Software Agents for Dynamic Supply Chain Management

Tom Wagner and **Valerie Guralnik Honeywell Laboratories**
3660 Technology Drive, MN65-2600
Minneapolis, MN 55418
{wagner_tom, guralnik_valerie, phelps_john}@htc.honeywell.com and John Phelps

Abstract

Some dynamic supply chain problems are instances of a class of distributed optimization problems that intelligent agents were made to address. Agents are thus a natural enabling technology for such problems. In this paper we describe the use of TÆMS agent technologies on a sterilized dynamic supply chain management problem.

Introduction

The *intelligent* agent paradigm is a natural fit to certain classes of dynamic supply chain problems because the paradigm focuses on coordinating the activities of loosely coupled distributed entities, e.g., raw material suppliers, shippers, manufacturers, distribution centers, and retailers (where each of these is represented by an agent). One goal of the paradigm is to enable agents to meet deadlines and resource constraints but also to be flexible, robust, responsive, and adaptive. With agents these behaviors are obtained without centralization and without assuming complete static knowledge *a priori*. Accordingly agents are well suited to dynamic supply chain problems in which frequent and timely adjustments to the flow of goods and production schedules are needed in order to leverage conditions in the marketplace or to take advantage of opportunities as they arise.

In this paper we present the use of TÆMS (Decker & Li 1998; Lesser, Horling, & et al) agent technologies on a sterilized version of a dynamic supply chain problem. In this application the agents work to manage the production schedule and material flow of a small build-to-order production line. It is important to note that the focus of this work is not on agent-based bidding schemes or price determination but is instead on reasoning about actual orders, production schedules, and material flow. This research also does not make strong assumptions about statistical characteristics of demand or product orders – the focus is not on setting up a steady state manufacturing schedule.

In this application we focus on the problem domain of manufacturing goods for the outdoor recreational market sector – specifically backpacks and sleeping bags. The actors in our example will include retailers and a fictitious

manufacturer called "MountainMan." Any resemblance to actual retailers or manufacturers in this sector or other market sectors is purely coincidental. The example itself is based on a real world situation though the work presented here is simulated and the problem specification should be viewed as an educated assessment of a real world situation.

It is important to emphasize that the focus of this work is not on negotiation protocols or market mechanisms – it is on the distributed optimization problem that occurs when agents have multiple interacting activities and the activities have individual deadlines and individual resource constraints. Even if we could build a single centralized representation of the problem space, the hybrid planning/scheduling (with interacting time limits, interacting resource constraints, and utility interactions) problem is intractable for any but toy problems. In this setting, provably optimal solutions require exhaustive search. Now consider what happens when the problem space is distributed and put into a dynamic setting. It is this space in which we operate. Complete discussion is beyond the scope of the paper (Wagner, Raja, & Lesser 2002) – the point is to understand that a large class of supply chain management problems fall into this very difficult problem space that has more in common with distributed scheduling than it does with market mechanisms or research whose focus is on communication protocols.

In this paper we examine the production scheduling problem of a small volume manufacturing line that produces build to order goods for private label customers. The line is managed by an agent that interacts with other agents that represent retailers. The retailer agents, raw material supplier agents, etc., all potentially need to solve the same class of problem as the manufacturing agent in this paper – we focus primarily on the small volume production line for simplicity and clarity. As shown in Figure 1 the small volume production line is an off-shoot from the primary production line of MountainMan. Generally with private label production a large order is produced via the primary production line and subsequent smaller orders, which are caused by unanticipated demand or customized variants of the goods, are routed to the smaller production line. The small volume production line supports small private label jobs, such as orders generated by unanticipated sales, that would interrupt the flow of the primary production line if scheduled for that line.
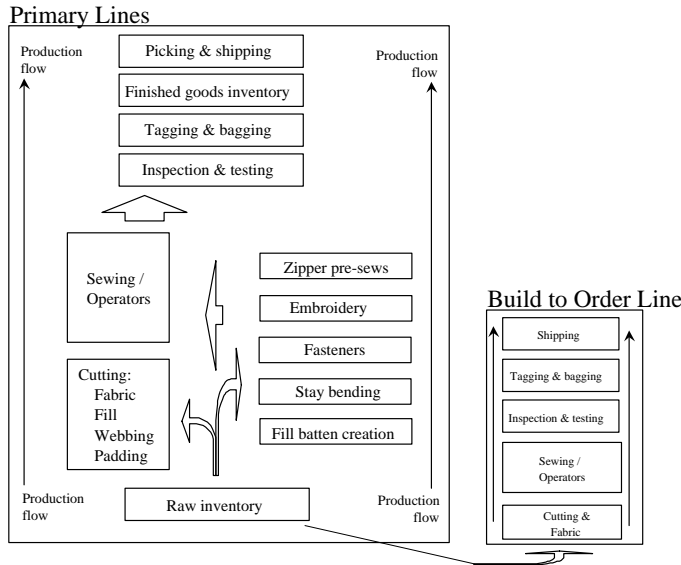
Figure 1: Primary and the Build-to-Order Production Lines

Figure 2: MountainMan's Overall Supply Chain

This is particularly true for custom variants or restocking orders in part because such orders are small but still require different fabric, fasteners, etc., but also because such orders generally have short term deadlines. In other words, when a retailer requests a small volume they generally want it in a few days rather than the period of weeks for which primary production line is geared. This type of production has different requirements than the standard main stream production operations. While minimizing raw and finished goods inventories are important for all manufacturing processes, this is particularly true for private label goods as the customer base is very limited (generally just one) and often portions of the raw materials are specific to the private label purchase. Additional requirements for the line include ending the day with the "tables empty" (no work-in-progress), maintaining zero inventory on finished goods, and building products to order only.

