Agent-Based e-Supply Chain Decision Support

Norman M. Sadeh[a] David W. Hildum[b] Dag Kjenstad[c]

[a]e-Supply Chain Management Lab. & Institute for e-Commerce

ISRI – School of Computer Science

Carnegie Mellon University, Pittsburgh PA 15213-3890 – U.S.A.

E-mail: sadeh@cs.cmu.edu

[b]Intelligent Coordination and Logistics Lab.

The Robotics Institute – School of Computer Science

Carnegie Mellon University, Pittsburgh PA 15213-3890 – U.S.A.

E-mail: hildum@cs.cmu.edu

[c]SINTEF Applied Mathematics, Department of Optimization

PO Box 124 Blindern, N-0314 Oslo, Norway

E-mail: dag.kjenstad@math.sintef.no

Abstract

The global Internet economy is expected over time to give rise to increasingly agile practices where new supply chain arrangements are dynamically set up in response to changing business conditions and demands for highly customized products and services. To implement these "e-Supply Chain" practices, enterprises need the ability to rapidly evaluate new business opportunities and dynamically identify the best potential supply chain partners to respond to them. They also need the ability to effectively coordinate production and delivery of goods or services across the resulting value chains. In this paper, we introduce MASCOT, an agent-based decision support environment for dynamic supply chain management. MASCOT agents help users distributed across multiple organizations and across different levels within a given enterprise to collaborate on the development and revision of supply chain solutions, as they evaluate different business opportunities (e.g. requests for bids from potential customers) and different sourcing options. The paper also discusses new coordination protocols, developed within the context of MASCOT and aimed at better exploiting the power of finite capacity scheduling functionality across the supply chain. Empirical results are presented quantifying the benefits afforded by these new protocols under different loads and supply chain configurations.

Keywords: Dynamic Supply Chains; Supply Chain Collaboration; Agent-Based Decision Support; Production Scheduling; Mixed-Initiative Decision Support; Workflow Management; Agile Business Practices; Blackboard Architecture

1 Introduction

The emerging global Internet economy will increasingly put a premium on the ability of companies to quickly and accurately evaluate new market opportunities, new product and subcomponent designs, and other strategic business decisions (e.g., make-or-buy) in *collaboration* with potential supply chain partners. It also requires that companies be able to effectively and efficiently coordinate activities such as production and transportation across supply chains that are dynamically set up in response to constantly changing and increasingly customized market demands. This paper provides an overview of MASCOT ('Multi-Agent Supply Chain cOordination Tool'), a reconfigurable, multilevel, agent-based environment for collaborative supply chain decision support. We review key innovative elements of the MASCOT architecture with a special emphasis on its support of real-time, mixed-initiative 'what-if' decision support. We show how MASCOT decision support agents enable end-users operating across different enterprises and different levels within their organizations to rapidly and selectively coordinate the evaluation of alternative market opportunities and associated sourcing requirements.

MASCOT agents are customizable, mixed initiative decision support tools that act as coordination and collaboration wrappers for the management modules associated with a given supply chain entity-whether an entire enterprise or a particular plant or shop within an enterprise. These management modules typically include bidding and order management modules, supply chain planning and scheduling modules as well as procurement modules. A MASCOT agent essentially serves two functions:

1. It provides an open and uniform communication and coordination interface between the management modules of different supply chain entities (e.g., between the sourcing module of one enterprise and the Available-To-Promise (ATP) or Capacity-To-Promise (CTP) module of another, or between the planning and scheduling modules of multiple facilities within a given enterprise)

2. It supports customizable, mixed-initiative functionality that enables users at different levels within the architecture to interactively manipulate and evaluate alternative supply chain decisions – or collaborate. Specifically, MASCOT agents can be customized to selectively coordinate the requirements of their solutions with those maintained by other agents. As a user interactively evaluates different options, his or her agent will selectively coordinate production and transportation plans with other relevant agents. For instance, when evaluating alternative sourcing options a user's agent might selectively coordinate its plans with those of relevant supplier agents. Or, when attempting to improve the delivery date associated with a potential bid, the agent might selectively coordinate with agents responsible for one or more bottleneck facilities within the enterprise to make sure its production plan is not overly optimistic.

