DIMS: Implementation of a Federated Information Management System for PRODNET II

Cesar Garita,Yasemin Ugur,Anne Frenkel,Hamideh Afsarmanesh,L.O.Hertzberger

University of Amsterdam, Faculty of Science
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{cesar, yasemin, annef, hamideh, bob}@wins.uva.nl

Abstract. The Esprit project PRODNET II[1] (Production Planning and Management in an Extended Enterprise) had as its main objective the development of a reference architecture and a support infrastructure for Industrial Virtual Enterprises (VEs). The Distributed Information Management system (DIMS) component of the PRODNET architecture supports the complex VE information management requirements and is based on a federated database architecture. In the design of DIMS, general concepts of federated database systems are specifically tailored and adapted to the specificities of the VE paradigm. In previous work, the federated information management requirements for industrial collaborative VE environments have been systematically analyzed. This analysis has identified the required data models and data manipulation operations for the PRODNET application domain. Based on this analysis, the need for federated sharing and exchange of information among the VE enterprises, while preserving their autonomy with proper information visibility rights, are also reported in earlier documents. The focus of this paper is on describing the internal kernel design and certain implementation aspects of the DIMS federated database architecture for PRODNET. In particular, the main components of the internal DIMS architecture are described in details, such as the interoperable server agent, the generic federated query processor, and the export schema manager among other development issues.

1 Introduction

The Virtual Enterprise (VE) concept can be briefly defined as an interoperable network of pre-existing enterprises collaborating towards the achievement of a common goal. As a whole, these enterprises can function together and be regarded as a single organization. Even though there are many definitions and ontologies

[1] This research was partially supported by the ESPRIT project-22647 (1996-1999), PRODNET II of the European Commission, and involved the following partners: New University of Lisbon, University of Amsterdam, ESTEC, Uninova, Lichen Informatique, ProSTEP, Federal University of Santa Catarina, Akros and CSIN.

around the VE paradigm, the fact is that any IT platform or infrastructure aimed at the support of these virtual organizations will certainly face an extremely complex and fractal-shaped problem domain. There already exist many software tools and standards that are able to cope with parts of the related interoperability issues, but there are a large number of challenges and open issues left unresolved. For instance, there is still no common and widely accepted definition of a flexible reference architecture for VEs that can properly support the VE coordination and interoperability requirements.

In this context, it is clear that advanced mechanisms must be designed and implemented in order to support the complex VE information management requirements [22], [5]. In PRODNET, the Distributed Information Management System (DIMS) is the component that encapsulates all the functionality to support these requirements. In order to implement the DIMS component, two major pre-development phases were carried out in terms of the requirement analysis and the general system design. The results of these activities are documented in other papers and briefly summarized in the next paragraphs (for more details see [7], [8], [15]).

First, in relation to the analysis phase, a profound study of the PRODNET VE application domain was performed and an initial classification of the information that needed to be modeled was defined. For this purpose, three main focus areas that represent the main kinds of interaction and exchange of information between different elements of the VE environment were delimited. Then, after further modeling and classification of the information involved in the focus areas, it was possible to identify which part of the information needed to be kept local, imported, exported, and accessed in an integrated way among the different enterprise nodes. Subsequently, the DIMS information management operations that were required for the VE interoperation layer were described.

In order to support all the information management requirements identified during the analysis phase, a federated database architecture was conceived during the design phase of the DIMS module [4],[6],[9]. The design of the DIMS federated layer is based on the definition of a PCL (PRODNET Cooperation Layer) *integrated schema* that is represented and handled in all nodes. Data can be exchanged and shared through this integrated schema, but the proper access rights are defined locally at every enterprise in order to precisely specify the rights of external nodes on the local information of every node. Therefore, the DIMS properly preserves the federated information access and visibility constraints by means of well-determined export schema definitions. The general design of the DIMS has also been influenced by the PEER federated system architecture [2],[3],[24],[25].

After this general design phase, which mostly focused on the specification of the federated schema integration approach, the internal DIMS kernel architecture itself still needed to be designed and implemented. Namely, the specific internal DIMS components needed to be conceptualized, designed, and implemented in order to support the general federated schema architecture. The internal system design and final implementation of the DIMS kernel represent the main focus of this paper.

The rest of this paper is organized as follows. Section 2 describes the general DIMS reference architecture. Section 3 describes in details the main DIMS functional components. Finally, Section 4 summarizes the achieved results after the implementation of the DIMS module.

2 General DIMS Implementation Approach

In order to illustrate the role of the DIMS in the PRODNET architecture, it is necessary to first introduce the general PRODNET node architecture (see Fig. 1). This architecture has been extensively reported in other papers [10], and here only the basic elements are described.

2.1 The PRODNET Node Architecture

Every enterprise in the PRODNET II network of potential VE-members is considered as a *node* consisting of three major components: an Internal Module, a PRODNET Cooperation Layer (PCL), and an Advanced VE Coordination Functionalities (ACFs) module. The Internal Module of a node basically consists of the internal information management systems of the company –such as its Production Planning and Control systems (PPC)-, necessary to accomplish its regular operations. The ACFs module provides some additional functionalities to extend the scope of the PCL, including the coordination of VE-related activities, and supporting tools for logistics operations. The Distributed Business Process Manager (DBPMS) module represents one of these ACFs. Finally, the PCL component is responsible for the actual functionalities for the inter-operation of nodes in the PRODNET network. The PCL is the fundamental component that allows the enterprise to interoperate with others in the context of the VE. The PCL itself consists of several internal *components* described below:

– The Human Interface module: supports the end-users interactions with the PCL.

