# Distributed Database Support for a Concurrent Engineering Environment

**Hamideh Afsarmanesh, Michiel Wiedijk**
Computer Systems Department
University of Amsterdam
Kruislaan 403,
1098 SJ Amsterdam
THE NETHERLANDS

**Aureo Campos Ferreira, Norton Paim Moreira**
Mechanical Engineering Department
Universidade Federal de Santa Catarina
88040-900 - Florianópolis/SC
BRAZIL

**Abstract**:Computer Integrated Manufacturing involves collaboration and exchange of information among several groups of engineers. A Concurrent Engineering environment uses CIM technologies to handle shared information. Such applications are both data-intensive and computation-intensive. A natural architecture to support the management and sharing of information for this application is a network (federation) of heterogeneous and autonomous agents that are some loosely and some tightly-coupled. This paper presents a preliminary study in the design of a distributed CIM database for an aerospace industry in Brazil. In this effort, first some information modelling requirements are distinguished. Then, a federated information management system PEER, that handles the requirements,is described.

**Keywords** : Concurrent engineering, object-oriented databases, federated databases, co-operation network

## 1. Introduction

The current production process, especially in the case of aerospace industry, presents a huge variety of entities and concepts, and a small batch production, involving the use of several design and manufacturing technologies. Considering additionally the complexity of this process, a very large amount of information is manipulated, requiring the sharing of activities among many industry areas. The traditional production system follows a sequence of activities. Each activity, being executed by a specific enterprise area, defines both the peculiarities of the product and the production procedures. This practice could lead to a sequence of errors caused by deficient information that comes from other areas. The Concurrent Engineering approach, more than proposing that these activities are developed simultaneously or made compatible, focuses on the factors that affect the whole product development and the role they play in every specific activity. In Concurrent Engineering, a team of experts is gathered from all fields involved in the design, engineering, manufacturing and marketing of a single product and the team is made responsible for its successful production. The activities involved will be performed with close cooperation among all the team members. For instance, production engineers in the team are able to look at early designs, just as well as marketing experts.

In order to improve the productivity and quality in a Concurrent Engineering environment, at the Federal University of Santa Catarina, Brazil, an information management system is under development, where each CAx (e.g. CAD, CAPP, CAQ, etc.) technology has an information management server for its own speciality [1, 2, 3]. For example, if a designer needs manufacturability information to help him in the design of a part, he can access that information in the CAPP system. This approach is currently applied to the Concurrent Engineering environment development for a Brazilian aerospace industry. The main entity in a mechanical industry environment is the product. There is an enormous amount of detailed information describing a product. We have introduced and used the Feature Technology to model the manufacturing information such as processes, tools, thermochemical processes, tolerances, etc.

The model implementation is characterised by the information heterogeneity and its distribution through several industry areas. A federated database system is the natural architecture to support the management and sharing of information in this application. It can efficiently support a network of cooperative engineers. For information management, each member needs have a large degree of autonomy in structuring and organising his own

information (drawings, documents, analysis results, etc.), while he must also consider the needs of the other team members. For example, every team member is an expert in his own field, and deals with the detailed views of his work, however team members with other expertise need a less detailed view or a different structure defined in his information.

In such cooperative domains, an agent represents an application environment activity that typically runs on a single workstation. Every agent stores some data locally, performs some local computations, and needs access some remote data from other agents. Usually, several agents that perform the same kind of activity (e.g. production planning, design, etc.) construct a team. A team is represented as a cluster in the federation network. An agent in this network is heterogeneous in the sense that it develops independently and has its own private representation of the information. An agent is autonomous in the sense that it can decide to conditionally share a part of its information with other agents and keep another part private and explicitly for its own local use.

