A Distributed Order Promise and Planning System for the Virtual Enterprise

Américo L. Azevedo(1,2), Jorge P. Sousa(1,2), João A. Bastos(2), César Toscano(2)
(1)FEUP – University of Porto
(2)INESC - Rua José Falcão, 110 - 4300 Porto, Portugal
email: {aazevedo, jsousa, jbastos, ctoscano}@inescn.pt

Abstract

This paper describes an «Order Promise» system to support decision making in the planning processes of Virtual Enterprises, viewed as networks of different units, namely plants, logistic centres and storage facilities. The system aims at improving customer due date calculation, and is intended to be a multi-site planning tool to co-ordinate local activities across the Virtual Enterprise network. It is based on the use of aggregate information and on a set of local rough capacity models.

This work is part of a broader project with a particular focus on the microelectronics industry. Companies of this industry are a good example of Virtual Enterprises, where a quick response to the customers needs and to unpredictable changes in production conditions is considered a major factor for success.

Keywords

Virtual enterprise, Decision Support Systems, available-to-promise, planning and control

1 INTRODUCTION

The current trend towards global markets (arising from the removal of trade barriers, the emergence of the newly industrialised countries as well as new

G. Jacucci et al. (eds.), *Globalization of Manufacturing in the Digital Communications Era of the 21st Century*

developments in manufacturing technologies) and the increasing customer orientation and expectations in terms of price, quality, and just-in-time delivery, give rise to new manufacturing paradigms and to challenging planning environments. Therefore, manufacturing management systems must have a much broader scope than in the past, and due to the increasing complexity of distributed production management, there is a strong need for powerful and flexible tools able to increase the degree of automation in the decision-making processes.

The rate of change in the market requires flexibility, not only in products but also in the business structure and business processes (Ross, 1997). Management faced with rapid changes must devise new strategies to deal with the competitive nature of the environment. The new strategy must enable flexibility, reduced time to market and reduced order cycle time.

Emphasis on customer needs is an emerging trend and it implies a close contact between the customer and the supplier. Whereas in traditional systems the supplier or manufacturer might be able to forecast customer needs, in today's environment this is frequently not possible. As we move from Make-to-Stock to Make-to-Order, the customer order decoupling point (where the processing order is pegged to a particular customer order) is tending to move towards earlier stages of the manufacturing process. In fact, until this point there may be a 'push' production system to deal with make to stock products and after this point, a 'pull' system to deal with the customer orders. The key point is that production of the customer specific item can only start when a firm customer order has been received.

In the past, the emphasis was on integration inside the manufacturing plant, namely CAD/CAM integration, integration of production planning and control systems, the development of new manufacturing processes and their control through shop floor control systems. Today, the emphasis has changed towards supply chain management and customer driven manufacturing, i.e. the integration of manufacturing and distribution planning and control systems. These systems were seen as separate and decoupled by storage points. The new approach takes a global logistic view, in order to get shorter product delivery lead-times and more accuracy in quoting due dates. A key element to allow this integration is the co-operation inter-enterprises.

That is why more and more companies are being organised as networks of different units, namely plants, logistic centres and storage facilities. In order to produce cost effective products within the time scales requested by the customers, many manufacturing enterprises are therefore becoming global businesses covering multiple manufacturing sites consisting of different shop floors, subcontractors and suppliers. The concept of Virtual Enterprise (VE) was born in this environment (Azevedo and Sousa, 1997, Soares et al., 1997).

Specifically, the work reported in this paper describes part of a Decision Support System that deals with the acceptance of customer orders and the aggregate

planning and control tasks in a virtual enterprise environment, comprising several distributed and co-operating entities. In this framework, planning and control activities are very complex, and have to take place both within the enterprise and across the whole supply network. The system, currently under development, enables the planners to model heterogeneous manufacturing systems, and transactions between customers, suppliers and subcontractors. In particular, the Order Promise System carries out the interaction with the customer, as a part of a more broad planning set of software modules encompassing fine planning, reactive control and monitoring (INESC et al., 1996) (INESC et al., 1997).

