Agent-based Manufacturing : a database point of view

C. HANACHI
University of Toulouse I
Place Anatole France 31042, France.
Direct phone number: (33) 5 61 63 35 60
Fax numbers: (33) 5 61 63 37 98
Email address: hanachi@univ-tlse1.fr

Abstract

This paper examines three advanced services provided by Database technology that can support agent-based manufacturing systems: temporal modelling, activity representation and meta-modelling. We will explain how the combination of these three services, based on well-defined models, enables the description of dynamic behaviour useful to cooperate with the environment, manage contracts and exploit efficiency data describing other manufacturing sites. We will illustrate this paper with the description of an information agent devoted to manufacturing sites cooperation.

Keywords

Active and Temporal databases, Meta-Modelling, Information Agent.

1 INTRODUCTION

Agent technology can be used in two different ways in the context of manufacturing systems: it can be used inside a system to achieve a precise function like scheduling, planning, or at a higher level to guarantee the cooperation between several manufacturing sites. Obviously, these two kinds of applications are not exclusive.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35390-6_58

L. M. Camarinha-Matos et al. (eds.), *Intelligent Systems for Manufacturing*

This paper will explain how Database technology can be used to support Agent-based Manufacturing. We will illustrate the paper in the context of cooperation between several manufacturing sites even if the concepts involved are general enough to be widely used.First let us precise why Manufacturing Agents should be based on Database technology:

- Pre-existing manufacturing systems are already organised around a technical database that manages information provided by CAE, CAD, Planning, Scheduling and Coordination.
- DBMSs guarantee good properties in a multi-user environment: data coherence, data interchanges between various applications, data availability, efficiency, reliability, concurrent accesses, integrity constraints checking.
- New concepts, techniques and tools mostly inspired by AI, are today available and can be coupled with Databases to provide high level services like cooperative answering to users, collaborative work support, heterogeneous and distributed Databases federating.

To our knowledge, most of the research aiming at developing Agent features on top of databases has concentrated on the use of Active Databases and most of the time it was applied to information retrieval. (Berndtsson 1996) studies how active rule paradigm may be used to support Cooperative Problem Solving and more precisely addresses the following issues: 1) Task Sharing by proposing a method to generate active rules from high-level protocol specifications, 2) Result sharing based on notification sending between producers and consumers of results, 3)Distributed and Federated aspects by stating that a Corba-like distributed computer infrastructure is required and that active rule paradigm must be extended to take into account global events. (Kinny 1995) makes a comparison between active Databases and Agent Systems with respect to purpose, structure, functionality and implementation. More precisely, It compares the two paradigms for expressing the Belief-Desire-Intention Agent Architecture.
This work differs from the works mentioned above in two ways. It first differs from a conceptual point of view, since our solution combines three complementary paradigms : Active rules, Temporal model and meta-model. This combination offers a lot of new services. Secondly, our objectives are also different as we aim at applying this work to the context of manufacturing systems where for example time management is obviously required.

This paper is organised as follows. The second section presents the architecture of an information agent devoted to the cooperation between different sites : the different components and how they interact with one another. Section 3 gives a meta-data model that supports the information agent reasoning. Section 4 deals with the data evolution modelling. We use the temporal data model of (Navathe 1987) to describe the contracts and the partners behaviour through time, and we also

precise how to use it during a contract life-cycle. Section 6 describes the information agent activity (control, coordination) with active rules.

2 ARCHITECTURE OF THE INFORMATION AGENT

To ease the cooperation between several manufacturing sites in the current context of the information age (network, web, ...), each company must advertise its capabilities, manage supply and demand, and share information with the partners. (Afsarmanesh 1997) has shown the importance of information in Virtual Organisations and the necessity to federate it. In this paper, we explain how to « agentify » databases in order to support the cooperation between manufacturing sites by providing cooperative information, and also by managing some activities included in a contract life-cycle.