In our implementation, agents are situated at each of the involved sites. For example, one agent is located at MountainMan and one agent is located at each retailer. The MountainMan agent's job is to interact with the retailer agents and to determine production schedules for the MountainMan small volume production line accordingly. Note that the overall supply chain, shown in Figure 2, involves raw material flows, shippers, and distribution centers. In this paper we focus on the problem of coordinating production to meet dynamic demand at the retailers. Recent work includes assigning agents to shippers and reasoning about the implications of raw material flow to production scheduling, though aspects of this work are beyond the scope of this paper. The general flow of events across the agents is that when inventory levels fall below the specified threshold at a retailer, the retailer's agent places orders with MountainMan for replacement goods. The MountainMan agent reasons about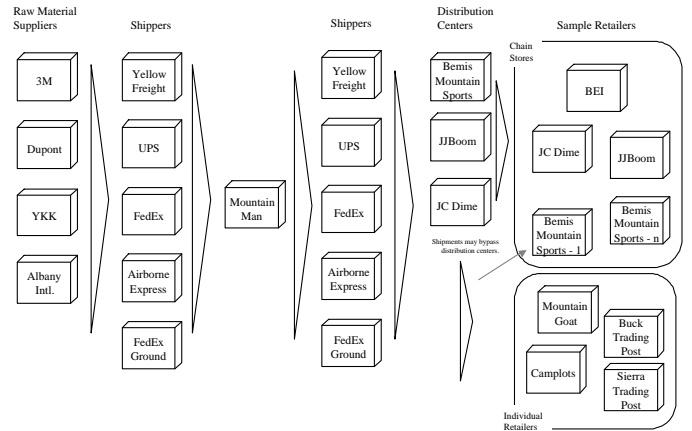 the new order and how it relates to currently scheduled production in terms of temporal constraints and overall value. It can then negotiate over delivery time with the retailer agent in order to optimize MountainMan's mix of goods. The use of agents for this example is what enables the retailers and MountainMan to coordinate dynamically and automatically to optimize their activities.[1]

In the next section we provide a high-level view of TÆMS agents and TÆMS technologies. In the section after that we return to the details of the application and illustrate the use of TÆMS agents for managing MountainMan's dynamic supply chain problem. We then discuss chains of interactions and identify selected related work, discuss limitations, experimental plans, and future work.

TÆMS Agents

We use the expression *TÆMS agents* to describe our agent technology because the cornerstone of our approach is a modeling language called TÆMS (Task Analysis Environment Modeling and Simulation). TÆMS is a way to represent the activities of a problem solving agent – it is notable in that it explicitly represents alternative different ways to carry out tasks, it represents interactions between activities, it specifies resource use properties, and it quantifies all of these via discrete probability distributions in terms of quality, cost, and duration. The end result is a language for representing activities that is expressive and has proven useful for many different domains including the BIG information gathering agent (Lesser *et al.* 2000b; 2000a), the Intelligent Home project (IHome) (Lesser *et al.* 1999), the DARPA ANTS real-time agent sensor network for vehicle tracking

---

[1]The issue of what criteria over which to optimize is deliberately unspecified. If we assume that MountainMan and the retailers have no direct relationship then MountainMan's goal is to optimize over its own local criteria. In general this is simply to maximize profit but other secondary items might be to keep the line fully utilized or to use particular machines on a regular basis. For this example the exact optimization criteria is unimportant because all of these aforementioned issues can be mapped into the TÆMS quality attribute over which the MountainMan agent optimizes. Subsequent sections contain more information on TÆMS and TÆMS agents.

(Vincent *et al.* 2001; Horling *et al.* 2001), distributed hospital patient scheduling (Decker & Li 1998), distributed collaborative design (Decker & Lesser 1995), process control (Zhang *et al.* 2001), agents for travel planning (Wagner *et al.* 2001), agent diagnosis (Bazzan, Lesser, & Xuan 1998; Horling, Benyo, & Lesser 1999), and others.