PEER is a federated object-oriented database management system designed and developed at the University of Amsterdam. In this paper, we give an overview of several modelling concepts developed in PEER [4, 5, 6, 7] that support vast and complex CIM application areas. PEER supports the autonomy of agents involved in the cooperation team allowing them to share and exchange information. This is achieved by a sophisticated schema derivation/integration mechanism, which supports importing remote information and restructuring and integrating it with the local information [6]. The sharing of information among team members in PEER is negotiated to preserve the referential integrities among the information of team members [4]. PEER also offers support for complex objects shared in the network. The subject of complex object-representation, -identification, and -boundary, and a linearization mechanism developed for complex object transformation into messages is fully described in [7,5]. Object identity and support for object naming is treated in [7]. A prototype implementation of the PEER federated system is developed in C that runs on UNIX, on a network of SUN workstations. PEER is partly implemented within the ESPRIT-Archon project (as the AIM information management system). ARCHON develops a system to integrate cooperating expert systems [8]. Within ARCHON, the object-oriented information management system supports the sharing and exchange of information among the complex expert systems. PEER has been tested at the control rooms of power distribution networks in two different industries, in Spain and England.

The remainder of this paper is organized as follows. Section 2 presents an approach to a Concurrent Engineering computational environment, under development in a Brazilian aerospace industry, and a first approach to the manufacturing information modelling. Section 3 presents an overview of the PEER architecture, and its data model and query language. Section 4 describes the object model and query language of PEER. Section 5 presents some PEER modelling concepts useful in a Concurrent Engineering environment, including schema integration and derivation, referential integrity in a network of PEERs, and complex shared object modelling and their linearization to interface to the application program environments. Finally, in Section 6, some conclusions and next steps are discussed.

## 2. A Concurrent Engineering Environment

Many approaches are made to the Concurrent Engineering implementation, and most of them are based on multifunctional teams. In this approach experts of different areas work together during the design phase to prevent mistakes. This practice however, solves only problems in the design phase. In order to help other production phases, such as process planning and scheduling, an abstraction approach must be used. The main objective in the entire production process is the improvement of productivity associated with quality and cost. To satisfy these goals, all information must be available within every phase of the process. To supply information and handle it over the whole production process, each phase must provide its information and query and

analysis mechanism to be used by other partners. This characteristic is modelled through the Computer Integrated Manufacturing tasks. As such, each CAx (CAD, CAPP, CAQ) must be a server, providing an information management model and language to support all needs of the clients.

### 2.1. Information Modelling

The main problem to implement this approach is the information heterogeneity, information complexity, and information interrelationships. Each production process phase uses different kinds of information like product definition, technological data, geometric data, algorithms, and process knowledge. Besides, each phase treats some specific private information. The information is distributed over different phases and must be carefully managed to avoid inconsistent redundancies.

An analysis of the production process information flow shows that the product is the main entity with which other manufacturing information is related. Marketing, design and manufacturing phases are involved in activities that all depend on the product definition. During these activities an enormous amount of information is generated and aggregated to the product definition. To use the product definition as an integration element, a common powerful model must be introduced for product representation. As such, the part model must integrate different kinds of information in a single description, so that this representation can work like an access key to manufacturing information. The Feature Technology is introduced to support this requirement by providing access to manufacturing information from any production process phase. Through the key feature a user is able to obtain the necessary manufacturing process information to produce a specific part.

Each feature represents the design function used by the designer at the product design stage. With this constructive element the part is built in a function assembly process. The feature manipulation is done through a library where the user may choose the best feature to represent his purpose. Figure 1 presents a part modeled by features. A feature can be defined by types (classes) using an object-oriented information management paradigm. This approach simplifies the representation and handling of features. As such, a feature is defined as an object with some attributes and methods. Multiple inheritance is defined among subclasses (subtypes) of features. In Figure 2, a description of feature is presented using an object-oriented approach. The feature model (functional information) associates technological information with abstraction concepts. In this figure, FEATURES are divided into PRIMITIVE and DETAILS. This abstraction is used to distinguish between the elements that constitute the body of the product and the details that are used within the body. The attributes of FEATURES are inherited by its subclasses. A STANDARDS object is a model of a set of kinds, for example a set of rules about some product, e.g. the thread direction. Any kind of attributes (relationships) may be defined between the STANDARDS and THR-DIR objects, as well as among different STANDARDS.

