# Enterprise Management Network Architecture
# The Organization Layer

**Michel Roboam, Mark S. Fox and Katia Sycara**

**CMU-RI-TR-90-22**

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

November 1990



Michel Roboam is currently visiting scientist in the Center for Integrated Manufacturing Decision Systems and is sponsored by the AEROSPATIALE Company (France).

This research has been supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

# Table of Contents

# List of Figures

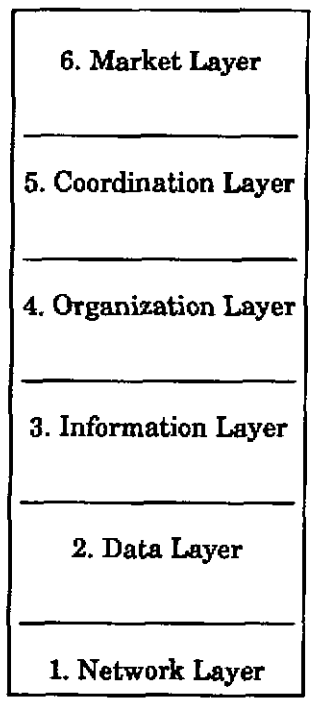
# List of Schemata

## Abstract

Achieving manufacturing efficiency requires that many groups that comprise a manufacturing enterprise, such as design, planning, production, distribution, field service, accounting, sales and marketing, cooperate in order to achieve their common goal. In this paper we introduce the concept of Enterprise Management Network (EMN) as the element to facilitate the integration of distributed heterogeneous functions of a manufacturing enterprise. The integration is supported by having the network first play a more active role in the accessing and communication of information, and second provide the appropriate protocols for the distribution, coordination, and negotiation of tasks and outcomes. The Enterprise Management Network is divided into six layers: Network Layer, Data Layer, Information Layer, Organization Layer, Coordination Layer, and Market Layer. Each of these layers provides a portion of the elements, functions and protocols to allow the integration of a manufacturing enterprise. The Organization Layer plays the central role in the EMN architecture by defining the model of a decentralized structure, and identifying its major components to be supported by the other layers. Methods of analysis and representation are defined at this layer to model manufacturing organization. This representation provides a definition of the decentralized agents and of their interactions. They will be the basis for identifying coordination and negotiation protocols to support distributed problem solving.

# 1. Introduction

In the first report [38], we have introduced the concept of Enterprise Management Network (EMN) architecture to support the integration of the manufacturing enterprise. We defined this architecture as a multi-layers system supporting both distributed knowledge base and distributed problem solving. We have identified six different layers:




The *Network Layer* provides for the definition of the network architecture (figure 1-1). At this level, the nodes are named and declared to be part of the network. Message sending (or message passing) between nodes is supported along with synchronization primitives (such as "blocking"). Security mechanisms are also provided such as message destination recognition.

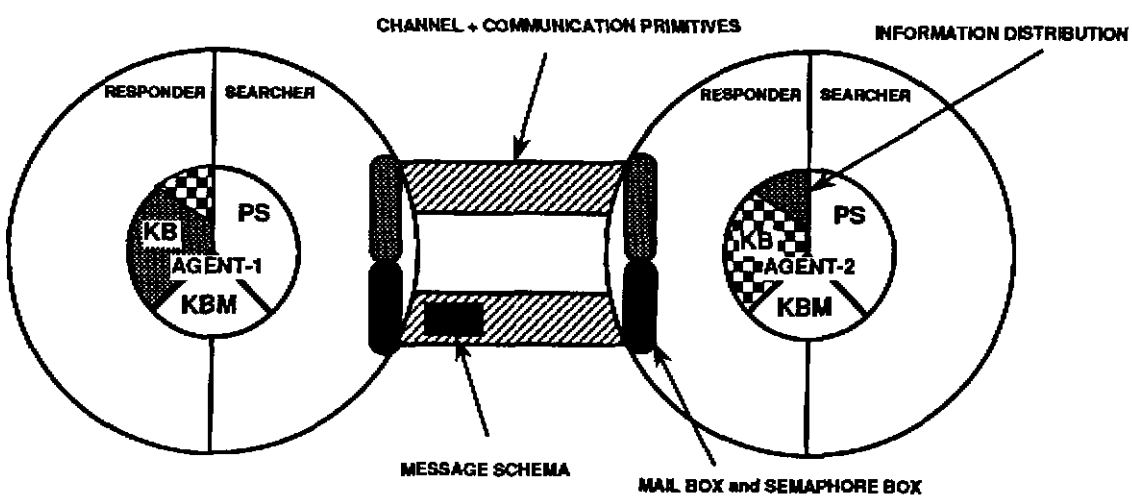



**Figure 1-1:** Network Layer Implementation Example

The *Data Layer* provides for queries and responses to occur between nodes in a formal query language patterned after SQL [7, 8] (figure 1-2).

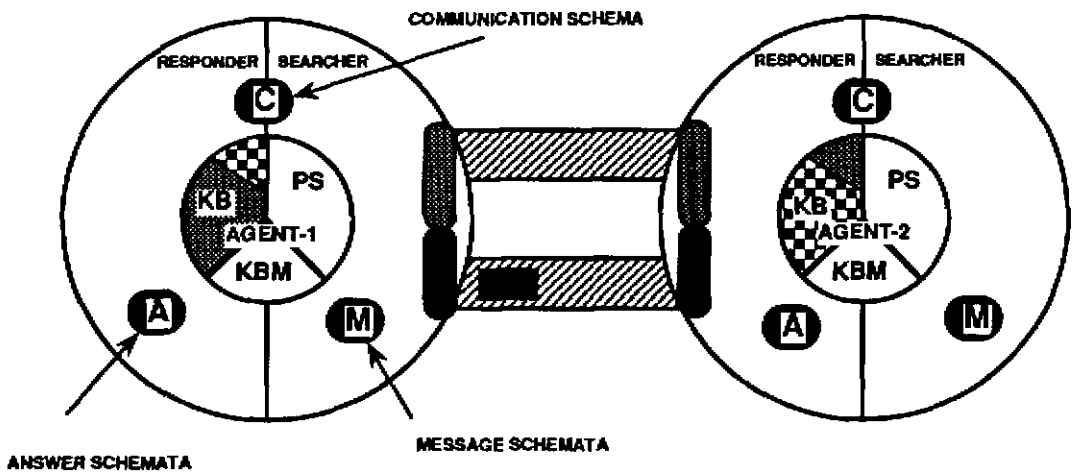



**Figure 1-2:** Data Layer Implementation Example

The *Information Layer* provides "invisible" access to information spread throughout the EMN (figure 1-3). The goal is to make information located anywhere in the network locally accessible without having the programs executed locally know where in the network the information is located nor explicitly request its retrieval. This Layer also includes information distribution focussed on data classes, keywords and content and security mechanisms such as agent blocking and unblocking and schemata locking and unlocking. All the information queries expressed at this layer use the query language defined at the data layer.

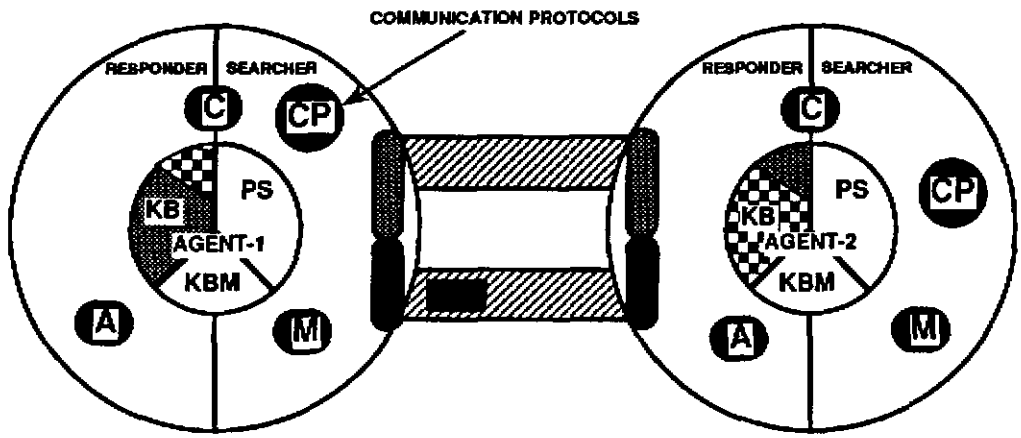



**Figure 1-3:** Information Layer Implementation Example

The *Organization Layer* provides the primitives and elements (such as goal, role, responsibility and authority) for distributed problem solving. It allows automatic communication of information based upon the roles a node plays in the organization. Each EMN-node knows its responsibility, its goals, and its role in the enterprise organization.

The *Coordination Layer* provides the protocol for coordinating the activities of the EMN-nodes through negotiation and cooperation mechanisms.

The *Market Layer* provides the protocol for coordination among organization in a market environment. It supports the distribution of tasks and the negotiation of change and the strategies to deal with the environment. In this report, we present in details the fourth layer of this architecture (Organization Layer). The aspects covered by this report mainly concern the distributed problem solving supported by the EMN architecture. In the previous report [38], we presented the problems of distributed knowledge base and how they are covered and supported by the EMN architecture (As another example for distributed knowledge base, we can refer to [25, 1, 34]). The implementation of this architecture and of the communication system is described in [39]. This Layer provides the support for distributed problem solving by defining the type of interactions we can have between EMN-nodes. In the next layers (Coordination and Market), we will define the protocols supporting these type of interactions.

We define at first the concept of distributed problem solving by identifying its characteristics such as coupling, grain size or degree of cooperation.

Then, after presenting the content of an EMN-node, we define the Organization Layer of our EMN architecture. This level is the platform on which we build the structure to support the distributed problem solving between the EMN-nodes. In addition, this layer provides information about the EMN-nodes to complete the three first layers of our architecture described in [38].

## 1.1 Enterprise Management Network Capabilities

The optimization of the manufacturing enterprise can only be achieved by greater integration of activities throughout the production life cycle. Integration must not only address the issues of shared information and communication, but how to coordinate decisions and activities throughout the firm.

Achieving manufacturing efficiency requires that the many groups that comprise a manufacturing enterprise, such as design, planning, production, distribution, field service, accounting, sales and marketing, cooperate in order to achieve their common goal. Cooperation can take many forms:

- **Communication of information** relevant to one or more groups' tasks. For example, sales informing marketing of customer requirements, or production informing the controller of production performances.
- **Feedback on the performance** of a group's task. For example, field service informing design and manufacturing of the operating performance of a new product.
- **Monitoring and controlling** activities. For example, controlling the execution of operations on the factory floor.
- **Assignment** of new tasks. For example, a new product manager signing up production facilities to produce a new product.
- **Joint decision making** where groups of "agents" have to negotiate and cooperate in order to achieve their tasks (which can be antagonistic or not). For example, an inventory manager and a scheduler negotiating to define the manufacturing activity.

An Enterprise Management Network is viewed as the "nervous system" of the enterprise, enabling the functions described above. It is more than a network protocol (e.g., MAP) in that it operates and

participates at the application level. Its philosophy is different in terms of participation and structuring. Such a system must be defined in such a generic way that it can be integrated with all kinds of applications an enterprise can use. The following describes the capabilities provided by the Enterprise Management Network:

- **Information routing**: given a representation for information to be placed on the network and a representation of the goals and information needs of groups on the network, the information routing capability is able to provide the following:
  - Static routing: transfering information to groups where the sender and the receivers are pre-defined.
  - Dynamic routing: transfering information to groups which appear to be interested in the information. This is accomplished by matching a group's goals and information needs to the information packet.
  - Retrospective routing: reviewing old information packets to see if they match new goals and information requirements specified by a group.
- **Closed loop system**: Often, the communication of information results in some activity, which the initiator of the communication may be interested in. The EMN will support the providing of feedback in two modes:
  - Pre-define feedback: operationalizes pre-defined information flows between groups in the organization. For example, production providing feedback to sales on the receipt of orders.
  - Novel feedback: Providing feedback for new and novel messages.
- **Command and control**: Given a model of the firm which includes personnel, departments, resources, goals, constraints, authority and responsibility relations, the EMN will support these lines of authority and responsibility in the assignment, execution and monitoring of goals and activities. In particular, it will manage the distribution of information and the performance of tasks.
- **Dynamic task distribution**: Supporting the creation of new organizational groups and decomposition, assignment and integration of new goals and tasks, contracting and negotiation are examples of techniques to be supported.

## 1.2 Distributed Systems Definition

The Enterprise Management Network Architecture provides the elements and functions to define, implement and support a **distributed system**. A distributed system is a system with many processing and many storage devices, connected together by a network.

### 1.2.1 Distributed Systems Advantages

Potentially, this makes a distributed system more powerful than a conventional, centralized one in two ways:

- First, it can be more reliable. Every function can be replicated several times. When a processor fails, another can take over the work. Each file can be stored on several disks, so a disk crash does not destroy any information. We call this property **fault tolerance**.
- Second, a distributed system can do more in the same amount of time, because many computations can be carried out in **parallel**[1].

[1]Note we are talking about **large grain parallelisms not connection machine** style parallelism.

### 1.2.2 Decentralized Systems top-level description

"In a very general terms, a system is said to be distributed when it includes several geographically distinct components cooperating in order to achieve a common distributed task" [2]. But this definition is not true for all the domains. If we consider, for example, games involving two players, the aim of each one is to win the game. So the two agents of this decentralized system do not cooperate, they compete (they cooperate in playing the game, i.e., they follow some rules, but they compete about sub-goals-winning).

The set of nodes in the system is usually organized according to various domain dependent topologies. Decentralized systems in every day life come from a wide variety of areas, e.g., a business firm, a system for traffic control, etc.

The processing nodes in a decentralized system may all be identical in their capabilities or they may each possess specific skills. Whatever the configuration is, in a decentralized system both the control (process) and the knowledge can be distributed throughout the system.

In actuality, there is a range of approaches for decentralized architecture, from an almost centralized system to a distributed system with a centralized planning and control element, to a distributed system with a distributed, hierarchical group of control elements, to a fully distributed, "flat" system in which each element is responsible for its own control.

Moreover, the organization amongst the elements may either be static, remaining the same as time elapses, or dynamic, adapting itself as the requirements of the environment needs it. In any case, the processing nodes, or agents, contain knowledge about themselves and their environment, and a logical capability to work on that knowledge. In other words, the agents have a memory and a processor.

But we have a limitation for the memory aspect: we cannot have in a decentralized agent all the needed information for completely autonomous running (the concept of bounded rationality [41]). This means that we must acquire some information from the other agents of the decentralized system: the agent must **communicate**. Bounded rationality implies that both the information a computing agent can absorb and the detail of control it may handle are limited.

### 1.2.3 Distributed System Dimensions

Since almost any real world system is decentralized and, moreover, open in nature [21, 29, 22], the spectrum of categories for decentralized system is infinite. But we can use two attributes to categorize decentralized systems along two continuous dimensions: the degree of **coupling** among the agents (or nodes), and the **grain size** of the processors of the agents.

**Coupling** is a measure related to links between the agents in the system. Loose coupling means that information exchange amongst the agents is limited. In loosely coupled systems the agents spend most of their time in local processing rather than in communication among themselves. Tight coupling, therefore, indicates that there is no practical physical limit on the bandwidth of the communication channel between the agents. Because of excessive communication, tight coupling also indicates that the concept of bounded rationality of computing does not completely apply [41].

The **grain size** of the processors measures the individual problem-solving power of the agents. In this definition, problem-solving power amounts to the conceptual size of a single action taken by an agent visible to the other agents in the system. If the grain is coarse then the processing nodes are themselves rather sophisticated problem-solving systems with a fair amount of complexity. In coarse-grained applications, the distribution may be characterized to be, therefore, at the task level. Fine grain often indicates that the individual processors are functionally relatively simple, i.e., they do not exhibit any "intelligence" per se, and that their number in the system is substantial. Thus, the distribution in fine-grained applications is at the statement level as opposed to task level distribution.

#### 1.2.3.1 Parallel Distributed Processing Systems

Decentralized, fine-grained systems with tight coupling are often referred to as parallel distributed processing systems [26, 9, 6, 21]. The processing aspect emphasizes concurrent execution of functionally decomposable tasks.

The objective in parallel distributed processing systems is usually load balancing of shared informational and physical resources. In distributed processing systems, the computational or syntactic motivations for decentralization are highlighted:

- speed,
- performance/cost,
- modularity,
- availability,
- scalability,
- reliability,
- extensibility,
- flexibility.

Although the current trends in the cost and availability of computer hardware would suggest that adding up enough conventional, low cost processors would result in an immense overall computing power with a reasonable investment, this has not proven to be the case. On the contrary, it has been recognized that a severe bureaucracy "bog-down" effect in multiprocessor systems calls for totally new architectural strategies to operate on the higher degree complexities in routine problem solving.

#### 1.2.3.2 Distributed Problem Solving Systems Definition

As the opposite of PDP, we have distributed problem solving systems. These are defined informally as networks of loosely coupled, relatively coarse-grained, semiautonomous, "artificially intelligent" asynchronous problem-solving agents, cooperating (or competing according to the domain) to fulfill their global mission. Asynchronous means that the agents are thought to function concurrently [26]. Cooperation means that because no node is capable of solving the entire problem by itself; the nodes have to work as a team and exchange knowledge about the tasks, results, goals, and constraints to solve the global problem or set of problems.

The **degree of cooperation** between the nodes in a decentralized problem-solving system may vary. On one extreme, the nodes may all be pursuing a common goal and be thus fully cooperative.

This assumption is often referred to as the benevolent agent assumption. On the other extreme of the cooperation continuum, the nodes are nonbenevolent, i.e., they are self-interested, possessing conflicting goals and preferences. Thus, a process of negotiation to resolve the conflicts becomes crucial.

Decentralized problem-solving architectures with the last set of characteristics mentioned above are often categorized as nearly decomposable systems. In **nearly decomposable systems**, the interactions among the components are weak but not negligible. The emphasis in studying coordination within nearly decomposable systems is on dealing with the problems arising from restricted communication and bounded rationality. In the case of decentralized problem solving, the semantic motivation to pursue decentralization are thus addressed in terms of complexity, possibility and natural decomposition.

### 1.2.4 Distributed Systems capabilities

As mentioned above, a distributed system has to be capable of parallel execution and of continuing in the face of single-point failures, so it must have:

- **Multiple processing elements** that can run independently. Therefore, each processing element, or node, must contain at least a CPU and memory[2].
- There has to be communication between the processing elements, so a distributed system must have **interconnection hardware** which allows processes running in parallel to communicate and synchronize.
- A distributed system cannot be fault tolerant if all nodes always fail simultaneously. The system must be structured in such a way that **processing elements fail independently**.
- Finally, in order to recover from failures, it is necessary that the nodes keep **shared state** for the distributed system.

### 1.2.5 Distributed Systems Problems

All these advantages of distributed systems cannot be satisfied due to the complexity of designing such systems [31, 20, 24, 29, 17]. Some examples of system problems are:

- the amount of **interconnections** and risk of failure,
- the **interferences** between processes,
- the problem of **propagation of effects** between processes,
- the **information inconsistency** due to its duplication,
- the **effects of scale** due to the dimension of distributed systems and
- the **partial failure** of one processor that can perturbate the other ones.

The EMN architecture we define in this paper covers most of these aspects. The utilization of Artificial Intelligence techniques to support communication and distribution offers help in solving most of these problems, especially propagation of effect, and information inconsistency.

---

[2]Note that multiple EMN-nodes may share a processor

## 2. Enterprise Management Network Node

The Enterprise Management Network links together two or more application nodes (EMN-nodes) by providing the "glue" that integrates the manufacturing enterprise through architectures and mechanisms to support decision making at all levels of the organization. For example, the CORTES system [18] is composed of an uncertainty analyser, a planner, a scheduler, a factory model and two dispatchers responsible for several machines (figure 2-1). Each is defined as an EMN-node.

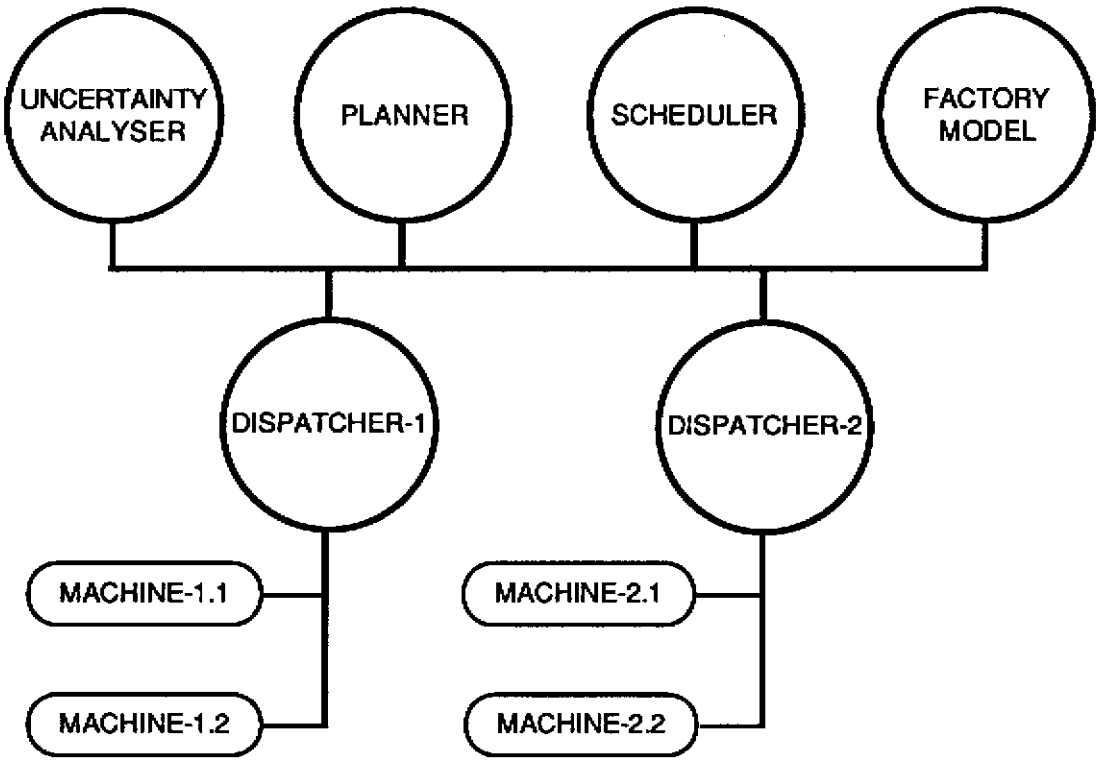



**Figure 2-1:** Example of decentralized system

Each EMN-node consists of the following subsystems[3] (figure 2-2):

- Problem Solving Subsystem,
- Knowledge Base,
- Knowledge Base Manager, and
- Communication Manager.