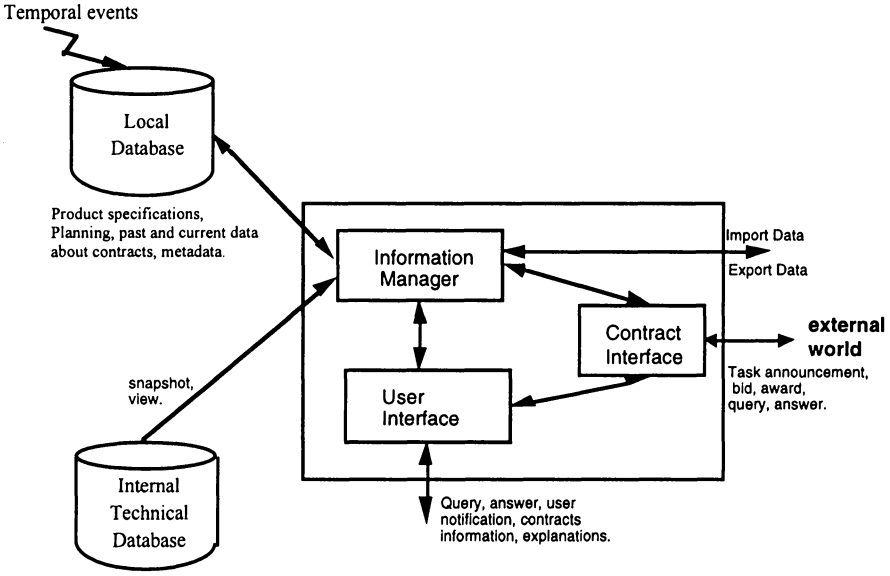

Figure 1. Information Agent architecture for Manufacturing sites cooperation.

Let us present the main features of an architecture facilitating such a cooperation and underline the essential role of Database technology to support such an architecture. This architecture is built around *an information agent* with the following features :

- it acts as an interface between a given company and the external world. It communicates with other companies information agents, provides advertising information to customers, and imports information from the environment to improve the knowledge the system has of the external world and about itself.

- it manages contracts (supply and demand) according to a precise protocol depending on the tasks and partners,
- its reasoning is based on a local database recording global and synthetic information about the company (its capabilities, its planning, ...) and information about the other companies of the market (capabilities, histories and states of the current and past contracts).
- it cooperates with the staff to inform them about important events (new contracts, deadlines, ...), to give them explanations, and to transform their needs into tasks announcement to be sent to suppliers.

To guarantee these functions, the information agent is organised around three functional components :

- **a contract interface,** which manages contracts both from a supplier and a customer point of view,
- **a information manager,** which records and provides information about companies and contracts. The information manager must be active : notify
- **a user interface,** which allows internal staff to communicate with the system in order to follow the contract during its life cycle and to get informed about important events.

The local database contains four types of data :

- information about the manufacturing system at a high abstract level : demand, shop capabilities (functions, limits, ...), quantitative goals, provisional plans upon large time horizons. This information is mainly extracted from the factory internal technical database. The stable data are stored in the local database in an appropriate format (snapshot, for efficiency reasons) while a view is created on the more variable data. For security reasons, it is important, at this level, not to export strategic information which could be used by competitive companies;
- Information about external environment : references on other companies(addresses, capabilities, histories, ...). This information may be imported by the system itself. Obviously, an update notification mechanism on remote information change is necessary to guarantee information freshness;
- Information about the current contracts as supplier and customer; and finally, meta-information describing the behaviour of each partner through time, the structure of each alliance and its evolution through time, and information on the contracts evolution.

From a database point of view, the effective implementation of this architecture raises several problems :

- **Data interoperability** between the different sites : the information that is exchanged between the different nodes must be understandable by each one and we must preserve the autonomy of each site. There are lots of solutions for that

(Common Internode Language, Mapping, STEP Files, ...) and different possible architectures (federation, multi-database, interoperable databases). Here, we will not focus on this interesting subject which has been well treated elsewhere : for example (Gardarin 1997) presents the IRO-DB project which is an object-oriented federated database and it explains how to integrate IRO-DB to the web. See also (Afsarmanesh 1997) for a discussion and solutions about data exchange in the context of virtual organisations.