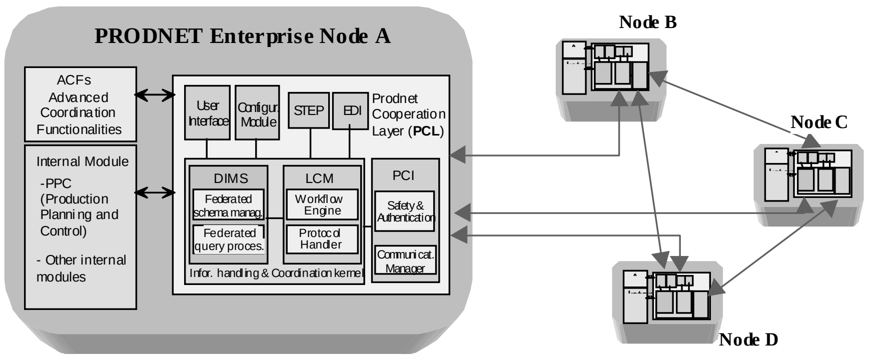

Fig. 1. Description of the PRODNET node architecture.

– The Configuration component: allows the set up of certain elements and functionalities of the PCL, for every VE in which a given enterprise is involved.
– The STEP and EDI modules: primarily support the exchange of technical product data and the commercial order-related data respectively.
– The Local Coordination Module (LCM): executes and controls the internal PCL workflow, which specifies the desired cooperation behavior of each PCL [11].
– The PRODNET Communication Interface (PCI): is responsible for the actual communication channel among nodes in the VE network [20].
– The Distributed Information Management System (DIMS): supports all the distributed information management requirements for the PCL operation.

In the next two sections, the details of the DIMS reference architecture and its internal components are described.

2.2 The DIMS three-tier architecture

In addition to the client-server kind of applications, some multi-threaded applications are conveniently modeled using a three-tier architecture, also called client-agent-server architecture. In this architecture the client is only concerned with presentation services. The agent (or application server) processes the application logic for the client tier, hiding the underlying implementation and access details of the server tier and adding higher level support functionalities for the client. The server tier includes the low-level implementation of the data management services required by the agent tier.

In this sense, the DIMS implementation approach follows a three-tier architecture of this type (please see Fig. 2). The client tier is represented by all the other PCL components that request DIMS services via a DIMS client library. The applications server (agent) is represented by the DIMS Server Agent, together with the other DIMS internal operational components. The DIMS Sever Agent acts as a client of an ORACLE database server, which in turn represents the server tier in this scenario. Bearing in mind this general three-tier architecture will help understanding the relationships among some DIMS components that will be addressed in coming sections of this document.

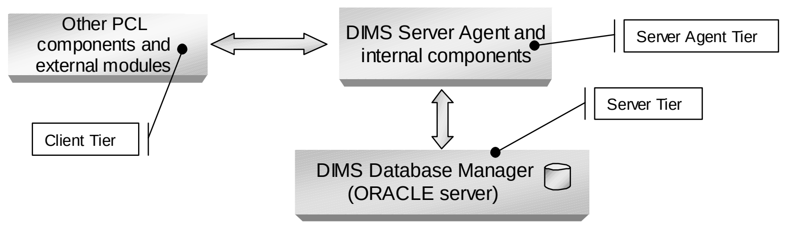
Other PCL components and external modules
DIMS Server Agent and internal components
Server Agent Tier
Server Tier
Client Tier
DIMS Database Manager (ORACLE server)

Fig. 2. General DIMS three-tier architecture.

2.3 The DIMS reference architecture

In this section, the main components of the DIMS applications server tier introduced in the previous section are described. The general reference architecture of this tier embodies the following components, as depicted in Fig. 3: the DIMS Server Agent, the Federated Query Processor, the Export Schema Manager and Tool, the Internal DIMS Database Manager, and the DIMS Kernel Configurator. A general description for every component is given in the paragraphs below and in Section 3, more details will be included:

- DIMS Server Agent: corresponds to the heart of the “agent tier” of the DIMS architecture and is responsible for receiving and dispatching all the DIMS service requests issued by the other PCL modules. The Server Agent determines the nature of the service requests and activates of the involved DIMS internal components.
- DIMS Federated Query Processor (FQP): its main objective is to transparently support the access to data distributed over the nodes of the VE network, taking into account the specific visibility access rights (represented by export schemas) defined for every node. The FQP functionality of DIMS enables end users such as the VE Coordinator to query the privileged proprietary VE related information for which the coordinator is authorized, while hiding the data location details.
- Export Schema Manager (ESM) and Tool (ESMT): encloses the functionality to create and maintain the hierarchy of export schemas that are defined on the PCL local schema, based on the visibility access that are specified for a given node. The ESM will ensure that the export schema hierarchy remains consistent, and that the schema definitions for every dependent partner export schema are properly created. The ESMT (Export Schema Manager Tool) developed for DIMS provides a user inteface to support the definition and creation of the export schemas.
- Internal DIMS Database Manager: the DBMS that was used as “construction ground” for the DIMS is the Oracle DBMS (version 7.3). This component represents the server tier of the DIMS which provides all the functionalities that are expected from a database management system including: transaction management, data storage and retrieval, stored procedures management, SQL support, triggers, etc. The Oracle Server is used from the DIMS internal components through a specific set of access mechanisms such as ODBC drivers, stored procedures and packages. All the low-level details to access these Oracle-specific tools are hidden from the PCL components, which do not access the Oracle server directly.
- DIMS Kernel Configurator: allows the user to specify the values of certain DIMS operation parameters, including the communication port number of DIMS server, and the timeout duration for distributed queries.

