered as the best choice for modellisation. It proposes several notations that covers the different model of a system (static , dynamic). Theses notations are unified, so the model can be understood by different development teams.

The proposed method is also enriched by three elements: A new phase for the re-engineering of the old system, the adaptation of existing steps to the requirements of the proposed integration approach, the use of a semantic mapping tree that provides a complex mapping between global and local attributes, and finally a new diagram to model the implementation of the integrated system. The implementation diagram is created using the UML extension mechanisms.

**The re-engineering phase:** The main goal of this phase is to model the behaviour and the role of each system and of each task in the system globally. This is possible by extracting the structures and the services of each system. The information will serve in the next phases.

**Step 1:** Each system is seen as an independent entity. Thus, we can precise the various systems involved in the integration operation and the global role of each one. This step allows to identify the application domain of each system and to understand its global working.

**Step 2:** The various databases model (relational, object-oriented, hybrid) must be known. Each database schema is captured according to its model. The schema extraction is difficult. It is facilitated if the system is documented. On the contrary, the developer can use technical tools to achieve this.

**Step 3:** Is involved with *services* supplied for each system application. The services are difficult to extract because they are most of the time integrated into the application part, but not into the DBMS. If the application is not documented, the goal can be achieved by scenario execution. the scenario is an important approach to find all the services proposed bay the system.

**Step 4:** Consists to model the system components using use case diagrams. The global comportment of the system is modeled by a unique diagram providing a high level view of the system.

**Step 5:** The developer classifies the behaviours by roles and groups the ones fulfilling a common objective into separate sets. The classification is accomplished according to the semantic of the roles. Each set becomes a possible "integration point". The latter, is considered as the beginning point of for the next phases.

**The analysis phase:** During this phase, a first model of the system is realized. This analysis is based both on the results of the previous phase (Re-engineering) and on the integration project requirements. The result of this phase provides the three analysis models i.e., object model, dynamic model and functional model. They give an abstract view of the global system. The detailed modeling is done latter.

**Building the object model:** During this step, the object model describing the static aspects of the systems (Plihon, 1994) is realized. It is considered as the canonical model for the systems to integrate. The creation of such a model follows the constraints defined in the client requirements and by the results of the re-engineering phase. The object model creation is achieved through these steps:

**Identifying the classes:** The system classes are identified. To satisfy the client constraints, the developer can add classes specific to the domain. The "integration points" that are created in the previous phase are the candidate classes.

**Modeling the integration points:** A class models each integration point. A global mediator class and two local mediators classes represent each set created in the re-engineering phase. One has in charge the data conversion into XML and the other one their modification.

**Identifying global attributes and associations:** The attributes of each class and he associations between classes are identified now. Latter, the unnecessary attributes and associations will be deleted. The attributes of the global classes are denoted global attributes or concepts. They can be created according to two techniques

- The union of the attributes of the various local classes belonging to the same group

- By merging attributes of the same sense. But the data conversion into XML document is not enough to ensure the efficiency of the integration. As a matter of fact, correspondences between local and global attributes must be done. To achieve this, correspondence trees are created between local and global attributes.

**Building the global dictionary:** It allows the clarification of the vocabulary used by the workers involved in the integration project. It constitutes the referential for the developers. The structure of each element of this dictionary is as follow: A name of the concept, a type and a description. The name of the concept is considered as a unique ID. The type determines if the concept is complex or simple. A complex concept is composed with other concepts. In our approach the dictionary is stored as a XML Document. Its structure is described within the Fig. 1 XML schema.

Defining a correspondence between the global concepts and the local ones. This is the role of the semantic correspondence trees that describe the semantic relation between the global concepts and local ones. In our case, one uses the linguistic approach to establish the synonym relation between the global concepts and the local ones. Each global concept is represented as a root of a tree. For each concept, we associate children, each children represents one of the local systems. Each system node contains also different children. The latter are local concepts of the local system. The correspondence between the global and the local concepts is not simply one-to-one. Generally, the global concept is mapped to several local ones (Fig. 2).

We remark that the mapping is not simply one to one. The global concept can be mapped to two local concepts in system1. XML is used to implement this tree. The chose of XML is motivated by the hierarchical characteristic of XML.