The first part of this paper provides an overview of the MASCOT architecture with a particular emphasis on its support for mixed-initiative collaboration functionality. The second part discusses different coordination policies studied within the context of MASCOT and reports on an experimental study conducted to evaluate the benefits of these policies under different operational assumptions, namely different loads and different supply chain configurations. The study suggests that new finite capacity coordination protocols can significantly improve the accuracy of ATP/CTP decisions, which in turn can translate into significant increases in profits and performance.

2 The Overall MASCOT Architecture

The MASCOT architecture provides a framework for the dynamic identification of supply chain partners (sourcing) and the coordinated development, evaluation, manipulation, and maintenance of supply chain plans across multiple entities and multiple levels of abstraction. MASCOT agents serve as wrappers for the supply chain management modules of a particular supply chain entity – whether an entire company or a single plant or shop within a given enterprise. These modules include bidding and

order management modules, planning and scheduling modules as well as sourcing modules. Lower-level MASCOT agents are typically wrappers for planning/scheduling modules that support individual facilities over short to medium-term horizons. Higher-level agents are coordination wrappers for tactical or strategic supply chain management modules that generally require looking over longer horizons and across multiple facilities (typically within the same company). Figure 1 provides an illustration of the interactions taking place between MASCOT agents sitting across different organizations and different levels of abstraction.

Each MASCOT agent plays several roles:

1. *Coordination:*

   MASCOT agents act as coordination wrappers for supply chain planning and scheduling modules, supporting both lateral and vertical coordination protocols. In our terminology, lateral coordination protocols support interactions between peer-level agents, whereas vertical coordination protocols support interactions between high-level agents and those lower-level agents responsible for a portion of the underlying supply chain. An example is a low-level agent responsible for the planning and scheduling of one of several facilities or shops under the responsibility of a higher-level agent. Vertical coordination typically takes place within a given enterprise, whereas lateral coordination can takes place either within a given enterprise or across enterprises. We further distinguish between (1) high-level lateral coordination protocols that involve, for example, requests for bids on subcomponents and exchange of ATP/CTP information, and (2) low-level lateral coordination protocols to dynamically establish and revise delivery dates across feeder and consumer facilities for individual batches. Vertical coordination protocols, on the other hand, help maintain consistency across multiple levels of abstraction (e.g. between coarse enterprise capacity models and finer shop-floor capacity models) and selectively validate high-level planning decisions through more detailed evaluation at the level of one or more bottleneck facilities. For example, an

order promising decision for a product that needs to flow across several internal facilities can be validated by carrying out a detailed analysis of the impact of the order on the load of a particular bottleneck facility through interaction with the agent responsible for that facility.

2. *Integration with Heterogeneous Planning, Scheduling, and Sourcing Modules:*

Each MASCOT agent is configured around a blackboard architecture [4, 6, 9] that allows for the easy integration of a number of supply chain management modules. These typically include an order management/bidding module, planning and scheduling modules, a sourcing/partner selection module, one or more analysis modules, etc.. Within a given agent, these modules can be activated to develop, evaluate, and revise integrated planning/scheduling solutions and associated supply chain arrangements. Solutions are stored in a data structure accessible to all the agent's modules called a *blackboard*. Specific modules differ from one agent to another depending on the facility(ies) for which an agent is responsible (e.g., legacy systems) and the level at which it operates within the architecture. Typically, low-level/facility-based agents might include a process planning/process selection module and a production scheduling module – e.g. see the IP3S system [9]. Higher-level agents will typically combine enterprise-wide order promising/bid submission modules along with high-level scheduling/capacity-allocation modules (e.g. Available-To-Promise/Capacity-To-Promise functionality), high-level process-flow/bill-of-material (BOM) development modules and sourcing modules.

3. *Mixed-Initiative Decision Support:*

An agent's blackboard is partitioned into *contexts*, each corresponding to a possibly different set of assumptions (e.g., capacity assumptions, sourcing assumptions, demand assumptions, make-or-buy

assumptions, internal process-flow/BOM assumptions) that enable the end-user to compare different options both locally and through coordination with agents responsible for other supply chain entities. Contexts can be created either (1) manually by the (agent's) end-user (e.g., to evaluate a particular assumption), or (2) automatically by the agent (e.g., in response to a query from another agent). Within each context, solutions are developed by the end-user or the agent (or a combination thereof) through the activation of services supported by the agent's supply chain management modules.