3 DIMS internal implementation

This section addresses more specific design and implementation details regarding the DIMS Server Agent, the Federated Query Processor and the Export Schema Manager components of the DIMS architecture, that were introduced in Section 2.

3.1 The DIMS Server Agent

The Server Agent is the gateway to the internal DIMS architecture, which encapsulates all the specific information management services for the PCL modules. The agent can be seen as a bi-directional gateway, since it also provides a mechanism that allows internal DIMS components to reach the service interface of other PCL modules when required. To support this interoperation mechanism, both the DIMS and the other PCL modules are extended (wrapped) with some kind of interoperation layer, through which services can be reciprocally requested and answered. This layer couples with the associated heterogenity problems among these modules.

The interoperation layer is actually composed of two main parts (see Fig. 4): the PCL Module Interoperation Layer and the DIMS Interoperation Layer. Each of these layers is in turn decomposed into two major components: the client component and the server (or proxy) component. This subdivision is due to the fact that the interoperation between the PCL module and the DIMS is managed by a dual client-server interaction, in which each interoperation layer needs to simultaneously act as client and server of the other layer. For instance, the DIMS is able to request services from the PCL module (PM) via the PM client interface. The PM client in

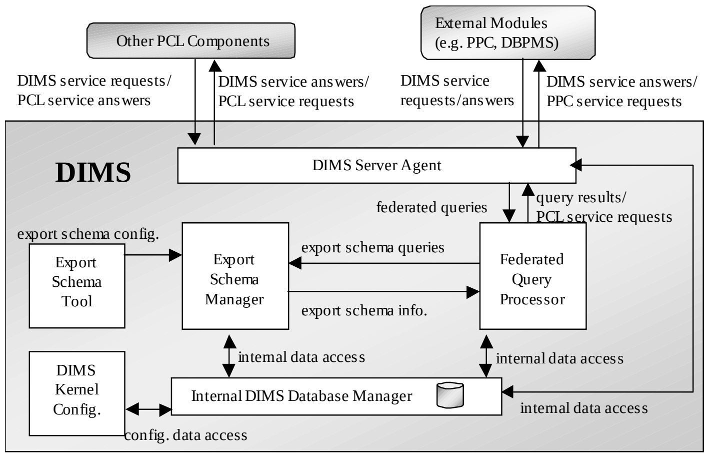
Other PCL Components
External Modules (e.g. PPC, DBPMS)
DIMS service requests/ PCL service answers
DIMS service answers/ PCL service requests
DIMS service requests/answers
DIMS service answers/ PPC service requests
DIMS
DIMS Server Agent
federated queries
query results/ PCL service requests
export schema config.
Export Schema Tool
Export Schema Manager
export schema queries
export schema info.
Federated Query Processor
internal data access
internal data access
DIMS Kernel Config.
Internal DIMS Database Manager
internal data access
config. data access

Fig. 3. . General DIMS architecture approach.

turn will contact the PM server that will carry out the service request. Similarly, the PM needs to be able to request services from the DIMS via the DIMS client. In this way, the DIMS client will in turn contact the DIMS server (proxy) that will carry out the service request, as shown in Fig. 4.

All PMs and DIMS client interfaces are provided as DLLs that are linked to the corresponding main application. The DLL supports the interface to specific services that must be implemented in the associated server. Each client DLL provider must implement a mechanism in order to establish the communication with the corresponding server. For the implementation of the communication mechanism for each module, the PRODNET approach does not impose any constraints about it. In the case of DIMS, the implementation was done using remote procedure calls (RPC) [23].

Furthermore, please notice that the communication mechanism to implement the functions provided in the client DLLs to request internal services, can be implemented either synchronous or asynchronous. In the synchronous approach, the requesting application program will not proceed with its execution until the request is fulfilled. The service request can also be satisfied asynchronously, which means that the issuing application will send the request and will be "released" to do other tasks while the service is carried out. Once the service request is accomplished, the answer is sent to the issuing application via a specific function. Both approaches can supported by the general PRODNET model, however in this document the asynchronous approach is assumed and described since it is the most commonly used approach.

In order to support the asynchronous approach, a pre-requisite for each PM (and the DIMS) is the implementation of an interface providing a pair of services required for the bilateral interoperation mechanism. These interface services are included in the client DLLs. The basic declaration for the PM interface services is as follows:

<*PM_ID*>**_ServiceRequest** (parameters)
<*PM_ID*>**_ServiceAnswer** (parameters)

The <*PM_ID*> is the PM unique identifier, an acronym used to uniquely represent each PM within a certain enterprise environment, such as: "LCM", "STEP", etc. For both of the request/answer functions, the parameters comply with a generic type definition that allows the transmission of elements of all the necessary types.

A basic interaction scenario of the general DIMS-PMs integration model using the service request/answer functions is also depicted in Fig. 3. For instance, let us suppose that the PM needs to request a DIMS service. Then the PM will asynchronously call the DIMS_ServiceRequest function of the DIMS client interface. After the invocation, the PM will be released to continue its regular execution, and the request will be transparently transferred to the DIMS server at the DIMS interoperation layer side. When the DIMS service request is fulfilled, the answer is sent to the PM via the <*PM_ID*>_ServiceAnswer function of the PM client DLL. This PM client interface will in turn seamlessly contact the PM server. It is also possible that the DIMS request a service from the PM in an equivalent way.