- Using the UML notation for the object model diagram. We replace the OMT object model diagram by an UML class diagram.
- Refining the previous steps. We refine the class diagram by introducing inheritance links between classes sharing common properties and by specifying the belonging of attributes to classes.

**Building the dynamic model:** The dynamic model shows the various system objects behaviour. The dynamic aspects of the behaviour are due to the various events rising over the time.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="GlobalConcepts">
  <xsd:complexType>
  <xsd:sequence>
          <xsd:element ref="Concept"/>
  </xsd:sequence>
  </xsd:complexType>
</xsd:element>
 <xsd:element name="Concept">
  <xsd:complexType>
  <xsd:sequence>
  <xsd:element ref="Name"/>
  <xsd:element ref="Type"/>
  <xsd:element ref="Definition"/>
        <xsd:element ref="Concept" minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Type"/>
<xsd:element name="Name"/>
<xsd:element name="Definition"/>
</xsd:schema>
```

Fig. 1: XML schema for the global dictionary

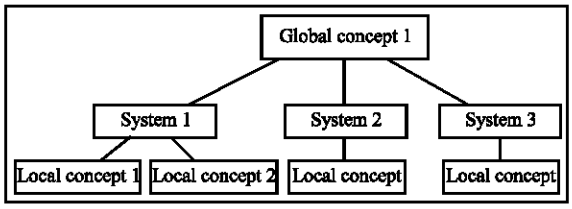


Fig. 2: Correspondences between local and global concept

The dynamic model is crucial for the interactive systems. It is adapted to the DB domain because interactions between the various modules and components exist. The steps are those of the OMT method.

**Building the functional model:** The UML do not support DFD (Data flow diagrams), Thus, we replace the DFD by an activity diagram, which can be used in the workflow system.

**The system design phase:** This phase allows one to choose the structure of the system. The OMT design steps are re-used and modified:

- Partitioning the system. The system is partitioned into modules. Each module contains a global class with two local classes respectively reserved to convert data into XML and to modify data.
- The OMT sub-systems are modeled here using UML packages. The concurrency is treated by the *Thread* approach.

- In order to optimize the network resources and to preserve the fault tolerance, the objects of the global class will be installed on the central computers. The objects of the local systems will be affected to users of the local systems.

**The object design phase:** This phase details the initial design and refines the system design. It proposes to apply the OMT steps; i.e., defining operations for the object model, creating an algorithm to implement the operations, adjust the classes' structure, choosing the representation of the attributes. Then the diagram built in the previous phase is refined. The model created until this step is platform independent. This PIM (platform independent model) (Miller and Mukergi, 2001) does not allow one to specify the needed technology. It will thus be translated into a PSM (Platform Specific Model) where the implementation technologies of the classes will be specified. In order to facilitate the implementation phase, a new implementation diagram is proposed. The latter uses UML notation enriched by some UML extensions to improve the semantic of the different entities.

**A new implementation diagram:** It is needed to have a diagram showing explicitly the data application and the used technologies. This diagram is useful for the code generation (automatic, thanks to the appropriate tools). In OMT, the implementation phase does not provide such a diagram. Defining an implementation diagram consists of defining the graphical notation used, the semantic of notation; the guidance for the code generation.

In order to propose such a diagram, we choose the UML notation together with its extension mechanism. The latter is considered as powerful solution to add semantic to UML notation and it allows more simplicity in code generation. In our research, the extension mechanism is used to mention the role of the different mediator and its implementation technology. According to (Conallen, 2000), an UML extension must contain a brief description of the extension and a list of stereotypes, tagged values and constraints.