Figure 3 shows a TÆMS task structure like that used by the MountainMan agent in this application. The structure is a hierarchical decomposition of a top level goal which is simply to `Produce`. The top level goal, or task, has two subtasks which are to `Make Bags` and `Make Back Packs`. Each of these tasks are decomposed into subtasks and finally into primitive actions. Note that most of these are omitted from the figure for clarity. The details are shown for the `Make BEI Jasper` pack task – it consists of four primitive actions that are picking zippers and fasteners, cutting the webbing, cutting the fabric, and sewing the pack. The inter dependence of these tasks is modeled in TÆMS using an *enables non-local-effect*. The dotted edges (enablements) from tasks like cutting and picking to the sewing tasks indicate that these tasks must be performed first and be performed successfully for the sewing task to be performed. Note that all of the primitive actions (leaf nodes) also have Q (quality) and D (duration) discrete probability distributions associated with them. For simplicity in this paper we do not use uncertainty and all values will have a density of 100%. Picking the zippers thus takes 10 minutes (.16 hours) and generates a quality of one. Cutting the webbing has a similar allocation while cutting the fabric takes longer (1/2 an hour) and produces a higher quality (four). Sewing the packs takes over an hour and also produces a higher quality (four). The `sum()` function under the `Make BEI Japser` task is called a *quality-accumulation-function* or *qaf*. It describes how quality (akin to utility) generated at the leaf nodes relates to the performance of the parent node. In this case we sum the resultant qualities of the subtasks – note that the cutting of the fabric and the sewing operations dominate how well the bags are made in this application. Quality is a deliberately abstract concept into which other attributes may be mapped. In this paper we will assume that quality is a function of the amount of profit generated by the production of a given good.

In the sample task structure there is also an element of choice – this is a strong part of the TÆMS construct and important for dynamic supply chains. The `Make Back Packs` task, for example, has two subtasks joined under the `sum()` qaf. In this case the MountainMan agent may perform either subtask or it may perform both depending on what activities it has time for and their respective values. The explicit representation of choice – a choice that is quantified by those discrete probability distributions attached to the leaf nodes – is how TÆMS agents make context dependent decisions. In supply chain applications this is how the MountainMan agent sees that it has a choice of which products to produce and when.

By establishing a domain independent language (TÆMS) for representing agent activity, we have been able to design and build a core set of agent construction components and reuse them on a variety of different applica-
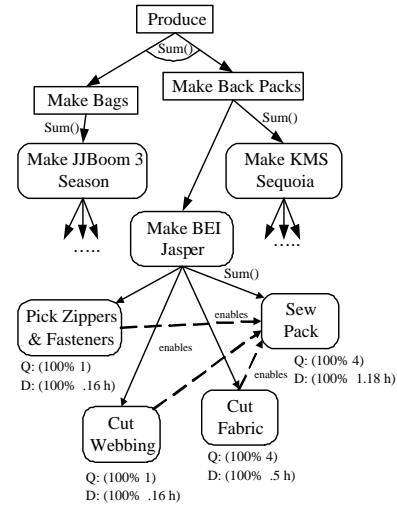

Figure 3: Sample TÆMS Task Structure for a Manufacturing Agent

tions (mentioned above (Lesser *et al.* 2000b; 2000a; 1999; Vincent *et al.* 2001; Horling *et al.* 2001; Decker & Li 1998; Decker & Lesser 1995; Zhang *et al.* 2001; Wagner *et al.* 2001; Bazzan, Lesser, & Xuan 1998; Horling, Benyo, & Lesser 1999)). TÆMS agents are created by bundling our reusable technologies with a domain specific component, generally called a domain problem solver, that is responsible for knowing and encapsulating the details of a particular application domain. In the BIG information gathering agent, for instance, the domain problem solver is a blackboard planner that knows how to model software products, build models of products from raw text data, and compare/recommend products to purchase. In another application the domain problem solver may be a process plan or a legacy database application. In each of these cases we abstract away from the details of the domain by creating a custom mapping function that translates the internals of the domain problem solver into TÆMS task structures that are then operated on by the rest of the system.

For this paper it is sufficient to know that TÆMS agents have components for scheduling and coordination that enable them to 1) reason about what they should be doing and when, 2) reason about the relative value of activities, 3) reason about temporal and resource constraints, and 4) reason about interactions between activities being carried out by different agents. A high-level view of a TÆMS agent is shown in Figure 4; everything except for the domain problem solver is reusable code. Note that each module is a research topic in its own right. The agent scheduler is the Design-to-Criteria (Raja, Lesser, & Wagner 2000; Wagner, Garvey, & Lesser 1998; Wagner & Lesser 2001; Wagner, Garvey, & Lesser 1997) scheduler and the coordination module is derived from GPGP (Decker & Li 1998). Other modules, e.g., learning, can be added to this architecture in a similar plug and play fashion.

In the supply chain application there are two types of domain problem solvers, those that manage the retailers' inventories and the MountainManagent that manages Mountain-
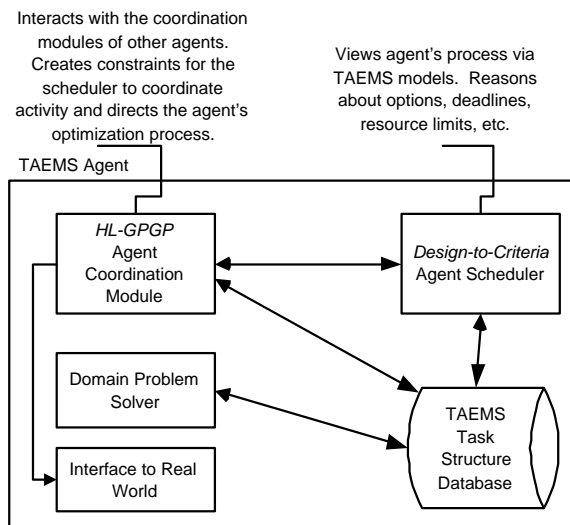
Figure 4: A Single TÆMS-based Agent Ready to Coordinate Its Activities With Other Agents

Man's production. The retailer problem solvers are of similar construction. Their function is to monitor purchasing activities and check inventory levels when purchases are made. If a good falls below a specified threshold, they reorder and negotiate with the MountainManagent to determine delivery times/dates. The MountainMandomain problem solver is different an instead reasons about MountainMan's production. It creates new candidate runs for new orders as they come in and remove jobs from the list of candidates if orders are canceled.