4. *Reconfigurability:*

MASCOT agents can be dynamically reconfigured to accommodate the introduction of, for example, new products, new product flows, new facilities, new suppliers, new make-or-buy arrangements, and new transportation arrangements. Lateral and vertical coordination protocols rely on explicit (declarative) product, process, and facility models to determine the agents with which to coordinate. Changes in these models, such as the identification of a new supplier for a particular component or a new product flow, trigger the automatic recalculation of relationships between agents in the architecture.

Insert Figure 1

Figure 1: Overall MASCOT Architecture

3 The MASCOT Agent Architecture

The use of blackboard architectures [4, 3, 1] as an effective vehicle for integrating multiple sources of knowledge to solve complex problems has been demonstrated in a wide range of application domains. Blackboard architectures emphasize the modular encapsulation of problem-solving knowledge within independent Knowledge Sources (KSs). KSs work collectively to develop solutions by communicating through a shared data structure, namely, the blackboard. By explicitly separating domain knowledge (e.g., in the case of MASCOT, planning, scheduling, supplier selection, analysis, coordination, and communication knowledge) and control knowledge, blackboard architectures offer several key advantages:

- *flexibility of the control mechanism*, making it possible for the user to select from among a dynamic set of control regimes (e.g., highly interactive control regimes where most decisions are made by the end-user versus more autonomous regimes where the end-user specifies high-level tasks or 'goals' and lets the system – the agent's controller - decide how to accomplish them)
- *extensibility of the architecture*, making it particularly easy to add and enhance knowledge sources
- *ease of integration with legacy systems* through the encapsulation of existing problem-solving systems as knowledge sources
- *reusability of knowledge sources* across multiple domains

Insert Figure 2

Figure 2: MASCOT Wrapper Agent Architecture

Figure 2 presents the MASCOT wrapper-agent architecture. Each MASCOT agent typically includes:

- a set of planning, scheduling, sourcing/partner selection, coordination, analysis, and communication modules – or *KSs*
- a *blackboard*, which serves as the repository of multiple partial and complete integrated planning/scheduling/sourcing solutions, each stored in a separate context
- a module-activation *controller* that orchestrates the construction and revision of solutions through the activation of services supported by different KSs, based on either direct input from the end-user or the agent's control heuristics (or a combination of the two)
- a graphical user interface (GUI)

3.1 Mixed-Initiative Asynchronous Problem Solving

The blackboard is a shared data structure on which KSs post solution components (e.g., BOM, sourcing decisions, production schedules, etc.), analysis results (e.g., resource/capacity utilization statistics), and coordination/communication status information. It is partitioned into an arbitrary number of contexts that correspond to different sets of working assumptions (e.g., sets of orders that need to be planned/scheduled, available/allocated resource capacities, sourcing arrangements) and different solutions. Within each context, a summary of the current state of the solution is maintained in the form of a set of *unresolved issues*. An unresolved issue is an indication that a particular aspect of the current context solution is incomplete, inconsistent, or unsatisfactory (e.g., a request for bid still needs to be evaluated, a promised delivery date is violated, a bid from a prospective supplier still needs to be evaluated, a subcomponent still needs to be procured).

Problem solving within a MASCOT agent is directed by either the agent's end-user or the agent itself (or a combination of the two). It is said to progress in a "mixed-initiative" fashion through cycles during which:

1. one or more unresolved issue instances are selected to be resolved (i.e. to be addressed)
2. a particular method of resolution (i.e., a service supported by a KS) is selected from among the set of methods applicable to the unresolved issue instance(s)
3. the selected method is executed by invoking the appropriate KS

Instances of unresolved issues are created and deleted as a result of (1) KS invocations, (2) the incorporation of external events into a context (e.g. a bid from a prospective supplier), and (3) the modification of assumptions within a context to perform 'what-if' analysis.

In the remainder of this section we describe the major features of the MASCOT agent architecture, with an emphasis on the mixed-initiative problem-solving and integration capabilities it supports.