Table 1: The UML Extensions for ELITX

| Basic UML element | Stereotype |
|---|---|
| Class | «DB2XML:Type»<br>Description:<br>This class is responsible for the DB conversion to XML using the technology indicated by "Type". |
| Class | «QueryManager:Type»<br>Description:<br>This class is the query manager using the technology indicated by "Type". |
| Class | «DBModifier:Type»<br>Description:<br>This class is a mediator for the data modification using the technology indicated in "Type". |

For each stereotype, we give in Table 1 the UML element. It is based upon; its properties and constraints and for its graphical aspect an icon if needed.

The "type" determines the technology used to implement the mediators. Theses technologies can be (CORBA, DCOM, RMI) or the Web service. Theses extensions are used to build the implementation diagram and to automat the code generation.

**The implementation phase:** OMT proposes implementation mechanisms for many targets (object-oriented programming, database…). In this research, the implementation is based on the implementation diagram and using the distributed objects and the web service technologies.

The steps of this phase are divided into two main categories:

**Mediators generation:** For each global class, we implement a special mediator named Query Manager. The overall global classes constitute a global object-oriented model for the various systems. The mediators have in charge the following operations

- Intercept the global queries of the clients and transmit them to the local mediators.
- If it is a selection query, recovering the results of the various local mediators. These results are coded in XML and have the same schema. The mediator merges the recovered documents and returns the global documents as the final result. If it is a modification query, nothing is returned.

For each local class, we implement two mediator objects. The generation of two classes for each local one seem to be hard coding solution. But, this operation is facilitate by the use of adequate framework. Also, this approach reduces the maintenance effort and provides a high degree of scalability, so whole system do not crashes if one system crashes.

- The first mediator, named *DB2XML* will have in charge the data conversion into XML. However, this conversion is not enough to ensure the integration task. This object will also do.
- The conversion of the global query into local query using the correspondence table.
- Extract data using the local query
- Executing the data transformation to an XML document including the local attributes
- Converting the document into an other one including global attributes.

- The second mediators, named DBModifier have in charge the DB modification operations (add, delete, modify). It converts the global query into local queries thanks to the correspondence table.

For the objects belonging to the DB access type, the mapping is different. It is done as follow:

- The class is implemented as a table in the target DB.
- The fields of the tables are the attributes of the classes.
- The methods of the class having DB oriented treatments are implemented as stored procedures with the same name and arguments. The procedures are written into a language supported by the DBMS.
- The associations are implemented as tables.
- For each table and each SQL selection query, it is needed to create an XML schema. Conversion objects from the DB to XML will employ this document.

All the classes are created after the mapping of the interface file. The implementation of methods follows two cases:

- If the treatment is exclusively DB oriented, the method calls the equivalent DBMS stored procedure.