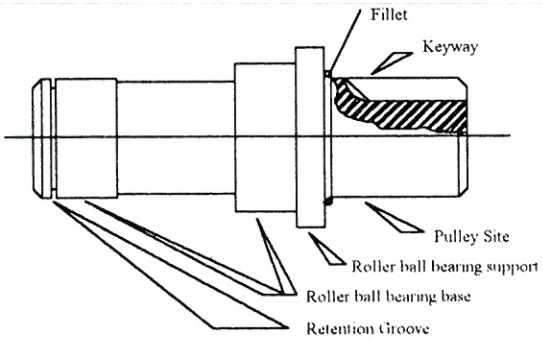


**Figure 1 . Design by Features**

A features object can be defined in terms of other objects. For example, finishing is an attribute of a thread's surface. Now consider FINISHING1 an instance of FINISHING and THREAD1 an instance of THREAD.Here the object FINISHING1 is a part of the definition of object THREAD1.

In mechanical parts' design, especially in aerospace industry, most of the elements are standardized, i.e. there are default values and relationships defined among them. The feature modelling supports the use of standards in the designed objects. For example, the object THREAD1 is an instance of DET-EXTERNAL, being simultaneously an instance of THR-RIGHT. As such, THREAD1 receives the attributes from both objects and among THR-RIGHT attributes, there are default values that characterize the standards. Similar to the way the information is represented for the above example, we can represent the process, quality, and cost information.

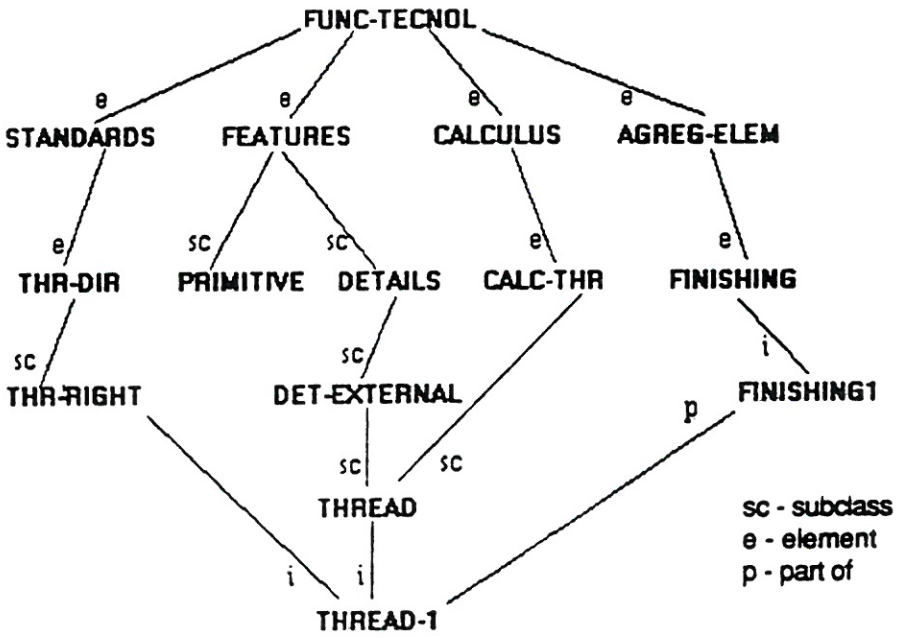


**Figure 2 Functional and Structural Information**

### 2.2. Aerospace Industry Application Development

The work under development in the Brazilian aerospace industry aims at designing a manufacturing information base to primarily support the design phase. It must be attached to the CAD system through a feature-oriented user interface. In order to support other production process phases, a distributed / federated information management system is needed to be developed that answers distributed queries over complex features. Each manufacturing phase must be able to access all the information available in the environment using a specific user interface [1, 2, 3].

