Appeared in Proceedings of WETICE '97, IEEE Computer Society Press, pp. 36 - 42, 1997

Agent Based Design and Simulation of Supply Chain Systems

Mihai Barbuceanu, Rune Teigen and Mark S. Fox
University of Toronto,
4 Taddle Creek Road, Rosebrugh Building,
Toronto, Ontario, Canada, M5S 3G9
{mihai,rune,msf}@ie.utoronto.ca

Abstract

We report on the use of our own agent and coordination technology to model, design and simulate global, distributed supply chains. We show by non-trivial examples how supply chains can be naturally modeled, simulated and improved in this way, within a short development time and with reduced human resources. As the agent technology was primarily built for implementation and control of distributed systems, the simulation models can be reused with minor modifications for actually controlling distributed supply chains. In this way, the presented agent technology gives us a powerful approach to life-cycle support of supply chain information architectures.

1 Introduction

The agent view provides both conceptualizations and technologies to construe and construct systems that interoperate across networks linking people, organizations and machines on a single virtual platform. Conceptually, agents bring a unifying view of software as autonomous, trusted, integrable and cooperative entities that work across the many boundaries that currently separate software systems. Technologically, agents bring about new communication, coordination and encapsulation frameworks aimed at providing value to users by unification and integration of the more or less disparate previous efforts in fields like artificial intelligence, databases, networks, user interfaces, etc.

We are interested in applying agent technologies to designing and controlling the dynamic behavior of the supply chain. The supply chain of a modern enterprise is a world-wide network of suppliers, factories, warehouses, distribution centres and retailers through which raw materials are acquired, transformed into products, delivered to customers, serviced and enhanced. In order to operate efficiently, supply chain functions must work in a tightly coordinated manner. But the dynamics of the enterprise and of the world market make this difficult: customers change or cancel orders, materials do not arrive on time, production facilities fail, workers are ill, etc. causing deviations from plan. In many cases, these events can not be dealt with locally, i.e. within the scope of a single supply chain "agent", requiring several agents to coordinate in order to revise plans, schedules or decisions.

Building on our previous work in agent architectures and coordination [1], in this paper we explore the application of agent technology to the design and simulation of complex supply chains. The goal is to use our agent framework to quickly prototype a supply chain (either modeling an existing one or designing a brand new one), put in place the required coordination protocols for agent cooperation and interaction and then simulate the operation of the system, collect measurements of relevant parameters, evaluate performance and finally modify the structure and dynamic behavior of components to improve the overall operation. The hypothesis that we test in this research is that the agent approach in general and our technology in particular are adequate and efficient for supply chain analysis and design, given that (i) closed form analytical solutions are too difficult for complex multi-tiered structures like supply chains and (ii) customized simulation analysis is very time consuming and usually non-reusable. The paper starts with an overwiew of the agent technology we are using, then presents the modeled supply chain and its analysis, and ends with evaluations, concluding remarks and future work.

2 Agent and Coordination Technology

While there are many facets of agenthood, for supply chain dynamic behavior we are especially interested

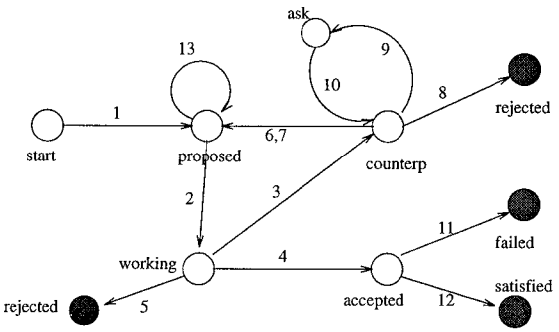

Figure 1. Graph representation of Customer-conversation.

```
(def-conversation-rule 'lep-1
 :current-state 'start
 :received '(propose :sender Customer
             :content(customer-order
                      :has-line-item ?li))
 :next-state 'order-received
 :transmit '(tell :sender ?agent
              :receiver customer
              :content '(working on it)
              :conversation ?convn)
 :do '(update-var ?conv '?order ?message))
```