- Otherwise, the method is made in the implementation class of the server while using data access techniques depending on the language and on the environment.

**Technological consideration:** Choosing the needed ORB (Object Request Broker): This choice is guided by several criteria. It is possible to:

- Choose the ORB by proposing the services we intend to use.
- Choose the ORB implemented under several platforms and allowing the mapping to several languages in order to bring the system as much open as possible.

Applying the implementation steps for the mediator classes that are taking into account the types of the classes (CORBA, RMI and DCOM…). The mapping is done first to a file of the interface type (CORBA IDL, DCOM MIDL, JAVA interface for RMI) then, the appropriate tools will make a second mapping to another target language (except for JAVA, as far as it is itself a target language).

The clients who are not web applications are created as proposed in the OMT method. Their methods contain the invocations to the server objects according to the chosen technology.

In the study of web clients, HTML pages are created. These pages cannot call directly the objects server. To do so, several solutions are proposed:

- The use of mobile code downloaded from the server. In this case, the applets or the COM objects can be used. This solution allows the use of HTML pages as classics applications. Unfortunately, downloading applets and COM objects is very time consuming for network resources and needs to specific configuration for the client side (the virtual machine for JAVA and windows for COM). These limitations lead us to adopt the following solution.
- The use of HTML forms calling the real clients of the distributed objects. In this solution, the HTML forms call the application on the server side. These applications are clients for the server objects. This property (to be at the same time client of distributed objects and server for the web pages) limits principally the technical choice to two technologies
- The servlets (JAVA on the server side) are adapted to the web environment through the management of the HTTP inputs/outputs. They follow the object-oriented paradigm by using JAVA language and are supported by the CORBA and RMI technologies
- The Microsoft ASP (Active Server Pages). The ASP manages the HTTP inputs/outputs and allows the calling the DCOM or CORBA (1999) distributed objects.
- The use the web service technology as a "window" for the distributed objects. This approach is motivated by the fact that the web service technology provides a high level of interoperability in web environment but the distributed object technology provides a good performance to achieve the task of integration. The combination of the two technology will provide a high level of performance, scalability and interoperability. So, a mapping must be done to generate a WSDL (Web Service description Language) corresponding to each classes with the stereotype "Web Service". The communication between the client and the server is done with the protocol SOAP (Simple Object Access Protocol) .
- For dynamic HTML pages, it is needed to add appropriate web server. For the servlets, the reference web server is APACHE/TomCAT, whereas for the ASP, it is IIS (Internet Information Server). In the case of "Web service", Microsoft .Net and J2EE are the most important tools to build a system according to the "web service" technology Fig. 3.

The next section presents an e-business application developed with the proposed method.

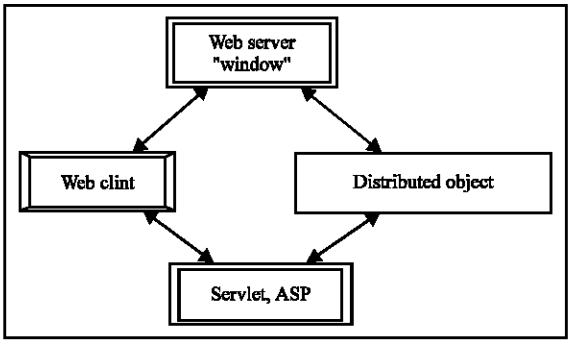


Fig. 3: The use of web service with the distributed object technology

## MATERIALS AND METHODS

The development of e-commerce is one of the most important challenges for all the concerned parts. Thus, many works aim at developing architectures, models and tools for sharing, exchanging and controlling data.