The design of the information management system for the aerospace industry is currently under way. So far, an inside factory wide-range information gathering has been done. In every production process phase, several activities related to the manufacturing information are identified. By analysing the survey results we were able to design an early system with its entities and entity relationships. Currently the feature definition for a product family is undertaken. It is the basis on which to identify other related information and model the relationships among them. The information modelling will follow the structure presented earlier.

### 2.3. Database Management System Requirements

In order to implement the information system, previously described, the database system needs satisfy the following requirements:

a) distributed and redundant information management;

b) object-oriented approach to represent many different kinds of information;

c) allow complex objects representation;

d) support complex queries in a distributed environment; and

e) provide information consistency over the distributed environment.

### 3.The PEER Architecture

PEER [6] is a federated database architecture that supports the management and sharing of information in a network of cooperating agents. The design of PEER is based on a global data model, the object-oriented PEER data model and a global language, the object-oriented query/update PEER language. By using the global model, agents' heterogeneous schema representations are made homogeneous. However, the homogeneity of schema representation does not address the semantic interrelationships (loose or tight integration) that may exist among the data and knowledge of different agents. These interrelationships are established systematically and incrementally through a set of derivation/integration operations defined for distributed schema management of PEER. Several schemas co-exist in every PEER, namely, the local, export, import, and integrated schemas. The integrated schema can be interpreted as one user's global classification of objects that are classified differently by the schemas in other databases.

The architecture of the PEER system is shown in Figure 3. The object-oriented database model and database language used in the design of PEER are extensions of the 3DIS and its language 3DIS/ISL [9, 7] that are briefly described in the next section. The query language and its capabilities play a major role in specifying the schema integration among the PEERs, as discussed in [6, 7].As represented in Figure 3, the nucleus of a PEER system for an agent in a CIM cooperation network is the extended 3DIS model and language and a layer of modelling constructs and operations that is defined specifically to represent the generic abstract data types used in engineering and industrial manufacturing industry. The distributed object management of PEER is handled by a layer on top of the nucleus, that supports the distributed schema management and the complex object sharing. For the exchange of complex objects as single entities, between PEERs and between a PEER and an application program, a linear representation is generated [4]. The last layer of the architecture is the PEER interface to support the access and communication of users and application programs to the agent