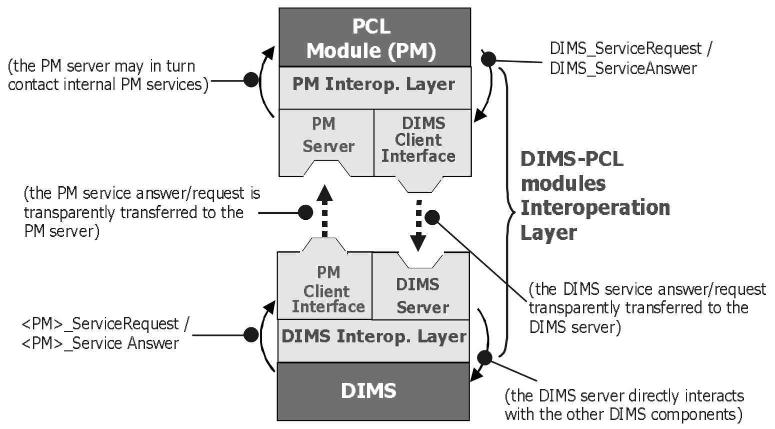
PCL Module (PM)
PM Interop. Layer
PM Server
DIMS Client Interface
DIMS_ServiceRequest / DIMS_ServiceAnswer
(the PM server may in turn contact internal PM services)
DIMS-PCL modules Interoperation Layer
(the PM service answer/request is transparently transferred to the PM server)
PM Client Interface
DIMS Server
(the DIMS service answer/request transparently transferred to the DIMS server)
<PM>_ServiceRequest / <PM>_Service Answer
DIMS Interop. Layer
DIMS
(the DIMS server directly interacts with the other DIMS components)

Fig. 4. – General DIMS - PCL module interaction.

About the parameters of the service request/answer functions, they consist of three main predefined types: a *token* parameter, a list of *PCL parameters* of an abstract PCL parameter type, and a result condition parameter. The token parameter type supports the *context* definition for the execution of the service request, and specifies for instance a unique service request identifier, the identifier of the specific service that is being requested from the target module, and a timestamp, among other fields. The PCL parameters list allows the specification of the actual parameters that the specific module service demands. For this PCL parameters list, an abstract PCL data type has been defined from which a large set of specific data types can be derived and used in any module service. Through this mechanism, the DIMS can offer high-level services involving the retrieval of distributed information along the VE network. For instance, to support the VE monitoring and coordination, certain DIMS services can be used by the DBPMS module to get specific information related to the purchase orders and internal production orders, that have been assigned to a given set of the VE partners.

3.2 DIMS Federated Query Processing

The PCL applications such as advanced coordination modules, and end users need to access VE-related data without worrying about the physical data distribution. At the same time, the owner of the data wants to share different parts of the local VE level data for different groups of users and keep other part confidential. The Federated Query Processor (FQP) component adds to DIMS the functionality to provide authorized access to proprietary VE-related data distributed over the VE network, depending on their visibility levels defined at the remote sites [7].

In this section, the main tasks of the FQP are detailed. Other approaches to the global query processing design and development in federated and multi-database systems can be found in [13], [16], [17], [18], [19] and [12]. Most of these works aim at a query processing mechanism to support a general multi-database architecture where normally there is a central interoperable layer to handle global queries, and a component layer at each participating database system to process the subqueries. However, considering the VE peculiarities, these generic approaches should be tailored and extended such as has been done for DIMS.

DIMS Federated Query Processing Internal Subtasks

The processing of federated queries in DIMS can be summarized as follows: when the query arrives at the DIMS, it is analyzed and decomposed into a set of single-site subqueries, each of which needs to be sent to only one site (VE node) to be processed. After that, the results of the sub-queries are gathered and merged into the final result. If necessary, the FQP interacts with the corresponding PPC to retrieve up-to-date local production data, during this process. More specifically, the main subtasks of FQP can be enumerated as follows: Query reformulation and decomposition, Subquery transmission, Local subquery rewriting and evaluation, Pull PPC data, Subquery result transmission, and Subquery result merge. Each of these tasks is depicted in Fig. 5 and described in details in the rest of this section.

Query Reformulation and Decomposition

DIMS supports a set of *high-level service functions* to be used by the other modules, based on the PCL interface standards, to hide the low-level database access details from the query requesters. When one of these functions is called, FQP reformulates it into an internal query format using the parameters specified for every function. This reformulated query is then analyzed to determine the specific VE partners involved in the original query. Further, the query is decomposed into a set of simpler subqueries, so that each subquery involves the retrieval of data from only one VE node. Namely, each subquery needs to be sent to only one corresponding partner to be processed locally at that side.

Subquery Transmission

This task sends subqueries to the necessary remote nodes. The subqueries, which are sent from one DIMS server to another DIMS server (in another node), comply with a specific format in order to facilitate the processing at the target node. A DIMS subquery request message format is composed by four fields. In the first field, the query message contains a tag field that specifies the type of message (e.g, subquery request or answer). The second and third fields correspond to the identifications of the origin and target nodes, respectively. Since one enterprise may involve in more than one VE at the same time, the node identification should contain VE identifier as well as the enterprise identifier. Finally, the last field is reserved for the content of the query itself. To transmit the query from one DIMS node to another one, the DIMS exploits the facilities of the Local Coordinator Module (LCM) and the PRODNET Communication Interface module (PCI) as will be illustrated later in this section.