3.1.1 Alternate Problem Instances and Solutions

The mixed-initiative decision-support capabilities of MASCOT rely heavily on the use of blackboard contexts to support the representation of multiple problem instances. Each context is intended to eventually contain all relevant problem data, though at a particular point in time some assumptions or part of the solution may still be missing. Typically a context description includes a collection of available resources (including information on available capacity and existing reservations), raw material supplies and components, finished-goods and work-in-process inventory quantities, a collection of firm orders and requests for bids together with their corresponding process flows, BOMs, procurement arrangements, and production schedules. In addition, each context also maintains a set of unresolved issues to represent inconsistencies in any partial solution that must be removed to produce a

complete and satisfactory solution. As assumptions are modified and a solution is developed within the context, the set of unresolved issues is updated to help both the agent and its end-user keep track of aspects of the current solution (within that context) that require further problem-solving attention.

Contexts can be created either by the end-user or automatically by the agent. It is through the creation of multiple contexts that 'what-if' analysis is supported. By creating multiple copies of a context, changing various assumptions within the copies and producing solutions for each, alternate solution paths can be explored. Either the end-user or the agent can leave a particular context at any point in time and explore other potentially more promising alternatives in other contexts. Changes in the assumptions within one context remain local to that context and do not affect other contexts that may share the same entities. When a KS is invoked, its results are visible only within the context in which the end-user or the agent is currently working. This context is called the 'current working context.'

3.1.2 Selective Problem Definition

The mixed-initiative power of the context mechanism comes from the capability it provides the user to define a problem progressively and to modify the underlying assumptions about a problem. This can be done through either the modification of problem assumptions within a context (e.g., by changing various order and resource attributes such as due dates, work shifts, and supply availability dates), or the incorporation of events into a context (e.g., messages from other agents such as bids from prospective suppliers, requests for bids from prospective customers, or status updates from a plant within the enterprise).

Events received from other MASCOT agents (e.g. actual or prospective suppliers or customers) or other systems (e.g., manufacturing execution systems) are posted on the agent's *event queue* in preparation for being incorporated within one or more contexts by the end-user or the agent itself. These events include the notification of incoming orders, requests for bids, bids from prospective

suppliers, and execution updates, whether from the shop floor or from other agents (e.g., suppliers). When an event is incorporated into a context, the blackboard translates the initial result (or implication) of the action described by the event into an appropriate set of unresolved issues. The objective for the end-user and/or the agent is to resolve each such issue through the activation and execution of one or more KSs until all events have been addressed and all unresolved issues have been resolved.

The MASCOT event-processing mechanism supports two important mixed-initiative capabilities:

1. It allows both the end-user and the agent to ignore events that are unlikely to affect the part of the solution upon which work is currently being done. For example, when revising a plan for a part that needs to be processed within the week, incoming-order events for new orders due three months downstream can be ignored.
2. It allows both the end-user and the agent to process conditional events, such as requests for bids. For example, upon receipt of a request for bid on a possible order, a copy of the current context can be created, within which the order can be planned and scheduled. The resulting solution showing the impact of the possible order can then be evaluated to determine a realistic completion date and decide whether or not to submit a bid.

3.1.3 Unresolved Issues

As the assumptions within a particular context are modified or as new events are incorporated into a context, the set of unresolved issues within the context is updated automatically (according to declarative domain and control information). The set of unresolved issues within a context defines areas in the current partial solution where further problem-solving effort needs to be directed to produce a complete, consistent, and satisfactory solution. Like the *sources of uncertainty* of Carver

and Lesser's RESUN control planner [2], unresolved issues provide a powerful 'workflow management mechanism' that helps identify, plan, and control the work that remains to be done in any given context.

The MASCOT architecture distinguishes between three types of unresolved issues, relating to:

- the *completeness* of the solution, such as a request for bid that still needs to be evaluated or a subcomponent that still needs to be procured
- *inconsistencies* within the solution, such as required supplies being delivered past their required date
- potential areas for solution *improvement*, such as an order with an excessively late completion date or long leadtime

3.2 Module-Activation Control

Each MASCOT agent is responsible for directing solution construction, revision, and analysis, either through close interaction with the end-user, or on its own with the help of a knowledge base of control heuristics. The MASCOT agent architecture control mechanism supports two key mixed-initiative capabilities:

1. *support for multiple control regimes*, ranging from a highly interactive mode where the user specifies each problem-solving action to an autonomous mode where the controller takes full responsibility for (1) the selection of which events to incorporate into the current context, (2) the determination of which unresolved issues to resolve, and (3) the selection of the specific methods for their resolution

2. *support for multilevel customizable problem-solving tasks* to provide a range of low to high-level modes of user interaction (e.g., the activation of a specific low-level KS service, the posting of high-level objectives (or 'goals'), the activation of a sequence of services and goals)

The MASCOT agent architecture allows an end-user to select from among different levels of interaction and different control regimes at any time. In addition, the set of high-level problem-solving tasks provided by the agent can be easily augmented to accommodate changing user-interaction patterns. Specifically, a hierarchy of high-level *goals* and *scripts* can be defined in terms of the basic set of services provided by the particular problem-solving systems encapsulated as KSs and incorporated within each MASCOT agent.