Figure 2. Conversation rule.

in modelling coordination, understood as the process of *managing dependencies between activities* [4]. We thus start with describing our coordination technology as embedded into our coordination language (named COOL). The major idea is that agents' behavior is guided by the plans they use to achieve their goals. Operating in a multi-agent environment, agents' plans explicitely represent interaction with other agents, by exchanging messages (hence the name conversation plan). Agents can not predict the exact behavior of other agents, but can delimitate classes of alternative behaviors to be expected. As such, agents plans are conditional over the actions/reactions of other agents. Finally, as agents' plans may be initially incomplete or inaccurate, our agents are endowed with knowledge acquisition capabilities enabling them to extend and modify plans during execution (these are reported in [1]). Details follow.

Conversation plans are descriptions of how an agent *acts* and *interacts* in certain situations. A conversation plan consists of states (with distinguished initial and final states) and rule governed transitions together with a control mechanism and a local data-base that maintains the state of the conversation (see [5] for theoretical accounts of similar organizations). The execution state of a plan is maintained in *actual conversations*. For example, the conversation plan in figure 1 shows how the `Customer` agent interacts with the `Logistics` agent in a supply chain, when the former proposes an order to the latter. After proposing the order, the `Customer-conversation` goes to state `working` where it waits for `Logistics` to either accept, reject or counter propose. If `Logistics` accepts, then the `Customer` waits for the finished order (which can end in success or failure). If `Logistics` counter proposes, a new iteration starts, or the counter proposal is rejected, or clarifications are asked. In each non-final states rules specify how the agent interprets incoming messages, how it updates its status and how it responds with outgoing messages. A conversation plan describes an interaction from the viewpoint of an individual agent (in figure 1 the `Customer`). For two or several agents to "talk", the executed conversation plans of each agent must generate sequences of messages that the others' conversation plans can process (according to a mutual comprehensibility assumption that we make).

Conversation rules describe the actions that can be performed when the conversation is in a given state. The rule in figure 2 for example, states that when `Logistics`, in state `start`, receives a proposal for an order from the `Customer`, it should inform the sender that it has started working on the proposal and go to state `order-received`. Note the liberal use KQML-like [3] communicative actions for describing the exchanged messages (but the approach is essentially independent from KQML).

Error recovery rules (not illustrated) specify how incompatibilities (caused by planning or execution flaws) among the state of a conversation and the incoming messages are handled: for example by changing the state, discarding inputs, changing the plan, starting new conversations, etc.

Control. The framework also defines mechanisms by which agents can carry out may conversations in parallel, a hierarchical organization of conversations allowing parent conversations to control the execution or child conversations, a more complex typology of rules including various forms of event/condition triggered rules, and, finally, a mechanism allowing a conversation to be suspended (with preserved state), other conversation to proceed and the suspended conversation to be resumed when certain conditions related to the agent environment and conversations are satisfied. All these provide flexible control handles allowing the use of conversations as generalized processes that capture

both interaction and local processing.

3 Supply Chain Design and Simulation

We are designing a brand new enterprise manufacturing personal computers and we wish to simulate its supply chain, measure and evaluate performance and improve behavior. The agent based design of the supply chain is represented in COOL, and all simulation is equally done in COOL using the above described mechanisms.

3.1 Enterprise Structure

The Perfect Minicomputer Corporation (PMC) (figure 4) is a small manufacturer of mother boards and personal computers situated in Toronto, Canada. The minicomputers are sold to customers in two markets, Canada/USA and Germany/Austria. To satisfy the different standards of keyboard and power supply in the two markets, the computers need to be slightly differentiated, and are regarded as two distinct products. The mother board is PMC's third product sold to the computer industry of the Canada/USA market.