Local Subquery Rewriting and Evaluation

The evaluation of the subquery at the external node is crucial from the secure and protected data access point of view. The PCL schema definition is the same in all nodes as described earlier. Therefore, any node can issue a query against its "imported" part of the schema. However, the access rights of every node to the data that it can import from another node are precisely specified in the individual export schema defined for the origin node in the target node. Therefore, the arriving query will be carefully evaluated against the corresponding export schema of the sender and all the visibility access constraints will be preserved. For this aim, the subquery needs to be rewritten by incorporating the operations that are used to derive the export schema. The FQP component operates on the export schema definitions from the Export Schema Manager, so that it can always reflect the updates in these definitions.

Pull PPC Data

DIMS applications and end-users may need to get the most recently up-to-date data from the local data sources inside the PPC system of the enterprise. To meet this need, DIMS communicates with PPC, through a specific Application Programming Interface (API) developed to accomplish this necessity. The functions in this PPC API allow the retrieval of data from the internal database system, and convert the result into the common data format defined by the PCL interoperability approach. The DIMS-PPC interaction is carried out using the workflow activities coordinated by LCM defined for data retrieval from legacy systems. This workflow plan enables

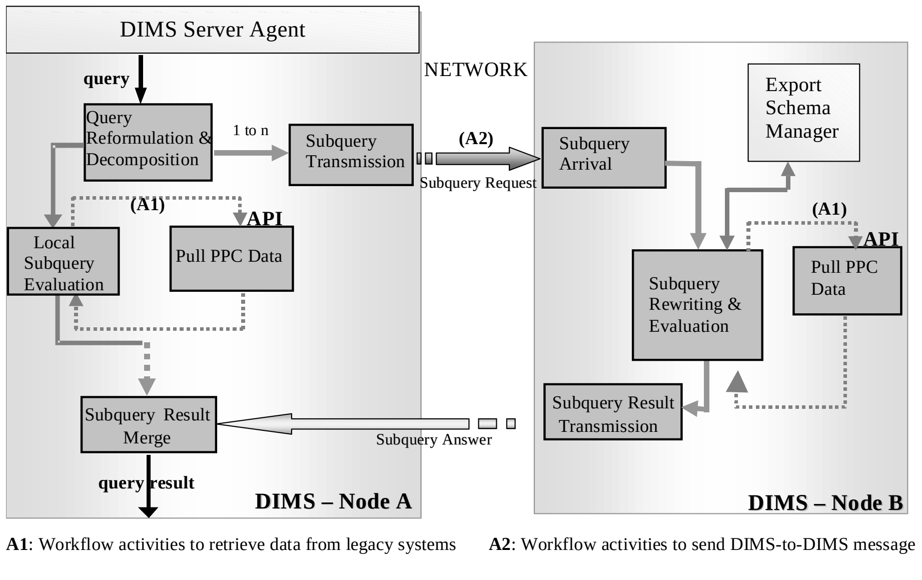
DIMS Server Agent
NETWORK
query
Query Reformulation & Decomposition
1 to n
Subquery Transmission
(A2)
Subquery Request
Subquery Arrival
Export Schema Manager
(A1)
API
Local Subquery Evaluation
Pull PPC Data
(A1)
API
Subquery Rewriting & Evaluation
Pull PPC Data
Subquery Result Merge
Subquery Answer
Subquery Result Transmission
query result
DIMS – Node A
DIMS – Node B
A1: Workflow activities to retrieve data from legacy systems
A2: Workflow activities to send DIMS-to-DIMS message

Fig. 5. - FQP Subtasks and Interaction

DIMS to get data from legacy systems of the enterprise, and any change in the activities can be easily adopted by using the workflow manager in LCM. Consequently, the DIMS gets the up-to-date data from local production system through the specific API and stores it in its internal database temporarily during the processing of the query. After this, the modified external subquery or local subquery is executed on the data stored temporarily in its internal database. An example of this functionality is included later in this section.

SubQuery Result Transmission
This step is similar to the step of sending the subquery to the remote nodes, except for minor differences in the format and the content of the inter-DIMS message. The first three fields of this DIMS subquery result message are defined earlier in the "SubQuery Transmission" task. The last field corresponds to the result of the query, which is composed of the identification of the subquery, the return code for the subquery evaluation, and the content of the associated result itself.

Subquery Result Merge
Once the subquery results arrive at the origin node which started the processing of the federated query being executed, the results of subqueries are kept in a separate FQP result-blob table. Each result blob has the identifier of the query that it belongs to, as well as the other information, such as its size. When all the results have arrived, the merging step starts and is achieved by the "union" operation of the individual results.

Federated Query Processing Steps
The FQP mechanism is implemented using multi-thread programming so that it can receive multi-requests simultaneously via the DIMS server agent, and consequently it can support the execution of different queries at the same time. In the general case, two kinds of federated queries may arrive at the DIMS of an enterprise, an *internal query* (a global query arriving from the VE coordinator module or an end-user) or an *external query* (a subquery arriving from VE member's DIMS). When an internal query is issued either by end-user or an application to the DIMS, the FQP of the DIMS involves the following simplified steps at the query issuer site.

```
1. Identify all the nodes (internal/external) to which the subquery must be sent
2. Decompose the query into subqueries where every subquery involves only one partner
3. For every node to which the subquery must be sent
    3.1. If the receiving node is this node itself
        3.1.1. If the query issuer asks the most up-to-date PPC generated data
                Invoke the workflow activity to retrieve updated data from PPC into the DIMS
        3.1.2. Evaluate the subquery and prepare the result
    3.2. If the receiving node is external node
        3.2.1. Prepare inter-DIMS query message (request)
        3.2.2. Invoke the workflow activity to send the DIMS query message to the remote node
4. Wait for the results of all subqueries evaluation from the external nodes (timeout is considered)
5. Process the partial results and merge them into final result
6. Return final result
```