2 MAIN SYSTEM FUNCTIONALITY AND ARCHITECTURE

The Order Promise System, as a global enterprise co-ordinator, ensures that the individual manufacturing points interact to meet an overall production plan. It offers the following functionality:

- Real-time management of customer orders in the VE;
- Capacity-checked determination of delivery dates;
- Integration with legacy systems, namely Business Systems and Manufacturing Execution Systems.

The VE is viewed by the customer as a single closed unit. The Order Promise System determines realistic delivery dates for customer requests and tackles customer orders in an optimised way. In this process, we can identify two main scenarios. Either the product requested is already available-to-promise and can thus be immediately promised or it must be included into the current production plan by the Rough Planner engine.

Architecture of the system

The Order Promise System architecture involves five distinct components, working in an integrated manner. These components are:

1. the order promise clients and server – with the purpose of receiving and submitting the requests from customers;
2. the rough planning engine – for choosing and generating new solutions proposals;
3. the rough capacity model – for evaluating the solutions feasibility and the effect of the changes in each manufacturing point of the global network;
4. the information manager (global – GIM and local components – LIM);
5. the logistic model

Figure 1 presents the global architecture of the system. All components are capable of communicating to each other using a communication infrastructure which is CORBA compliant. Thus each component is able to directly interact in an objected-oriented way with other components, by using a CORBA IDL interface.

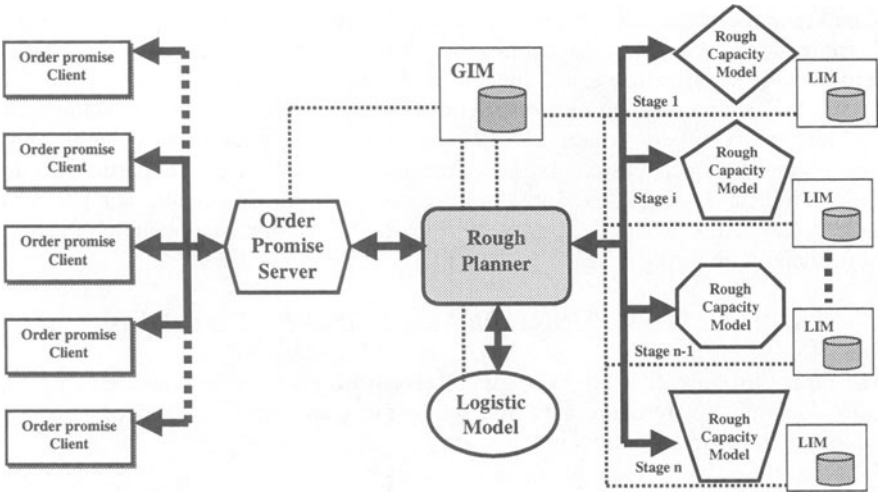

Figure 1 - Global Architecture of the Order Promise System

Several users are allowed to work simultaneously. In order to have this capability, the Order Promise system consists of a Server and several Clients. Customer requests are entered into the system using the Order Promise Clients. Since orders may be entered simultaneously, they have to be assigned priorities before the system tries to allocate any *available-to-promise* quantities or triggers the rough planning procedure.

Sequencing procedures for ordering customer orders are performed by using marketing priorities such as customer tiers or user defined priority rules (e.g. the order with highest priority is processed first, or among orders with equal priority, the one that has arrived first is chosen – First Come First Served).

3 PROCESSING OF A CUSTOMER REQUEST

The main objective of the Rough Planner engine is to determine feasible and reliable delivery dates for customer requests. As inquiries are received from customers, the system calculates the best order item delivery dates and according to the customer's acceptance, capacity is reserved. Usually, when a customer order comes into the sales department, the Order Promise System checks the availability of the requested product through *available-to-promise* (ATP). If the customer order can not be assigned to the ATP, the Rough Planning functionality is triggered in order to allocate the order, therefore creating a new plan.

An incoming request is divided into several suborders according to the VE production stages, the routing of the requested product and the capabilities of the units comprising the VE. There are a number of possibilities to process an order, using different intermediate schedules, and splitting order quantities over time

and/or virtual enterprise units (figure 2). This allocation is performed by checking feasibility through several Rough Capacity Models - one for each unit in the VE.