Here, we apply the proposed method to the case of a computer material sale enterprise. For several reasons, each site proposes a proprietary solution to manage its sales. The new manager wants a new computer service supervising three others while proposing to the clients of the enterprise several possibilities: Having a global catalogue of the products with their properties, recording the clients, registering the commands, accessing to various services from the Internet.

These new requirements involve the need to propose a solution for the integration of data sources of the three sites. They are distributed and heterogeneous.

**The re-engineering phase:** The result of this phase is the structure of each site and the behaviour of each system and of each task.

Analyzing each system. The enterprise hopes to integrate its three systems. After analyzing each system, we have this result:

- The first one is a sale management system. It maintains product lines, manage clients and suppliers and register commands and sales. It is based on the ORACLE 9i Enterprise DBMS.
- The second one, based on MS SQL server 2000, manages products, clients and sales.
- The third one is based on Paradox 7 and also manages sales.

The various systems have all in charge sale management with products and clients. The global system must be conforming to these objectives.

Table 2: The tasks realized by the three systems

| The first system | The second system | The third system |
|---|---|---|
| • Managing the products | • Managing the clients | • Managing the clients |
| • Managing the clients | • Managing the suppliers | • Managing the products |
| • Managing the suppliers | • Managing the products | • Recording sales for a client |
| • Recording sales for a client | • Recording sales for a client. | |
| • Displaying the list of products for a client. | • Establishing a command | |
| • Establishing a command. | | |

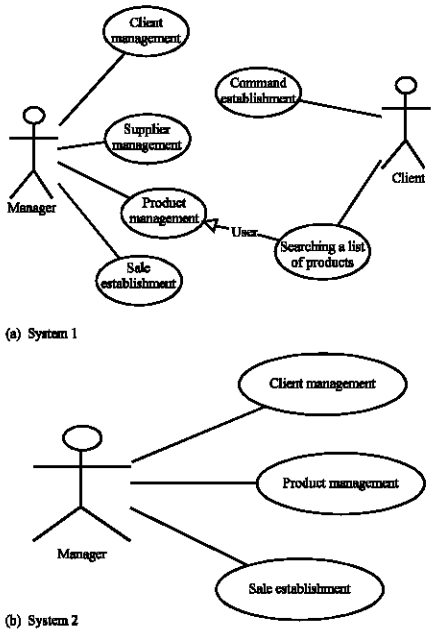


Fig. 4: The use cases diagrams of the three systems

Extracting the database structure. We use adequate tools to achieve this extraction. So, for the first system we use "Jdeveloper" tool for the ORACLE9i database. For the second, "Server Manager", a tool of MS SQL server 2000 is used and for the last one, the tool "Data Base Desktop" found in the Delphi and C++ builder environment is used.

Extracting the treatment philosophy

Consulting the existing documentation for each system, we found that each system realizes a set of tasks summarized in the Table 2.

The three systems have common points. They are all belonging to the same domain (Commerce).

Building the use cases diagrams. Figure 4 shows the use cases diagram of system 1 and 2.

Grouping the components. In Fig. 4, many use cases are similar. During this step, we group the use case presenting the same behaviour into packages. The use cases are integration points and serve as a base for the creation of the classes in the next step.

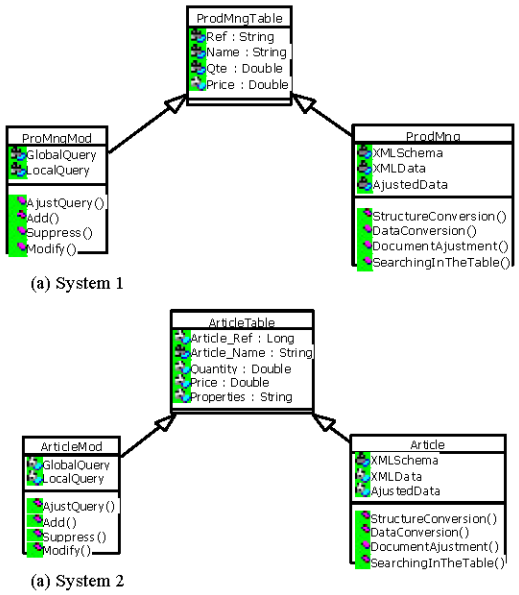


Fig. 5: Hierarchies of local classes for the system 1 and 2

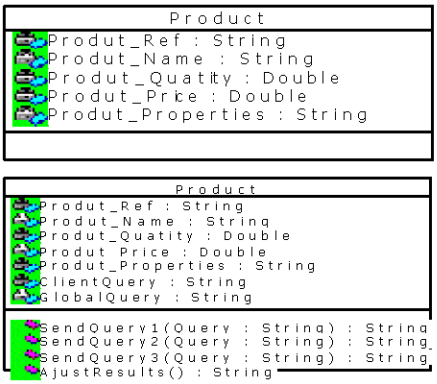


Fig. 6: The class Product before and after enrichment

**The analysis phase:** Constraints have been identified for this system: A client can register himself using the Internet. The list of products searched should be sent by Internet.

**Building the object model:** Identifying classes. According to the information of the previous phase, the candidate classes are Product, Client, Supplier, Command, Sale and Sale Line. We also add for each candidate class, which plays the role of mediator two classes: one for the conversion of data into XML and the other one for the data modification. The names of the classes are the same than the ones of the local systems. For instance, for the global class "Product", we also have: "PrdMod", "Prd"; "ArticleMod", "Article", "ProdMngMod", "ProdMng".