- **Time** is also an essential aspect as a lot of information about contracts and the market are related to date and duration (deadline, delay, ...). We need to control the evolution of the current contracts during their life-cycle and record their history in order to improve the quality of the future contracts and master the relationships with partners and competitors. At another level, the market also evolves since new products, new customers and suppliers may arrive and the role of a given company can change through time (a company may be some times partner and other times competitor)... More precisely, we have to manipulate past, current and pro-active information concerning contracts and companies. The management of time will be assigned to the information manager. We will use, for that, the Temporal Relational Model (TRM for short) of (Navathe 1987) which integrates a Temporal Query Language (Temporal-SQL).
- **Information freshness and reactivity** : The local database must be updated automatically when important changes occur in the internal or remote. database. The staff must also get informed rapidly of new events (contract announcement, deadline, arrival of bids, ...). This event-driven behaviour can be implemented by active rules which are integrated in Active Databases.
- **Meta-reasoning and reflexive reasoning** : to select the partners and manage its relationship with them, to explain its reasoning, and to estimate its own competence the information agent needs meta-reasoning and for that global and meta-information about itself and the environment. We provide in next section a meta-data model to ease this reasoning.

3. A META-DATA MODEL TO SUPPORT THE MANUFACTURING-AGENT REASONING

We call it meta-model because it gives a global and time-oriented view of the structure and of the whole system behaviour. This meta-model records and structures information about :

- Different manufacturing sites : their role and behaviour in the current and past projects(as manager as well as contractor), the financial and social relationships between them (coalition, shares, owner, concurrent, ...), and their competencies and resources.

- Guidelines to choose a protocol according to the partner. The protocol used with a given company may change with time and according to the opportunities offered.
- The evolution and structure of each project (tasks).
- The alliances and their evolution.

All this information is recorded with their history (see section 4.) so that each partner behaviour is clear. Consequently, the information agent is able to 1) select partners 2) reject companies 3) choose the good protocol with each one 4) explain its past and current behaviour to internal staff.

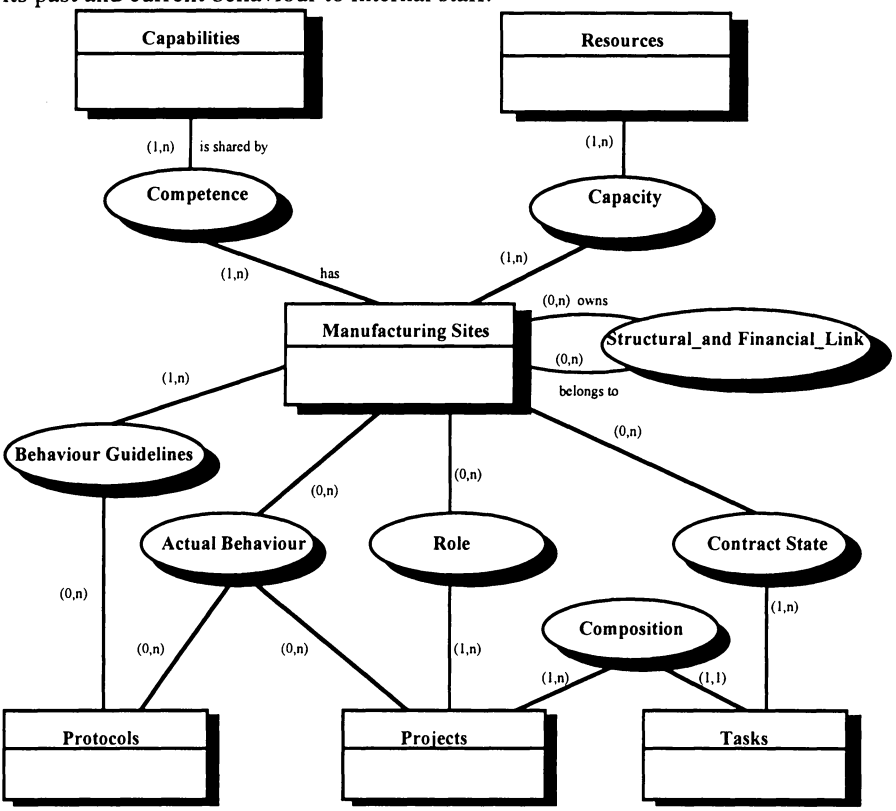

Figure 2.
Meta-Model supporting the information agent.

4. TIME MODELLING WITH TIME RELATIONAL MODEL (TRM)

To improve the temporal capacity of our system, we need temporal abstraction and high level language which, when integrated in active rules (see section 5), help to express system behaviour.

Temporal DBMSs have come to maturity in the context of Relational model as stated by the two following facts : the publication of a consensus glossary on temporal database concepts (Jensen 1994) and the definition of the TSQL2 language (Snodgrass 1995). At the opposite, in the Object-Oriented context there is no current consensus : researches and developments are still being active. (Fauvet 1997) provides a brief and current survey on Temporal-Object DBMSs such as OODAPLEX, OOTempSQL, OSAM/T, TOOSQL, TEMPOS.