Dynamic Supply Chain Example

As mentioned in this example each retailer has a TÆMS agent that manages its local interests and orders products when appropriate. MountainMan also has an agent that interacts with the retailer agents, responds to order requests, negotiates delivery times, and manages MountainMan's production.[2] In this paper we focus on a subset of the supply problem and do not address interacting directly with shippers or raw material suppliers. Current work involving chains of interactions (discussed later) will be disclosed at a later date once corporate IP policies are cleared. The agent network is shown in Figure 5. This example has specific properties, requirements, and assumptions that frame the problem. A subset of the more notable ones are:

- Production is a single shift and scheduled from 8am to 4pm. However, the agents operate both day and night and adjust production schedules as necessary.
- All goods are shipped over night via an express carrier.
- Raw materials are always sufficient to support production.
- Orders may arrive at any time day or night.
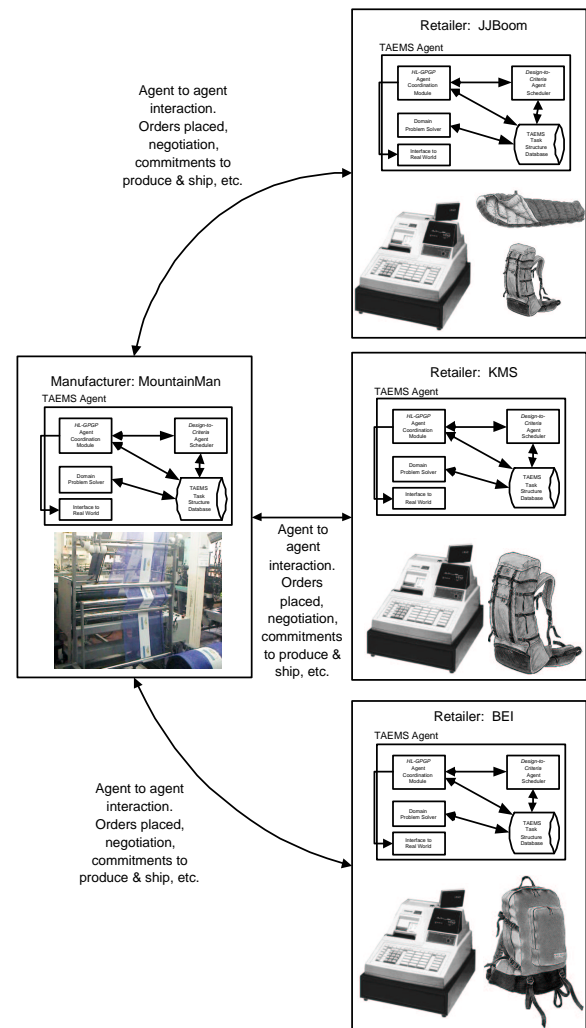- Retailers order goods in lots.

Figure 5: Each Company Has Its Own Agent That Manages Its Local Interests

- No WIP (work in progress) is left on the tables at the end of the day.
- Orders are not interruptable once they have begun.
- The TÆMS quality associated with production tasks is a function of the margins produced by different products.
- Production activities will be modeled as primitive actions in TÆMS at the grainsize of `Make Product X`.[3]
- All customers are equally valuable. If this were not the case, it too could be mapped into TÆMS quality associated with the production tasks.
- When orders arrive they have a desired delivery date/deadline (that is specified by the retailer agents).
- Production specifics: sleeping bag lots require four production hours, backpacks require two hours per lot.
- All TÆMS distributions are 100% certain (single valued functions).

---

[2]The actual computation about which items to produce and when is performed by the DTC (Raja, Lesser, & Wagner 2000; Wagner, Garvey, & Lesser 1998; Wagner & Lesser 2001; Wagner, Garvey, & Lesser 1997) TÆMS agent scheduler.

[3]This is sufficient for scheduling and selection in this example. The focus is on the interaction across agents – the more detailed MountainMan job shop scheduling problem is addressable with the TÆMS technology and other well defined techniques.

The current simulated world time is 10am. MountainMan's TÆMS task structure, which describes MountainMan's current production options and requirements at 10am, is shown in Figure 6. MountainMan's current schedule is also shown in the figure. In the task structure MountainMan has two orders – one for `JJBoom 3 Season` sleeping bags and one for `BEI Jasper` backpacks. (A different TÆMS task is associated with each order.) The `JJBoom` lot has a higher expected quality because the margins are better on the `JJBoom` product than they are with the `BEI Jasper`. However, the `JJBoom` sleeping bags take longer to produce than the `BEI Jasper` backpacks. If the MountainMan agent were optimizing over the quality/duration ratio rather than maximizing quality, and the two orders were mutually exclusive, the agent would choose the `BEI Jasper` run over the `JJBoom` sleeping bag run. In this case as both the orders can be satisfied and the agent is maximizing total quality both production runs are scheduled and both orders are set to be filled. MountainMan is currently two hours into the `JJBoom` sleeping bag production run and is planning to produce `BEI Jasper` backpacks after the sleeping bag run (at 2pm).