```
xml version="1.0" ?>
-<GlobalConcepts
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file://f:\ysoft\Articledi\
   Xmlexemple\conceptshema.xsd">
 - <Concept>
  <Name>Product</Name>
  <Type>Complex</Type>
  <Definition>Management of the Product</Definition>
 - <Concept>
  <Name>Product_Ref</Name>
  <Type>Simple</Type>
  <Definition>The key of the Product</Definition>
  </Concept>
 - <Concept>
  <Name>Product_Name</Name>
  <Type>Simple</Type>
  <Definition>The name of the product</Definition>
  </Concept>
 .....
  </Concept>
   </GlobalConcepts>
```

Fig. 7: Global dictionary using XML language

Table 3: Correspondences between local and global attributes for the class "GestProd"

| | Local attributes | | |
|---|---|---|---|
| Global attributes | ProdMng | Article | Prd |
| Product_Ref | Ref | Article_Ref | PrdR |
| Product_Name | Name | Article_Name | PrdDesignation |
| Product_Quantity | Qte | Quantity | PrdQte |
| Product_Price | Price | Price | PrdPrice |
| Product_Preperties | - | Properties | - |

Identification of the attributes and associations. Here, we add the attributes and associations according to the requirements. In the following, the *Product* is used as an example of global class. At the local level, three local classes are used to manage the products.

The operations identified into the use case "Product Management" are added to the local class. They allow adding, suppressing and modifying the product properties. These information are added to the "ProdMngMod" class as presented in Fig. 5.a. Inheritance is use to factories the common properties of the both local classes. "ProdMngTable" is the super class. The same mechanism is applied for each local system (Fig. 5.b).

The identification of the global class properties is achieved by two mechanisms: union and merge. Operations are then added Fig. 6 to the global class in order to intercept the client query and send them to the various global classes, receiving the result, adjust and return them.

Building of the data dictionary. Its structure is based on the global classes of the system. As it is an XML document it is created according to the XML schema presented in Fig. 1. Here is an example of global dictionary.

Attributes correspondence Tables between local and global classes are then created.

The creation of this table follows the semantic of each global attribute and the local ones. This table is implemented as XML document that is proposes a good solution for representation the tree structure.

**Building the dynamic model:** Using the scenarios, the construction of the dynamic model will describe the system behaviour in the time Fig. 7.

**Building the functional model:** The functional model is represented here through an UML activity diagram.

**The system design phase:** Each step of the phase is now detailed on our example.

**System partitioning:** The system is partitioned into modules represented as "UML packages" (Fig. 8.a). On the (Fig. 8.b) Product package is detailed.

**Concurrency identification:** Our system should be able to offer services to several clients at the same time. As all the middleware have an internal task manager, this requirement is achieved by the broker configuration.