The **Problem Solving Subsystem** represents all the rules and functions which allow the EMN-node to solve any problems related to its domain. The local execution cycle is triggered either by the internal transactions generated during local problem solving, or by external events forwarded to the EMN-node by the Communication Manager.

Each EMN-node contains a locally maintained **Knowledge Base** to support its problem solving. It is composed of entities (or objects) which may be either physical objects (products, resources,

[3]Currently implemented in CommonLisp

operations, etc) or conceptual objects (customer orders, process plans, communication paths, temporal relations, etc). The knowledge base is expressed as CRL[4] schemata [28].

The **Knowledge Base Manager** manages information exchanges between the problem solving subsystem and the knowledge base, maintains the consistency of the local knowledge base, and responds to request made by other EMN-nodes. In the Enterprise Management Network, knowledge and data may be distributed throughout the network. It is the philosophy of the system that knowledge does not have to be available locally in order for it to be used by the EMN-node. Therefore, knowledge, in the form of schemata, fall into one of two classes: that *owned* by the knowledge source which must be stored locally, and knowledge *used* by the knowledge source, in which the original is stored at another EMN-node and a copy is stored locally.

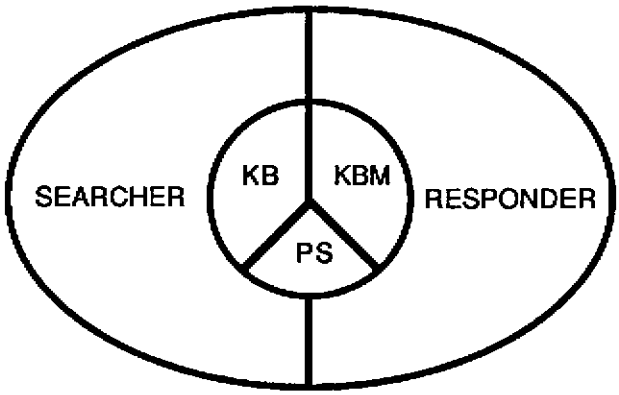



**Figure 2-2:** The elements of an EMN-node

A problem that arises in supporting the exchanges between the problem solving subsystem and the knowledge base is the unavailability of schemata locally. The problem solver often refers to knowledge that cannot be found locally, but may be found in another EMN-node's knowledge base. At the time of reference, the problem solver may or may not know where in the Enterprise Management Network the knowledge resides. It is the responsibility of the Knowledge Base Manager to "hunt down" the missing knowledge and to respond to like requests from other EMN-nodes. To accomplish this, the Knowledge Base Manager has as part of it a **Communication Manager**. It both manages the search for information in the EMN and responds to like requests from other EMN-nodes. To perform these activities, the Communication Manager has two modules:

- The **searcher** corresponds via message sending with other EMN-nodes. The searcher peforms two tasks: searching for knowledge not available locally, and the updating of knowledge changed and owned by the EMN-node.
- The **responder** answers messages originating from other EMN-nodes' searchers, and updates the local knowledge base according to updating messages.

The communication manager manages four types of interaction:

- ***Triggering***: information that triggers the node's processing.
- ***Dynamic retrieval***: Requests for information not available in its knowledge base and

[4]CRL stands for Carnegie Representation Language.

necessary to perform its task. This information needs appear during the internal activity (processing) of an EMN-node.

- ***Updating information***: When an EMN-node, as the owner of some schemata, modifies these schemata, the searcher dispatches the modifications to other EMN-nodes that have local copies of these schemata. The responder may or may not update a local copy depending on the usage at the receiving EMN-node. Being the owner of a schema means, the EMN-node is the only one allowed to globally modify the content of a schema. But each EMN-node having a local copy of a schema can locally modify the content of that schema.
- ***Transaction request***: Similar to remote procedure calls.

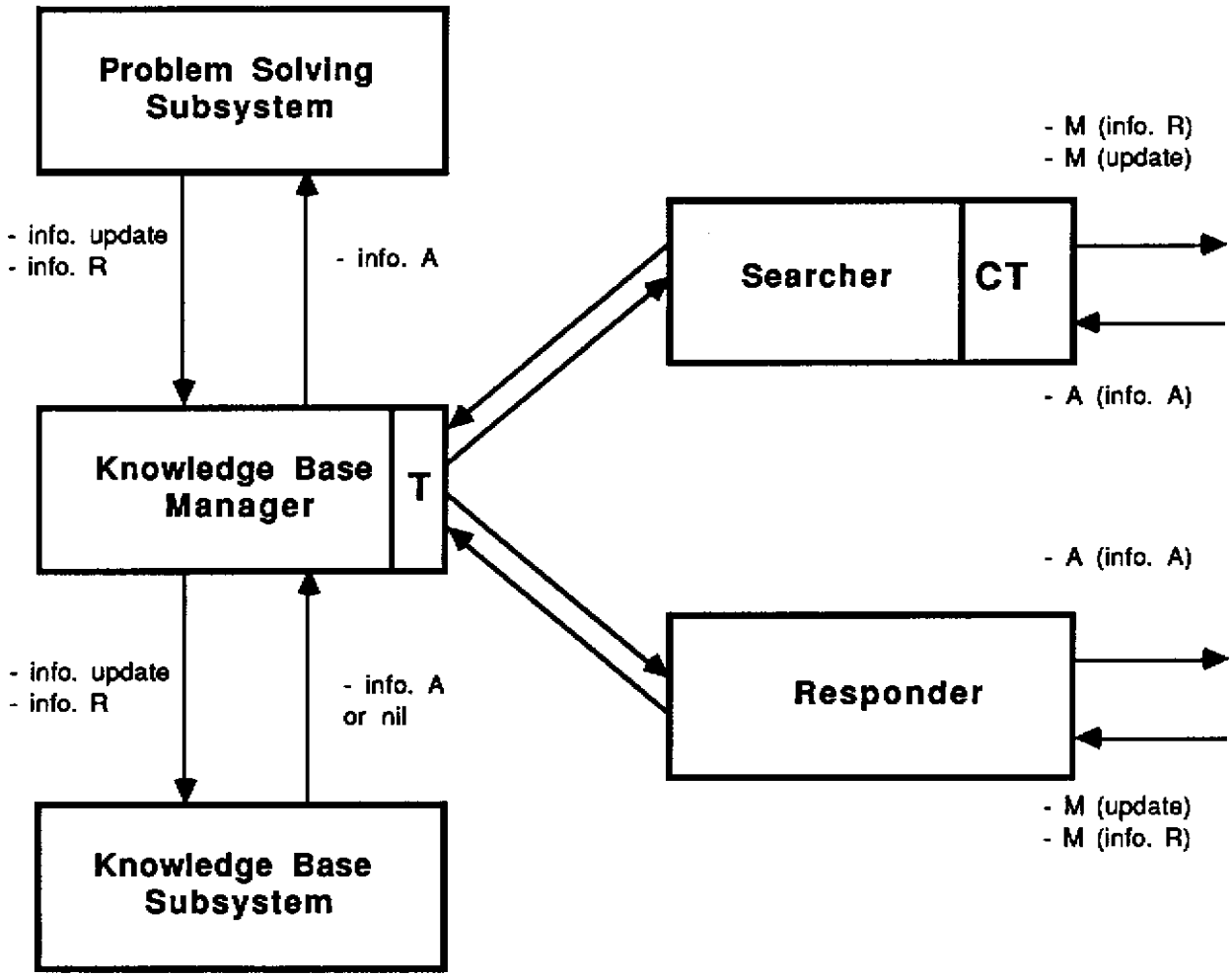



**Figure 2-3:** Information exchanges overview

We summarize all these exchanges between the modules of an EMN-node in figure 2-3. This figure shows the different types of information sent and received by each module (M stands for Message, A stands for Answer, R stands for Request, T stands for Translator and CT stands for Correspondance Table). We will discuss in the next sections the content of these informations.

The three upper layers of the Enterprise Management Network architecture are defined in the remaining sections. Each layer provides further detail on the functionality and operation of EMN-nodes. To illustrate the specific content of these layers, we will take an example. We will consider a decentralized system composed of three agents, connected by a network. Each agent has a specific Problem Solving subsystem (PS) and a specific Knowledge Base subsystem (KB). We also assume that the three first layers of the EMN architecture have been implemented in each EMN-node (figure 2-4). We will extend this example by adding the specific schemata, functions and protocols provided at the organization layer.

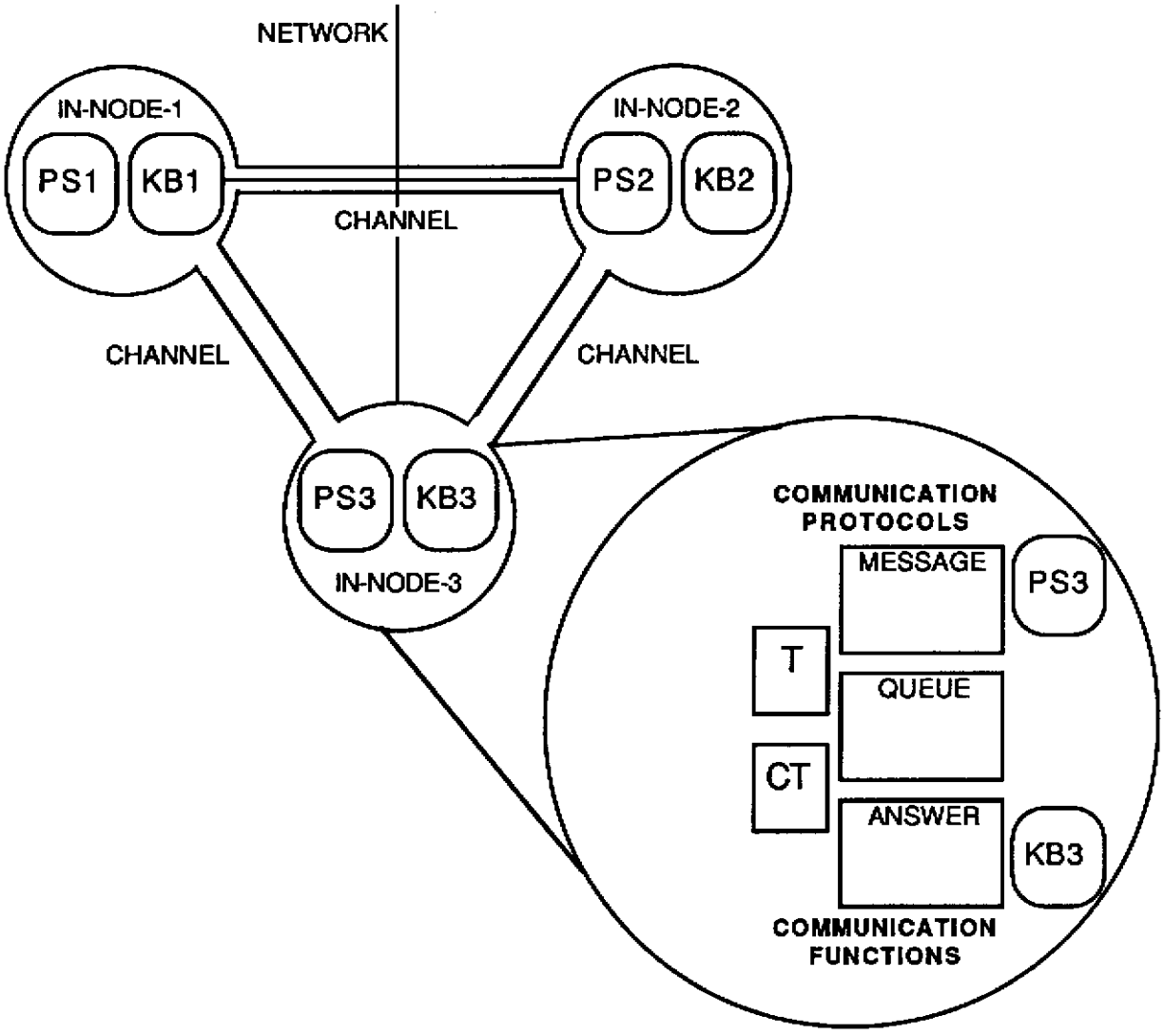



**Figure 2-4:** Decentralized system example

# 3. Organization Layer

The Organization Layer provides the primitives that define an agent's goals, roles, responsibilities and authority in an organization. These primitives are used to support distributed problem solving, that is the definition of both structure and the support of different methods of coordination, and to determine to whom information is to be communicated automatically.

Our approach to modeling an organization is to start with its structure [32]. The three main aspects are:

- ***Physical***: all the physical resources of the organization, such as the machines, the personnel, the tools, etc.
- ***Decisional***: all activities related to the control of the physical system which need a decision to be taken. The main caracteristic of these activities is the possibility of multiple choices.
- ***Informational***: the control of the physical activities by the decisional system is done by exchanging information between them. Information exchanges are also present in each systems. We include in the informational system all the information processing such as the Material Requirement Planning.

We add to this structural information, the lines of authority, goals, roles and responsibilities of each organizational entity. The model is then further refined with the information flows that are necessary to support decision making, and the temporal horizon over which decision are to be made or actions performed.

Our modeling methodology utilizes the GRAI [11, 36] and GIM [36, 37] graphical modeling tools as a means of specifying an organization[5]. The advantages of using such graphical tools are in the clarity of the conceptualization of the real environment. They provide a strict formalism of the different systems we intend to model. These models, once created, will allow a better understanding of what the inter-actions and hierarchical links are.

The interactive graphical specification of organization is automatically translated into the underlying organization, information, and network layer schemata and protocols. These schemata allow the definition of links and inter-actions between the EMN-nodes. Mechanism are defined to complete, using the content of these schemata, the communication schema, to create channels and decision frames, to define, using the informational links, what are the updating sequences and potential users of the information. In addition, the hierarchical structure of EMN-nodes will be defined through the hierarchy of decision frame we define in the organizational model. This hierarchy will be used at the coordination layer to define coordination and negotiation protocoles.

In figure 3-1, we define the different sequences and functionalities supported by the Organization Layer of the EMN architecture. Starting from a specific enterprise, the first step performed at the Organization Layer is to build, using a graphical editor, a model of the organization of this enterprise. This model uses both GRAI and GIM modeling formalisms. The GRAI model structures a manufacturing organization according to a decision point of view. It defines the production management of a manufacturing organization. The GIM model supports the data modeling. It

---

[5]GIM: GRAI-IDEF0-MERISE or GRAI Integrated Methodology has been created in our PhD thesis [36] and in the Europeen ESPRIT Project 418 Open CAM System [15, 35]

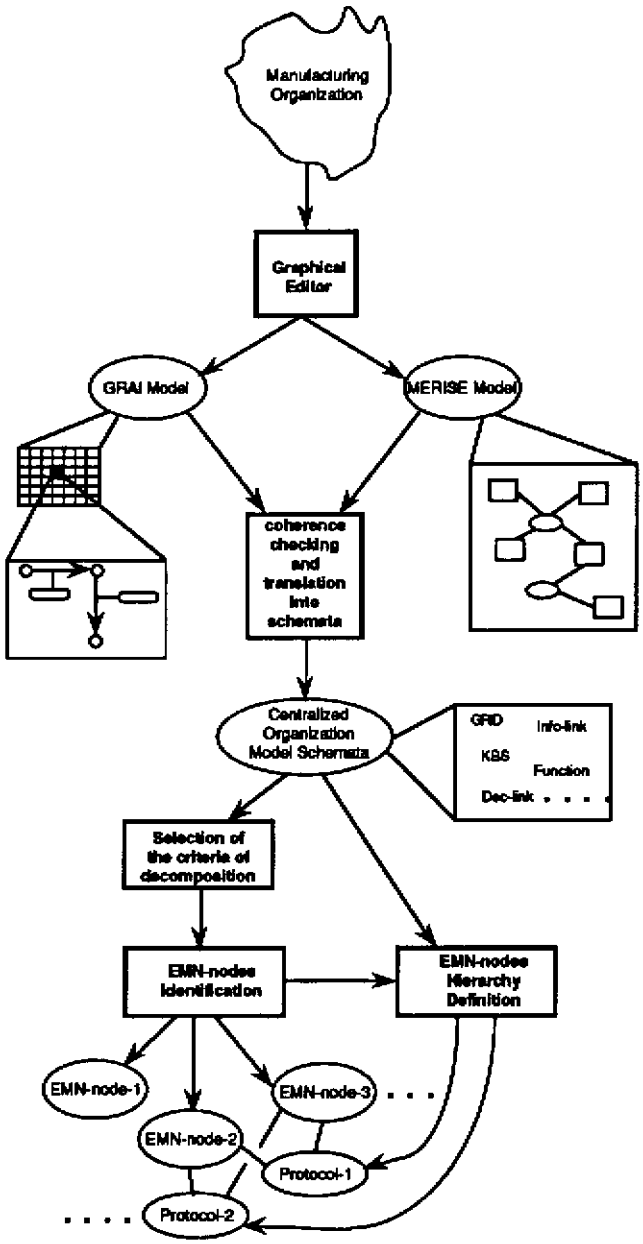



**Figure 3-1:** EMN architecture instantiation

identifies the entities and their relationships. After a consistency checking between these two models (using coherence tools defined in paragraph 3.2.6), they are automatically translated into the underlying organization and information schemata. These schemata support the definition of the centralized structure of the manufacturing organization. The next step is to split up this centralized structure into a decentralized one. For that purpose, we select one or several criteria of decomposition. The definition of the decentralized structure of the enterprise is derived from these schemata according to specific criteria. The instantiation of the different EMN-nodes, of their interactions and content is defined using the GRAI grid, the information and decision links, etc. The instantiation of an EMN-node means the creation of a decentralized agent and the initialization of the different schemata defined at the three first layers of the EMN architecture in this agent. In

addition, the used, shared and owned information specific to this EMN-node are identified, the channels between this new EMN-node and the already existing ones are created. Then, the hierarchy and interactions between this new EMN-node and the already existing ones are identified and the corresponding coordination and negotiation protocols are applied to support the distributed problem solving among the different EMN-nodes. The definition of the EMN-nodes and their interactions are determined using both GRAI and GIM models. After defining one or several criteria of decomposition, we can derive using the different rules specified in paragraph 3.2.2 the structure of the decentralized production management system of a specific organization. This structure is also supported by schemata we present in section 3.3. The negotiation and coordination protocols are defined at the coordination layer. But their application is determined by the interactions identified at the organization layer.

In this section, we define the concept of a modeling tool. Then, we present the modeling tools of the GRAI and GIM methods and their application to define the structure of the organizational model. In the last part, we present the schemata to support the implementation of this organization layer.

## 3.1 Modeling tool selection

Modeling is a difficult task; the domain we intend to model is complex. The goal of modeling is not to simplify but to better represent the complexity in order to support analysis [30]. Simon suggests analyzing a problem by splitting it up into "action and goals" [42]. Titli suggests decomposing and aggregating hierachically a structure in order to identify modules and analyse their inter-actions [49]. We have selected the GRAI methodology for modeling organizations. Our choice is based on an existing classification [35, 15] of the current methods and tools which use the following criteria:

- What aspects of the system modification life cycle is supported by the methodology,
- What abstractions of the system the methodology is able to model, and
- What types of subsystems can be modeled.

For our purposes, we can ignore the life cycle modeling criterion[6].

The complexity of a manufacturing organization is great, thereby precluding its modeling in complete detail. Consequently, a methodology must support the modeling of an organization at different levels of ***abstraction***. Three abstraction levels have been identified:

- The **conceptual** level defines a system in terms of entities, activities, and their relationships.

---

[6]In modifying an organization, there are five recognized phases that make up the system life cycle:

- **Analysis phase**: we study the situation of the existing system and we try to define its inconsistencies. The Constraints and goals are also defined.
- **Design/Specification phase**: the functional specifications, the basic framework and the general behaviour of the futur system are defined.
- **Development phase**: based on the choices made at the previous step, this phase concerns the technical choices and the realization of the prototype of the futur system.
- **Implementation phase**: integration and adaptation of the prototype in its real environment.
- **Operating phase**: utilization, control and readjustment of the implemented system.

- The **organizational** or **structural** level models both the system's structure, such as departmental hierarchies, authority relations, etc., and the modeling of technologies being used such as network and database types.
- The **realizational** or **physical** level defines the physical implementation of the system defined at the previous level. Choices for software packages and hardware components are made.