For our purpose, and given the previous remarks, we will use the TRM relational model of (Navathe 1987) combined with TSQL.

TRM allows both ordinary relations (static relations) and time-varying relations (TVR). A TVR is a relation (that has, as any relation, a time invariant key, TIK) to which two time-stamp attributes are added, T_{start} and T_{end}. For a given value of TIK, there may be several tuples whose time-stamp attributes define the time interval during which the value is continuously valid. (TIK, T_s) and (TIK, T_e) are both candidate keys and the time intervals [T_s, T_e] must not overlap, since an attribute has a single value at any moment. Time-stamp attributes do not refer to the physical time (the date, when an attribute value is inserted or updated in the DB), but to logical time (i.e. the date provided by the user when the attribute value is valid in the real world). Thus both proactive and retroactive values may be represented in the DB. Dates are defined with any granularity (year, month, date, hour, minute, second).

TSQL is an extension of the SQL language dealing with the temporal aspects of queries. It allows the usual operations on dates and time intervals. Time-points are treated as degenerated time-intervals, and may be compared with the following operators : « before », « precedes », « after » , « follows » (just after), « adjacent » (precedes or follows), « overlap », « during » and « equivalent ».
In a query, the « WHEN » clause specifies the temporal relationships of the participating tuples, and is evaluated by examining the relative chronological ordering of tuples time-stamp. « INTERVAL » refers to the time-interval of a tuple, « DURATION » to the length of this interval, « T_start » and « T_end » to its bound. We will not present this language exhaustively, but only illustrate its capabilities by two examples based on the following relational schema, describing customers and their contracts according to the contract net protocol.

ConTract (CTname, CompanyName, BidExpirationDate, TaskExpirationDate, Task_specification, Eligibility_specification, Bid_specification...)
ConTractState (CTname, State : (Received, Possible, Bidding, Ignored, Rejected, Awarded, Working, Finished), T_s, T_e).
Companies (CompanyName, state : (partner, competitor, ...), T_s, T_e) /*the role of a company may evolve : sometimes partner, other times competitor*/

Query 1 : Possibly good customers : Company having awarded more than X contracts during the last three years ?

```
Select C1.CompanyName, count(*)
from ConTract C1, ConTractState C2
where C1.CTname=C2.CTname and State=« Awarded »
group by  C1.CompanyName
having count(*) > X
TIME_SLICE year [Now-3years, Now]
```

The time-slice clause states to what period of time the research is applied, with the required time-granularity.

Query 2 : History of contract CN97 since its bid expiration date ?

```
Select C2.State, C2.INTERVAL
from ConTract C1, ConTractState C2
where C1.CTname=« CN97 » and C1.CTname=C2.CTname
when C1.bidexpirationdate < C2.T_S
```

Query 3 : Doubtful Customers : Which company became competitor more than 3 times in any time interval of a month during last year ?

```
select Companyrname
from Companies
where  Companies.state = « competitor »
moving window 1 month
having count(*) >3
TIME_SLICE year [Now-1, Now]
```

The moving window clause enables to consider all time-intervals having a given length. For each time-interval having the length specified by the moving window clause, a group is formed of tuples which fall within this interval, and the having clause apply to these groups.

5. EVENT-DRIVEN BEHAVIOUR MODELLING WITH ACTIVE RULES

The dynamic control of a contract during its life-cycle will be implemented with the concept of Active Database where a rule based language is added on top of a conventional database. This rule-based language allows one to express behaviour and reactions to events occurred to the objects stored in the database or the environment (temporal event, signals from other softwares, ...). It can have varied applications among which (Widom 1996) : application invocation or user notification whenever a relevant situation appears; automatic checking of integrity constraints; automatic computing of derived data when base data are updated; etc.

All these functions are guaranteed without user intervention. This mechanism is interesting when one bears in mind that with conventional DBMSes, the user has to query the database periodically to know if an interesting event has occurred !

For that purpose, the rules have an Event-Condition-Action(ECA) form, of the following type : **When** <E> **if** <C> **then** <A> which may be interpreted as When the signal <E> occurs, if condition <C> is true then action <A> must be performed.