**Affecting modules to materials:** The global classes are affected to the central computer whereas the local ones are affected to computers where the local systems are localized.

**The objects design phase:** Here the class diagram elaborated before is detailed to ensure a simple and direct implementation. A class materializing the attribute correspondence table is first added for each local system. An example is shown in Fig. 9.

The method "SearchingInTheTable()" permits to find the global attributes corresponding to a local attribute, given the local attribute as parameter. *DB2XML* and *DBModifier* use it.

The steps of this phase are:

- Adding a "Registration" operation into the global class *Client*. It allows registering a new client and calling existing methods such as *Add*.
- Adding an operation "GetTheList" into the *Product* Class in order to know the list of all products proposed by the enterprise.
- The algorithms describing the conversion of the data into XML are presented in Djaghloul (2003).
- Creating the final class diagram includes the UML extensions proposed by the ELITX method. The adequate technologies for each class will be chosen. Let's start by the *Product* class and its local classes.

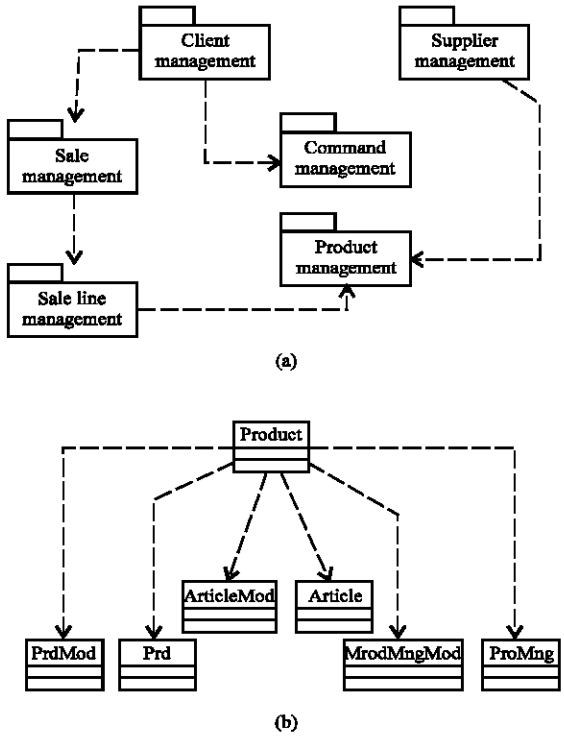


Fig. 8: Modeling the system through packages

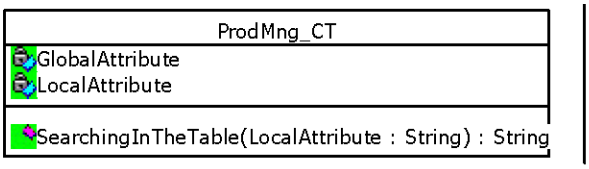


Fig. 9: The class diagram for "ProdMag-CT"

- Using of Web service for Product; CORBA with ORACLE is the best solution for the first system. For the second one, we choose DCOM and for the third RMI is chosen.
- Using of the UML extensions allow to precise the role of each class the content of "Product Management" package is presented in Fig. 10.

**Implementation phase:** Following the steps of this phase ensures an immediate code production, simpler and more consistent.

- Choosing the ORB (only for the CORBA class). In our case, we choose the (VisiBroker 4.0) middleware with the development environment of the Imprise society. Visibroker allows the use of Delphi, C++Builder and Jbuilder.

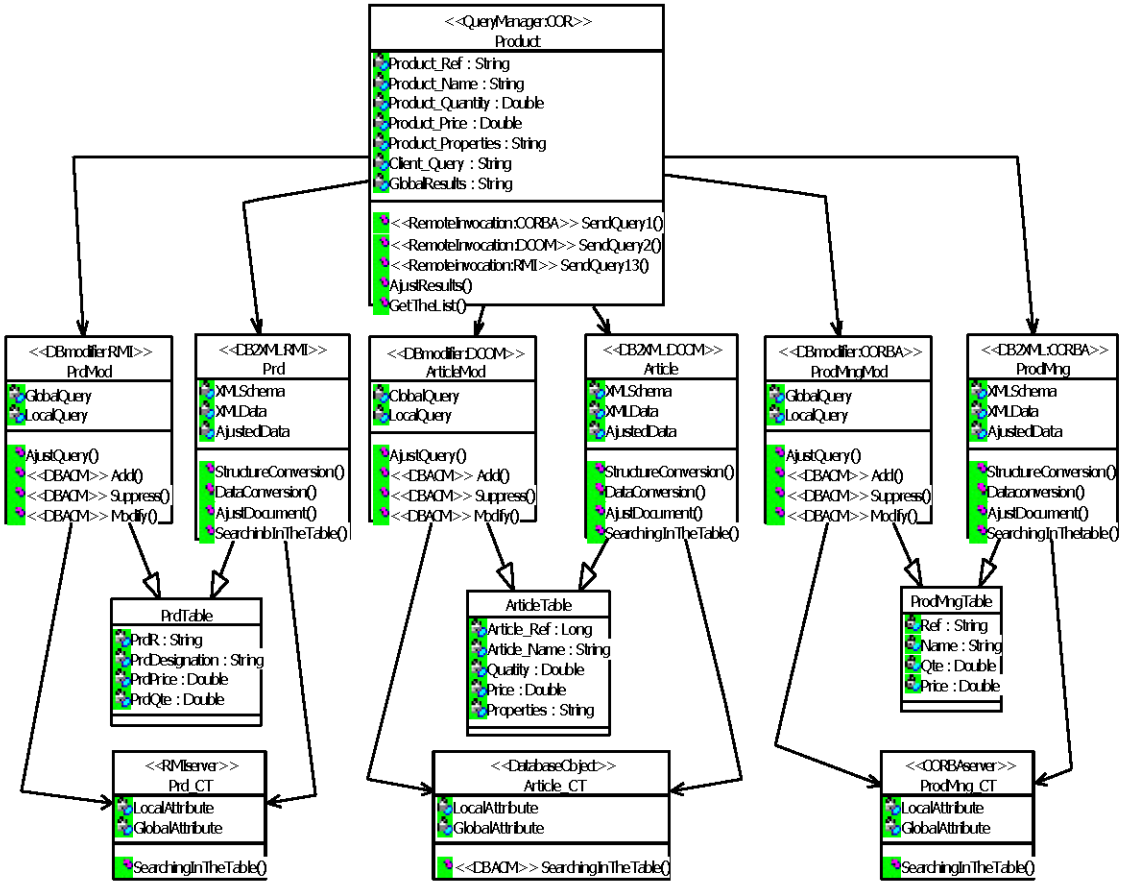


Fig. 10: The implementation diagram of the "Product Management" package

- Creating the interface for classes according to their type.
- Coding the conversion methods according to the algorithms specified.
- Creating the table Article_CT defined as a data table by the stereotype DataBaseObject. This class contains the method "SeachInTheTable" with the stereotype DBACM doing the treatment dedicated to data management. As this class is assigned to the second system using SQL server, we created stored procedures.
- The stereotype of Product class is "QueryManager:WS". So, the class is implemented with Web Service technology. A WSDL file is first created, then the implementation is done with C# language.
- Creating the client part. According to the method, the client is an HTML file containing a form. It is a client for the a web service that is considered as client to the CORBA server. The web server used is Internet Information Server 5.1 with Microsoft .Net SDK.