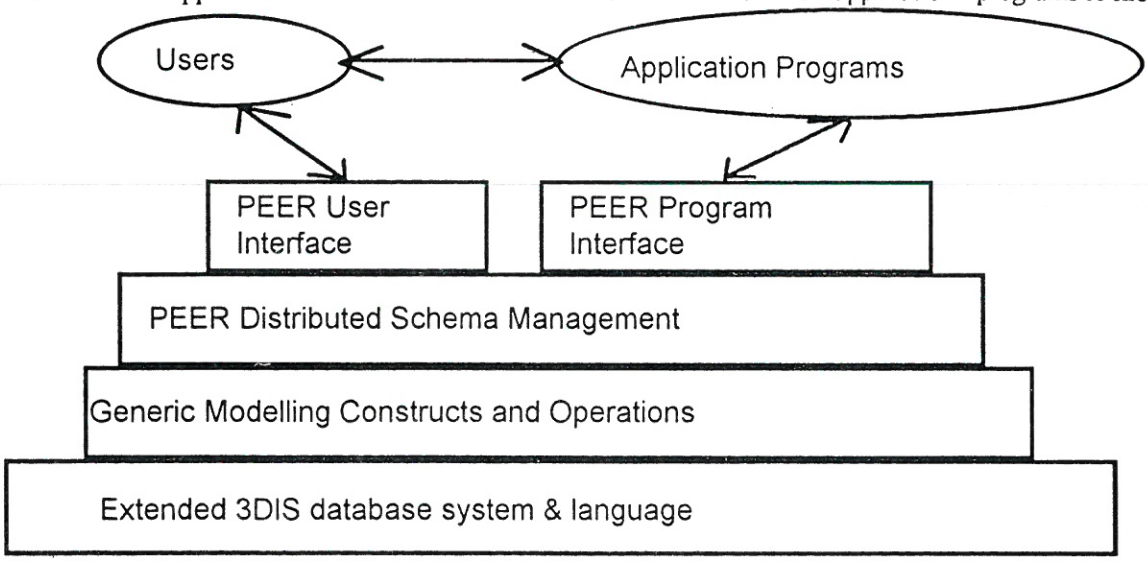


**Figure 3. Architecture of an Individual PEER Agent**

## 4. PEER Object Model and Query Language

The data model of PEER is primarily based on the 3DIS [9,10], an object-oriented database model. However, this model has been vastly extended to support more semantics, to represent the specific concepts and entities of the manufacturing industries, and to support the kernel structure for the distributed architecture of PEER. The PEER data model supports the fundamental abstractions of instantiation, generalization, and aggregation. Any identifiable piece of information is uniformly represented as objects. Different kinds of objects are distinguished by the set of structural and non-structural (data) relationships defined on them. The PEER data model represents atomic, composite, and type objects. Atomic objects are strings of characters, numbers, booleans, text, etc. usually referred to as *values* in other systems. Composite objects are non-atomic entities and concepts of the application environments and can be decomposed into further objects. Mapping objects, usually referred to as *attribute* in other documents, are a special kind of composite objects. Mappings can be further decomposed into their domain type object, range type object and inverse mapping object. Mappings can be single-valued or multi-valued and represent both the descriptive characteristics of an object as well as its association with other objects. A type object is a structural specification of a group of atomic or composite objects. It denotes a collection of database objects called its members (instances). Mappings (attributes) common to members of a type are defined by the type definition. The 3DIS supports multiple inheritance; the subtype/supertype relationships defined among types form a Directed Acyclic Graph (DAG).

The query language designed for the 3DIS, called the 3DIS/ISL [7], is simple but powerful. This language is the base for the design of the PEER database language. The retrieval of information is mostly based on asking for binary relations among objects, since any relationship defined among objects is always broken into a number of binary relationships defined among those objects [9]. Therefore, the retrieval of information is always done through a triple with the domain-object, mapping-object, and range-object being its three elements, where one or more of these elements can be replaced by a "?" to represent the query. A major difference between the PEER and most other database models is in the uniform treatment of objects and values. Instead of considering an attribute of an object to have a certain value (asymmetric relationships between objects and values), PEER always considers two objects to be related through a mapping. This symmetry is reflected in the query language, as illustrated by the basic example: retrieve (part-22, ?, ?). This query retrieves all the attribute values of the object part-22, as it retrieves all other objects in the database that are related to part-22; for instance, part-22's *subparts* (part37 and part51) that are other objects in the database. PEER extends the query language of 3DIS by allowing path expressions as the mapping element of a retrieve command. In the query: retrieve (?, designer.education, "mechanical engineering") all objects will be retrieved that are related by *designer* to a designer object whose *education* background is "mechanical engineering".

## 5. PEER Support for Cooperative Applications

PEER offers a number of facilities that are useful in cooperative applications such as Concurrent Engineering and CIM environments that are briefly described in this section. Information management in a network of agents is supported by PEER through distributed schema management including integration and derivation of local information and information that is available from other PEER agents. Reliability is supported by enforcing the integrity of relationships among local and remote information. For the interaction with end-user and applications PEER offers support for object naming, complex objects and interface support through linearization of complex objects.

### 5.1 Schema Integration / Derivation

The local schema in each PEER agent specifies the type structure of the information stored locally at that agent. Part of the local information can be made available to other PEERs, by specifying one or more export schemas, that define a view on the local information. An export schema in a PEER can restrict the exported information to a subset of local information to make it available to other PEER agents. Other PEER agents can import these export schemas as their import schemas. This makes the information of other PEERs available locally.