As previously said, the events may be related to Database operations (updating, deleting, creation), temporal events (duration analysis, time-out, frequency analyse), or signals from other applications. The condition part describes a situation that must be satisfied by the DB. It is expressed with a query addressed to the database. In our case, we will use a Temporal Data Model TSQL which is described in the previous section. The action part is a program which may contain : user notification, other tools invocation, database manipulation, ... Besides, a lot of Active Database Systems are currently available. They have been implemented on top of conventional DBMS (Starburst, Postgress, Hipac, Ode, O2, Samos, ...). For a review see (Widom 1996)).

Here are some examples of active rules which should ease the management of the contracts :

Rule 1 : Controlling the customers (Impossible contracts)
When a new contract CT is announced /* it is a database event captured after a SQL insert order*/
If the originator is doubtful /*can be computed by a temporal SQL query which analyses the behaviour of this customer during a significant period : see query 3 in section 4*/
then update the tuple corresponding to the announcement of CT : Time_End to Now (table ConTractState, see §4)
insert a new tuple stipulating that CT is « ignored » and Time_start =Now
inform the originator /*external program*/
inform the staff concerned with CT/*external program*/

Rule 2 : Possible Contracts (Rule 1 is supposed to have a higher priority than Rule 2).
When a new contract CT is announced /* it is a database event captured after a SQL insert order*/
If the Eligibility Specification are included in the Capacity of the company (SQL query)
then update the tuple corresponding to the announcement of CT: Time_End=now
insert a new tuple stipulating that CT is right now « possible »
Time_Start= now

Rule 3 : Extending Contracts Deadlines
When to day = deadline of bids submission of a contract CT /* it is a temporal event which generates a database query*/
If only 30% of the suppliers sent a bid/* it is a sql query*/
then inform the staff responsible of the contract CT /*external program*/
update the deadline of task CT /*SQL query*/
notify the suppliers /*external program*/

6. CONCLUSION

This paper has examined advanced concepts from Database technology to support the design and development of Agent-Based Manufacturing. The main research contribution of this paper is to combine three complementary paradigms (meta-modelling, temporal models and active rules) which jointly provide advantages to describe the dynamic and temporal behaviour of the agent. From the numerous examples we have given, we can conclude that these concepts are expressive enough to be considered seriously for the development of manufacturing agents. Besides, we believe this solution is easily implementable either from scratch (a simple DBMS) or by using a pre-existing active and/or temporal database.

7. REFERENCES

Afsarmanesh, H., and Camarinha-Matos, L. M. (1997) Federated Information Management for Cooperative Virtual Organizations, *in Proceedings of 8th international Conference on Database and Expert Systems Applications* (ed. A. Hameurlain & A. Min Tjoa , LNCS 1308), Toulouse.

Berndtsson, M and Chakravarthy, S. and Lings, B. (1996) Cooperative Problem Solving: A new direction for Active Databases, *International Symposium on Cooperative Databases for Advanced Applications*, Japan.

Fauvet, M.C. and Canavaggio, J.F. and Scholl, P. C. (1997) Modeling Histories in Object DBMS, *in Proceedings of 8th international Conference on Database and Expert Systems Applications* (ed. A. Hameurlain & A. Min Tjoa , LNCS 1308), Toulouse.

Gardarin G. (1997) Multimedia Federated Databases on Intranets : Web-Enabling IRO-DB », *in Proceedings of 8th international Conference on Database and Expert Systems Applications* (ed. A. Hameurlain & A. Min Tjoa , LNCS 1308), Toulouse.

Jensen, C.S. and Clifford, J. and Elmasri, R. and Gadia, S. and Hayes, P. and Jajodia, S. (editors) (1994) A consensus glossary of temporal database concepts. *ACM SIGMOD Record*, 23 (1).

Kinny, D. and Georgeff, M. and Bailey J. and Kemp B. D., Ramamohanarao K. (1995) Active Databases and Agent Systems-A Comparison, *Proceedings of the second Rules in Database SystemsWorkshop*, Athens.

Navathe, S. B. and Ahmed, R. (1987) TSQL- A language interface for histiry Databases », *in proceedings of IFIP conference on Temporal Aspects in Information Systems*, Sophia Antipolis.

Snodgrass, R.T., editor (1995) The TSQL2 temporal query language. *Kluwer Academic Publishers.*

Widom, J. and Ceri S. (1996) Active Database systems, .Morgan Kaufmann Publishers.