At 12pm, when MountainMan is two hours into the `JJBoom` sleeping bag production run, a new order arrives. It is for the very high profit `KMS Sequoia` backpacks. The arrival of the new order causes the domain problem solver for the MountainMan agent to produce a new candidate production task, `Make KMS Sequoia`, associated with the order as shown in Figure 7. The new order has a desired delivery date for the subsequent day. Because the production schedule for today is full, the MountainMan agent must either negotiate with the KMS retailer that placed the order or find some other way to produce the desired goods. Note that at this time (12pm) the `BEI Jasper` packs are scheduled for production at 2pm. Thus the MountainMan agent actually has four possible choices: 1) it can reject the `KMS Sequoia` order because production is full for the current day, 2) it can reject the existing `BEI Jasper` order and do the `KMS Sequoia` run instead, 3) it can negotiate with the KMS retailer agent to obtain a delivery deadline that it can meet more easily, 4) it can negotiate with the BEI retailer agent to obtain a later delivery date for that order.

In this case we assume that KMS orders are generally non-negotiable and the MountainMan agent considers rescheduling accordingly. Upon consideration the agent itself detects that the `KMS Sequoia` order can be filled and that it should bump the `BEI Jasper` because the `KMS Sequoia` production run is more profitable. (This analysis is performed by the TÆMS Design-to-Criteria agent scheduler.) In making this determination the agent could reason about the cost of decommitment for the existing order and compare said cost to the higher value associated with the new order. In this example decommitment cost is not used. Given the higher value of the new order, the agent reschedules as shown in Figure 8. When the agent reschedules it sends the BEI retailer agent a decommitment message that indicates the BEI order will not be filled as expected. Note that the `JJBoom` run that is currently in production proceeds uninterrupted. If runs were interruptable (they are not – see
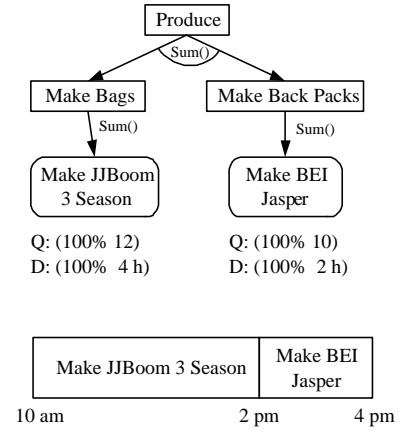

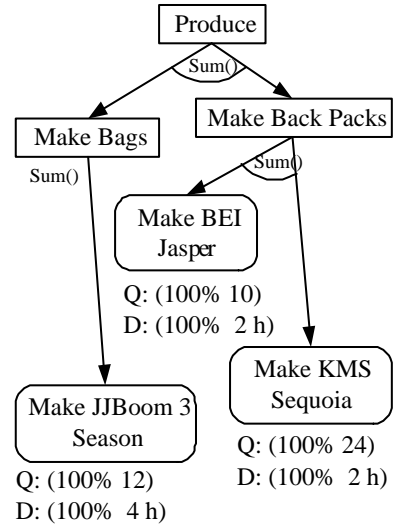Figure 6: MountainMan's TÆMS Task Structure And Associated Schedule At 10AM

Figure 7: MountainMan's Modified TÆMS Task Structure Reflects The New Order

the assumptions above) the agent would consider aborting the current run and could even evaluate taking the run off the line at some cost (overhead) and putting it back on during a future time when the line was idle or constraints more relaxed.

In response to the notice that MountainMan will not fulfill its order the BEI retailer agent examines its own local TÆMS task structure (not shown) and because there are no other orders that are competing for financial resources (shelf space could be considered here also) it re-issues its order with a later delivery date. The MountainMan agent reschedules again, as shown in Figure 9, and decides to 1) complete the `JJBoom` run (as it should given the requirements above), 2) then do the `KMS Sequoia` production run, 3) and then tomorrow (at 8am) to do the `BEI Jasper` run.

This small example illustrates an important class of capabilities for dynamically managing a small supply chain. First it shows autonomously making a quantified choice be-
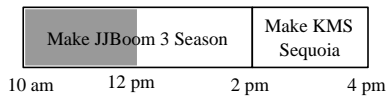
Make JJBoom 3 Season	Make KMS Sequoia

10 am   12 pm   2 pm   4 pm

Figure 8: MountainMan's Production Schedule Modified To Leverage The New Order

Make JJBoom 3 Season	Make KMS Sequoia	Make BEI Jasper

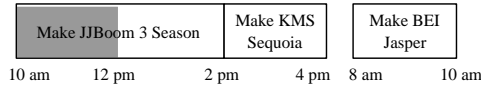10 am   12 pm   2 pm   4 pm 8 am   10 am

Figure 9: MountainMan's Production Schedule Is Revised Again To Support The KMS Order And The Revised BEI Order

tween candidate activities as the situation changes. On a given day there could be many such events and many such exchanges. Automating some or all of this process enables the aggregate system to optimize continuously to improve efficiency, lower costs, maximize profit, or whatever the objective criteria is appropriate. This example also illustrates how *intelligent* agent technology that incorporates temporal reasoning maps to supply chain problems where deadlines (delivery times) and other related constraints are present. The example also identifies many areas where application specific sophistication can be added. For instance the agents could engage in a complex negotiation process to determine appropriate delivery times or could predetermine a price for decommitment (failing to fulfill an order after a guarantee has been given) that would be considered by the MountainMan agent before a decommitment action was taken.