Organizations can be viewed in many ways, each having different representations, methods of design and analysis, and separate criteria they must satisfy. The ***sub-systems*** of a manufacturing system we wish to model are:

- The **Physical** subsystem which includes the men, the machines, the material flows, etc. of a manufacturing system.
- The **decisional** subsystem which controls the physical system by triggering and readjusting its activities. We introduce the concept of operating level which links the decisional and physical level (it includes the control of machines, the security procedures, etc.).
- The **informational** subsystem corresponds to all the information and information processing which can occur between or inside of the two previous systems.

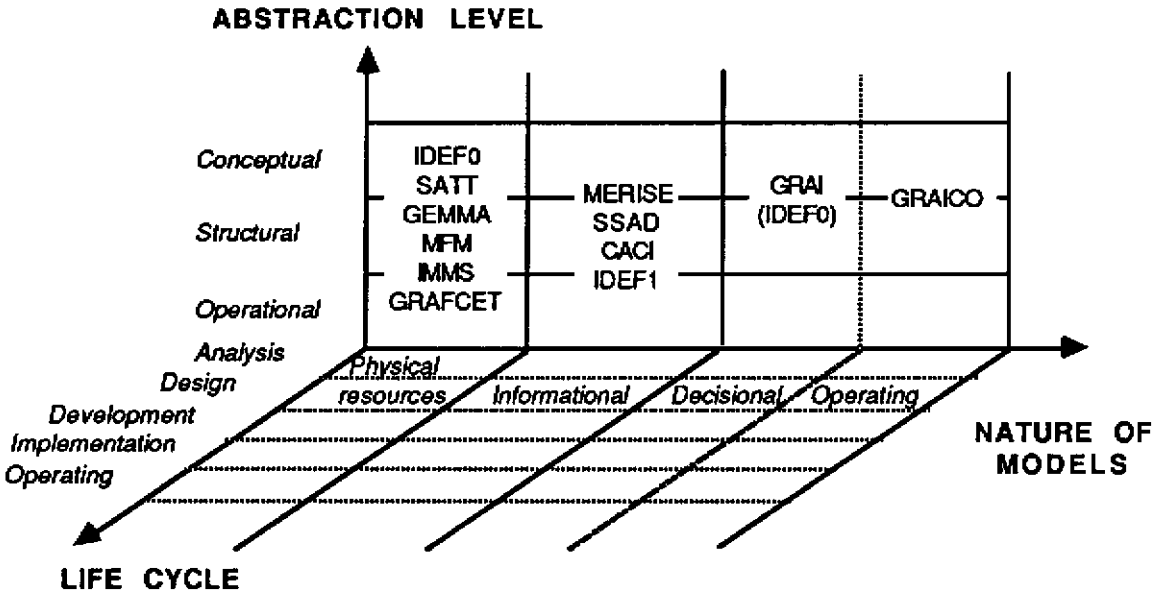



**Figure 3-2:** Methodologies typology

These elements allow to define a methodology typology, we represent in figure 3-2. Other criteria can be added to this typology such as the pragmatic, semantic and syntactic characteristics of the methodology tools where:

- the **syntactic** aspect covers the problems of vocabulary,
- the **semantic** aspect covers the problems of structure, and
- the **pragmatic** aspect covers the "problem solving" power of these tools.

## 3.2 Enterprise Modeling

In this section, we define the content and use of the tools used to acquire the description of the organizational model of the Enterprise Management Network (EMN) architecture. The first graphical tools we describe are the GRAI [11, 36] tools. Then we define the data modeling tool of the MERISE method [47, 48] which uses the entity/relationship model originaly created by Chen [5]. These two models span both aspects of problem solving and knowledge: the GRAI model describes the enterprise's decision making processes and supporting knowledge, and the MERISE data modeling tools define the data structure used in the decision processes. In the last part of this section we propose two coherence tools to support the integration of both GRAI and GIM models.

### 3.2.1 The GRAI Methodology

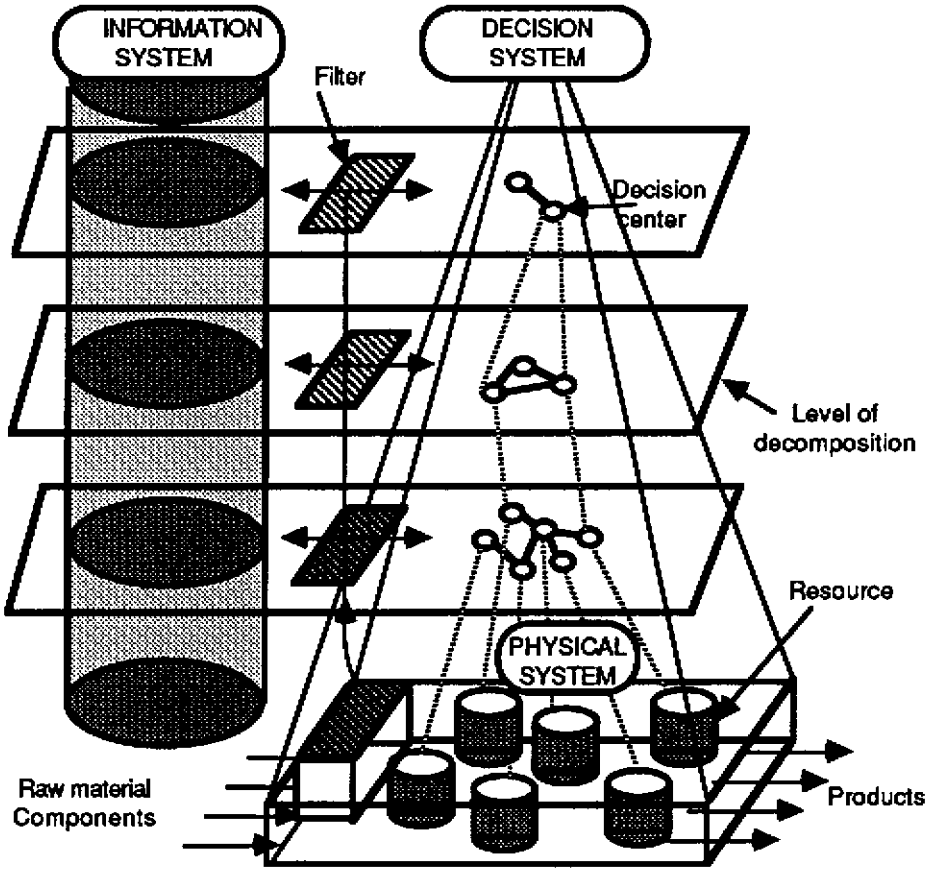



**Figure 3-3:** Global conceptual model of the GRAI method

The GRAI methodology approaches the problem of modeling complex enterprises by viewing them as being composed of the following systems:

- The physical subsystem which represents the machines, tools, men, products, components, etc. of a manufacturing organization. Its purpose is to transform the raw material, parts, components, etc. into products the company can sell.
- The decision subsystem drives the physical subsystem to perform the orders. It is defined as a hierarchical structure composed by a set of decision centers.

- The informational subsystem is the link between the two previous subsystems. All information exchanged, manipulated, transformed, created, etc. are part of this subsystem.

Figure 3-3 provides a graphical depiction of an enterprise's systems from the GRAI perspective.

A decision system can be decomposed into *decision centers* at different levels of the enterprise's hierarchy. Tasks are passed among decision centers in the form of *decision frames* which define the goals, decision variables, rules, etc. The elements of a decision center are depicted in figure 3-4.

These two conceptual models define the concept behind the modeling tools of the GRAI method. They introduce the notion of system, hierarchical decomposition, decision center and decision frame.

In the next section we describe the tools available in GRAI for acquiring and instantiating a specific enterprise model. These tools are restricted to modeling the decisional subsystem and parts of the other two subsystems relevant to the decision processes.

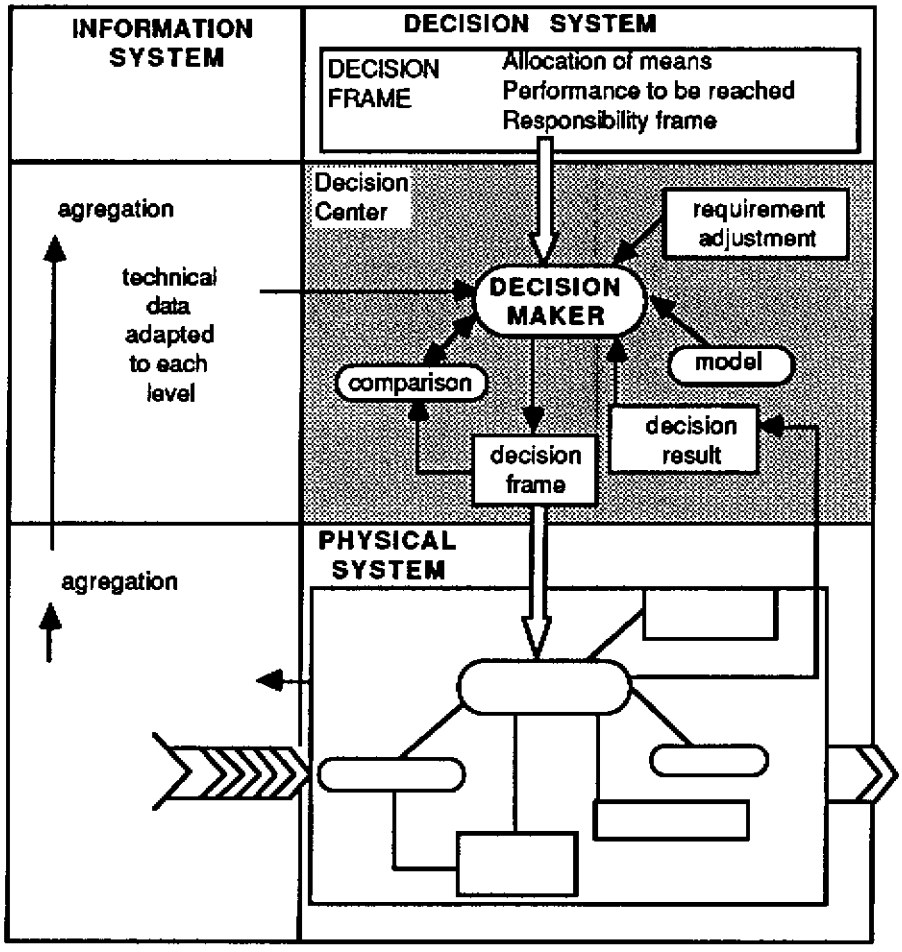



**Figure 3-4:** Structure of a Decision Center

### 3.2.1.1 The GRAI method modeling tools

GRAI has two graphical tools for modeling decision subsystems: **GRAI grid** and **GRAI nets**. The **GRAI grid** (figure 3-5) provides a hierarchical representation of decision activities that spans the entire decision system. The grid has two axes:

- The horizontal axis indicates the **functions** of a production management system. For example, planning, purchasing, supply, quality control, engineering, etc.
- The vertical axis defines a **temporal decomposition** of these functions, defined by two parameters:
  - The **Horizon** which is the duration of which a decision is valid (for example, establishing a budget for one year => H = 1 year).
  - The **Period** is the time after which you revise your decision (for example, I make a schedule for the week and I readjust it every day => H = 1 week, P = 1 day).

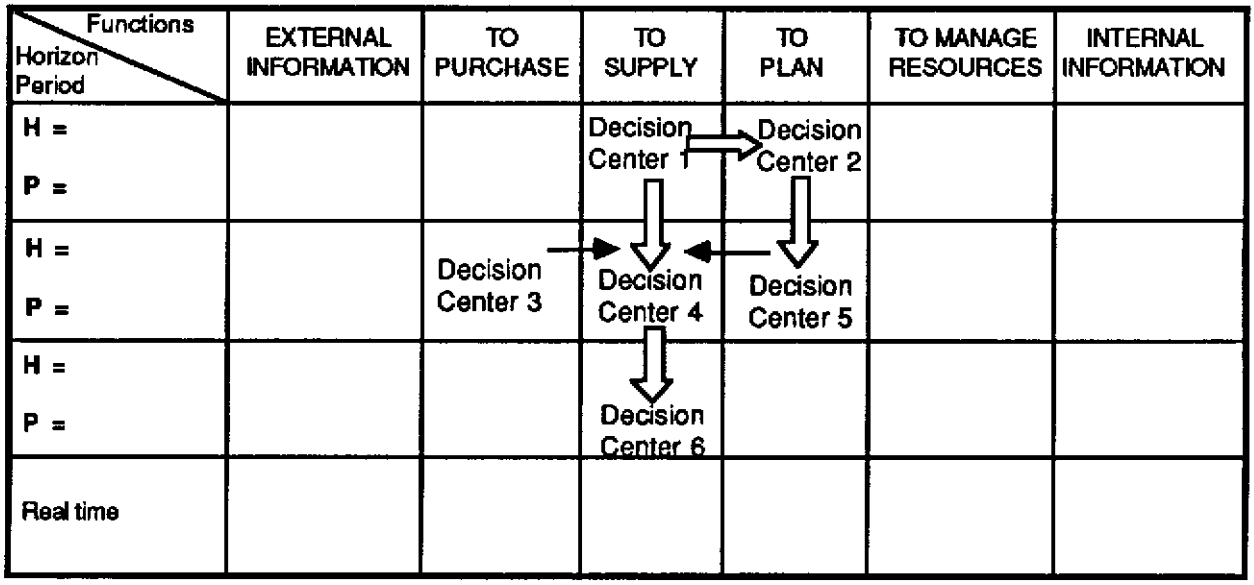



**Figure 3-5:** GRAI grid

Each "box" in the grid defines a *decision center* (for example, "to make the Master schedule", "to make the schedule", "to define the supplying parameters", etc.). Decision centers can be linked as follows:

- The **information link**, drawn with a single arrow, represents the transmision of information between two decision centers (for example, the engineering decision center provides the process plan to the scheduling decision center).
- The **decision frame**, drawn with a double arrow, defines the goal, decision variables and rules transmission. It defines the hierarchical task allocation link between two decision centers.

Figure 3-6 is an example of a GRAI grid. In this example, four decision levels are defined: (1 year, 3 months), (1 month, 1 week), (2 weeks, 1 day) and Real time. Four different functions have been taken into account: to purchase, to supply, to plan and to manage resources. The two columns: internal information and external information are just information supports providing knowledge about the source of information used by the Production Management System and which are not part

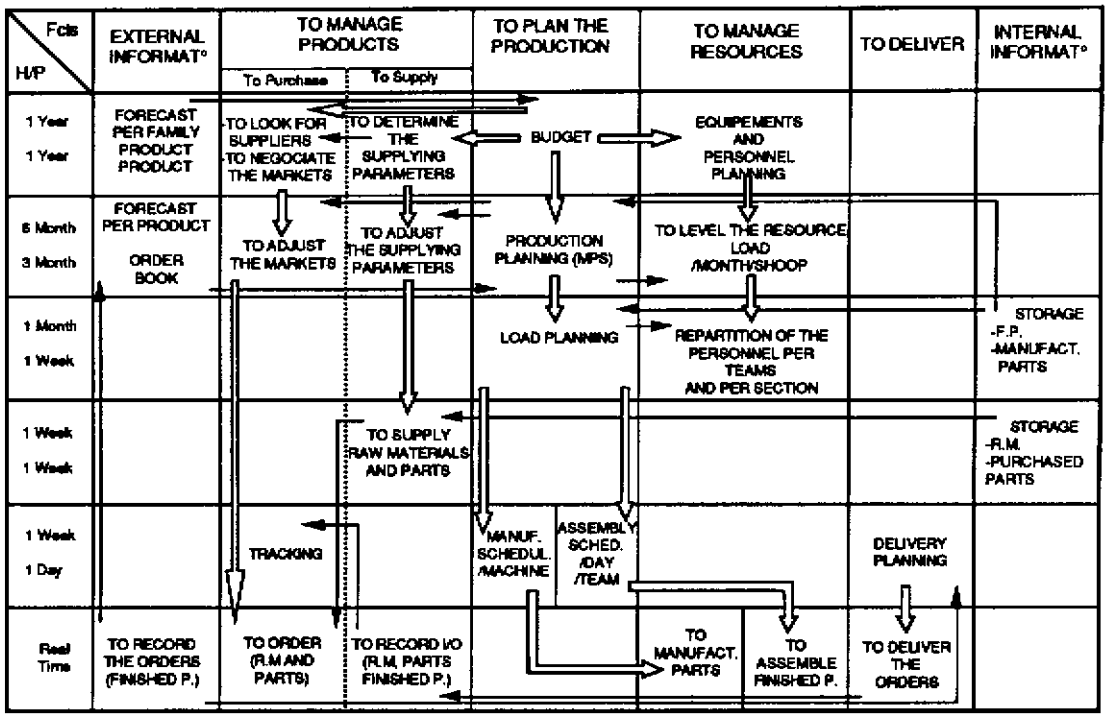



**Figure 3-6:** GRAI grid example

of this system. A GRAI grid is read from the top to the buttom. In the example we start from the "master schedule" part of the planning function and performed every three months with a one year horizon. Based on this "master schedule", the supply function defines at the same H/P level the supplying parameter, the purchase function negotiates with suppliers, and the resource management function determines the planning for the men and machines. The master schedule uses the forecasts as a basic input. Then weekly (P= 1week), a "load planning" is determined based on the master schedule and adjusted with the real orders. According to the part availability, provided by the supplying function, this load planning is adjusted. Its horizon is one month.

Each "box" of the grid is decomposed into a GRAI net (figure 3-7) (or several, depending on the level of detail needed). A **GRAI net** (figure 3-8) defines the sequence of activities performed in a decision center, and the information, resources, etc., used.

The decomposition process begins by splitting a decision center into two or three macro activities. This first level is also called a macro GRAI nets. At least one of the activities must be a decisional activity (implying a choice). Then, each of these activities can be decomposed into another GRAI net we call micro GRAI net. This hierarchical decomposition of activities is equivalent to what we can find in other structured methodology such as IDEF0.

A distinction is made between decision activities and execution activities. The first type implies that a choice is to be made according to some goals and the values of "decision variables". Each decision activity uses some knowledge, possibly in the form of rules. Decision activities are drawn with vertical arrows in the GRAI net. The execution activities imply no choice. They are information processing activities and are drawn as horizontal arrows in the GRAI net.

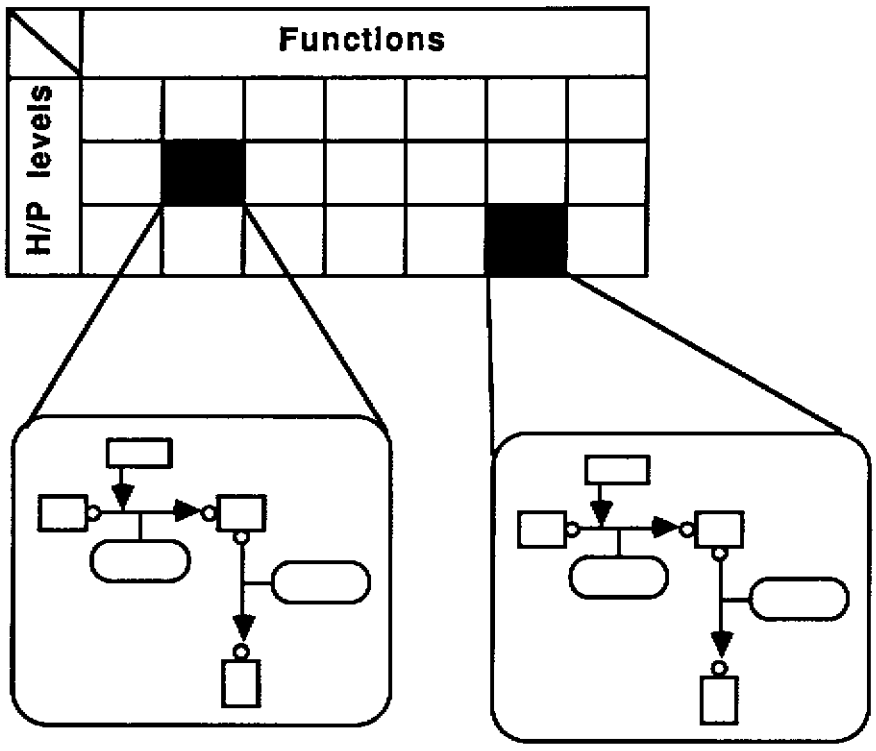



**Figure 3-7:** GRAI grid decomposition in GRAI nets

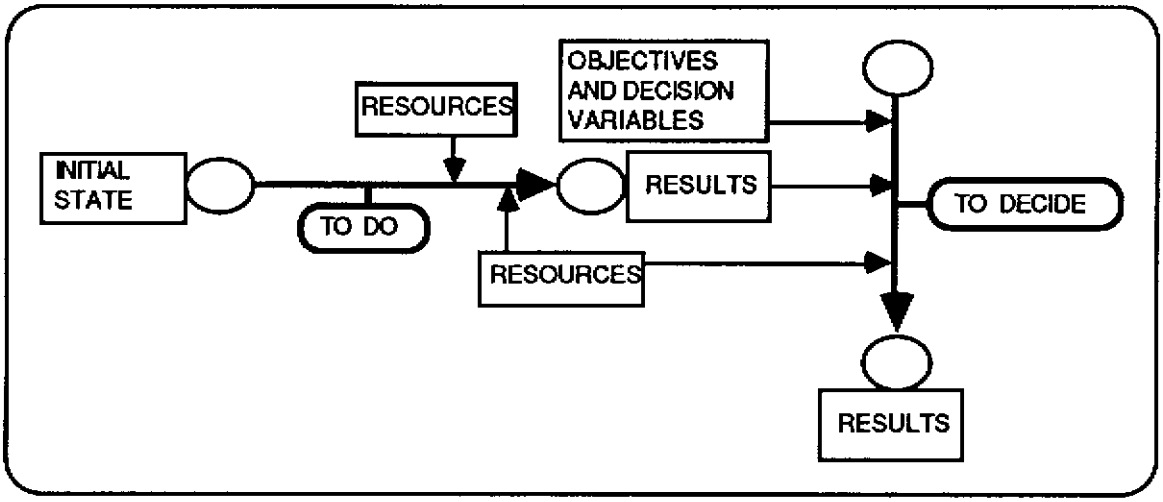



**Figure 3-8:** GRAI net

We give an example of GRAI net in the figure 3-9. This example represents the macro GRAI for the "dispatching" decision center. Two different activities have been identified: "to update schedule" and "to select next order". The first activity is an execution activity. The previous schedule is updated according to what has been performed in the shop floor. In the example, workstation 7 has completed its order and waits for the next one. The purpose of the "to select next order" activity is to select the next order. Based on the updated schedule, taking into account the shop floor status, the