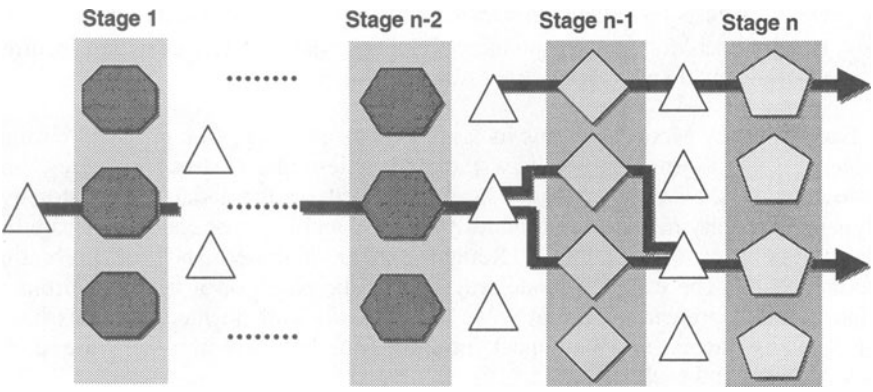

Figure 2 – Alternative order flows across the VE

After a negotiation process between the Rough Planner and each Rough Capacity Model, and after the merging of the partial offers received from each unit, a global solution is proposed, based on the evaluation of a previously defined objective function. Then, the system returns a set of feasible answers, with planned delivery dates and costs. As a result of this analysis, there may be a need for changes in the plan, involving one or more units of the VE. In the case the order is committed to the system, capacity or inventory is reserved (ATP is updated), so that future quotations are accurate. In order to ensure that the negotiations lead to a satisfactory result, the Rough Planner engine takes into account the counter-offers proposed by each Rough Capacity Model.

Therefore it is possible to promise a capacity-checked delivery date within a short time interval after the customer request. In fact, to satisfy the time constraint imposed by a customer request, the Rough Planner does not establish a new plan from scratch, but instead updates the current plan with the insertion of each suborder generated during the negotiation process.

In order to optimise and co-ordinate the order flow across the manufacturing network, the Rough Planner engine needs to have access not only to the costs of each VE unit involved (local level) but also to the costs and lead-times between stages (global level). Considering the VE as a matrix of nodes, where the columns are stages and the rows are plants (processing sites), and also with eventually some stock points between those stages, the following information is needed (and should be provided by the logistic model):

- Transportation and time costs, transport availability between each node of one stage and all nodes of the following stage;

- Inventory costs between stages. These costs are the holding costs in logistic nodes and they arise if, after completion of the work at one stage, an order must wait until the work at the next stage starts;
- Delivery costs (associated to packing and delivering of products);
- Penalty costs for non-compliance of delivery dates. These costs are incurred when orders do not respect due dates.

Each Capacity Model performs its tasks with local information provided on-line. Mainly, this information includes the description of resource capacities, and resource loads over the planning horizon. Based on these data, a constructive dynamic capacity model is continuously updated and used for checking feasibility or unfeasibility of the global Rough Planner inquiries and for presenting counteroffers. The capacity model engines can be based on a large spectrum of algorithmic approaches. Currently, we have already implemented solutions based on queuing theory and bottleneck optimisation, but several others are under development and evaluation.

4 ROUGH CAPACITY MODELS

The main purpose of the Rough Capacity Models is to provide a capacity checked answer to customer inquiries throughout the Virtual Enterprise. Obviously each site that belongs to the VE is singular. The role of each capacity model is exactly to capture its specific features, creating an interface mechanism with the higher level Rough Planner, to provide a planning mechanism to the entire Virtual Enterprise.

Therefore the main functionality of each Rough Capacity Model is to:

- Represent each site capacities in an easy to integrate software model;
- Evaluate customer inquiries through a centralised Rough Planner;
- Insert any new local order in the capacity model and pass it to the local scheduler.

4.1 Component Interface

The Rough Capacity Model component has the following interfaces defined to exchange data and control signals.

Interface with Rough Planner

A generic interface was defined between the rough planner and the rough capacity model components supporting the integration of quite different capacity models with the rough planner engine. These capacity models can implement many different concepts to represent the features of a specific site, but it should always be possible to interface them with the rough planner using this common CORBA's IDL interface.