Note that the key properties of the supply chain problem space represented here is that control is *distributed* and the situation is *dynamic* as the orders are driven by actual consumer demand and not by estimates that are computed *a priori*. Other supply chain problems that map directly to this space include automatically changing production schedules to take advantage of spot market materials where the suppliers are represented by agents or shifting activities at both the manufacturer and the retailer to adapt to changes in shipping times or even a shipper going on strike. Another mapping is automatically modifying production to take advantage of changes in the marketplace as communicated by other agents, e.g., new customers, a change in product mix, a change in the product design itself, etc. In general by adding dynamic control to the problem space the entire supply chain becomes more flexible and potentially more efficient. Note also that the use of agents at all of the involved parties is what produces the increase in flexibility – because the agents automatically negotiate over time, and potentially quality and costs, and because they communicate and convey information as it happens, they converge on an optimization across the network of interested parties. With respect to control of actual business processes, particularly when large dollar figures are involved, the agents can fill a support/advisory role and still leave the ultimate decision making capabilities with a trusted human.

Chains of Enablement

In the previous example we discussed a coordination episode between retailer agents and a manufacturing agent. Consider if the same problem is expanded to include raw and intermediate material suppliers who ship to the manufacturer in response to the order placed by the retailer. In a conventional setting, most if not all of the suppliers will maintain an inventory of raw materials and will simply ship as needed from the inventory (scheduling production to generate more inventory as needed). However, in our problem space the raw material suppliers are running with a set of requirements akin to those of MountainMan in the previous example – everything is build to order (or the inventory levels are so small as to have the same effect).

The implications of this are not obvious. If everything that is produced along an entire chain is custom (for only one possible customer) and no inventories are maintained, it means that if the retailer cancels the order while it is in the process of being filled, those who have produced goods for said order lose money. Figure 10 sketches the situation. Each production activity only has value if the retailer completes its purchase of the finished good.

For coordination the implications are pronounced. GPGP coordination is, in general, a peer-to-peer coordination process. (The rationale for this is beyond the scope of the paper.) If a peer-to-peer process is used to coordinate over any chain (not just those that have the very unusual property just described) it may take many iterations back and forth across the chain to resolve all of the temporal interactions and converge on a solution that works throughout the chain. If you add in the characteristic that each activity *only has value* if the entire chain is performed from start to finish the agent must conceptually wait until the entire chain has solidified (or converged) before beginning production. This "global" commitment is very different than a conventional GPGP style commitment – it has more in common with a commitment to a particular course of action regardless of the actual bindings as described in (Castelfranchi 1995).

To address the chains of interactions, where tasks along the supply chain only have value if the retailer purchases the product, we have created a new distributed coordination algorithm that uses value and decommitment cost to achieve global coherence across the chain. We currently cannot present the algorithm due to corporate policies regarding intellectual property but it will be presented at the workshop and in subsequent publications.

Related Work

Huhns et al (Huhns, Stephens, & Ivezic 2002) implemented several methods to automate the construction of agent-based supply chains by translating UML diagrams and Business Object Documents (BODs) into state machines that model the conversations necessary for supply chain management and that can be then wrapped in agents. (Figure 7 in their paper shows their process well.) Their work differs in that they are not solving the problem of how agents decide which requests to fulfill in a collaborative setting beyond the protocol level. In our work there is a strong element of quantified
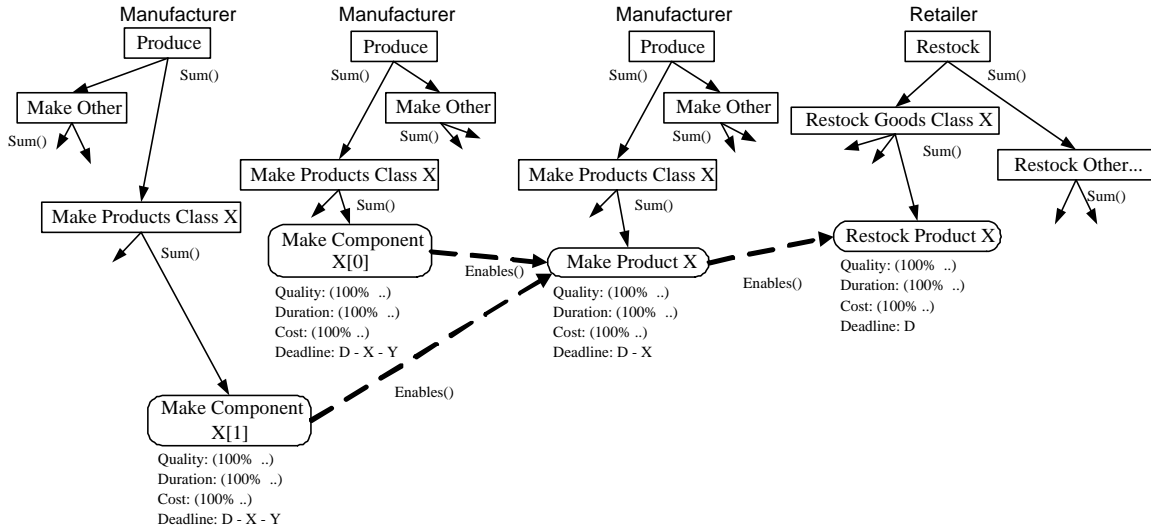
Figure 10: Chains of Interactions Where Tasks Only Have Value if Chain is Completed