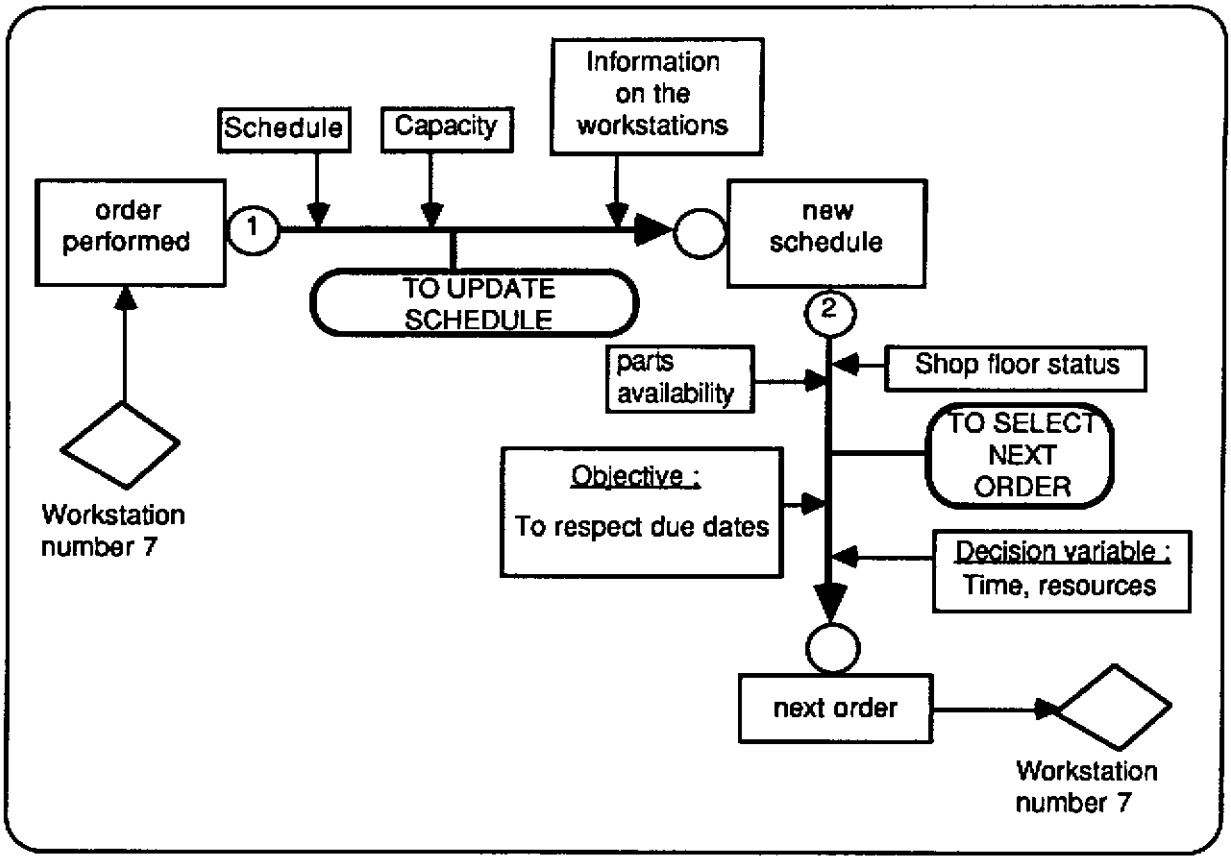



**Figure 3-9:** GRAI net example

part availability and the feasibility of the schedule, this activity selects from the list of orders allocated to workstation 7 the next one to be executed. This decision is made by trying to satisfy the due date and start time of each order.

### 3.2.2 Using the GRAI Model

#### 3.2.2.1 EMN-nodes identification

We use the GRAI grid to identify EMN-nodes (figure 3-10). An organization can be divided in many ways; it can be decomposed by decision center, groups of decision centers, by function, etc., each corresponding to an EMN-Node. Once identified, channels and network layer attributes can be defined and instantiated.

Consider a Production Management System (PMS), that can be structured according to following functions:

1. **Resource management**: This function provides manufacturing with the "resources" it needs at the right "time". These include, technical (machine) and human (personnel) resources. This function is divided into two sub-functions: technical resource management and human resource management.
2. **Product management**: This function provides the manufacturing activity the "products" it needs at the right "time". These include, parts, raw materials,

components, etc. that are used, manufactured, supplied, etc. This function is divided into 2 sub-functions:

- Supply: Determine the needed quantity of "products" and the date of this need for the manufacturing activity.
- Purchase: Acquire needed products from suppliers.

3. **Planning function**: This function synchronizes manufacturing activities. It plans and schedules the production of the "products" using "resources" of good quantity and at the right "time".

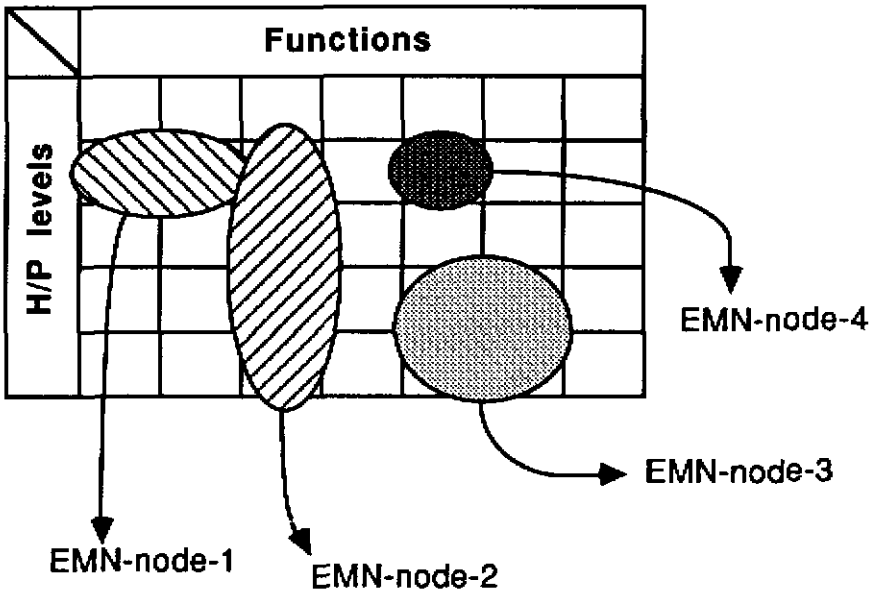



**Figure 3-10:** EMN-nodes identification using the organizational model

All these functions are performed at three levels:

- **Strategic** (S): which defines the objectives of the function,
- **Tactical** (T): which establish plans according to the objectives,
- **Operational** (O): which applies plans and re-adjusts them according to perturbations.

Additional functions include maintenance, quality control, distribution, design, etc.

According to this functional decomposition and the three identified decision levels, each function or sub-function can be split up into several activities. For example if we decompose the planning function we can identify six main activities:

The first activity performed in the planning function is **to do Production Planning**. Production planning forecasts customer demand and determines the manufacturing activities required to satisfy them, including budgets and capital investments.

**Master Production Scheduling** (MS) refines the Production Plan in more detail over a shorter horizon, with specific products and using firm orders. This is used as input to Material Requirement Planning (MRP). The MRP system produces three plans:

- a supply plan which is given to the supply function,

- a subcontracting plan which is given to the purchasing function and
- a manufacturing plan which is given to the planning function.

With the manufacturing plan, a **Load Plan** (LP) is developed by comparing the required demand against the theoretical capacity of the resources. In situations where demand exceeds capacity, load levelling is peformed in order to create a feasible plan. Leveling can be achieved by subcontracting, moving activities backward or forward in time, adding capacity through overtime, etc.

Given a load plan, **Scheduling** sequences the activities using detailed information about setup and run times, tooling and personnel requirements, etc. Once sequencing is completed, jobs are dispatched to the factory floor and schedules are adjusted in light of unplanned for events that may occur, such as machines failures.

A Production Management System can be viewed as a tree composed of several levels, each level corresponding to a criteria of decomposition (figure 3-11).

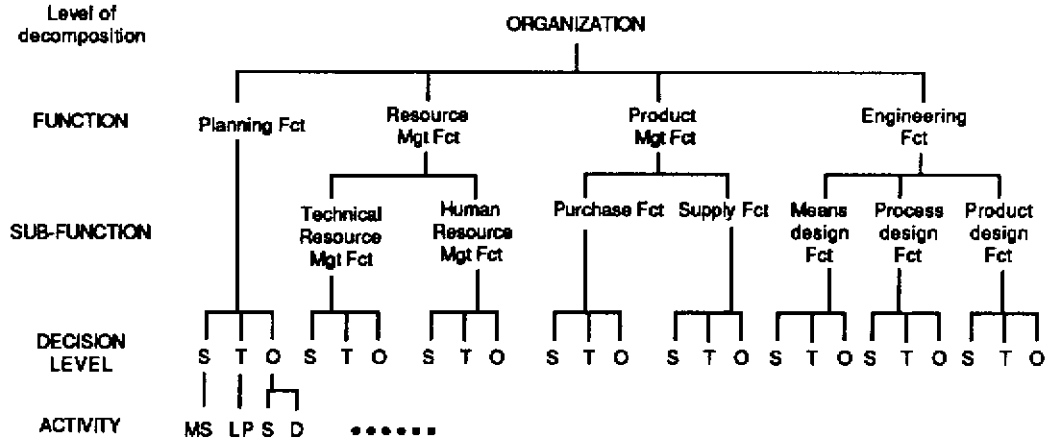



**Figure 3-11:** The organization tree

All these functions and activities can be identified on a GRAI grid.

Once the GRAI net and grid have been constructed, we can now map the organization onto EMN Agents. There exists more than one way in which to divide the organization, these criteria include:

- decomposition by function.
- decomposition by decision level (H/P level).
- decomposition by decision center.

Choosing a criterion depends upon how we value the degree of coupling and grain size of activities. The resultant decomposition spans a variety of problem solving organizations, but will contain all the activities present in the centralized structure described by the grid.

Additional elements can be added to distinguish these functions, such as:

- **Resources** (R): for example, machines and personnel.
- **Product** (P): generically, all raw materials, components, parts, finished products, etc. that are manufactured, supplied or sold by the company.

- **Time (T)**: such as duration, due date, starting date, etc.

For example, the three main functions of a PMS can be distinguished as follows:

- The Product Management function provides products to manufacturing at the right time and quantity. So, the elements manipulated by this function are P and T.
- The Resource Management function provides resources to manufacturing at the right time and capacity. So, the elements manipulated by this function are R and T.
- The Planning function synchronizes the production of products with the resources at the right time. So, the elements manipulated by this function are P, R and T.

In the ESPRIT project 418 [15, 35], *physical levels* are used as additional criterion for decomposition:

- factory level,
- shop level,
- cell level,
- workstation level,
- equipment level.

It is possible to build for each of these levels a decentralized structure with their own decentralized knowledge-base and problem solving subsystems. Such a decomposition has the advantage of being coherent and easily "coordinated" because it follows the production management hierarchical flow of decisions.

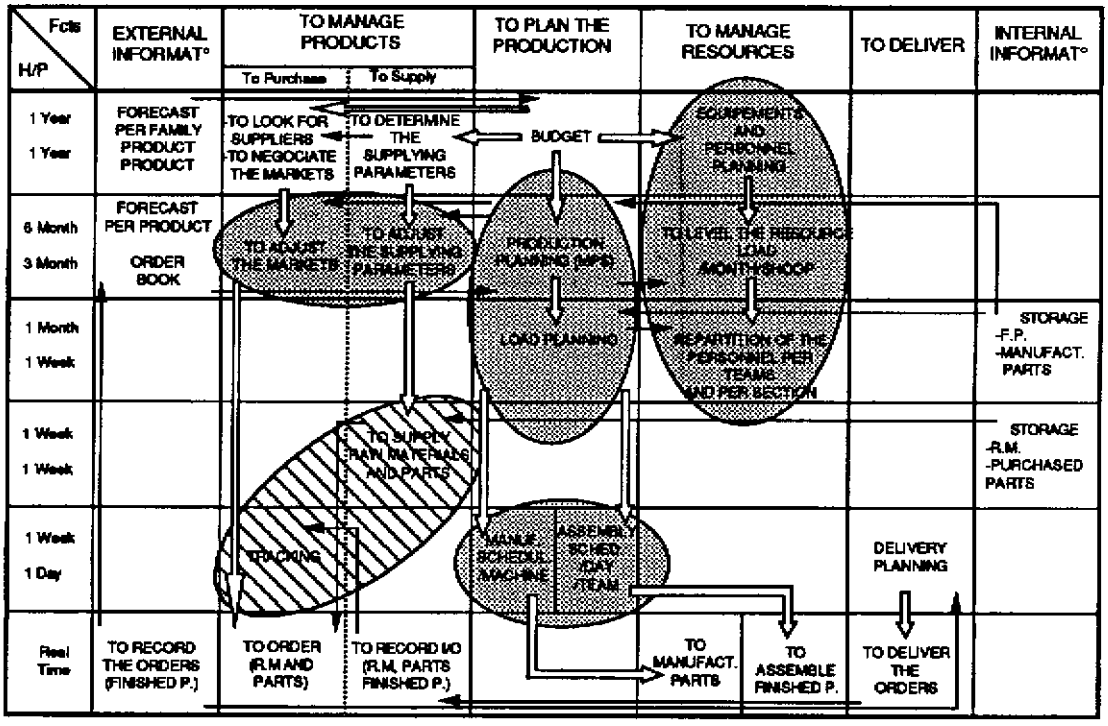



**Figure 3-12:** Example of EMN-nodes identification

As we can see, a wide variety of criteria is available to define the hierarchical structure of a manufacturing system. The selection of a criterion is the key issue for identifying the EMN-nodes of our structure (figure 3-12).

### 3.2.2.2 EMN-nodes hierarchy and inter-actions identification

The GRAI grid specifies the links between decision centers of a manufacturing organization. As the smaller grain size for the definition of the EMN-nodes is the decision center, we can easily make the correspondance between decision center links and EMN links.

The GRAI grid defines two link types:

- The information links, and
- the decision links.

The **information link** defines the information exchanges between decision centers (figure 3-13). Using this aspects, we can derive the owner, the user and the shared information. The origin of the information link can be defined as the information owner and the destination as the information user. By analyzing all these links, we can easily derive the content of the communication schema, defined at the data layer, for each different EMN-nodes. This derivation will be supported by some Lisp functions which will, using the schemata supporting the organizational model, complete the different slots of the communication schema of all the different EMN-nodes. Consistency checking will be also ensured.

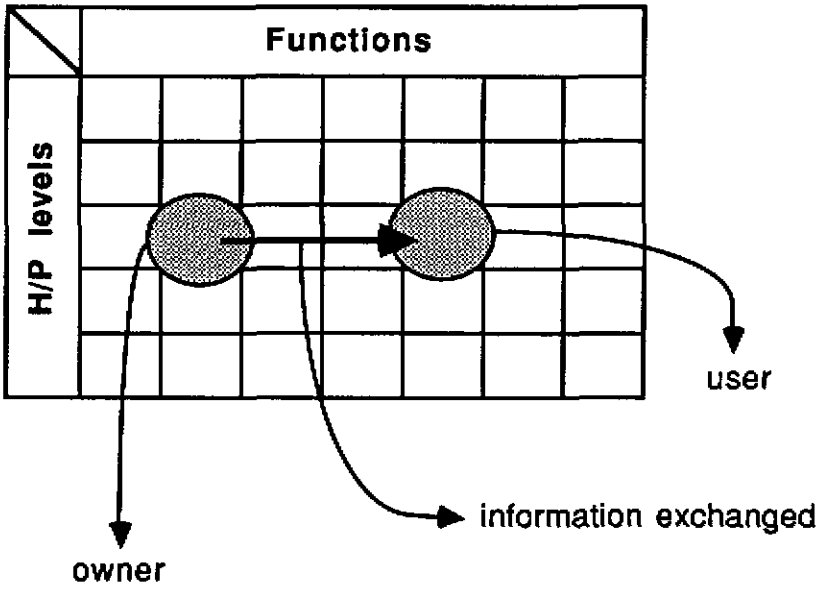



**Figure 3-13:** Identification of the information exchanges

The **decision link** defines the hierarchy of decision centers (figure 3-14). A decision frame or decisional link between two decision centers (or EMN-nodes in case of direct correspondance) defines the transmission of goals and decisional variables from one decision center to another. A decision frame is used as a platform to support decision activities. They define the decision centers hierarchy. In addition, elements such as goal, decision rules, responsibility, etc. are specified.

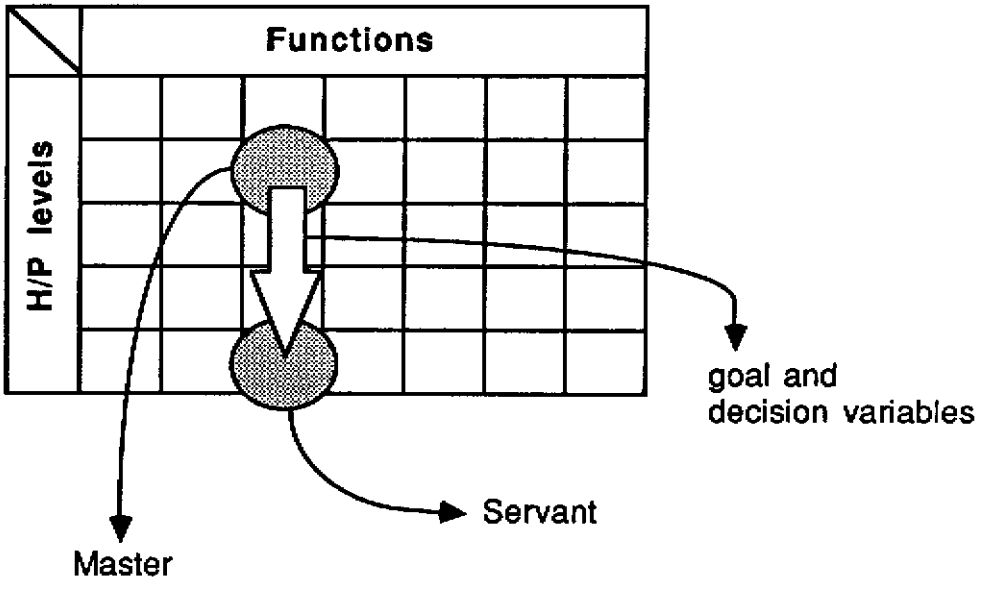



**Figure 3-14:** EMN-node hierarchy identification

We use these links to define the hierarchy of EMN-nodes once these EMN-nodes have been identified on the GRAI grid. These links allow to establish between pairs of EMN-nodes the type of inter-action it exists between them. According to this type, we can select a negotiation protocol, defined at the Coordination layer, to support the distributed problem solving between EMN-nodes.