Plants and Production. PMC is a vertically integrated company. In addition to the assembly of the finished computer systems (computer, monitor and keyboard), they assemble the motherboard and the computer boxes (without power supply) themselves in separate plants in Toronto. Each plant has a `Planning`, a `Materials`, a `Production`, and a `Dispatching` agent. The `Planning` agent is responsible for production planning. The `Materials` agent handles raw product inventory (RPI), the on-order data base for raw products, and all reception of raw products. The `Production` agent handles production and the work in progress inventory, and has the knowledge of the plant architecture. The `Dispatching` agent handles the finished goods inventory (FGI) and all shipments from the plant. In each plant we also have a set of workstations, bins, and stocks. The workstations are production units with a set number of lines giving the number of units that can be processed simultaneously, a scrap rate (in percent), and a production time for each unit of a given product. The production capacity of the workstation will be given by the number of lines times throughput rate (1 / production time) minus scrap. Each workstation is modeled as an agent. The storage areas between workstations are modeled as *bins*. Each bin has a maximum inventory level, which is the inventory level where the bin is full, hence no further products can be entered. There is a single bin agent in each plant, which is responsible for all bins in the plant. Each plant has two stocks areas, the RPI for incoming components or raw materials, and the FGI on the other end of production. Production is modelled as strictly pull production, where workstations finish products as long as the output bin is not full, and start products as long as the input bin is not empty. Production ceases when weekly production goals are achieved.

Markets and Distribution Centers. PMC also owns and operates their two distribution centers, one in Detroit for the Canada/USA market (`dc-us`), and one in Hamburg for Germany/Austria (`dc-ger`). All computers are distributed through these two distribution centers. All mother boards sold to external customers are distributed through the Detroit distribution center. Each DC is modeled as an agent.

Suppliers and Customers. Each external supplier is modeled as an agent. PMC has a `Purchasing` agent which is responsible for communication with suppliers. The `Purchasing` agent has knowledge of which parts to order from which suppliers. Three types of customers are identified for each product in each market, a, b, and c-customers, with a-customers being most important. Customers are modeled in one `Customer` agent for each market. The `Sales` agent in the company is responsible for communication with customers.

Transportation. A `Transport` agent is defined to handle transportation. This agent has knowledge of transportation times and capacities, and damage rates where applicable. It also keeps logs on tranports currently underway. Deliveries from plant to distribution centers is modeled with uncertain tranportations times (normally distributed), and in some cases limited capacity. Three types of carriers are used; boat, truck, and plane. Internal transportation from plant to plant is modeled as instantanious, and with unlimited capacity. All transports from external suppliers are the responsibility of the suppliers and are therefore not addressed in the model.

3.2 Coordination Processes

Production Planning. Production is planned through lists of goals for this week and a number of future weeks. These plans propagate upstream through the internal supply chain, and come back downstream as plans of delivery. On the way upstream each agent contributes with its own knowledge.

To examplify the use of conversation plans and rules, let's look at the issuing of demand-forecasts, which start production planning. (The demand-forecast gives the expected number of units ordered for this and coming weeks.) The `Sales` agent has a conversation plan for distributing demand-forecasts to the distribution