and temporal decision making or choice that is lacking from theirs. Note, however, that some of their ideas could be used to frame the communication process of our work.

Shen et al. (Shen *et al.* 1999) detail a general, domain independent, collaborative agent system architecture which incorporates standard agent services such as ontology, yellow pages, and centralized local coordination managers as well as the the notion of a dynamic cooperation domain abstraction for groups of cooperating agents. While this work identifies the importance of cooperation, it does not describe or implement quantified choice / coordination technologies.

Zeng and Sycara (Zeng & Sycara 1999) define a model that can be evaluated to identify efficient combinations of supply-chain activities. The model consists of and/or task decomposition trees. Parts of a supply chain are represented by these nodes. These models are translated into problems for which decisions can be made following inventory theory models. Said models do not appear to consider commit/decommit problems, or the explicit modeling of one tasks' properties versus another. It is thus unclear how flexible the system is – certainly it does not leverage quantified choice or selection.

Barbuceanu (Barbucceanu 1999) gives a representation for tasks and constraints on the execution of tasks (behaviors) called a *goal network*. Obligations and interdictions in his framework roughly correspond to commitments (part of the TÆMS coordination process) or the facilitates/hinders interactions in TÆMS (not shown in Section but related to TÆMS enablement). One agent's authority over another is required to set an obligation. The author also describes a way reasoning about the representation, which is branch and bound, to find the right commitment for goals which optimizes the utility of the task network.

Collins et al. (Collins & Gini 2001) describe a MultiAgent NEgotion Testbed (MAGNET) which implements collaboration via an auction model. Agents which require ser-

vices request them via a task network that includes task descriptions and time constraints. Provider agents then send back bids with the tasks that they are willing to undertake, when they can do them, and at what price. A bid manager fills in a requesting agent's task structure with an appropriate schedule from the bids by using either an integer programming or a simulated annealing evaluator. Their framework is not as rich as TAEMS, and they are not dealing with commit/decommit issues though they are considering temporal constraints and a choice mechanism – features we consider important.

In general our work differs in the inherent richness of TÆMS and the explicit effort of all of the TÆMS technologies to support dynamic adaptation to situations as they evolve. Thus both the agent scheduling and coordination technologies are designed not to rely on *a priori* or off-line computations and designed specifically to always evaluate options from a qualified perspective.

Limitations, Experimental Plans, and Future Work

This paper describes TÆMS agent technology and shows its use on an implemented supply chain management problem. The technology used here currently has a few limitations – some of which are being addressed and some of which are larger issues. In this paper, the coordination process used is pair-wise. This technique will not map to larger supply chains without introducing inefficiency and potential financial loss due to agents making inappropriate decisions. Recently we have created a distributed coordination process that handles chains of interactions using value and decommitment cost. This process will be presented in subsequent publications.

Another limitation of our current implementation is that we do not coordinate over resources. The chains of interactions described above and easily envisioned by raw-

to-manufacturer-to-retailer chains are not actually chains from task-to-task but are chains from task-to-resource-to-task. In other words, MountainMan produces a good that is consumed by the retailer. If we modeled and coordinated over that good, rather than using task interactions, the system would automatically handle situations in which MountainMan had a requested good in inventory or lacked a raw material needed for production. Currently this functionality is partly implemented in the domain problem solvers of the retailer agents and the MountainMan domain problem solver could be extended to provide this functionality as well, e.g., `if goods in inventory, ship and charge (do not make new production task)`. However, resource coordination has been done before in TÆMS and the explicit representation of the resources potentially introduces an additional level of flexibility and simplifies the construction of the domain problem solvers.

The larger issue that may not be obvious is that when control is decentralized in this fashion, and the problem decomposition itself is not structured but instead evolves, this type of distributed optimization is not always guaranteed to be optimal. When can it fail? When the problem spaces get large it is occasionally difficult for the Design-to-Criteria agent scheduler to produce an optimal solution. (The general case of the problem it solves is not computable – it uses approximation techniques to make the space tractable and operate on-line in soft real time.) Another case in which it may not be optimal is when the constraints fall in a particular way – the partial and distributed views held by the agents may not always contain enough information for them to fully optimize. With this caveat mentioned, it is also important to recognize that most human-centered business processes today are *far from* optimal. In this work we are seeking to improve efficiency and reduce costs – making gains over the approaches currently used.