## CONCLUSION

The study described in this article is a methodological work. The proposed method is based on UML for the diagram and also on OMT for the process. UML diagrams are even extended to answer the integration problems of heterogeneous data sources. This is done by specifying the role of appropriate mediators. Our method is described and then exemplified on an e-commerce application.

The main benefic of our method is that it covers the whole process of data integration by providing appropriate phases with specific steps. The use of UML and its extension mechanism improve the clarity of different models and the proposed implementation diagram automates the code generation of the appropriate mediators. The correspondence tree provides a complex semantic mapping between the locals and the global concepts. This mapping is used to translate the global queries to local ones and to translate the resulting local data to global ones. The use of distributed objects and

web services in implementation phase improves the scalability, the performance and the interoperability of the system, especially in web environment.
Our research can be improved by:

- Taking into account the referential integrity into the databases, refining conversion algorithms.
- Treating the conversion of data of object-oriented DBMS.
- Increasing the power of correspondence tables, to manage more complex structures.
- Providing a framework to facilitate the integration tasks.

## REFERENCES

Batini, C., M. Lenzerini and S.B. Navathe, 1986. A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, 18: 323-364.

Bergamaschi, S., I. Benetti, D. Benventano, F. Guerra and M. Vincini, 2002. An information integration framework for E-Commerce. IEEE Intelligent Systems.

Conallen, J., 2000. Bulding Web Application with UML. Addison-Wesley.

Djaghloul, Y., 2003 Une méthode pour l'intégration des données, basée OMT/UML et utilisant les objets distribués et XML. Master thesis, University of Constantine, Algeria.

Garcia-Molina, H., J. Ulman and J. Widom, 2002. Databases systems: The complete Book Prentice Hall.

Hayne, S. and S. Ram, 1990. Multi-User View Integration System (MUVIS): An Expert System for View Integration. ICDE, pp: 402-409.

Miller, J. and J. Mukerji, 2001. Model Driven Architecture (MDA), Document Number Ormsc.

Plihon, V., 1994. The OMT Methodology, Deliverable D.P.2 Formal Definition of Contextually Based Process Model, Project ESPRIT NATURE (N° 63 53).

Rahm, E. and P. Bernstein, 2001. On matching schemas automatically. VLDB J., 10: 4.

Rumbaugh, J., M. Blaha, F. Eddy, W. Premerlani and W. Lorensen., 1995. OMT. Modélisation et conception orientées objet. Masson Paris and Prentice Hall London.

Rumbaugh, J., I. Jacobson and G. Booch, 1998. The unified Modeling Language Reference Manual, Addison-Wesley.

Sheth, A. and S. Gala., 1989. Attribute Relationships: An Impediment in Automating Schema Integration, In: Proceedings of the NSF workshop on heterogeneous databases.

Spaccapietra, S. and C. Parent, 1991. Conflicts and Correspondence Assertions in Interoperable Databases, SIGMOD Record, 20: 49-54.

Tukwila, The Tukwila Data Integration System, University of Washington.

Web Services Architecture.