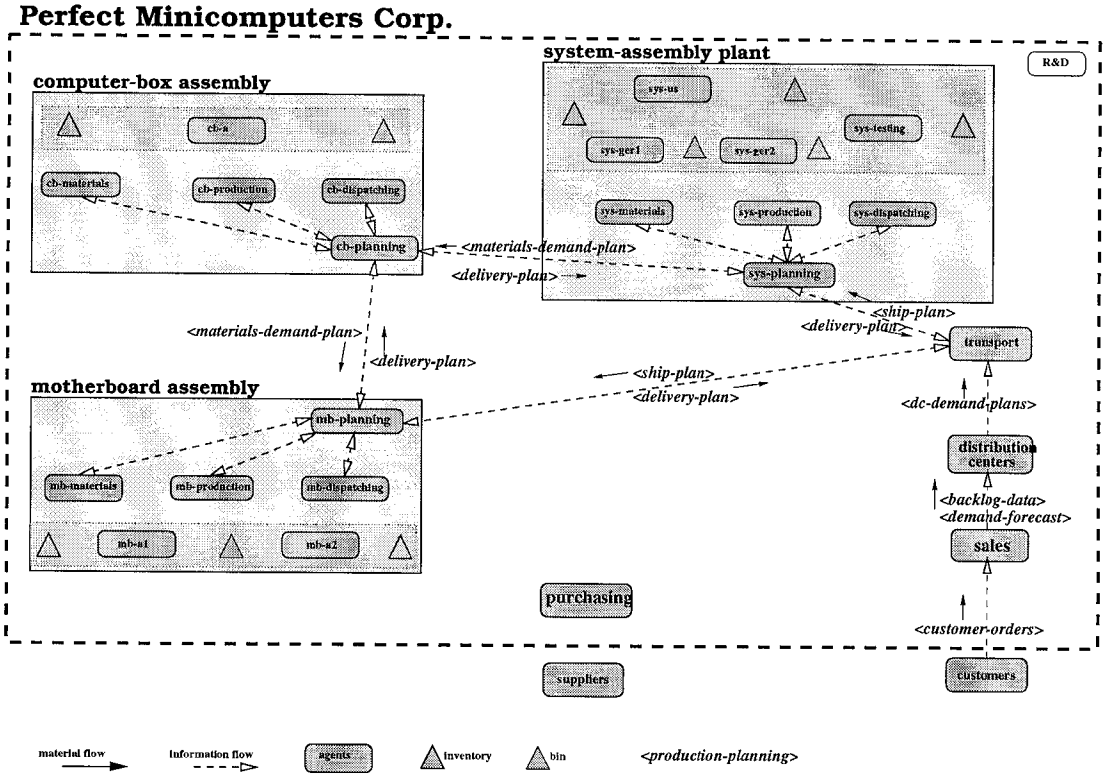

Figure 3. The Perfect Minicomputer Corporation.

centers. When a *demand-forecast-conversation* is created, the first rule of the conversation plan applies a specific method to compute the demand-forecast. The next rule of the plan prepares the data for sending, and rule `dfc-3` (figure 5) sends the message. The ?next-dc-forecast variable contains the demand-forecast for the market of the DC agent that is bound to the ?next-dc variable.

A demand-forecast message from `Sales` creates a *demand-plan-conversation* in the DC's. The rules of these *demand-plan-conversation*-s use knowledge of the DC's inventory levels. *DC-demand-plans*, defining the targetted quantity of each product arriving at the DC at the end of this and coming weeks, are made and sent to the `Transport` agent (and similarly creates a corresponding conversation in the `Tranport` agent). `Transport` knows how much is onway to the DC, and can therefore make *ship-plans*, defining the quantity of each product that should be shipped from a plant to a given DC at the end of this week and a number of future weeks. The *ship-plans* are sent to the planning agents of the plants concerned.

The aim of a plant's `Planning` agent is to convert the incoming *ship-plan* (if it has external customers) and *materials-demand-plans* from the next downstream plants (if it has internal customers) to the plant's

```
(def-conversation-rule 'dfc-3
 :name 'dfc-3
 :current-state 'sending-forecasts
 :such-that
   '(and (get-conv-var ?conv '?dc-left)
      (get-conv-var ?conv '?ready-to-send))
 :transmit
  '(tell
    :sender ?agent
    :receiver ?next-dc
    :content (:demand-forecast
                ?next-dc-forecast)
    :intent sending-demand-forecast
    :conversation ?convn)
 :do-after
  '(progn
    (put-conv-var ?conv '?dc-left
      (rest (get-conv-var ?conv '?dc-left)))
    (put-conv-var ?conv '?ready-to-send nil))
 :next-state 'sending-forecasts)
```

Figure 4. Sending Forecasts Conversation Rule

own *materials-demand-plan*-s for all internally supplied parts. These are sent to the next plants upstream. A *materials-demand-plan* defines the number of units of a given product the plant needs this week and a number of future weeks. To calculate the *materials-demand-plan*-s the `Planning` agent will use data from the other agents in the plant.