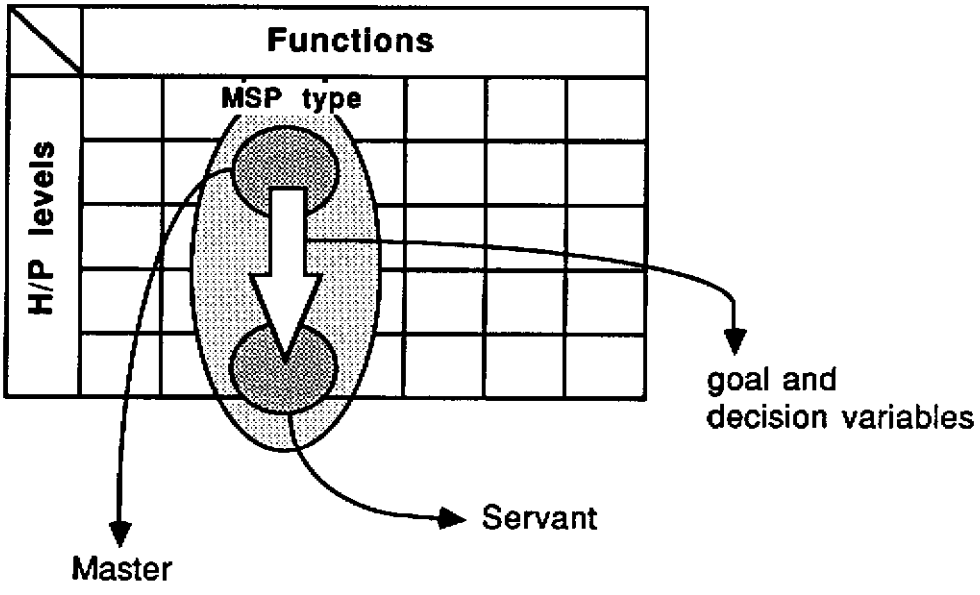



**Figure 3-15:** MSP type identification

We have identified three different type of inter-actions between EMN-nodes on a GRAI grid:

- The MSP (Master-Servant Protocol): when we have a decision frame between two decision centers which are in the same function but at different levels of decision (figure 3-15). This type of relation can be identified as a global goal transmission between two EMN-nodes of the system. The "servant" performs its activity based on the

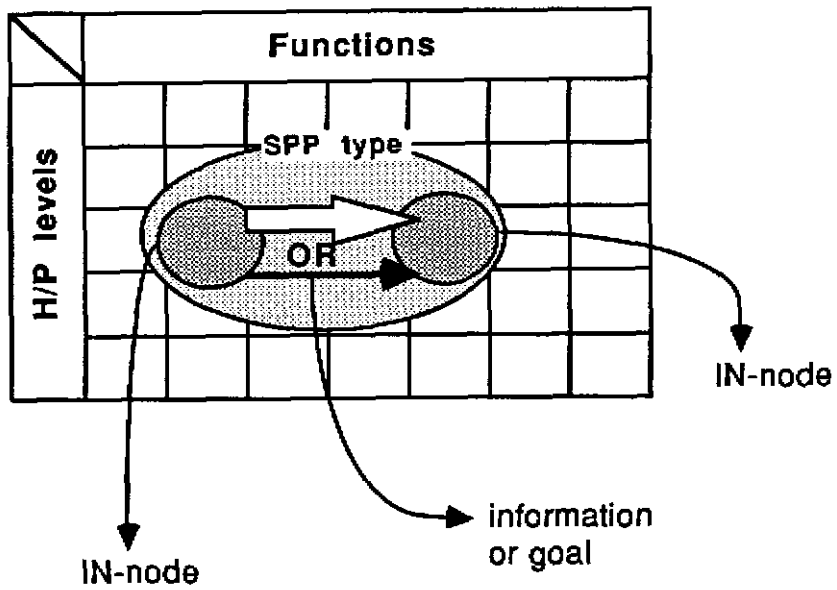



**Figure 3-16:** SPP type identification

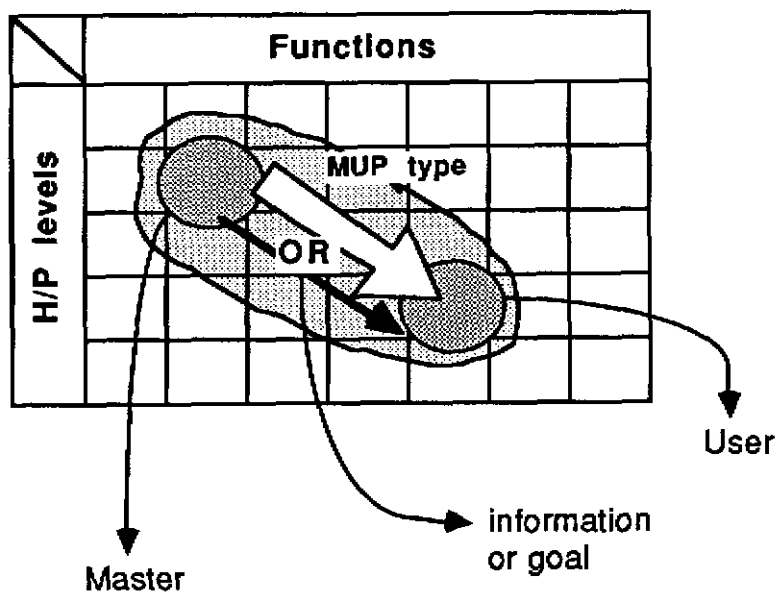



**Figure 3-17:** MUP type identification

*decision-frame* it receives from the "master". This *decision-frame* contains the goals and plans to follow. The interaction is mainly unidirectional (from the "master" to the "servant"). The "servant" only sends feedback to the "master".

- The SPP (Same-Power Protocol): when two decision centers are at the same level of decision but in different functions and linked by an information link (figure 3-16). This is the more complex type of relation. In that case, the EMN-nodes have to cooperate because they are performing an antagonistic task. The goals of their activities can be different but they are manipulating common resources. As an example, we can refer to [13, 33, 27, 10, 19, 12, 3, 4, 43, 46, 45] for a more complete description and study of coordination and negotiation mechanisms. Based on this literature, we will propose different protocols at the coordination layer, to support this aspect of coordination and negotiation of antagonistic EMN-nodes.

- The MUP (Master-User Protocol): when two decision centers are at different level of decision, in different functions and linked either by a decision frame or by an information link (figure 3-17). In such a case, a partial goal and plan transmission is done between the "master" and the "user". The "user" performs its activity by taking into account the partial goals and plans provided by the "master" but completed by information coming from an EMN-node located at a higher level of decision in the same production management function.

The identification of the different inter-actions between EMN-nodes is supported at the organization layer (figure 3-18). But the specification and implementation of the different identified coordination protocols are presented at the Coordination Layer.

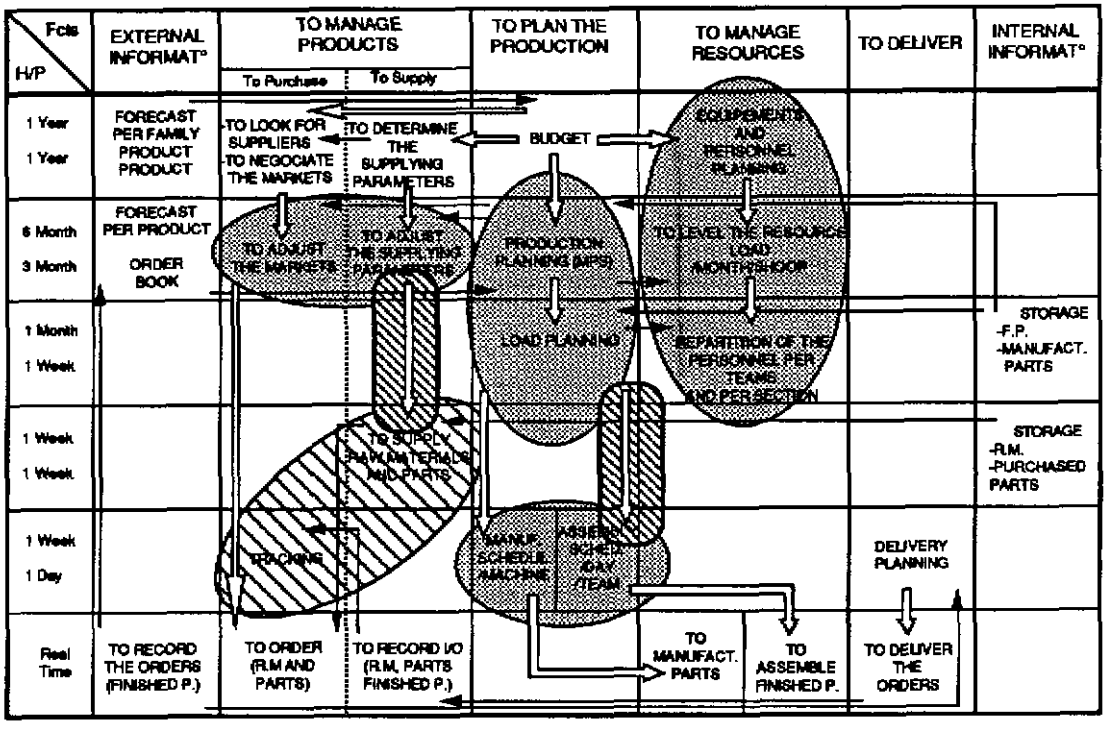



**Figure 3-18:** Example of EMN-node links identification

### 3.2.3 A generic organizational model

We use the GRAI grid to define the model of a manufacturing organization according to a decisional point of view. In this section, we give a generic example of what could be an organizational model (in this model, Cs is the supplying cycle and Cm is the manufacturing cycle).

This model (figure 3-19 represents the generic view of a MRP type manufacturing organization. Its definition has been initialized in [36] and completed in this project. The purpose of such a model is to provide a platform for manufacturing organization design.

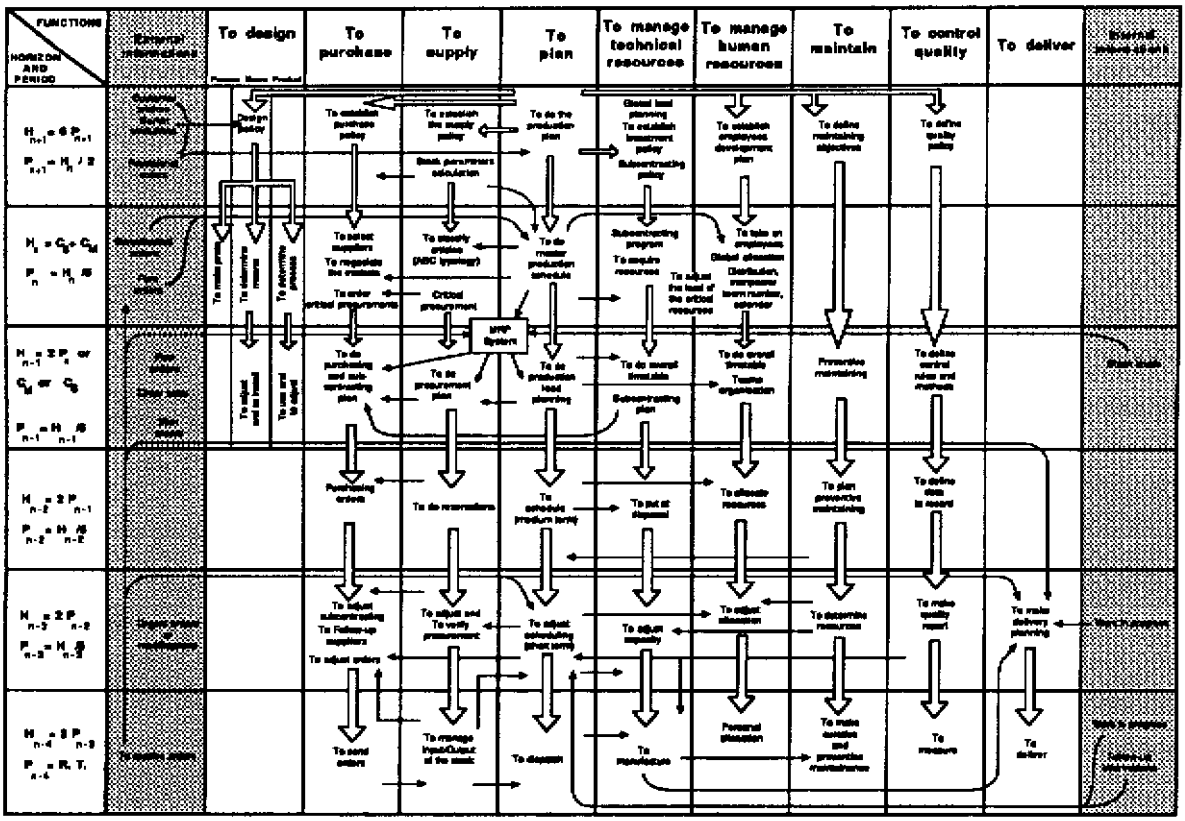



**Figure 3-19:** Organizational model: decisional point of view

### 3.2.4 The MERISE data modeling tools

The MERISE methodology uses entity/relationship model for data modeling. Data analysis is a methodology which links together the analysis of functions and data in an integrated and structured model. In this section, we define the concept of entity/relationship model to build a Conceptual Data Model (CDM). Then, we introduce the Logical Data Model (LDM) derived from the CDM using some translation rules we define.

#### 3.2.4.1 Entities and entity types

The building block upon which all the entity analysis is based, is called an **entity**. An entity is "anything relevant to the enterprise about which information could be or is kept". An entity represents data but is not itself a data. For instance, a drilling machine exists as a machine but its capability, number of tool, availability and so on are just characteristics which may or may not be represented as data. A second term used in entity analysis is entity type (figure 3-20). An entity type covers all entities relevant to the enterprise, which have a given common definition.

We can determine several types of **entity type**:

- real entity types: these are tangible objects or things, such as machines, people, buildings, etc.
- activity entity types: these are activities of interest to the enterprise, about which data could be kept, for instance: accident, inquiries, etc.

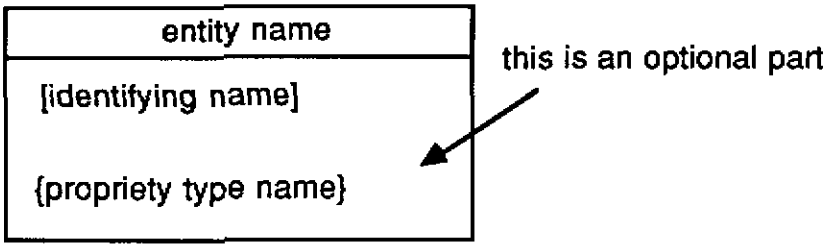



**Figure 3-20:** The entity content

- conceptual entity types: a business can invent or use purely conceptual entity types, both intangible and in some cases unique to the business, which might be: employment, cost center, shop order, etc.

#### 3.2.4.2 Relationship

A **relationship** is "an association between two or more entities which is of interest to the enterprise". Anything that shows or sharpens a connection between two or more entities may be thought of as a relationship.

The associated entities may be of one or two types, but not more than two. A relationship type comprises "all the relationship occurences which fit a given definition" (figure 3-21). A relationship type does not denote direction. If one were to draw a parallel between relationship types and language the relationship type would be the verb and the two entity types the subject and predicate nominative noun. In language these are reversible using a different verb construction (active and passive). In other words we could just as easily have reversed the relationship type to read and mean exactly the same thing.

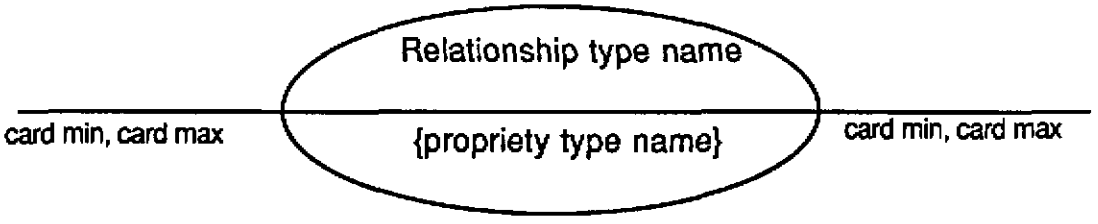



**Figure 3-21:** The relationship content

We can introduce the concept of degree in relationship. This concept is called **cardinality**. It exists several possibilities of expression to describe this degree. We present the three main found:

- One to one: one entity of one entity type may have that relation type with one entity of another or the same entity type,
- One to many: one entity of one entity type may have that relation type with one or more entities of another or the same entity type,
- Many to many: many entities of one entity type may have that relation type with one or more entity of another or the same entity type.

### 3.2.4.3 The MERISE data models

There exists one model per abstraction level and per life cycle step. For "data analysis", MERISE has determined three different models corresponding to the level of details:

- The Conceptual Data Model (CDM) (figure 3-22),
- The Logical Data Model (LDM) (figure 3-24),
- The Physical Data Model (PDM).

To build the CDM model we use the entity/relationship model (figure 3-22). The first step consists in determining a list of the vocabulary used within the company. Then we compare all these "words" between them to exclude all the synonymous, ...

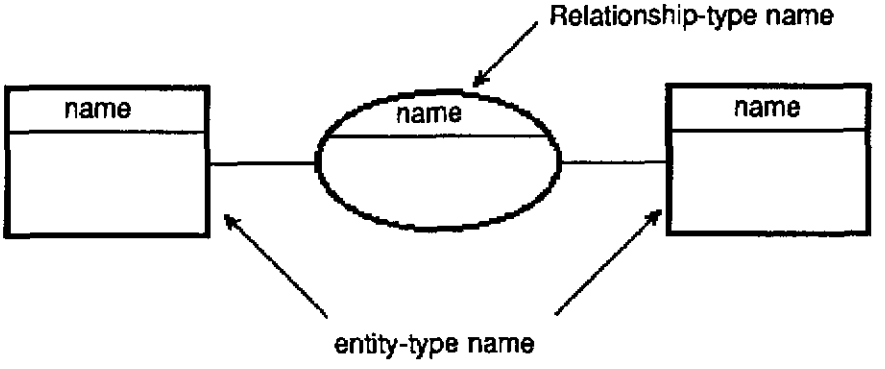



**Figure 3-22:** The entity/relationship model

The list of purified vocabulary represents the list of the entity-types (example: the entity-type workstation). For each entity-type we determine the attributes which allow to specify the content of the entity-type (example: the attributes of the entity-type workstation can be: name, capacity, identification, ...). The second step is the determination of the relationship between each entity. We establish a list of links and we give to each one a name. This list corresponds to the list of the relationship-type.

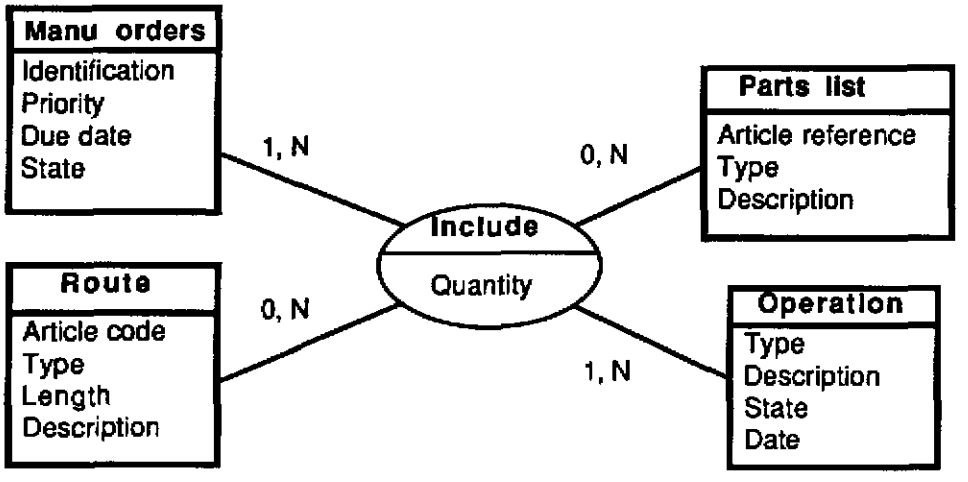



**Figure 3-23:** Example of CDM

With these two lists, we build the first draft of the CDM. We indicate for each link the cardinality of the relation. Then step by step, the final version of the CDM (example: figure 3-23) is built and adjusted. The MERISE methodology provides some rules to build the first draft and to revise the CDM. In addition, a methodology step by step is also define.

The **LDM** is a modification and adaption of the CDM according to the technological constraints on data base or files. The LDM is an adaptation of the CDM to the existing technology in term of data bases and knowledge bases. At this level we make the choices for the future structure of the data system. We have several possible choices according to the existing technology: relational data bases, hierarchical, network, object, ...

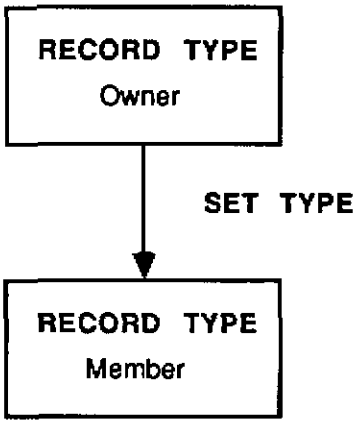



**Figure 3-24:** The Logical Data model

Once we get the final version of the CDM, a choice is made in the data base type we are going to use for this specific implementation. According to this choice, the LDM is build derived from the CDM. If we select for example a CODASYL Data base type we have to modify the CDM according to some rules (figure 3-24) to build the corresponding LDM (see translation rules for MERISE-CDM into MERISE-LDM in the next section).

The **PDM** corresponds to the realization of data base. It is in fact the implementation of the data bases according to the specification defined in the LDM.

### 3.2.4.4 Translation rules for MERISE-CDM into MERISE-LDM

The conceptual model has a too rich formalism to be translated into a data definition language of a data base management system. We have to fit this conceptual model according to the computer constraints without losing the signification of this model. To reach this objective, some formalism must be used to translate the CDM into the LDM.

The concepts of this logical internal formalism are:

- the **field**: It is the smallest part of a named data (we can compare the field to a small file part),
- the **record**: It is a named collection, without repetition of one or many field types (we can compare the record to a file),

- the **set**: It is a qualified relation between a record type which is declared as set master and a record type which is declared as member. It is a binary functional relation (we can compare a set to a data processing pointer).