The work presented here is also not heavily verified empirically. We will conduct experiments, after the interaction-chain protocols are constructed, that measure the proximity to optimal that the system can reach in this distributed fashion. Note the implication of this class of experiments are that the problem space has to be relatively small so that the centralized optimal (exhaustive) computation can be performed. Other appropriate experiments include testing the boundary conditions / extremes in terms of number and type of (conflicting) constraints. Metrics typically evaluated include the overall quality obtained by the agents, the number of messages passed, the number of decommitment operations required, and the number of times the agents schedule/reschedule while considering their options.

Acknowledgments

References

Barbucceanu, M. 1999. A negotiation shell. In *Proceedings of the 3rd International Conference on Autonomous Agents*, 348–349.

Bazzan, A. L.; Lesser, V.; and Xuan, P. 1998. Adapting an Organization Design through Domain-Independent Diagnosis. Computer Science Technical Report TR-98-014, University of Massachusetts at Amherst.

Castelfranchi, C. 1995. Commitments: From individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, 41–48.

Collins, J., and Gini, M. 2001. Exploring decision processes in multi-agent automated contracting. In *Proceedings of the 5th International Conference on Autonomous Agents*, 81–82.

Decker, K. S., and Lesser, V. R. 1995. Coordination assistance for mixed human and computational agent systems. In *Proceedings of Concurrent Engineering 95*, 337–348. McLean, VA: Concurrent Technologies Corp. Also available as UMASS CS TR-95-31.

Decker, K., and Li, J. 1998. Coordinated hospital patient scheduling. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, 104–111.

Horling, B.; Benyo, B.; and Lesser, V. 1999. Using Self-Diagnosis to Adapt Organizational Structures. Computer Science Technical Report TR-99-64, University of Massachusetts at Amherst. [http://mas.cs.umass.edu/ bhorling/papers/99-64/].

Horling, B.; Vincent, R.; Mailler, R.; Shen, J.; Becker, R.; Rawlins, K.; and Lesser, V. 2001. Distributed sensor network for real-time tracking. In *Proceedings of Autonomous Agent 2001*.

Huhns, M. N.; Stephens, L. M.; and Ivezic, N. 2002. Automating supply chain management. In *To Appear in the Proceedings of the 6th International Conference on Autonomous Agents*.

Lesser, V.; Atighetchi, M.; Horling, B.; Benyo, B.; Raja, A.; Vincent, R.; Wagner, T.; Xuan, P.; and Zhang, S. X. 1999. A Multi-Agent System for Intelligent Environment Control. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*.

Lesser, V.; Horling, B.; Klassner, F.; Raja, A.; Wagner, T.; and Zhang, S. X. 2000a. BIG: An agent for resource-bounded information gathering and decision making. *Artificial Intelligence* 118(1-2):197–244. Elsevier Science Publishing.

Lesser, V.; Horling, B.; Raja, A.; Wagner, T.; and Zhang, S. X. 2000b. Sophisticated Information Gathering in a Marketplace of Information Providers. *IEEE Internet Computing* 4(2):49–58.

Lesser, V.; Horling, B.; and et al. The TÆMS whitepaper / evolving specification. http://mas.cs.umass.edu/research/taems/white.

Raja, A.; Lesser, V.; and Wagner, T. 2000. Toward Robust Agent Control in Open Environments. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents2000)*.

Shen, W.; Ulieru, M.; Norrie, D.; and Kremer, R. 1999. Implementing the internet enabled supply chain through a collaborative agent system. In *Proceedings of the 1999 Autonomous Agents Workshop on Agent Based Decision Support for Managing the Internet Enabled Supply Chain*.

Vincent, R.; Horling, B.; Lesser, V.; and Wagner, T. 2001. Implementing Soft Real-Time Agent Control. In *Proceedings of Autonomous Agents (Agents-2001)*.

Wagner, T., and Lesser, V. 2001. Design-to-Criteria Scheduling: Real-Time Agent Control. In Wagner/Rana., ed., *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, LNCS. Springer-Verlag. Also appears in the 2000 AAAI Spring Symposium on Real-Time Systems and a version is available as University of Massachusetts Computer Science Technical Report TR-99-58.

Wagner, T.; Phelps, J.; Qian, Y.; Albert, E.; and Beane, G. 2001. A modified architecture for constructing real-time information gathering agents. In *Proceedings of Agent Oriented Information Systems*.

Wagner, T.; Garvey, A.; and Lesser, V. 1997. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 294–301. Also available as UMASS CS TR-1997-10.

Wagner, T.; Garvey, A.; and Lesser, V. 1998. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling* 19(1-2):91–118. A version also available as UMASS CS TR-97-59.

Wagner, T. A.; Raja, A.; and Lesser, V. R. 2002. Modeling Uncertainty and its Implications to Design-to-Criteria Scheduling. Under review. An early version is available as UMASS CS TR#1998-51.

Zeng, D., and Sycara, K. 1999. Agent facilitated real-time flexible supply chain structuring. In *Proceedings of the 1999 Autonomous Agents Workshop on Agent Based Decision Support for Managing the Internet Enabled Supply Chain*, 21–28.

Zhang, S.; Raja, A.; Lerner, B.; Lesser, V.; Osterwiel, L.; and Wagner, T. 2001. Integrating high-level and detailed agent coordination into a layered architecture. In Wagner, T., and Rana, O., eds., *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, LNCS. Springer-Verlag. Abstract also appears in ICMAS-2000.