To support these mixed-initiative capabilities, the MASCOT agent architecture control mechanism follows an *execution profile* that records the assignment of various problem-solving tasks (e.g., the incorporation of events, the selection of unresolved issues to resolve and the methods for their resolution) to either the agent or its end-user. The assignment of tasks can be changed at any point by modifying the execution profile. To provide multiple levels of interaction with the system through the definition and activation of aggregate and goal-oriented problem-solving tasks, each MASCOT agent maintains its own declarative *control knowledge base* that links each unresolved issue to the set of appropriate problem-solving options for its resolution. The control knowledge base also contains the collection of generic and domain-specific control heuristics that are used by the agent to perform the tasks assigned to it, as recorded in the execution profile.

An *agenda* mechanism is employed by each MASCOT agent to keep track of the problem-solving tasks remaining to be executed. When a particular course of action is selected, either manually by the end-user or automatically by the agent (through consultation with the appropriate control heuristics), one or more problem-solving task items are placed on the agenda, describing an action or sequence of

actions to be performed by the agent. The MASCOT agent architecture control mechanism supports three types of agenda items:

1. *service* activations, which correspond directly to the specific problem-solving services provided by the MASCOT KSs (e.g. incorporating a new order into the current production schedule, or requesting bids from possible suppliers)
2. *goal* activations, which are used to specify high-level problem-solving objectives that can be achieved by the execution of either a service or (more likely) a sequence (or 'script') of services and subgoals (e.g. trying to improve the delivery date in a bid before sending it to a prospective customer)
3. *scripts*, which specify a predefined sequence of KS services and goals generally known to accomplish a particular problem-solving task. For instance a script to process a request for bid will typically involve steps such as (a) creating a copy of the current working context, (b) incorporating the request for bid in the new context as a new order, (c) updating the production schedule to accommodate the new order, (d) requesting bids from prospective suppliers, (e) revising the production schedule to reflect the delivery dates received from prospective suppliers, (f) analyzing the resulting solution and deciding whether or not to submit a bid, and, possibly, (g) submitting a bid.

3.3 Problem-Solving Flow

All problem-solving activity within the MASCOT agent architecture is triggered by either:

- the incorporation of a new event (such as a request for bid, an actual order, a bid from a supplier, a status update from another agent or one from the supply chain entity the agent is responsible for) into the current working context, or

- the modification of an assumption within the current working context (e.g., 'what-if' analysis to evaluate a new request for bid or the benefits of adding extra work shifts or purchasing new equipment)

Both can be performed by either the end-user or the agent (as specified by the execution profile).

Insert Figure 3

Figure 3. Problem solving flow within MASCOT agents

The flow of problem solving is summarized in Figure 3. It proceeds from the modification of the current working context in a clockwise direction through the following steps:

1. updating the set of unresolved issues within the current working context to reflect the initial problem-solving action
2. selecting an unresolved issue to resolve
3. selecting a resolution method for the selected unresolved issue
4. activating the selected resolution method
5. executing the problem-solving service that corresponds to the activated resolution method

The MASCOT module-activation controller is invoked whenever there are problem-solving tasks on the agenda remaining to be executed, or, when running automatically (and depending on the execution profile), there are events to incorporate or unresolved issues to resolve.

3.4 Problem-Solving Modules

Knowledge sources serve as the primary problem solvers in MASCOT. They communicate their results by posting new information to the blackboard (e.g., new BOMs, selection of a new supplier, new production schedules) and modifying existing information (e.g., updated product flow or reoptimized production schedule). Each domain-level KS acts primarily as a server that supports a variety of problem-solving services.