Interface with the Local Information Manager
Another important interface provides the RCM (Rough Capacity Model) with the necessary data to build the model. We have defined three types of data: *Static Data* – Product and processes data which are very stable over time; *Dynamic Data* – Orders and Capacity data that vary over time; *Configuration Data* – User knowledge used to configure the capacity model.

The main source of data is the object oriented Information Manager component, which is responsible for providing all requested data and to collect and maintain it for the entire system.

Interface with the local Scheduler
Capacity models are connected to each site scheduler through specific interfaces with the purpose of providing the data needed to update the capacity model. Since the capacity model must reply to any inquiry with a capacity checked offer, and since that, each time a new schedule is released, the capacity profile is changed, the schedule data must be used to update the capacity profile in the RCM. This data connection is needed in order to guarantee that the RCM responses do not become worse over time, due to the lack of feedback.

4.2 Algorithms

The system architecture was designed in order to accommodate different algorithm implementations and instances. These instances can range from simple forward and backward strategies to queue theory based procedures, or more elaborated options such as local search heuristics. Currently, we are successfully testing an approach based on the bottlenecks as capacity constraints.

The bottlenecks choke the flow of materials in the factory and dictate the final throughput. Therefore they need to be fully utilised in order to maximise the resource utilisation. In a first stage, bottleneck optimisation focus on the bottlenecks and tries to optimise their operation. Then, for the operations that are before the bottleneck, it performs a backward schedule allocation. For the operations after the bottleneck, a forward schedule is used.

The main difficulty on the use of this approach is how to know what are the bottleneck resources prior to the full load of the model, mainly because the capacity model does not know the full load of orders that are going to be inserted in the future (since they are inserted sequentially as they arrive online).

The solution we have implemented is based on a simple concept, the dynamic bottleneck detection. This mechanism works as follows: at the beginning, an expert planner is asked to define a default bottleneck and also several bottleneck candidates. After this configuration is completed and the RCM component is initialised, the capacity model is ready to respond to inquiries and to insert new orders. To perform these tasks, the model uses the default bottleneck resources.

Therefore, when the component receives a new order to insert, it loads the default bottleneck and uses its capacities to make offers. In the meantime, the algorithm loads each of the bottleneck candidates in a faster and more aggregate way.

When the model reaches a near complete load state, the dynamic bottleneck detection mechanism checks if any of the bottleneck candidates has reached the condition of actual bottleneck of the system, for any of the specific periods of time, and if its load is larger than the current default bottleneck. If this condition is met, the current bottleneck is replaced by this new bottleneck candidate as default bottleneck. This action means that, for the next inquiry or order insertion, the resource capacity used will be computed for the new default bottleneck. Therefore, this dynamic bottleneck mechanism performs a continuous online seek of bottlenecks for all the periods of the planning horizon.

4.3 User configuration

Finally, it should be noted that the user plays a major role in this process, by interacting with the models through graphical user interfaces (GUI). A first GUI is concerned with the static configuration of the model, namely with the definition of the default and bottleneck candidates and with the definition of the level of data aggregation used in the construction of the model. A second GUI is used dynamically during run-time as a monitoring tool of the behaviour of the capacity model and as a fine tuning mechanism to the structure of the capacity model.

5 INFORMATION MANAGEMENT

Management of the object world is the main purpose of the Information Manager. Concurrent access to data, data persistency and coherency, interface to legacy systems, distribution of data among the various virtual enterprise units and an easy and object-oriented access to data are the main aspects addressed by this component (Orfali et al., 1996a).

The information management functionality is distributed across the global system infrastructure and is structured in two layers. The top layer, implemented by the Global Object Server, manages global data, i.e. data that aggregates local information that must be accessed in a global and simple way by global components. The lower layer is composed of a set of Local Object Servers, one for each virtual enterprise unit. Each of these object servers manages data that is local to the virtual enterprise unit: the objects describe the stage process specification for each intermediate product. An important aspect of this structure is the fact that local information is managed locally and by local people. The Global Object Server simply gets the aggregation of that data, namely customer orders and their splitting across the various virtual enterprise units.