PEER offers two approaches to integrate the local information and remote information (of other PEERs) imported into import schemas. The first approach is to define an integrated schema, derived from local and import schemas. An integrated schema defines a single uniform type structure on the information available locally and remote. Since the integrated schema is defined local to a site, different PEERs may establish different correspondences between their schema and other sites' schemas, thus there is no single global schema for the network. Queries in PEER are always evaluated in the context of a schema. The default context for a query is the integrated schema, but another context can be specified in the query. Distributed query evaluation in PEER is discussed in [7].

### 5.2. Remote Referencing

Another integration approach is directly interrelating local information and remote information. Figure 4 shows a relationship (Catalog92, has-entry, Refrigerator-B403), where Catalog92 is in the OM1's PEER and Refrigerator-B403 is in the OM2's PEER. The local schema of a PEER can be extended by extending the local type definitions with mappings that can refer remote objects defined in an import schema. The integrity of objects that contain references to remote objects however, can be violated when these are deleted by a remote PEER. For example, the deletion of Refrigerator-B403 violates the referential integrity. PEER preserves referential integrities by requiring a reference access right for remote references of a certain imported type. For example, OM1 must first request a reference access right on the type APPLIANCE-R (of which Refrigerator-B403 is an instance) that is represented in an import schema obtained from OM2. Together with this access right, OM1 can also ask for an action level (delete condition) to be followed by OM2, in the case that OM2 decides to delete an instance of APPLIANCE-R to which OM1 has a remote reference. There are three possible action levels that OM1 can request: warning that an object is deleted, warning that an object will be deleted after some preset time, deletion only after remote permission. These delete conditions are of different severities and must be negotiated by the two PEERs.

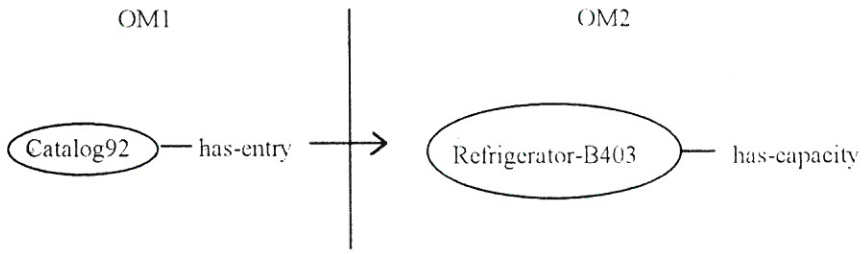


**Figure 4. Relationships between Objects in Different PEER Objectbases**

### 5.3. Complex Objects

A complex object represents a single entity that makes a cluster of objects together. As an example of a complex object, a bike can be defined to be composed of a frame and its front and back wheels, and wheels are composed of a rim, a nave and spokes. Complex objects are defined in PEER by a template. A template for a complex object type is defined by a type object-identifier, that is the *root* of the template, and a *derivation expression*.

The derivation expression is defined similarly to the map derivations described in [6], as an ordered set of mappings defined among different types, starting at the root and continued on to the next type using a mapping on the path. For example, the complex object template can be defined for Bike-CT as follows:

```
Bike-CT: BIKE,
         {has-frame,
          front-wheel.{has-spokes,has-rim,has-nave},
          back-wheel.{has-spokes,has-rim,has-nave}}
```

A complex object is identified by a tuple, with the object-id of its root object as the first element and the name of its template as the second element. For instance, for a bicycle bike-14 we can identify its Bike-CT complex object by the tuple (bike14 , Bike-CT).

Complex objects in PEER are supported for their retrieval and update. Modification of complex objects has to be described as both changing relationships between intermediate objects and the process of deletion and creation of affected objects. Deletion of intermediate objects is part of the process of changing a complex object. For example, deleting a nave means changing the description of the wheel that uses it. If other references may exist to a nave object, then the nave object may only be deleted after having checked that no such references exist.

### 5.4. Complex Object Linearization

A common linear representation of complex objects in our belief is more useful than a full integration of some programming language(s) with the objectbase. The requirements made by the industrial engineering applications are such that the disadvantages of an exclusive support for just one language outweigh the advantages. The schema derivation primitives, in combination with various options available in making linearizations, provide a lot of flexibility in converting from an abstract object definition in a database to a concrete data structure in a programming environment; the effect is that most of the retrieval operations required by application programs can easily be specified and supported.