The translation from an entity formalism structure to an equivalent structure in logical internal formalism is completely algorithmic. It is not a reversible translation. The translation rules are:

**Rule 1:**

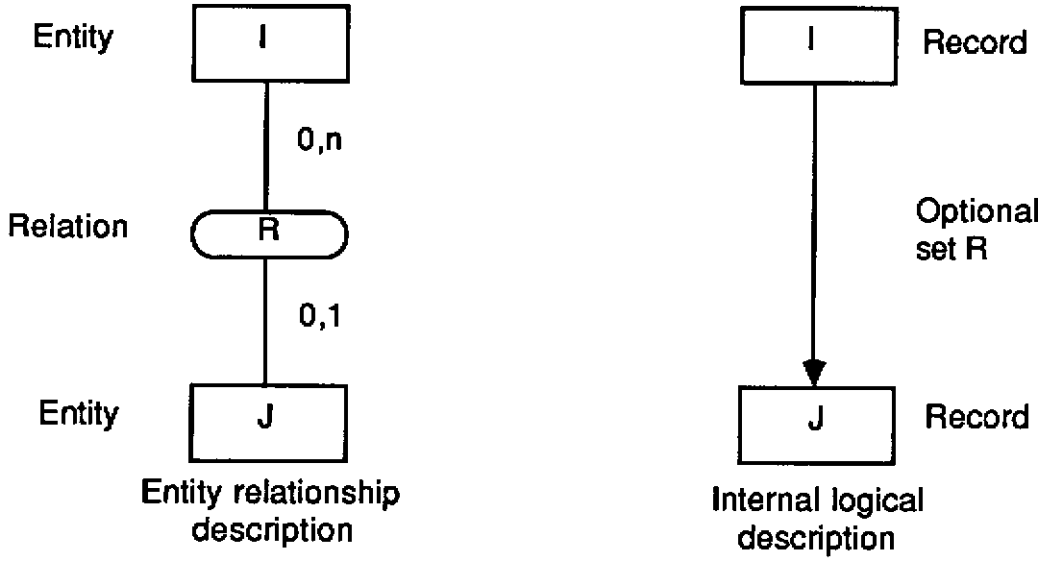
Property: each property (or attribute) in the CDM becomes a field in the LDM.

**Rule 2:**

Individual: each entity type in the CDM becomes a record type in the LDM.

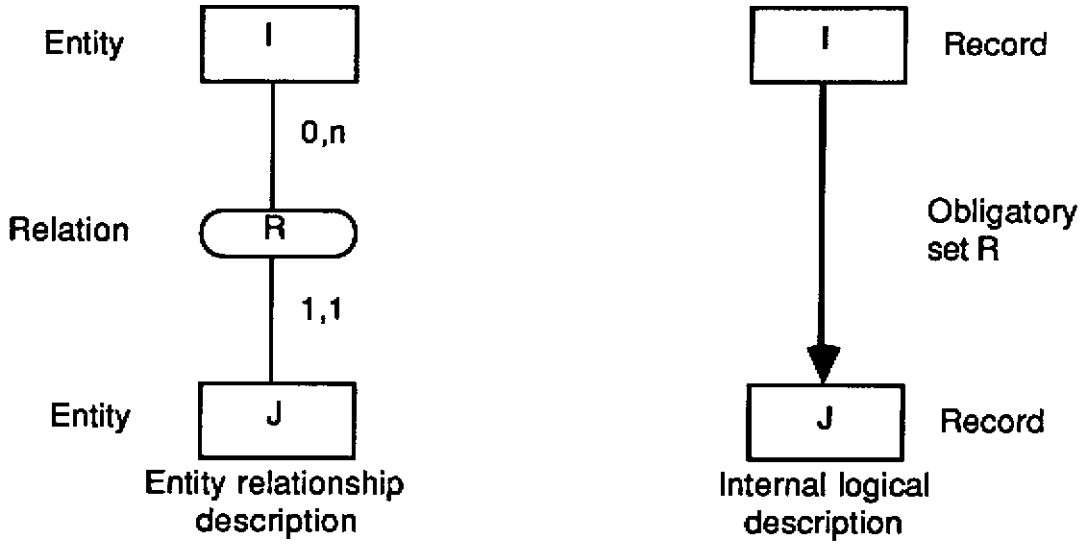
**Rule 3:**

Binary relation 0,n-0,1 or 1,n-0,1: all binary relations 0,n-0,1 or 1,n-0,1 in the CDM become an optional set type in the LDM.



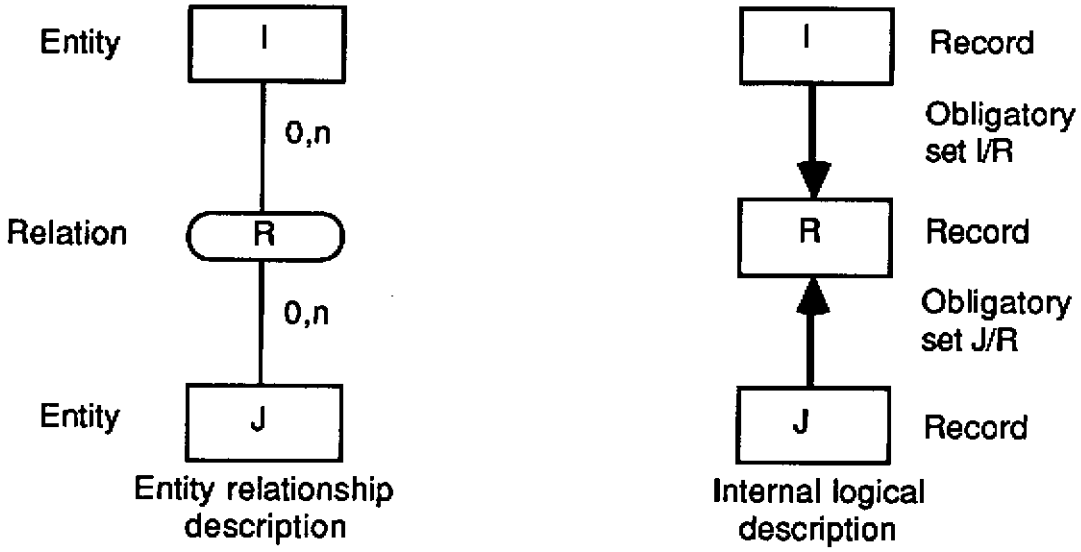
**Rule 4:**

Binary relation 0,n-1,1 or 1,n-1,1: All binary relations 0,n-1,1 or 1,n-1,1 in the CDM become an obligatory set type in the LDM.



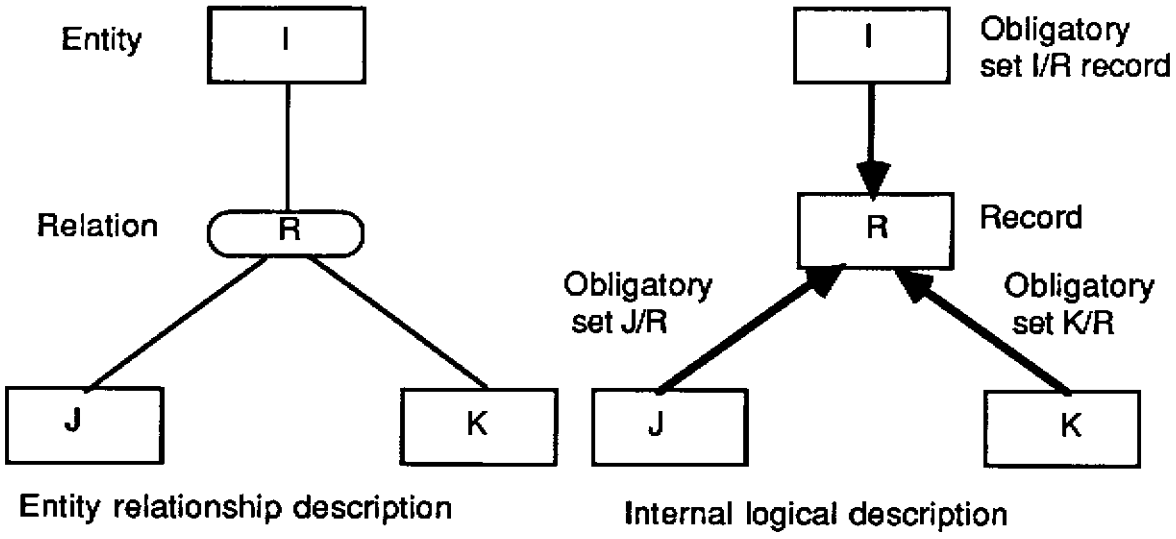

**Rule 5:**

Binary relations 0,n-0,n or 1,n-1,n: These relation types in the CDM are transformed in one record type and two set types in the LDM.




**Rule 6:**

Relation which involves more than two entities: This type of relation in the CDM is transformed in one record and there are as many sets as entities which participate in the relation in the LDM.




### 3.2.5 A generic data model supporting a manufacturing organization

In this section, we define a generic data model, using the entity/relationship modeling tool, which can support manufacturing organization. This model is dedicated to the job shop type of manufacturing process. Based on this model, the decentralized subsystems can be derived. We give an example of derivation for the supplying function.

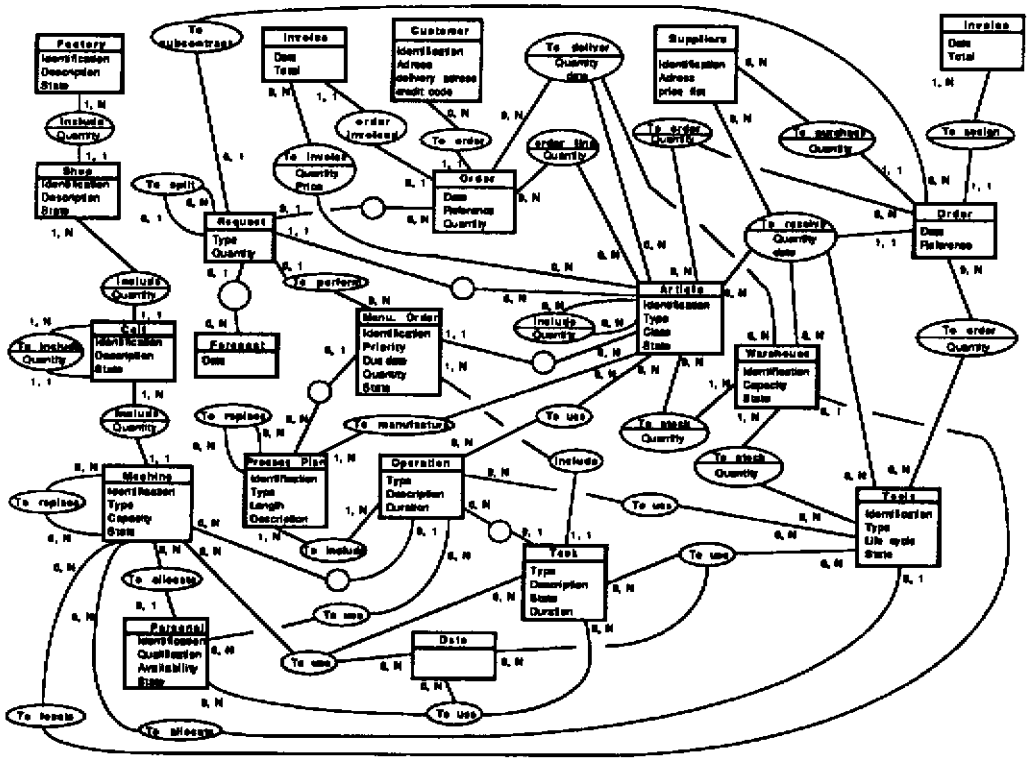


**Figure 3-25:** Organizational model: data point of view

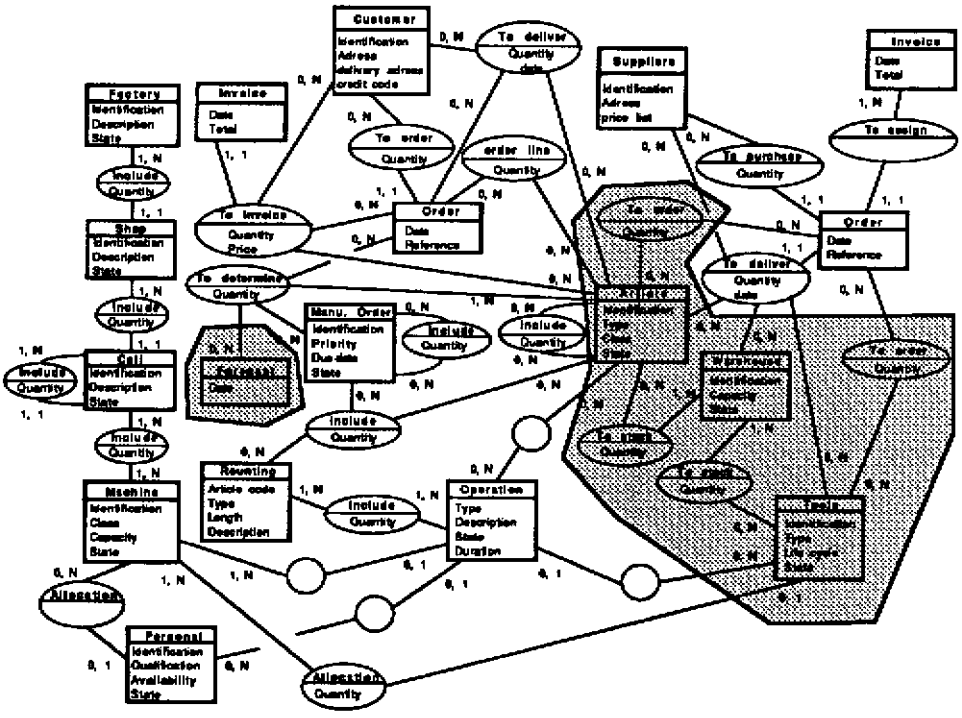



**Figure 3-26:** The supplying function conceptual data model

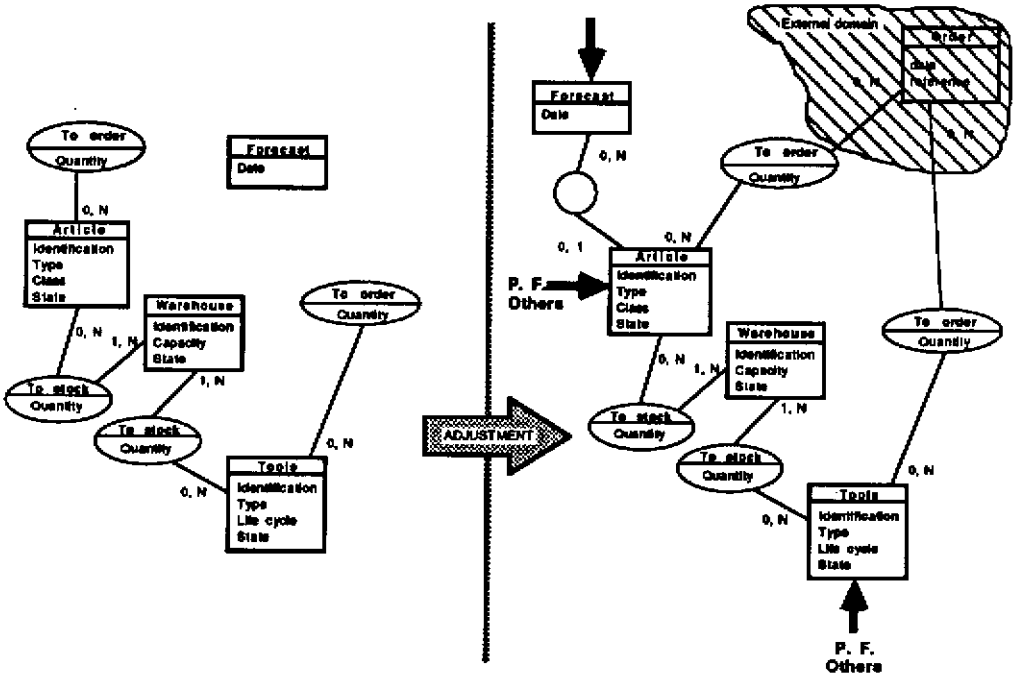



**Figure 3-27:** The logical data model derivation

The supplying function example (figure 3-26) shows the domain covered by this activity on the integrated model. Based on this domain, we define the submodel derived (figure 3-27). This submodel needs to be adjusted in term of coherence, consistency and completeness. In figure 3-27, we adjust the conceptual submodel defined as the basis of the logical data model of the supplying function. In figure 3-28, using the translation rules presented in the previous section, we determine the logical data model of the supplying function (in this example, we use the CODASYL standard).

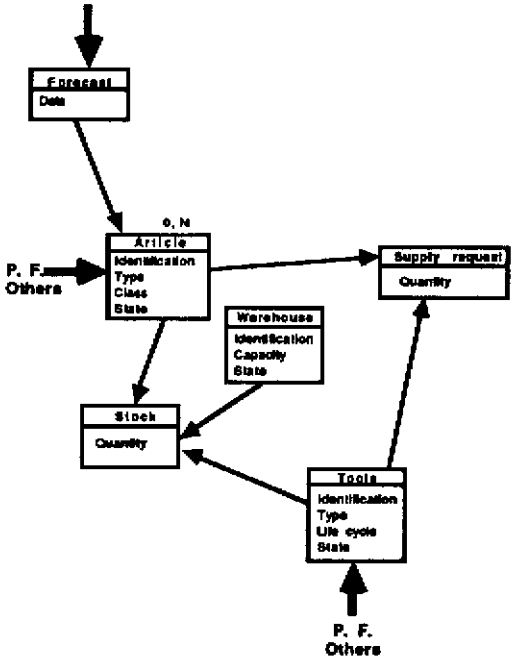



**Figure 3-28:** The supplying function logical data model

### 3.2.6 Coherence tools

The two models we define in the previous sections model a manufacturing organization according different points of view. As they will both support the definition of the corresponding decentralized system, they should be coherent. For that purpose, in this paragraph, we define two different coherence tools which ensure the mutual consistency of the GIM data model and of the GRAI decisional model:

- The Data/Process coherence tool, and
- The Process/Data coherence tool.

The D/P coherence tool (figure 3-29) consists in making the data model complete and coherent using the decisional model. The data model contains the entities and relationships which are supposed to be necessary for the running of the decision system. The D/P coherence tool creates for each decision process an external data model which represents the information necessary for that specific decision process. Then it checks the existence of all the entities and relationships of this external data model into the internal one (the GIM data model). This mechanism is applied to all the decision processes.

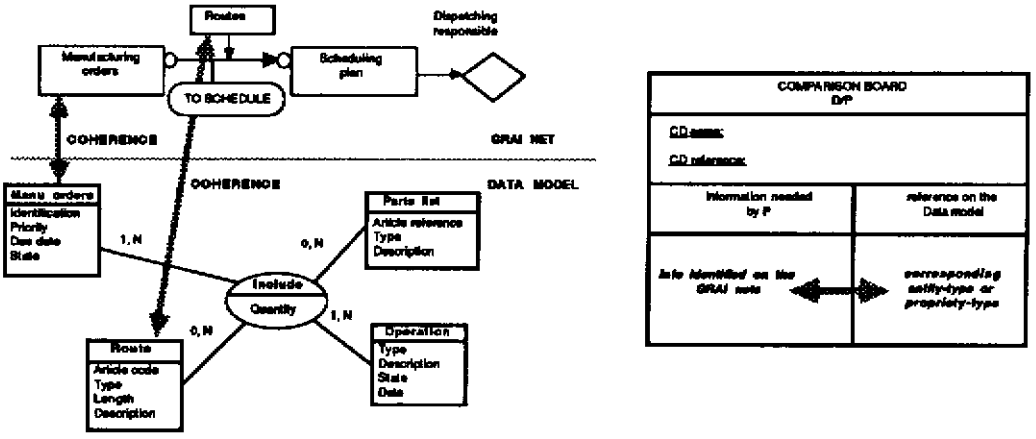



**Figure 3-29:** Data/Process coherence tool

The P/D coherence tool (figure 3-30) consists in making the decision model coherent such as all the information contained by the data model are created, modified, exploited and suppressed in the right sequence. For that purpose, the P/D coherence tool defines in the chronological order the different decision and information processing described in the GRAI model then the information creation, modification, exploitation and suppression is derived. By checking for each information the order of appearance, the decision and information processing can be adjusted.

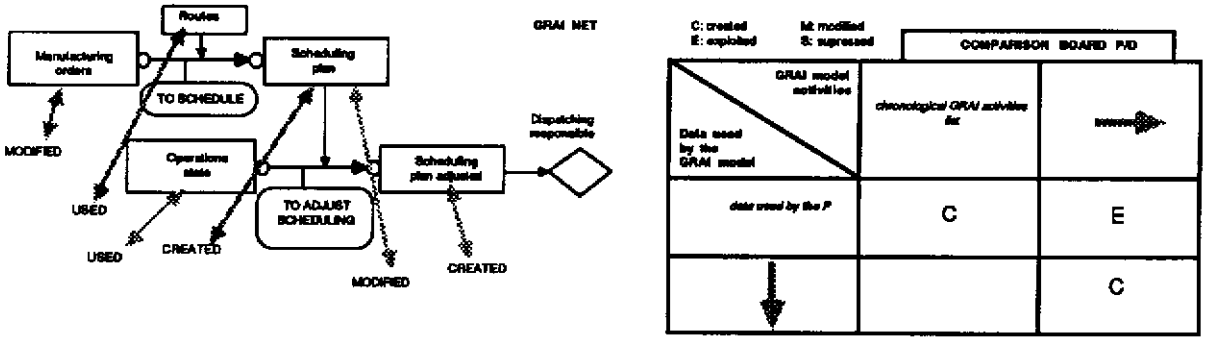



**Figure 3-30:** Process/Data coherence tool

## 3.3 Schemata defined at the organizational level

To be able to describe such a complex system, we must have a global and a detailed description of its components. In the Organization Layer, we focus our description on the EMN-node concept. At the upper level, as we describe the different types of organization, we provide tools to support the description of EMN-node interactions and coordination. The global view of an organization is given by the GRAI grid. The detailed view (EMN-node) is given the decision center description. The data model provides the support for all the activities identified in the GRAI model.