The typical MASCOT agent relies on a number of KSs:

- A **Bidding/Order Management KS** that manages requests for bids, bid submissions and incoming (firm) orders. Processing one or more requests for bid typically involves creating a copy of the current working context and incorporating the requests as fictitious orders into the new context. In some cases requests for bids can directly be satisfied out of existing finished-goods inventory. In other cases, they require developing new planning, scheduling and sourcing solutions. Either way, the KS has to determine whether or not to go ahead and submit bids. This generally involves looking at the impact of accommodating the fictitious order(s) on prior delivery commitments (for existing firm orders) – in other words, will the supply chain entity fail to meet other delivery commitments. Ultimately, the KS should determine whether submitting a bid is likely to help improve the entity's bottom line, taking into account profit margins on different orders, the chances of missing delivery commitments and associated penalties/consequences. The final step involves the computation of bid parameters (e.g. proposed delivery date or schedule, asking price as well as possibly other parameters such as late delivery penalties or even specific product configuration).

- A **BOM/Flow Configuration KS**: In the case of requests for bids that allow for several possible Bills of Material (BOMs) or process flows, a special knowledge source can be

used to explore different alternatives. Ideally, this module should select BOMs and process flows that lend themselves to competitive and profitable production schedules and supply chain arrangements (e.g., process flows that avoid bottleneck resources and BOMs with supcomponents that are easy to procure).

- A **Demand Explosion KS**: Most facilities maintain some level of inventory, whether in the form of finished goods, semi-finished goods, components or raw materials. Given a set of orders (including some possible requests for bids), a Demand Explosion KS will (1) determine how to best allocate existing inventories and (2) compute the remaining ("net") demand for subcomponents and raw materials, namely the demand that cannot be satisfied through allocation of existing inventories.

- A **Scheduling KS**, which is typically (though not necessarily) implemented by the *Micro-Boss* system [7, 8], a dynamic finite capacity scheduling tool that has been shown to support efficient just-in-time operation in complex, large-scale manufacturing environments. In particular, Micro-Boss supports anytime Available-To-Promise/Capacity-To-Promise functionality capable of taking into account high fidelity cost models (including tardiness penalties and both in-process and finished-goods inventory costs) along with a broad range of production constraints. In other words, given one or more new orders (e.g. requests for bids) and some prespecified amount of time to come back with an answer, the system is capable of selectively reoptimizing a given production schedule to accommodate these new order(s). The new production schedule can then be used to (1) submit requests for bids to potential suppliers and (2) determine the impact of accommodating these new orders (e.g. should the agent submit a bid for a new order and, if so, when should it promise delivery of the new order and at what price).

- **Analysis KSs**: An example is a resource utilization KS that estimates resource contention resulting from both current and projected reservations and whose results can help select promising process flows.

- **RFQ/Procurement KS**: This KS is responsible for interacting with prospective suppliers, issuing requests for quotes/bids on critical components and evaluating bid submissions. The results are used to refine local planning/scheduling solutions.

- A **Communication KS**, which facilitates coordination between other planning/scheduling agents and information sources.

4 Empirical Evaluation of Finite Capacity Lateral Coordination Protocols

While the MASCOT architecture aims at supporting more flexible and efficient supply chain practices, it does not assume any particular set of coordination protocols or policies. Instead, it is capable of supporting traditional leadtime-based coordination as well as policies aimed at taking advantage of finite capacity scheduling functionalities. This section highlights a few results of an empirical study conducted using MASCOT to compare several lateral coordination policies under different sets of business assumptions [5].

Lateral coordination is particularly critical when it comes to generating robust yet competitive and cost-effective promise dates in response to requests for bids from prospective customers. Once a delivery date has been quoted to a customer, lateral coordination can help dynamically resynchronize production across the supply chain to best meet prior commitments in the face of contingencies such as delays in raw material/subcomponent deliveries, production delays, quality problems, and the arrival of new orders. Traditionally, supply chain coordination has relied on standard leadtimes to determine synchronization dates across the supply chain. These leadtimes are insensitive to actual load

conditions. As a result, depending on the situation, they will be either (1) overly conservative, yielding uncompetitive promise dates, long leadtimes, and excessive inventory, or (2) overly optimistic, resulting in low levels of customer satisfaction and late delivery penalties. While finite capacity considerations offer the promise of more competitive and more realistic estimates, they can also be expected to yield more brittle solutions, which are easily invalidated by contingencies. A key question in this regard is whether, through dynamic coordination, it is possible to reap the benefits of finite capacity computations without suffering from the brittleness of their underlying solutions.