Data described by the Object Model is further classified as static and dynamic. Descriptions of the production processes are very stable over time and this kind of data can be seen as static.

The interactions between the Information Manager and information access clients take place via three kinds of interfaces: the *Static View*, the *Action View* and the *Notification View* interfaces. The *StaticView* interface offers a request/response data retrieval mechanism: the client requests a specific class of objects and gets a copy of it as response. This type of data access is mainly used to retrieve static data from the object repository. Actions on the object world are accomplished through the *ActionView* interface: any object can be inserted, updated or removed by a client request. The last type of interface offers a more complex and powerful data access mechanism. During start-up, each external component subscribes to a certain class of objects. After that moment, any insert, update or remove action made to one of the objects of that class triggers an event that is transmitted to the component. The event attributes characterise the action and the new value for the object.

A typical interaction between an external component and the Information Manager is as follows:
1. During start-up, the component retrieves static data objects from the Information Manager through the *StaticView* interface;
2. Initial retrieval of dynamic data happens as a side effect of the Notification View interface creation. The component's own object model is then completely built. As time goes by and system activities take place, changes are made to the data managed by the Information Manager. All subscribed components receive in an automatic manner the right notification events and immediately update their models. In this way, the component specific object model is always consistent with the system object model.
3. Actions made to the object world are accomplished via the *ActionView* interface.

This implementation is characterised by the fact that the objects do not reside on the computer's main memory but on the relational data base system. Access to static data is not very efficient but this happens not to be an issue since it only occurs during the start-up of the components. Notification events are indirectly triggered from the data base system itself. Any modification that is made to the database triggers a notification event to all subscribed components. Basically, the object world maintained by the Information Manager is virtual as object oriented views of data are only built when requested by the component.

6 DESIGN AND IMPLEMENTATION ISSUES

An information system for this distributed environment should be able to support the activities outlined above. The performance of the system can be measured by the capability of providing fast response to customer inquiries and allow order acceptance with absolute commitment to due date and quantity, adjust orders to

changing customer requirements, optimising the order flow through the VE to increase reactivity to unplanned events, shorter throughput time and reduced WIP.

Although the importance of these features is generally recognised, the information systems that are commercially available seldom satisfy the planning and control needs of distributed enterprises. Moreover, the development of a system to fulfil these requirements and to work with increasingly distributed organisations (in order to respond to the complexity of their business operations) goes clearly beyond traditional client-server technology. We have therefore considered an infrastructure that can support the new emerging paradigm that brings together client-server and object-oriented technologies: the Distributed Object Computing. The use of a distributed object-oriented architecture (CORBA standard) provides features that significantly simplify the integration of applications into a distributed system (Otte et al., 1996) (Siegel, 1996) (Orfali et al., 1996b).

The virtual enterprise environment was modelled according to the OMT methodology (Rumbaugh et al., 1991) and the SEMATECH CIM Framework for Enterprise Modelling (Sematech, 1996). The resulting object model and its subsequent design describes the virtual enterprise in terms of virtual enterprise units, products, customer orders, demand plan, production orders, lots and process specifications. Basically, the model identifies the products offered to the customer and the way they are manufactured, stocked and transported throughout the production chain. Examples of data objects are customer orders, forecast orders, finished goods, intermediate products and stage process specifications. The object model describes all data objects and relationships that are of concern to the planning system.

The object model instantiation gives rise to objects that are concurrently accessed by all the components during their planning activities. Processing customer inquiries imply read and write access to data contained in this repository of objects. At any time, data views of each component should be coherent and consistent. The component responsible for the information management aspect of this planning system is the Information Manager component.

The modelling and design of the system was based on standard modelling patterns usable in several contexts (Gamma et al., 1995). With this approach it is possible to define a family of algorithms, encapsulate them separately, and make them interchangeable. Another idea followed in the design was the use of state machines. The key idea was the introduction of an abstract class to represent the states of the Rough Planner engine describing different operational states. Each subclass implements state-specific behaviour. This approach puts all behaviour associated with a particular state into one object. Because every state-specific code resides in a State subclass, new states and transitions can be added easily by defining new subclasses. Encapsulating each state transition and action in a class assigns to the idea of an execution state, a full object status. This feature leads to a

more structured and clear code. Introducing separate objects for different states clearly makes the transitions more explicit.