When an *external* query arrives from another node, the query is evaluated against the export schema defined for the query sender node. When the result is obtained, it must be returned to the sender of the query. This process is described as follows.

```
1. Interpret the inter-DIMS query message
   1.1. If the query issuer asks the most up-to-date PPC generated data
            Invoke the workflow to ask PPC store most recently updated data into the DIMS
        Otherwise, evaluate the rewritten subquery and prepare the result
2. If the workflow to pull PPC up-to-date data is invoked
      2.1.  Wait for the PPC to finish the task (timeout is properly considered)
      2.2.  When the response arrives from PPC, evaluate the rewritten subquery and prepare the result
3. Prepare the inter-DIMS message including the result of the subquery and invoke the workflow to send
   the result back to the query issuer node
```

PCL modules interactions to support FQP

The activity of sending and receiving Inter-DIMS messages is performed through both the workflow management mechanism provided by the Local Coordinator Module (LCM), and the communication means provided by the PRODNET Communication Interface module (PCI). With this strategy, the LCM workflow management mechanism is exploited to support flexible definition and changes in the process of sending/receiving DIMS to DIMS (enterprise to enterprise) messages depending on the business processes and procedures applied at every enterprise [11]. Besides, the advanced and safe communication facilities are used from the PCI module. In Fig. 6, the sequence diagrams of inter-communication between several PCL modules involved in federated query processing is shown, for the case where a subquery is sent to another remote node and where the most recently updated data from the PPC is demanded. This scenario can be extended for the general case that

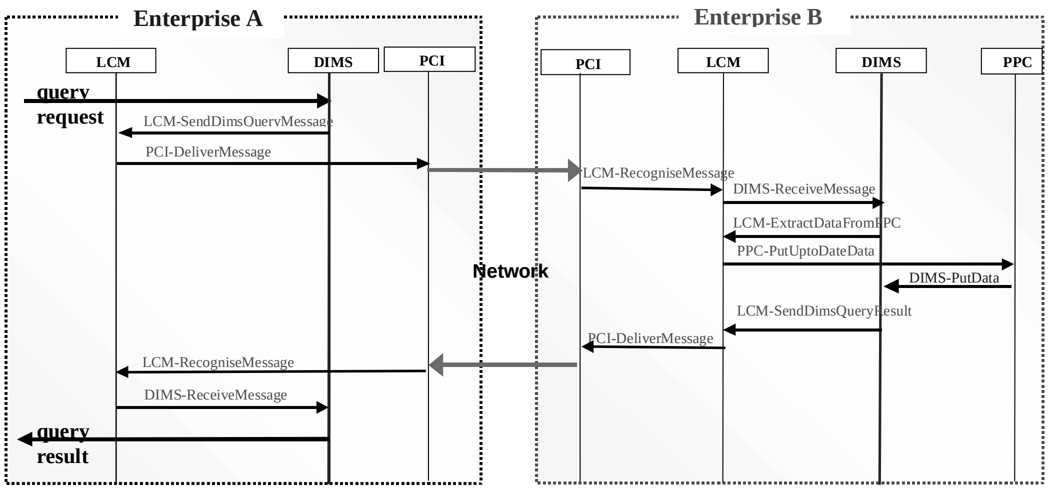

Fig. 6. –PCL Module interactions to support FQP

involves a set of remote nodes and one sender node. The sequences in this diagram can be described as follows: at the sender node where the query is generated (Enterprise A), DIMS processes the federated query and invokes the workflow activity (SendDimsQueryMessage) to send an external subquery to Enterprise B. This activity involves sending an inter-DIMS subquery message, embedded in the PCI message format, to the remote node. When the PCI module at Enterprise B receives this message, informs the DIMS at the same enterprise through LCM. The DIMS extracts the external subquery and rewrites it against export schema defined for Enterprise A. After the execution of the workflow activity, which extracts PPC up-to-date data, DIMS evaluates the rewritten query and returns the result to the sender by using a workflow activity (SendDimsQueryResult). The DIMS at Enterpise A processes and merges the results from other nodes and returns the final result.

3.3 DIMS Federated Export Schema Manager

In the Virtual Enterprise environment, every node must be able to give different visibility levels and access rights to its local information to every other partner in every particular VE that it is involved. The level of visibility and access that other nodes have on the local information of a given node will be determined by the role these nodes are going to play in the VE. To accomplish this objective following a federated database approach, every node can protect its autonomy and privacy by defining one detailed *individual export schema* based on its local schema, for every other node with which it shares information [7], [8]. Another approach to export and integrated schema management in the context of federated databases can be found for instance in [14]. This work defines a general federated database approach, which provides an ODMG interface to federate heterogeneous DBMS. However, in contrast to DIMS, this architecture does not aim at VE specific support. In [21] and [1], other approaches to view definitions based on underlying database schemas are described.

In DIMS, besides allowing the enterprise nodes to define individual export schema definition on the local schema for every external “partner”, we have generalized this basic idea to the definition of a complete *hierarchy of export schemas* based on the role of a given company in a VE.

The decision of every node in the federation, on what part of its local information to make available to the other nodes in the VE, will be based on the role that each of these nodes is going to play. Every partner of this enterprise in a given VE, will be associated with a role, and each role is related to an individual schema in the hierarchy of export schemas handled at this node. This hierarchy allows the grouping and classification of common export schema characteristics, facilitating the control and management of the individual export schema definitions. The objective of this approach is to avoid creating an export schema for each one of the nodes involved in the VE, since every node will give the same access rights to two or more of these nodes, e.g. these nodes will have the same role in the VE.