The *materials-demand-plan*-s will move upstream till they meet a last planning agent in the internal supply-chain. This agent will make *delivery-plan*-s for each customer (next plants downstream, or transport for deliveries to DC-s) , defining the number of units the plant will deliver this week and a number of future weeks. This is of course the total demand limited by part availibilities and production capacities. Upon receiving *delivery-plan*-s from upstream internal suppliers, a planning agent has the knowledge it needs to decide the *actual-build-plan* of the plant, i.e. the production goals for this and coming weeks. Thereby it will also make its own *delivery-plan*-s, and these plans will flow down-stream to the end of the supply chain.

Materials Ordering, Delivery, and Reception. From the *actual-build-plan*, via the BOM, the materials agent can calculate a *materials-order-plan* for externally supplied parts. The plans are sent to the purchasing agent, who transforms them to part orders for the suppliers. The supplier agents will send acknowledgment messages to the materials-agents. The materials agents update their on-order data base. Materials-shipments arriving at the plants are modeled as a messages sent by the suppliers to the materials agents. The materials agents update inventory and on-order.

Products Dispatching, Transportation, and Reception. Product tranportation from plant to DC is started through messages from dispathcing agents to the `Transport` agent. Arrivals at DC are done by messages from `Tranport` to the DC agent.

3.3 Dealing with Unexpected Events

Each agent within the enterprise records its own relevant data every week, building a data base that will be communicated to a `Simulation` agent at the end of the simulation and saved for later analysis. We measure parameters related to inventory levels and customer satisfaction. Examples include the values of all inventories, the company backlog, the incoming orders, the shipments from plants to DC-s, the average time from order arrival till product delivery, the percentage of shipments delivered on-time. We are especially interested in understanding the value of various coordination structures when unexpected disruptions occur and how coordination can be used to reduce the negative

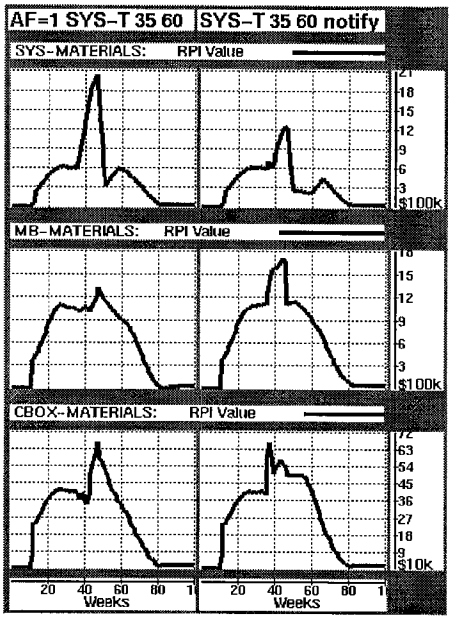

Figure 5. Effect of breakdown notifications over inventory levels

consequences of these events. A typical disruption is a machine breakdown. Such an event tends to increase the level of raw product inventory (RPI) in the plant where the breakdown occured as well as the inventories of the upstream and downstream plants, requiring more coordination to atenuate these effects.

To see how coordination can be used to deal with this problem, we have performed a series of experiments involving breakdowns of workstations in several plants and using various coordination mechanisms for dealing with them. In figure 5 for example, we assume a breakdown occured in the system test and assembly plant (in week 35, taking 12 weeks to repair and causing 80% of plant's capacity to be lost) and we show the RPI levels in case the plant's planning agent is not notified (left side) and in case it is (right side). (We assume the normal enterprise coordination strategy in which demand flows upstream and delivery plans flow downstream). The results show that the simple notification introduced reduces the average value of the raw product inventory at the system test plant with 26%. It also shows that for the upstream plants there is a noticeable increase of the same inventory. Globally however, the total inventory decreases with about 4% in average. But the most important consequence is avoiding the sudden take-off of the system test plant's stock which, in the non-notification case is more than tripled in the ten week period following the breakdown. The notification reduces the magnitude of the peek by

almost half.