To answer this and related questions, our study, which is detailed in [5], looked at a number of different supply chain configurations (e.g., different numbers of tiers, tiers with one or more suppliers) as well as different ATP/CTP scenarios (e.g., incoming orders with non-negotiable due dates, incoming orders that can be turned down, incoming orders with negotiable due dates, competition with other possible suppliers). A supply chain simulation testbed was used within which MASCOT agents were responsible for coordinating their operations in response to stochastic requests for bids and stochastic production contingencies.

The testbed consisted of a problem generator and a supply chain simulator:

- The problem generator took as input a supply chain configuration (number of products, number of tiers, number of entities in each tier, and number of resources in each entity) and generated stochastic variations of supply chain models based on this data. Variations among model instances included differences in product types, process routings, bills of materials, numbers of resources within each supply chain entity, processing times, and price and cost assumptions).

- The testbed simulated operations at each of the entities across the supply chain while generating random events such as requests for bids (including product types, quantities, requested delivery dates, and prices), and execution contingencies such as resource breakdowns and variations in processing times.

. Insert Figure 4

Figure 4. A Three-tier supply chain configuration

While a number of different scenarios were considered, the results reported below were obtained by simulating variations of the supply chain depicted in Figure 4. At any point in time, each supply chain entity can have both multiple suppliers and multiple customers. In addition, process routings include assembly operations that require synchronization of subcomponent deliveries from multiple customers. This provides for situations where finite capacity lateral coordination can help alert a subcomponent supplier that delivery of another subcomponent from a second supplier has been delayed, allowing the first supplier to reprioritize its schedule accordingly.

Additional assumptions used in the experiments reported in this section included the following:

- The supply chain operates in a pure make-to-order lot-for-lot fashion. Facilities that are the farthest upstream are assumed to have instantaneous access to raw materials (e.g., from ample stock).
- Within each supply chain entity/agent, finite capacity schedules are regenerated on a daily basis and jobs queuing in front of machines are executed starting with the job that is scheduled to run first on that machine. In other words, the schedule is interpreted as a priority mechanism rather than being fully enforced. All schedules were generated using the Micro-Boss finite capacity scheduling system [7, 8].

A request for bid can be accepted by an agent with its requested delivery date (i.e., unconditional ATP/CTP bid), it can be accepted subject to a relaxed delivery date (i.e., conditional ATP/CTP bid), or simply rejected (i.e., no bid). The chance that a bid is accepted depends on how close it is to the requested delivery date. Agents only send bids on orders that are expected to increase their profit. Profit from an order is measured as the difference between the sales revenue and costs associated with that order. Costs include fixed costs such as fixed production costs and subcomponent/raw material procurement costs as well as inventory costs and late delivery penalties (see [5] for further details).

Before regenerating a new schedule, each supply chain agent incorporates new incoming requests for bids into a copy of its working context (where unresolved issues are created accordingly). Requested delivery dates are interpreted as internal due dates. Similarly, promised delivery dates from suppliers, if available, are interpreted as earliest possible release date constraints. Each time a supply chain agent generates a new solution it communicates the resulting start time requirements to its suppliers and the expected completion dates to its customers. To avoid creating unrealistic synchronization dates, some padding is selectively added in the process as further detailed in [5].

Below, we report the performance of each of the following three sets of lateral coordination policies:

- **Leadtime-based**: a reference policy relying on historical leadtime data to estimate delivery dates for prospective orders
- **Local FCS**: similar to Leadtime-based, except that it relies on finite capacity schedules in its top-tier estimates (i.e., the agent that directly receives a request for bid generates a local

finite capacity schedule but relies on leadtime assumptions as far as its suppliers are concerned)

- **Coordinated FCS**: relies on finite capacity calculations at each tier across the supply chain to respond to requests for bids

Insert Figure 5

Figure 5. Performance comparison of 3 lateral coordination policies

The results shown in Figure 5 were obtained by averaging performance over twenty simulation runs (each simulating 150 days of execution for a particular model instance) for each policy. The high-low vertical lines in Figure 5 show the estimated mean profits and their corresponding 95% confidence intervals. The vertical bars represent average leadtimes across the entire supply chain. Profits obtained with the finite capacity lateral coordination policy (Coordinated FCS) are clearly higher than those obtained with the other two policies. The confidence interval is also considerably smaller, indicating that this policy is also more consistent/reliable. Leadtimes obtained with this policy are also significantly shorter. Additional results reported in [5] show that these performance improvements can be attributed to a higher accuracy in the ATP/CTP process combined with a better ability to dynamically adjust for contingencies and satisfy prior delivery commitments.