All interactions between the Rough Planner and other components, namely the Rough Capacity Models is carried out by a common IDL interface. Internal structures of these models and their required data may be completely heterogeneous, but as the interface is common, the internal operation and details are completely hidden from other components. This approach naturally creates a full 'plug and play' capability.

7 CONCLUSIONS

In this paper we have presented an «Order Promise» system intended to be used as a multi-site planning tool to co-ordinate local activities across the Virtual Enterprise network. The system aims at helping the planners to improve the efficiency and reliability of customer due date calculation. Several capacity models have been developed and implemented and are under preliminary evaluation by industrial companies.

As it was based on Object Oriented Technologies, we believe that the design is flexible and modular, allowing an easy integration with legacy systems, and the incorporation of particular «customised» models and algorithms. In practice, this may be an important feature, particularly because VE are in general very complex production systems, with heterogeneous management software applications and difficult to integrate «information islands».

8 REFERENCES

Azevedo, A. and Sousa, J. P. (1997) On The Design of an Order Promise System for Virtual Enterprises In *MCPL'97- Conference on Management and Control of Production and Logistics*IFAC Publications, Elsevier Science Ltd, Campinas -SP- Brazil.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns - Elements of Reusable Object-Oriented Software,* ADDISON-WESLEY.

INESC, IC and IPA (1996) *.X-CITTIC Design Overview* - Deliverable 2.2a

INESC, IC and IPA (1997) . *X-CITTIC Design of Global, Proactive Controller* - Deliverable 2.3

Orfali, R., Harkey, D. and Edwards, J. (1996a) *The Essential Client/Server Survival Guide,* John Wiley & Sons, Inc.

Orfali, R., Harkey, D. and Edwards, J. (1996b) *The Essential Distributed Objects Survival Guide,* John Wiley & Sons, Inc.

Otte, R., Patrick, P. and Roy, M. (1996) *Understanding CORBA (The Common Object Request Broker Architecture),* Prentice-Hall.

Ross, D. F. (1997) *Competing Through Supply Chain Manag.,* Chapman & Hall.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) *Object-Oriented Modeling and Design,* Prentice-Hall International, Inc.

Sematech (1996) SEMATECH, Austin - TEXAS.
Siegel, J. (1996) *CORBA Fundamentals and Programming,* J. Wiley & Sons, Inc.
Soares, A. L., Sousa, J. P., Azevedo, A. L. and Bastos, J. A. (1997) Using an Informal Ontology in the Development of a Planning and Control System In *ISIP'97 International Conference on Integrated And Sustainable Industrial Production,* Chapman & Hall, Lisbon - Portugal.

9 BIOGRAPHY

Américo Lopes de Azevedo obtained the degree of Electrical and Computers Engineer at the Faculty of Engineering of the University of Porto and is doing research work at INESC-UESP. He has been teaching in the Electronics and Computers Department where is assistant since 1988. Currently, he is preparing his PhD with a work in the area of manufacturing systems. His main interests lie in the Integrated Manufacturing area.

Jorge Pinho de Sousa is an Assistant Professor at the Faculty of Engineering of the University of Porto and researcher at INESC-UESP. He holds a Ph.D. in Operations Research (Université Catholique de Louvain, Belgium, 1989). His main interests are on Operations Research, Combinatorial Optimization, Project Management, Industrial Organization, Production and Operations Management.

João Bastos obtained the degree of Mechanical Engineer at the Faculty of Engineering of the University of Porto and is doing research work at INESC-UESP. He holds a MSc. in Industrial Informatics and is currently undertaking his PhD work in the area of intelligent manufacturing systems.

César Toscano obtained the degree of Electrical Engineer at the Faculty of Engineering of the University of Porto in 1985. He has been doing research and development activities at INESC since 1985 and teaches in the Computer Engineering Department of Instituto Politécnico de Gaia since 1996. His main interests are on Distributed Systems and Object Oriented Methodologies.