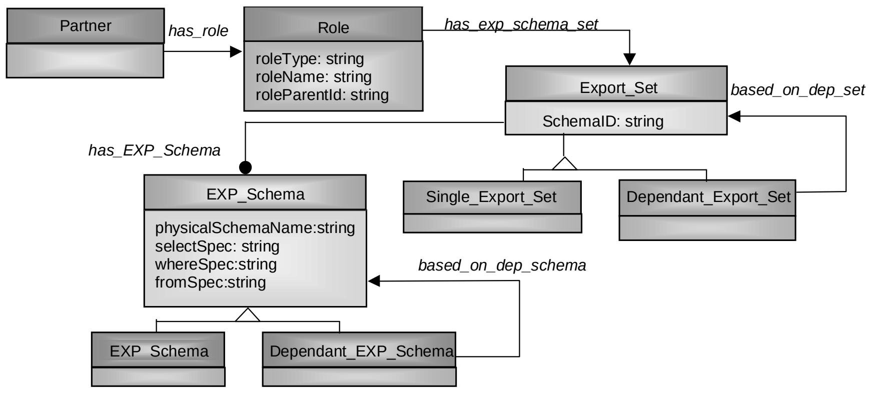

Fig. 7. Schema definitions for partner export schemas management

For example, let us assume that for a VE there are three different kinds of roles that a given enterprise can play: the *coordinator*, *supervisor* (subordinated to coordinator but enabled to monitor certain VE activities), and *regular* VE partner (subordinated to supervisor and enabled to perform certain restricted VE activities). Clearly, for every role, different information items must be made accessible from other nodes. For example, a VE coordinator needs to know information, which for another regular VE partner may even be a secret. The support for a fine-grained visibility level mechanism is required to model this situation. For more information on the concept of role and its relation with export schema definitions please see [8], [15].

In Fig. 7 the design of the database schema related with export schema management is presented. Through this schema, the recursive definition of elements of the export schema hierarchy and the role hierarchy are supported. For every VE role, an external schema set (Export_Set) is defined, which at the end corresponds to the partner's export schema. Through the Export_Set, the proper visibility levels for the partners on the local schema of the enterprise are specified. An Export_Set can be either a single or a dependent export set depending if they are based on other export sets or not. With this approach, on one hand, support for the general export schemas definition is provided, where not only the pre-defined export schema definitions at the level of VE, coordinators, supervisors, and partners, are considered, but also other hierarchies can be defined and supported as necessary. On the other hand, an Export_Set consists of a set of schemas, which in turn can be single schema (EXP) or dependent schema (Dependent-EXP) following this definition strategy.

Besides the definition of the export schema hierarchy, it is necessary to define the hierarchy of roles. For every different function that is going to be played in the VE a role (ROLE) is defined. Every ROLE has as attributes: the general type of the role (e.g. coordinator, supervisor, regular), the name that identifies the role and the identification of the parent of the role in the hierarchy.

To operate on the described schema, an "Export Schema Manager" (ESM) module has been developed. The ESM is used to create a basic export schema, and then, to define dependent partner export schemas based on it, is also used to create the hierarchy of roles. The ESM will ensure that the export schema and the role hierarchies remain consistent, and that the schema definitions for every dependent partner export schema are properly created.

The Export Schema Manager Tool (ESMT) is a grahical user-friendly application developed on top of the ESM that helps the human operator of PCL to define and create the export schemas, during the configuration phase of the VE. The main window of the ESMT interface tool contains a menu bar that enables the user to perform different operations, such as create an export schema for every database table, define the EXP/Dependent-EXP set, define the role export schema hierarchy, and create the export schema for an enterprise among others. For a more detailed description of the ESMT, please see [8].

In relation to the DIMS implementation environment and tools, the DIMS was implemented on Windows NT using Microsoft Visual C++ (Professional Edition 5.0.) Other tools used to support the DIMS implementation include: Microsoft Foundation Classes (MFC), MFC Database classes, ODBC drivers, and RPC support tools.

4 Conclusions

The implementation of the distributed/federated architecture of the DIMS in PRODNET, has proven to properly support the cooperative information sharing and exchange, node autonomy, information visibility levels and access rights for exchanged data among the VE nodes. The DIMS, assisted by the workflow management engine of LCM, acts as a real backbone in order to support the entire PCL operation and as a result, as a backbone to support the VE operation itself.

The implemented DIMS server represents a three-tier architecture with multi-threading capabilities, which efficiently support the interaction between the DIMS kernel and the other PCL modules. The DIMS Server Agent design provides a high degree of flexibility for future extensions, and at the same time it allows the invokation of the DIMS services from phisically distributed machines.

The DIMS Federated Query Processor that has been developed provides access to the proprietary VE information for the authorized enterprises, while hiding the data location details from the end user. The defined interoperation scenarios between the FQP and the workflow management engine of LCM represent one of the novel implementation strategies of the DIMS.

The DIMS Export Schema Manager properly supports the definition of visibility levels and access rights for the information accesses from other VE nodes. The Export Schema Management tool incorporates advanced user interface graphic elements and provides a comprehensive and friendly environment for the end users.

Finally, the implemented DIMS module satisfies all the information management requirements that were identified within the context of the PRODNET project, and provides a solid platform that can be extended in order to address future VE life-cycle support enhancements to the current PCL implementation.