4 Conclusions

The above supply chain system has 40 agents and about the same number of conversation plans. The entire specification takes about 7,500 lines of COOL code, plus about 2,000 lines for GUI-s. A typical simulation run over 100 weeks generates many thousands of message exchanges and takes less than 1 hour to complete (no optimizations attempted, and the system runs in an interpreted mode). The system was writen by one author, who hasn't a computer science background, in less than 2 months. Learning the underlying agent and coordination technology was done in another 2 months, during which time a simpler supply chain was built. (Some limited code sharing between these systems occured). Previously, we have built supply chain models that exercised coordination mechanisms for dynamic team formation and management [2] with similar results in terms of conciseness of representation and efficiency of development. We take these data as indications that (1) the agent and coordination model itself is natural and understandable and (2) the agent and coordination model is adequate to modeling systems like the supply chain. Being able to quickly model a supply chain as an agent community, encode it in COOL, run, evaluate and improve it is one advantage of the approach. Unlike other simulation or modeling techniques, the agent model constructed for analysis purposes can be reused to a very large extent for actually controlling the supply chain which has been modeled. The agent technology was built for this purpose in the first place. The agent structure, the format of messages, the coordination plans and rules can all be reused in the real supply chain environment with at most minor modifications. Similar reuse is to be expected across application domains, prompted by the explicit, structured object representations of agents, conversation plans, conversation rules, etc. This makes it possible to envisage libraries of such components being built and used to speed up and provide quality guarantees for modeling supply chain systems. Similar advantages with respect to inter-application reuse have been reported by [6] who have developed a similarly aimed system, based however on totally different agent technology that was not reported to handle migration towards system control as well.

Because the agent language supports distributed execution of agents (anywhere reachable by TCP/IP), the simulation can be run distributedly as well, which can be exploited in the global enterprise environment.

As for future work, there are many interesting directions that we intend to pursue. Along one direction, we are interested in balancing the internal "intelligence" of agents (given e.g. by the sophistication of the local scheduler used by the production planning agents) and the complexity of agent coordination. A more intelligent agent for example, by being able to solve more difficult problems locally, may perturbate less other agents by requesting them to modify their plans. Along another direction, we are expanding the study of coordination mechanisms able to handle unexpected events in a manner that minimizes perturbation and distraction.

5 Acknowledgments

This research is supported, in part, by the Manufacturing Research Corporation of Ontario, Natural Science and Engineering Research Council, Digital Equipment Corp., Micro Electronics and Computer Research Corp., Spar Aerospace, Carnegie Group and Quintus Corp.

References

[1] M. Barbuceanu and M.S. Fox. Capturing and Modeling Coordination Knowledge for Multi-Agent System. International Journal of Cooperative Information Systems, Vol. 5, Nos. 2 and 3 (1996) 275-314.

[2] M. Barbuceanu and M.S. Fox. Coordinating Multiple Agents in the Supply Chain. Proceedings of WET-ICE'96, IEEE Computer Society Press, 1996.

[3] T. Finin et al. Specification of the KQML Agent Communication Language. The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, 1992.

[4] T. W. Malone and K. Crowston. Toward an Interdisciplinary Theory of Coordination. Center for Coordination Science Technical Report 120, MIT Sloan School, 1991

[5] F. vonMartial. Coordinating Plans of Autonomous Agents, Lec ture Notes in Artificial Intelligence 610, Springer Verlag Berlin Heidelberg, 19 92.

[6] J. Swaminathan, S.F.Smith, N. Sadeh. A Multi Agent Framework for Modeling Supply Chain Dynamics. Robotics Institute, Carnegie Mellon University, December 1996.