To illustrate the advantages of developing a representation to support distributed problem solving lets consider the following example: the schemata we define to support distributed problem solving can be used to create for each specific problem involving several EMN-nodes a blackboard [14, 23].

For example, if a problem to be solved involves three different EMN-nodes, they will all have a local blackboard dedicated to that problem. Each time one of the three agents will modify something in its local blackboard, modifications (or updates) will be sent to the two other blackboards (of the two other EMN-nodes). Each EMN-node will have one blackboard per antagonistic task and among the decentralized system, for a specific antagonistic task, there will be as many blackboards as involved EMN-nodes (figure 3-31). These blackboards can be build according to the organization model and data model we define at this layer. The updating mechanism will support the adjustment of the shared blackboard data structure.

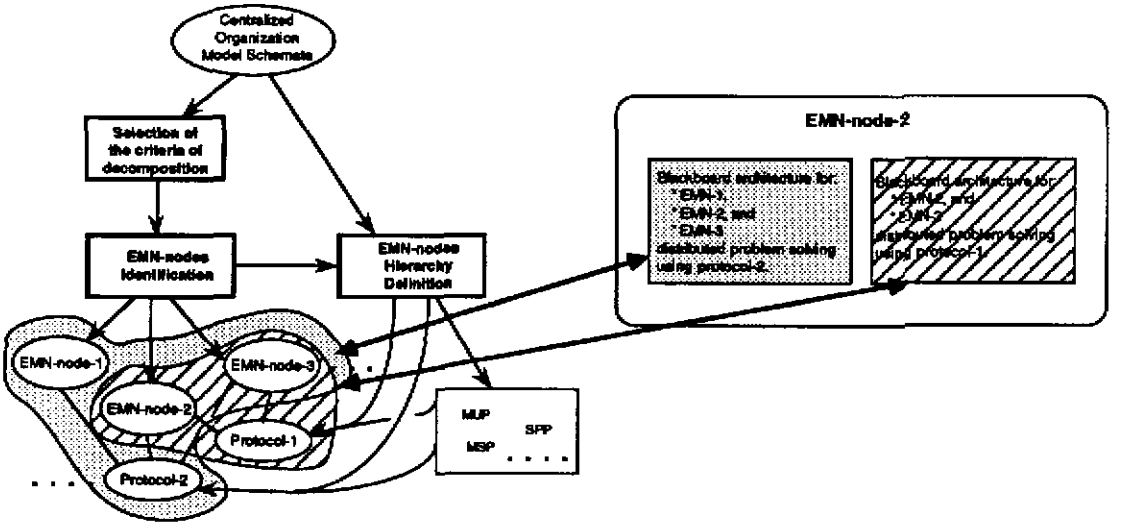



**Figure 3-31:** Example of task decomposition

In the previous paragraph and in figure 3-31, we define coordination and negotiation protocol as a basis for our distributed problem solving architecture, this in addition with the schemata describing the decentralized organization and the task blackboards. These protocols can be viewed as generic rules to follow for negotiating and coordinating decentralized EMN-nodes. These protocols should be general and must cover a class of problems instead of being too precise and restrictive. Our idea for the coordination layer is to define generic protocols which can allow agents to start working and to add learning mechanisns so that the protocols can be improved during their execution. As an example of generic protocol, we can refer to [44]. In this paper, a protocol for distributed scheduling system is presented. Distributed scheduling is a process carried out by a group of agents each of which has (a) limited knowledge of the environment, (b) limited knowledge of the constraints and intentions of other agents, and (c) limited number and amount of resources that are required to produce a system solution. Some of these resources may be shared among many agents. Global system solutions are arrived at by interleaving of local computations and information exchange among the agents. There is no single agent with a global system view.

The multi-agent communication protocol is as follows:

I. Each agent determines required resources by checking the process plans for the orders it has to schedule. It sends a message to each monitoring agent (as specified in a table of monitoring agent) informing it that it will be using shared resources.

*II.* Each agent calculates its demand profile for the resources (local and shared) that it needs.

III. Each agent determines whether its new demand profiles differ significantly from the ones it sent previously for shared resources. If its demand has changed, an agent will send it to the monitoring agent.

IV. The monitoring agent combines all *agent demands* when they are received and communicates the *aggregate demand* to all agents which share the resource[7].

V. Each agent uses the most recent aggregate demand it has received to find its most critical resource/time-interval pair and its most critical activity (the one with the greatest demand on this resource for this time interval). Since agents in general need to use a resource for different time intervals, the most critical activity and time interval for a resource will in general be different for different agents. The agent communicates this reservation request to the resource's monitoring agent and awaits a response.

VI. The monitoring agent, upon receiving these reservation requests, checks the resource calendar for resource availability. There are two cases:

1. If the resource is available for the requested time interval, the monitoring agent (a) communicates "Reservation OK" to the requesting agent, (b) marks the reservation on the resource calendar, and (c) communicates the reservation to all concerned agents (i.e. the agents that had sent positive demands on the resource).
2. If the resource had already been reserved for the requested interval, the request is denied. The agent whose request was denied will then attempt to substitute another reservation, if any others are feasible, or otherwise perform backjumping.

VII. Upon receipt of a message indicating its request was granted, an agent will perform consistency checking to determine whether any constraint violations have occurred. If none are detected, the agent proceeds to step II. Otherwise, backjumping occurs with undoing of reservations until a search state is reached which does not cause constraint violations. Any reservations which were undone during this phase are communicated to the monitor for distribution to other agents. After a consistent state is reached, the agent proceeds to step II.

The system terminates when all activities of all agents have been scheduled Backtracking, with this version of the protocol, is based on the following design decisions: 1) Once an agent has been granted a reservation, this reservation is not automatically undone when some other agent who had to backtrack now needs the reservation. This can lead to situations where one agent solves its local scheduling problem but the other agent cannot due to unresolvable constraint violations. 2) If an agent backtracks, it frees up resources but the reservation of other agents on these resources remain as they were. This policy may result in non-optimal reservation for other agents since it denies the other agents greater opportunity to take advantage of the canceled reservations of the backtracking agent, but it results in less computationally intensive performance.

At the Organization Layer, we must structure an organization. The **grid** schema supports such a description. It partially specifies the decision center, the modules, the data-modules and the links between them. The distinction is made between decisional and informational links. Both are supported by schemata. The grid provides the global view of the organization we want to structure. This model will be the basis in the definition of the decentralized EMN-nodes. The granularity of this model is the decision center. This graphical tool produced from the GRAI method [36], supported by a schema, describes the main characteristics of the decision system of this specific organization. It shows the links between the EMN-nodes, as well as those with the environment of the system. It provides a decisional and global description of the organization.

---

[7]With the exception of the first time demands are exchanged, agents do not wait for aggregate demands to be computed and returned prior to continuing their scheduling operations (although they can postpone further scheduling if desired).

**Schema 3-1:** Grid

| Grid | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Is-a | *Restriction:* | model |
| Functions | *Value:*<br>*Restriction:* | type string* |
| Decision-levels | *Value:*<br>*Restriction:* | type (horizon/period)* |
| Decision-centers | *Value:*<br>*Restriction:* | type decision-center-name* |
| Decisional-links | *Value:*<br>*Restriction:* | type decision-frame* |
| Informational-links | *Value:*<br>*Restriction:* | type informational-link* |

The x-axis of the GRAI grid is composed by a set of **functions**. Each function can be described by an instance of the schema 3-2. This schema defines the goals and decision centers composition of each function. In addition, a description of the purpose of each function is provided.

**Schema 3-2:** Function

| Function | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Description | *Value:*<br>*Restriction:* | type string* |
| Goals | *Value:*<br>*Restriction:* | type goal* |
| Has-modules | *Value:*<br>*Restriction:* | type decision-center* |

The y-axis of the GRAI grid is defined by a set of **decision levels**. A decision level is a pair (horizon, period). We describe each decision level by an instance of the schema 3-3. Each decision level schema includes the value of the pair H/P and also an identifier which is generaly determined according to the following rules:

- Each decision level is identified by a multiple of 10.
- The decision levels are classified by decreasing period.
- At equivalent period, the decision levels are classified by decreasing horizon.

**Schema 3-3:** Decision-level

| Decision-level | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Identification | *Value:*<br>*Restriction:* | type string |
| Horizon | *Value:*<br>*Restriction:* | type string |
| Period | *Value:*<br>*Restriction:* | type string |

In this paragraph, we have to provide the elements to define the content of the two specific subsystems of an EMN-node:

- a domain modeling subsystem,
- and a problem solving subsystem (figure 3-32).

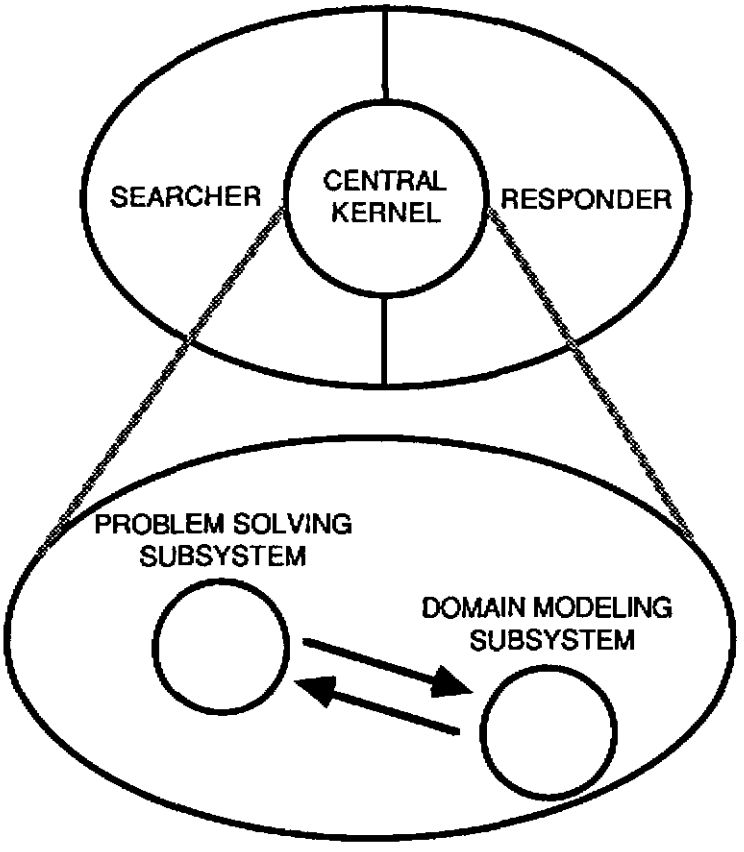



**Figure 3-32:** Content of the central kernel

The determination of the domain modeling subsystem can be done by using the model we defined in the previous section. We have defined in figure 3-25 the structure of centralized data base. We must identify on this global model the subdomain of each functions. By this way we identify the content of the decentralized data base or domain modeling subsystem.

To complete this work we must reorganize the elements of the subdomain (figures 3-26, 3-27 and 3-28 shows an example for the supplying function) to have the "best" and more efficient organization in a decentralized utilization.

For the problem solving subsystem, we have to build in the same way a decentralized structure able to have an autonomous running and capability to react to the perturbations related to the subdomain.

We can start our description with the centralized process model (figure 3-19) and to define the subdomain. We have to identify the elements of the decentralized problem solving subsystem. We must have a hierarchical decomposition to be able to respect the coordination aspect of an EMN-node. We have seen previously that several criteria can be used to split up such a global structure into a set of decentralized elements. To follow the hierarchical view of the grid, we can split up the problem solving subsystem into several hierarchical levels (figure 3-33).

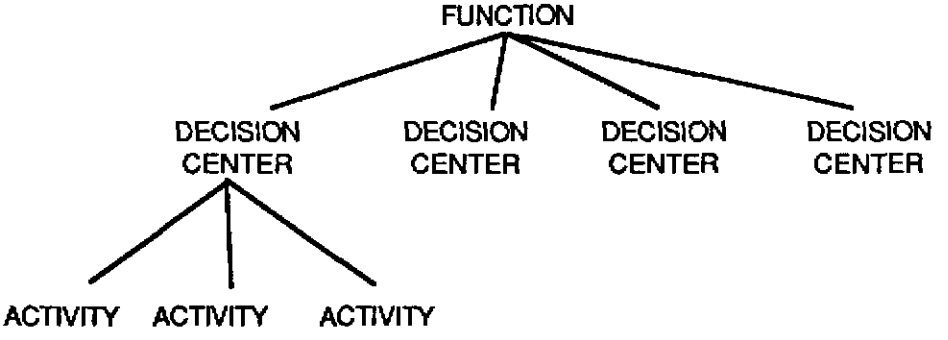



**Figure 3-33:** Problem-solving hierarchical levels

For each of these elements, we can build a schema defining their charasteristics and content. The first element is the **function**. A function represents a column of the grid (figure 3-25). A function is composed of several **decision centers**. A decision center is a "box" of the grid. It is in fact the intersection of a function and a decision level (H/P level). A decision center can be split up into several **activities**. Each activity can be define as an object. We can identify two kind of activities: the decision activities and the execution activities.

A **decision activity** implies a choice. This choice is done according to some rules or knowledge rules. For each choice, we have to respect a local-goal and our choice is done by determining the value of decision variables.

An **execution activity** is a calculus, an information processing we can define by an algorithm.

All these elements (EMN-nodes, Function, Decision center, Activity, Decision activity and execution activity) are objects. For all these objects, we can build a schema. If we want to implement this structure into knowledge craft, we must identify the schemata of such a structure. Figure 3-34 provides an overview of these schemata.

The basic element of the organization level is the EMN-node. As an EMN-node is responsible for a specific task, we represent it as a **decision center**. The concept of decision center comes from the

GRAI method. A decision center contains all the elements needed to perform a specific decision activity.

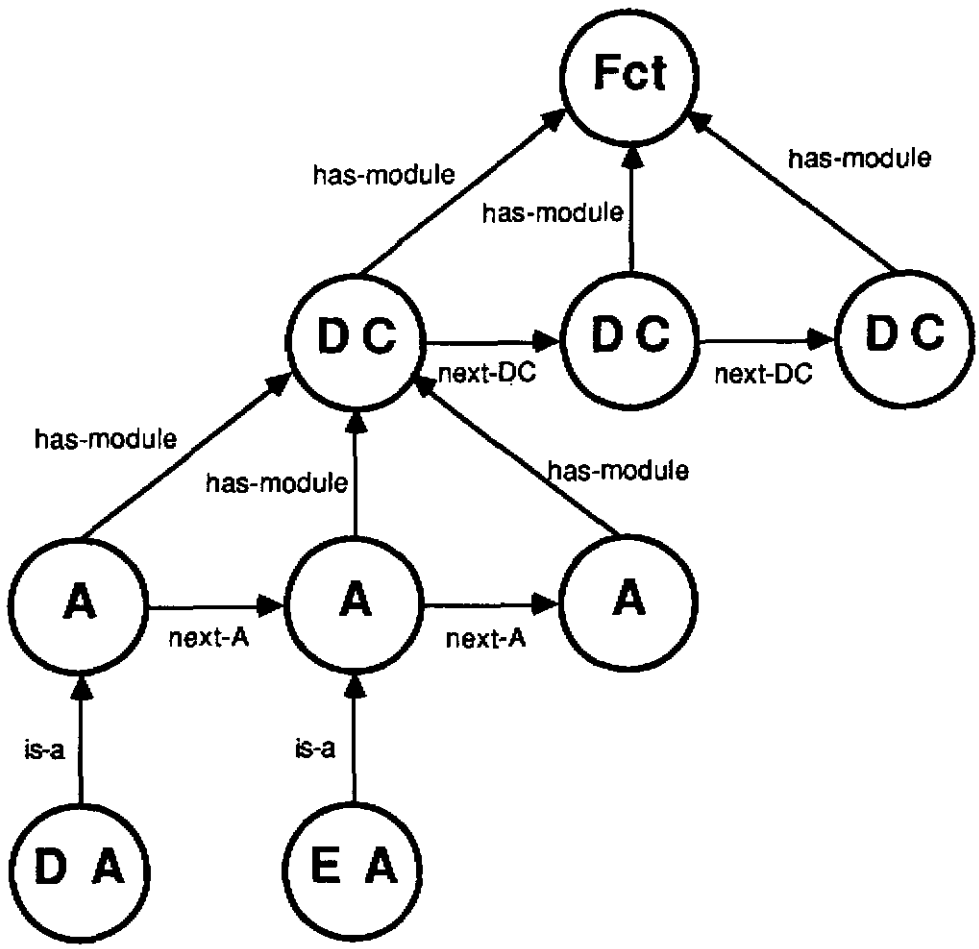



**Figure 3-34:** Problem solving hierarchical decomposition

We create for each decision center an instance of the schema 3-4. This one contains knowledge related to the decision aspect. The decision center is the basic element of our organizational model. The granularity used to define the EMN-nodes is the decision center. Generally, a decision will represent an EMN-node. But, in some structure, an EMN-node can be defined as a combination of several decision centers.

A decision center has a specific role (described in the role slot), performs its activity according to one or more goals (described in the goal slot) and determines the value of certain decision variables (listed in the decision-variables slot). To perform its activity, a decision center has a specific Knowledge Base, a specific problem solving sub-system and can get information (schemata) from the other decision centers.

As we have seen, each EMN-node possesses a Knowledge Base sub-system and a Problem Solving sub-system. Both of them are **models**. A model can be viewed as an abstraction of a specified object [16]. In each model, an abstraction is composed of states and transitions between them.

**Schema 3-4:** Decision-center

| Decision-center | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type EMN-node-name |
| Decision-variables | *Value:*<br>*Restriction:* | type string* |
| Goal | *Value:*<br>*Restriction:* | type goal-name* |
| Role | *Value:*<br>*Restriction:* | type role-name* |
| Decision-rules | *Value:*<br>*Restriction:* | type string* |
| Decision-level | *Value:*<br>*Restriction:* | type decision-level |
| Period | *Value:*<br>*Restriction:* | type time |
| Function | *Value:*<br>*Restriction:* | type function |
| Has-module | *Value:*<br>*Restriction:* | type activity* |
| Previous-decision-center | *Value:*<br>*Restriction:* | type decision-center* |
| Next-decision-center | *Value:*<br>*Restriction:* | type decision-center* |
| Inputs | *Value:*<br>*Restriction:* | type information* |
| Outputs | *Value:*<br>*Restriction:* | type information* |
| Knowledge-Base-subsystem | *Value:*<br>*Restriction:* | type data-model-name |
| Problem-solving-subsystem | *Value:*<br>*Restriction:* | type Problem-solving-name |

A **state** in the computation is defined by a subset of **state-variables** with a particular position in the object's code. A model is the generic entity which represents an abstraction of a real object. All the other specific models we will describe will be linked with that one with the IS-A relation.

The Knowledge Base sub-system and the Problem Solving sub-system are both models. We create a schema for each one which describes their specific elements.

The **Knowledge-Base** schema is a collection of data-objects and knowledge objects. The purpose of this schema is mainly to identify a KB as member of one EMN-node.

**Schema 3-5:** Model

| Model | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| State-variables | *Value:*<br>*Restriction:* | type string |
| States | *Value:*<br>*Restriction:* | type string |
| Abstraction | *Value:*<br>*Restriction:* | type string |

**Schema 3-6:** Knowledge-Base

| Knowledge-Base | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Is-a | *Restriction:* | model |
| Knowledge-objects | *Value:*<br>*Restriction:* | type knowledge-object* |

Each knowledge-object is also described by a schema which defines its content, attributes and relations with the other knowledge-objects. The **Knowledge-object** schema describes a specific piece of data or a specific piece of knowledge in an EMN-node (We can identify this piece of data as a schema or as a rule).

**Schema 3-7:** Knowledge-object

| Knowledge-object | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Is-a | *Restriction:* | model |
| Description | *Value:*<br>*Restriction:* | type string* |
| Attributes | *Value:*<br>*Restriction:* | type (name, value [, value, ...])* |
| Relation | *Value:*<br>*Restriction:* | type (knowledge-object-name, cardinality)* |

For the problem solving subsystem, we use the first schema: **Problem-solving**. This schema IS-A model, and it describes the procedures specific to an EMN-node. Each procedure is also defined as a

schema. The **procedures**[8] are subsets of the Problem-solving schema. Each of them represents a specific function or functionality. The procedures manipulate the knowledge objects of the Knowledge Base.

**Schema 3-8:** Problem-solving

| Problem-solving | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Is-a | *Restriction:* | model |
| Procedures | *Value:*<br>*Restriction:* | type procedure-name* |

**Schema 3-9:** Procedure

| Procedure | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Is-a | *Restriction:* | model |
| Description | *Value:*<br>*Restriction:* | type string* |

Each decision center can be split up into several **activities**. Two activity types are identified: the **execution** and the **decision** activities. Each activity is defined by an instance of the activity schema (schema 3-10). The activity is defined as one of the module of a decision center. The definition of an activity is the link between the organization modeling representation and the CARMEMCO factory model [40].

In the decentralized system, each EMN-node (or decision center) has a specific purpose and role to play in the organization. A hierarchy exists in the organization. In this hierarchy, each specific decision center has some responsibility and authority over other decision centers. Similarly, each decision center also receives some orders and commands from the upper level of this hierarchy. The decision centers are linked together. We can distinguish two kinds of links: information links and decision frame links.

The first kind just concerns exchanges of information needed for the internal processing of the EMN-node. We define for each **informational link** a schema which contains the information exchanged between two EMN-nodes. The second kind of link concerns the decisional activity. A decision frame contains elements concerning goals, decision variables and objectives. To allow the transmission of coordination aspects through out the entire organization of EMN-nodes.