References

1. Abiteboul, S; Bonner, A. - Objects and Views, in Proceedings ACM SIGMOD91, pages 238-247, May 1991.
2. Afsarmanesh, H; Tuijnman, F.; Wiedijk, M.; Hertzberger, L.O.- Distributed Schema Management in a Cooperation Network of Autonomous Agents. Proceedings of the 4th International Conference on Database and Expert Systems Applications (DEXA'93), Lecture Notes in Computer Science 720, pages 565-576, Springer-Verlag, Sept 93.
3. Afsarmanesh, H. et al. Flexible and Dynamic Integration of Multiple Information Bases, Proceedings of the 5th IEEE International Conference on "Database and Expert Systems Applications DEXA'94", Athens, Greece, Lecture Notes in Computer Science (LNCS) 856, Springer Verlag, p. 744-753, Sep. 1994.
4. Afsarmanesh, H; Camarinha, L. - Federated Information Management for Cooperative Information - in proceedings of the 8th International Conference on Database and Expert Systems Applications (DEXA'97), September 97.
5. Afsarmanesh, H; Garita, C; Hertzberger, L.O.; Santos, V. - Management of Distributed Information in Virtual Enterprises:The PRODNET Approach –in proceedings of the International Conference on Concurrent Enterprising (ICE'97), Nottingham,UK, October 97.
6. Afsarmanesh, H., Garita, C., Hertzberger, L.O., Virtual Enterprises and Federated Information Sharing. Proceedings of the 9th IEEE International Conference on "Database and Expert Systems Applications", DEXA'98, Lecture Notes in Computer Science, Vienna, Austria, August 1998.
7. H. Afsarmanesh, C. Garita, Y. Ugur, A. Frenkel, and L.O. Hertzberger. "Federated Information Management Requirements for Virtual Enterprises". In Infrastructures for Virtual Enterprises - Networking Industrial Enterprises (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, ISBN 0-7923-8639-6, 1999.
8. H. Afsarmanesh, C. Garita, Y. Ugur, A. Frenkel, and L. O. Hertzberger. "Design of the DIMS Architecture in PRODNET". In Infrastructures for Virtual Enterprises - Networking Industrial Enterprises (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, ISBN 0-7923-8639-6, 1999.
9. Camarinha-Matos, L; Afsarmanesh, H; Garita, C.; Lima, C - Towards an Architecture for Virtual Enterprises. Special issue of the journal of Intelligent Manufacturing with the focus on Agent-based Manufacturing, Volume 9, Number 2, Pages 189-199, Chapman and Hall publications, March 1998.
10. Camarinha-Matos, L; Afsarmanesh, H. "The PRODNET Infrastructure". In Infrastructures for Virtual Enterprises - Networking Industrial Enterprises (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, ISBN 0-7923-8639-6, 1999.
11. Camarinha-Matos, L; Lima, C.P. "PRODNET coordination module". In Infrastructures for Virtual Enterprises - Networking Industrial Enterprises (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, ISBN 0-7923-8639-6, 1999.
12. Dayal, U. "Query Processing in a Multidatabase System". In "Query Processing in Database Systems" (W. Kim, D. S. Reiner and D. S. Batory, Editors), Springer 1985.
13. Elmagarmid, A; Rusinkiewicz, M; Sheth, A. "Management Of Heterogeneous and Autonomous Database Systems", Morgan Kaufmann Publishers, 1999.
14. Fankhauser, F. et al. Experiences in Federated Databases: From IRO-DB to MIRO-Web. Proceedings of 24th International Conference on Very Large Data Bases, New York, USA, pages 655-658, Morgan Kaufmann, August 1998.

15. Garita, C.; Afsarmanesh, H.; Hertzberger, L.O. – “The PRODNET Cooperative Information Management for Industrial Virtual Enterprises”, submitted to the International Journal of Intelligent Manufacturing, February, 2000.
16. Jonscher, D. and Dittrich, K.R. An Approach For Building Secure Database Federations. Proceedings of 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, pages 24-35, Morgan Kaufmann, September 1994.
17. Kapsammer, E; Wagner, R.R. “The IRO-DB Approach Processing Queries in Federated Database Systems”, in Proc. of Eight International Workshop on Database and Expert Systems Applications DEXA’97, IEEE Computer Society Press, Toulouse, France, 1997.
18. Meng, W.; Yu C. T. “Principles of Database Query Processing for Advanced Applications”, Morgan Kaufmann Publishers, 1998.
19. Nural, S; Koksal, P; Ozcan, F; Dogac, A. "Query Decomposition and Processing in Multidatabase Systems", in Proceedings of OODBMS Symposium of the European Joint Conference on Engineering Systems Design and Analysis, Montpellier, 1996.
20. Osorio, L.; Antunes, C.; Barata, M. "Communication Infrastructure". In Infrastructures for Virtual Enterprises - Networking Industrial Enterprises (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, ISBN 0-7923-8639-6, 1999.
21. Rosenthal, A.; Sciore, E.; First-Class Views: A Key to User-Centered Computing, ACM Sigmod Record, Volume 28, Number 3, September 1999.
22. Silberschatz, A.; Zdonik, S. – Database Systems: Breaking out the box, SIGMOD Record, v. 26, n. 3, September 97.
23. Sinha, Alok. Network Programming in Windows NT. Addison Wesley, 1996.
24. Tuijnman, F.; Afsarmanesh, H. - Management of shared data in federated cooperative PEER environment. International Journal of Intelligent and Cooperative Information Systems (IJICIS), 2(4): 451-473, December 1993.
25. Wiedijk, M.; Afsarmanesh, H.; Hertzberger, L.O. - Co-working and Management of Federated Information-Clusters. Proceedings of the 7th International Conference on Database and Expert Systems (DEXA'96), Lecture Notes in Computer Science 1134, pp 446-455. Springer Verlag, September 1996.