The conversion from abstract to concrete data structure is addressed in PEER using the complex object definition. A retrieval of a complex object is interpreted as making a linear concrete representation of the object graph denoted by the complex object specification. One example of several different linearization techniques supported by PEER [3],is given below. A complex object can be represented by a list of lists of mapping-value pairs. When the range element of a mapping refers to another object, then the linearization can represent the object cluster at once. For example, the Refrigerator B-403, with a subpart of Cooling system F-111 is linearized as follows:

```
((name "Refrigerator B-403")
 (designer John)
 (subpart ((name "Cooling System F-111")
           (designer Mary))) )
```

### 6. Conclusions

Although the Concurrent Engineering environment development is in its inception , its basic premises are presented in this paper. Several information modeling requirements for supporting concurrent engineering applications are distinguished. Then, the PEER federated object management system is described addressing those information management requirements. This paper presents a first attempt towards a cooperative work between two of the research teams involved in the ESPRIT basic research CIMIS.net project ECLA 004:76102. So far, we have studied a Brazilian aerospace industry application environment, determining ,on the one hand, the requirements and on the other hand the information management and modelling concepts to satisfy them. To develop a CIM database for this industry the next step is to design PEERs representing the complex application information and implementing different CIM activities.

## REFERENCES


1. MOREIRA, N., **An Information Modelling Proposal to the Manufacturing Integration and Concurrent Engineering**. Master Thesis. Universidade Federal de Santa Catarina, 1993 (in Portuguese).
2. MOREIRA, N. and FERREIRA, A., **Production System Integration: A Feature Approach**. VII International Symposium in Computer Applications. Antofagasta, Chile, 1992.
3. MOREIRA, N., FARIA, P. and ROSA, E., **An Information Modelling for CIM.** I Congreso IberoAmericano de Ingenieria Mecanica, Madrid, Spain, 1993.
4. TUIJNMAN, F. and AFSARMANESH, H., **Sharing Complex Objects in a Distributed PEER Environment**. 13th Int. Conf. on Distributed Computing Systems, IEEE, May 1993.
5. TUIJNMAN, F. and AFSARMANESH, H., **Distributed Objects in a Federation of Autonomous Co-operating Agents**. Int. Conf. on Intelligent and Co-operative Information Systems, IEEE/AAAI, May 1993.
6. AFSARMANESH, H., TUIJNMAN, F., WIEDIJK, M. and HERTZBERGER, L.O., **Distributed Schema Management in a Co-operation Network of Autonomous Agents**, Proceedings of the 4th IEEE International Conference on Database and Expert Systems Applications (DEXA), Lecture Notes in Computer Science 720. SPRINGER-VERLAG, September 1993.
7. TUIJNMAN, F. and AFSARMANESH, H., **Management of Shared Data in Federated Co-operative PEER Environment**, INT. JOURNAL OF INTELLIGENT AND CO-OPERATIVE INFORMATION SYSTEMS, Vol. 2, No. 4, December 1993
8. T. Witttig (Ed.) **ARCHON: An Architecture for Multi-agent Systems**. ELLIS HORWOOD, 1992
9. AFSARMANESH. H. and MCLEOD, D., **The 3DIS: An Extensible Object-Oriented Information Management Environment**. ACM TRANSACTIONS ON INFORMATION SYSTEMS, 7:339-377, October 1989.
10. AFSARMANESH, H., KNAPP, D., MCLEOD, D. and PARKER, A., **An Extensible Object-Oriented Approach to Databases for VLSI/CAD**, Proceedings of the 11th International Conference on Very Large Data Bases, 1985, pp.13-24. Reprinted in S. Zdonik and D. Maier (Eds.) Readings in Object-Oriented Database Systems, MORGAN KAUFMANN PUBLISHERS, 1990..