[8] In our current implementation, procedures are CommonLisp functions

**Schema 3-10:** Activity

| Activity | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Input | *Value:*<br>*Restriction:* | type data* |
| Output | *Value:*<br>*Restriction:* | type data* |
| Description | *Value:*<br>*Restriction:* | type string* |
| Previous-activity | *Value:*<br>*Restriction:* | type activity* |
| Next-activity | *Value:*<br>*Restriction:* | type activity* |

**Schema 3-11:** Decision-activity

| Decision-activity | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| is-a | *Restriction:* | activity |
| Goal | *Value:*<br>*Restriction:* | type goal* |
| Decision-variables | *Value:*<br>*Restriction:* | type information* |

**Schema 3-12:** Execution-activity

| Execution-activity | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| is-a | *Restriction:* | activity |
| Algorithm | *Value:*<br>*Restriction:* | type string* |

We have just described the structure of a decision center. These schemata are connected by channels. Channels allow the exchange of schemata. To this point, we have developed information exchange. The coordination of the decentralized structure needs goal, decision-variable and rule exchanges as well. The purpose of the **Decision-frame** schema is to support such exchanges. In

this way, we can describe the organization structure of a manufacturing system. The content of a decision frame is as it has been described in section 3.2. We define goals, decision variables, and some rules used in decision process.

**Schema 3-13:** Informational-link

| **Informational-link** | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Provenance | *Value:*<br>*Restriction:* | type decision-center-name |
| Destination | *Value:*<br>*Restriction:* | type decision-center-name* |
| Information | *Value:*<br>*Restriction:* | type schema slot value |

**Schema 3-14:** Decision-frame

| **Decision-frame** | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Provenance | *Value:*<br>*Restriction:* | type decision-center |
| Destination | *Value:*<br>*Restriction:* | type decision-center |
| Decision-variable | *Value:*<br>*Restriction:* | type string* |
| Goals | *Value:*<br>*Restriction:* | type goal* |
| Decision-rules | *Value:*<br>*Restriction:* | type string* |

An EMN-node uses another aspect: the EMN-node goal. This element is described by a specific schema. The description of a goal is of primary importance to an organization. We can refer to [16] to find the description of the **Goal** schema. In addition, a **Role** schema can be defined to provide the link between the EMN-node activity definition and the local goals.

**Schema 3-15:** Goal

| Goal | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Type | *Value:*<br>*Restriction:* | type string |
| Precondition | *Value:*<br>*Restriction:* | type string |
| Postcondition | *Value:*<br>*Restriction:* | type string |
| Resource-consumption | *Value:*<br>*Restriction:* | type string |
| Resource-production | *Value:*<br>*Restriction:* | type string |
| Resource-transformation | *Value:*<br>*Restriction:* | type string |
| Initiation | *Value:*<br>*Restriction:* | type string |
| Goal-model | *Value:*<br>*Restriction:* | type string |
| Ports | *Value:*<br>*Restriction:* | type string |
| Objects | *Value:*<br>*Restriction:* | type string |
| Organization-membership | *Value:*<br>*Restriction:* | type string |

**Schema 3-16:** Role

| Role | | |
|---|---|---|
| SLOT | FACET | VALUE |
| Name | *Value:*<br>*Restriction:* | type string |
| Description | *Value:*<br>*Restriction:* | type string |

## 3.4 Example

In this example, we define the hierarchical structure of the planning function. The figure 3-35 provides a global view of this structure and establishes the link between the generic schemata defined in the previous section and their instantiation for the planning function.

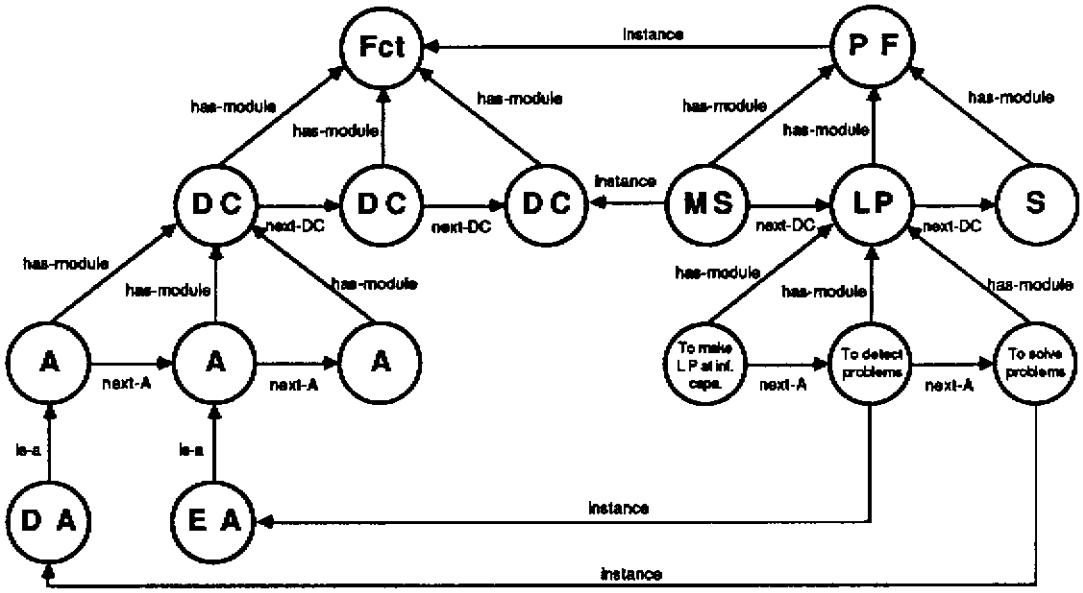



**Figure 3-35:** Problem solving hierarchical decomposition example

Then, we define the content of some schemata part of this hierarchical structure. We present the content of the load planning decision center. We detailed its content by defining two of its activities: "to-detect-problems" and "to-solve-problems".

```
{  PLANNING-FUNCTION
        INSTANCE: Function
        NAME: Planning
        DESCRIPTION:  to synchronize the manufacturing activity
        BASIC-ELEMENTS: P and R and T
        GOALS: to satisfy due-date and delay of the customers
        FRIENDS:        1 - Resource-Management function
                        2 - Product-Management function
                        3 - Engineering function
        HAS-MODULES: PP, MPS, LP, S, AS, D.     }

PP: Production Plan
MPS: Master Production Schedule
LP: Load Planning
S: Schedule
AS: Adjust Schedule
D: Dispatch
```

```
{  LOAD-PLANNING
        INSTANCE:  Decision-center
        NAME: Load Planning (LP)
        IS-MODULE-OF: planning-function
        DESCRIPTION: to adjust load according to capacity
        DECISION-LEVEL: Tactical
        GOALS: satisfy due-date
        INPUTS: MRP calculus
        OUTPUTS: load planning at finite capacity
        PREVIOUS-DECISION-CENTERS: MPS
        NEXT-DECISION-CENTERS: S
        HAS-MODULES:     - to make load planning at infinite capacity
                         - to detect problems
                         - to solve problems     }
{  TO-DETECT-PROBLEMS
        INSTANCE:  execution-activity
        IS-MODULE-OF: Load-planning
        NAME: to detect problems
        DATA-INPUTS: entities Machine, operation, routing,
                     task and date.
        DATA-OUTPUTS: task and date
        PREVIOUS-ACTIVITIES: to make load planning at infinite
                             capacity
        NEXT-ACTIVITIES: to solve problems
        ALGORITHM: to compare previsional load to capacity.
                        IF (load > capacity) THEN  problem
                        IF (load < capacity )  THEN  nil     }
{  TO-SOLVE-PROBLEM
        INSTANCE: decision-activity
        IS-MODULE-OF: Load-planning
        NAME: to solve problems
        DATA-INPUTS: entities Machine, operation, routing,
                     task and date.
        DATA-OUTPUTS: task and date
        PREVIOUS-ACTIVITIES: to detect problem
        NEXT-ACTIVITIES: to translate load into operations
        RULES: < if overload then subcontract the task>
        LOCAL-GOALS: to keep a regular manufacturing activity
        DECISION-VARIABLES: internal or external machine     }
```

## 3.5 Organization Layer example

In this Layer, we add to figure 2-4 the definition of roles, responsibilities, authority and goals specific to each EMN-node to get figure 3-36. With these elements, the EMN-node knows exactly its place in the organization of the decentralized system.

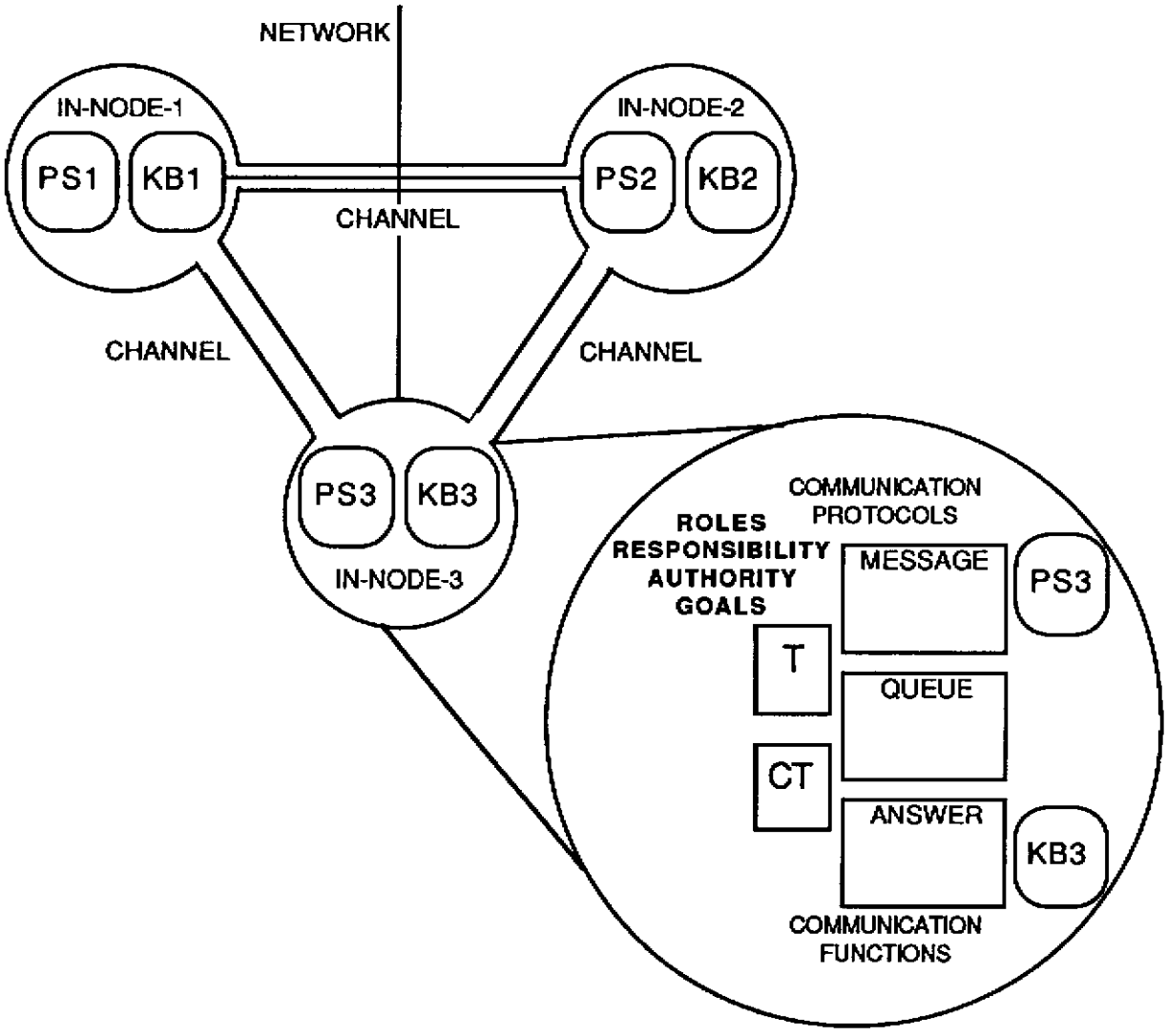



**Figure 3-36:** Organization Layer implementation example

## 4. Conclusion

The Enterprise Management Network is designed to facilitate the integration of heterogeneous functions distributed geographically. Integration is supported by having the network first play a more active role in the accessing and communication of information, and second by providing the appropriate protocols for the distribution, coordination and negotiation of tasks and outcomes.

As described in this paper, the Organization Layer plays a central role in the EMN architecture. It is the connection between a real manufacturing environment and its implementation as a multi-agents system. This layer is also a platform for the negotiation and coordination activities between antagonistic EMN-nodes. The different mechanisms defined in the three first layers of the architecture provide the support for distributed knowledge base but also for all types of communication. They are instantiated according to the EMN-nodes identified at the Organization layer. In addition, the organization model provides conceptual links betwen the EMN-nodes and identifies interactions between them in order to make them solve antagonistic problems. The resolution of distributed problem solving is done by applying the coordination and negotiation protocols defined at the Coordination Layer according to the identified EMN-node interactions on the organization model. The schemata we define at this layer are the main elements to support distributed problem solving. The way we intend to use them is presented at the Coordination Layer.

## Acknowledgement

We would like to thank the CORTES and CARMEMCO project members which have contributed through their comments to the development of this Enterprise Management Network Architecture.

## References


[1] Adler, M.R., and Simoudis, E.
Integrated Distributed Expertise.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[2] Alford, M.W., and all.
*Distributed Systems - Methods and tools for specification.*
Springer-Verlag, 1985.
Lecture notes in Computer Science 190.

[3] Barr, A., Cohen, P.R., and Feigenbaum, E.A.
*The Handbook of Artificial Intelligence, Volume 4.*
Addison-Wesley Publishing Company, Massachusetts, 1989.

[4] Bond, A.H., and Gasser, L.
*Readings in Distributed Artificial Intelligence.*
Morgan Kaufmann, 1988.

[5] Chen, P.P.
The entity/relationship model: toward a unified view of data.
*ACM Transaction on Database Systems* 1(1):9-36, 1976.

[6] Corkill, D.D., and Lesser, V.R.
The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 748-755. Morgan Kaufmann Publishers, Inc., 95 First Street, Los Altos, CA 94022, 1983.

[7] Date, C.J.
*An introduction to data base systems.*
Addison-Wesley Publishing Company, Massachusetts, 1981.

[8] Date, C.J. and White, C.J.
*A guide to SQL DS.*
Addison-Wesley Publishing Company, Massachusetts, 1989.

[9] Davis, R., and Smith, R.G.
Negotiation as a Metaphor for Distributed Problem Solving.
*Artificial Intelligence* 20:63-109, 1983.

[10] Decker, K., and Lesser, V.
A Scenario for Cooperative Distributed Problem Solving.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[11] Doumeingts, G.
*GRAI method: A design methodology for computer integrated manufacturing systems (in French: Methode GRAI: methode de conception des systemes en productique).*
PhD thesis, Laboratoire GRAI, Universite de Bordeaux, Bordeaux, France, 1984.

[12] Durfee, E.H., and Lesser, V.R.
Using Partial Global Plans to Coordinate Distributed Problem Solvers.
In *Proceedings of 10th International Joint Conference on Artificial Intelligence.* Milan, Italy, 1987.

[13] Durfee, E.H., and Montgomery, T.A.
A Hierarchical Protocol for Coordinating Multiagent: An Update.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[14] Engelmore, R., and Morgan, T.
*Blackboard Systems.*
Addison-Wesley, 1988.

[15] Erkes, K., and Clark, M.
*Public domain report number 1.*
Technical Report, ESPRIT Project 418, Open CAM System, April 1987.

[16] Fox, M.S.
*Organization Structuring: Designing Large, Complex Software.*
Technical Report CMU-CS-79-155, Computer Science Department, Carnegie Mellon University, 1979.

[17] Fox, M.S.
An Organizational View of Distributed Systems.
*IEEE Transactions on Systems, Man, and Cybernetics* SMC-11(1):70-80, 1981.

[18] Fox, M.S., and Sycara, K.
Overview of the CORTES project: a Constraint Based Approach to Production Planning, Scheduling and Control.
*Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.* , May, 1990.
Submitted for publication.

[19] Gasser, L., and Huhns, M.N.
*Distributed Artificial Intelligence, Volume II.*
Pitman Publishing & Morgan Kaufmann Publishers, 1989.

[20] Gasser, L., Braganza, C., and Herman, N.
MACE: A Flexible Testbed for Distributed AI Research.
In Michael N. Huhns (editor), *Distributed Artificial Intelligence*, chapter 5, pages 285-310. Pitman Publishing & Morgan Kaufmann Publishers, 1987.

[21] Hayes-roth, F.
Towards a framework for distributed AI.
1980
In: Randy Davis Ed, Report on the workshop on distributed AI, SIGART Newsletter.

[22] F. Hayes-Roth and V.R. Lesser.
Focus of Attention in a Distributed Logic Speech Understanding System.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 27-35. 1977.

[23] Hayes-Roth, B.
A Blackboard Architecture for Control.
*Artificial Intelligence 26* , 1985.

[24] Huhns, M.N.
*Distributed Artificial Intelligence.*
Pitman Publishing & Morgan Kaufmann Publishers, 1987.

[25] Huhns, M.N., Bridgeland, M.L., and Arni, N.V.
A DAI Communication Aide.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[26] Hynynen, N.J.
*A framework for coordination in distributed production management.*
PhD thesis, Acta Polytechnica Sandinavica, Helsinky, Finland, 1988.

[27] Klein, M.
Supporting Conflict Resolution in Cooperative Design Systems.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[28] *Knowledge Craft*
Carnegie Group Inc, Five PPG place, Pittsburgh PA 15222, 1985.

[29] Lesser, V.R.
Cooperative Distributed Problem Solving and Organization Self-Design.
*SIGART Newsletter* :46, October, 1980.

[30] Meleze, J.
*Approches systemiques des organisations.*
Paris, Editions hommes et techniques, 1979.

[31] Mullender, S.
*Distributed Systems.*
ACM Press, New York, N.Y., 1989.

[32] Panse, R.
CIM-OSA - A vendor independent CIM architecture.
In *Proceedings of CIMCON '90,* pages 177-196. Gaithersburgh, MD 20899, 1990.

[33] Parunak, H.V.D.
Toward a Formal Model of Inter-Agent Control.
In *Proceedings of 10th International Workshop on Distributed Artificial Intelligence.* Bandera, Texas, 1990.

[34] Ram, S., Carlson, D., and Jones, A.
Distributed Knowledge Based Systems for Computer Integrated Manufacturing.
In *Proceedings of CIMCON '90,* pages 334-352. Gaithersburgh, MD 20899, 1990.

[35] Roboam, M., Doumeingts, G., Dittman, K., and Clark, M.
*Public domain report number 2.*
Technical Report, ESPRIT Project 418, Open CAM System, October 1987.

[36] Roboam, M.
*Reference models and analysis methodologies integration for the design of manufacturing systems (in French: Modeles de reference et integration des methodes d'analyse pour la conception des systemes de production).*
PhD thesis, Laboratoire GRAI, Universite de Bordeaux, Bordeaux, France, 1988.

[37] Roboam, M., Zanettin, M., and Pun, L.
GRAI-IDEF0-MERISE (GIM): integrated methodology to analyse and design manufacturing systems.
In *Computer-Integrated Manufacturing Systems, Volume 2 Number 2,* pages 82-98. PO box 63, Westbury house, Bury street, Guilford, GU2 5BH, UK, 1989.

[38] Roboam, M., Fox, M.S., and Sycara, K.
*Enterprise Management Network Architecture - Distributed Knowledge Base support.*
Technical Report CMU-RI-TR-90-21, CIMDS, Carnegie Mellon University, Pittsburgh, PA, 1990.

[39] Roboam, M., and Fox, M.S.
*Distributed communication system: user manual.*
Technical Report, CIMDS, Carnegie Mellon University, Pittsburgh, PA, 1990.

[40] Sathi, A., Fox, M.S., and Greenberg, M.
Representation of Activity Knowledge for Project Management.
*IEEE, Transactions on Pattern analysis and Machine Intelligence* : 531-552, 1985.

[41] Simon, H.A.
*Model of man.*
John Wiley, 1957.

[42] Simon, H.A.
*The science of the artificial.*
Cambridge, Massachussets, The MIT Press, 1969.

[43] Sycara, K.
Resolving Goal Conflicts via Negotiation.
In *Proceedings of the Seventh National Conference on Artificial Intelligence [AAAI-88]*. 1988.

[44] Sycara, K., Roth, S., Sadeh, N., and Fox, M.S.
Distributed Production Control.
In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*. 1990.

[45] Sycara, K. and Roboam, M.
Intelligent Information Infrastructure for Group Decision and Negotiation Support of Concurrent Engineering.
In *Proceedings of the 24th Hawaii International Conference on System Sciences*. 1991.

[46] Sycara, K.
Negotiation Planning: An AI Approach.
*European Journal of Operational Research* (46):216-234, 1990.

[47] Tardieu, H., Rochfeld, A., and Colletti, R.
*La methode Merise, principes et outils.*
Paris, Les editions d'Organisation, 1983.

[48] Tardieu, H., Rochfeld, A., Colletti, R., Panet, G., and Vahee G.
*La methode Merise, demarche et pratiques.*
Paris, Les editions d'Organisation, 1985.

[49] Titli, A.
*Analyse et commande des systemes complexes.*
Toulouse, Cepadues Editions, 1979.