Results in Figure 5 were obtained for an average nominal load of 80%. This is the average load of the most significant bottleneck area (across the entire supply chain) if all requests for bids were to result in actual orders.

Insert Figure 6

Figure 6: Sensitivity to the average external demand as a percentage of the supply chain's nominal capacity.

Figure 6 shows similar results for nominal loads varying between 40% and 120%. These results indicate that:

- All three policies result in similar performance when the nominal load is low (i.e., less than 60%). Most orders can then be scheduled just-in-time without much resource contention. The benefits of finite capacity lateral coordination become more obvious as the nominal load increases.
- In fact, the finite capacity lateral coordination policy is able to consistently maintain a high profit as the nominal load increases above 0.6, while performance of the other two policies deteriorates. This is not really a surprise: as the nominal load gets close to or greater than 100%, finite capacity coordination ("coordinated FCS") makes it possible to selectively and accurately turn down requests for bids that do not contribute to the company's bottom line and, when necessary, submit bids with relaxed delivery dates, which the other two policies cannot really do.

5 Concluding Remarks

The emerging global Internet economy will increasingly put a premium on the ability of companies to quickly and accurately evaluate new market opportunities, new product/subcomponent designs, and other strategic business decisions (e.g., make-or-buy) in coordination with potential supply chain partners. It also requires that companies be able to effectively and efficiently coordinate activities such as production and transportation across supply chains that are dynamically set up in response to constantly changing and increasingly customized market requirements. In this paper, we introduced MASCOT, an architecture that aims at providing a framework for dynamic selection of supply chain partners and for coordinated development and manipulation of planning and scheduling solutions at multiple levels of abstraction across the supply chain. MASCOT places a particular emphasis on mixed-initiative problem solving, enabling users to flexibly select among a broad range of interaction

regimes. It also supports powerful workflow management capabilities that allow users spread across multiple tiers in the supply chain and operating at different levels of abstraction to collaboratively assess complex tradeoffs. An initial version of the MASCOT architecture was developed and validated in the context of an environment requiring coordination between a machine shop and a tool shop at Raytheon. Recently, the architecture has been further validated through experimentation with more complex supply chain configurations and through development of coordination policies that attempt to dynamically take advantage of anytime Available-To-Promise/Capacity-To-Promise functionality. The MASCOT architecture is now being refined to support more complex multi-attribute supply chain negotiation.

References

[1] N. Carver and V.R. Lesser. The evolution of blackboard control architectures. CMPSCI Technical Report 92-71, Department of Computer Science, University of Massachusetts, Amherst MA, October 1992.

[2] N. Carver and V.R. Lesser. A planner for the control of problem-solving systems. *IEEE Transactions on Systems, Man, and Cybernetics* Special Issue on Planning, Scheduling and Control, 23(6), November 1993.

[3] D.D. Corkill. Blackboard systems. *AI Expert*, 6:40-47, September [99].

[4] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213-253, June 1980.

[5] D. Kjenstad. Coordinated supply chain scheduling. PhD thesis and NTNU Report 1998:24, Department of Production and Quality Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, November 1998.

[6] V.R. Lesser and D.D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15-33, Fall 1983.

[7] N.M. Sadeh. Micro-opportunistic scheduling: The Micro-Boss factory scheduler. In M. Zweben and M.S. Fox, editors, *Intelligent Scheduling*, chapter 4, pages 99-135, Morgan Kaufmann, San Francisco, 1994.

[8] N.M. Sadeh. Micro-Boss: Dual-use ARPI scheduling technology helps improve manufacturing performance, *IEEE Expert*, February 1995.

[9] N.M. Sadeh, D.W. Hildum, T.J. Laliberty, J. McA'Nulty, D. Kjenstad, and A. Tseng. A blackboard architecture for integrating process planning and production scheduling. *Concurrent Engineering: Research & Applications (CERA)*, 6(2):88-100, June 1998.

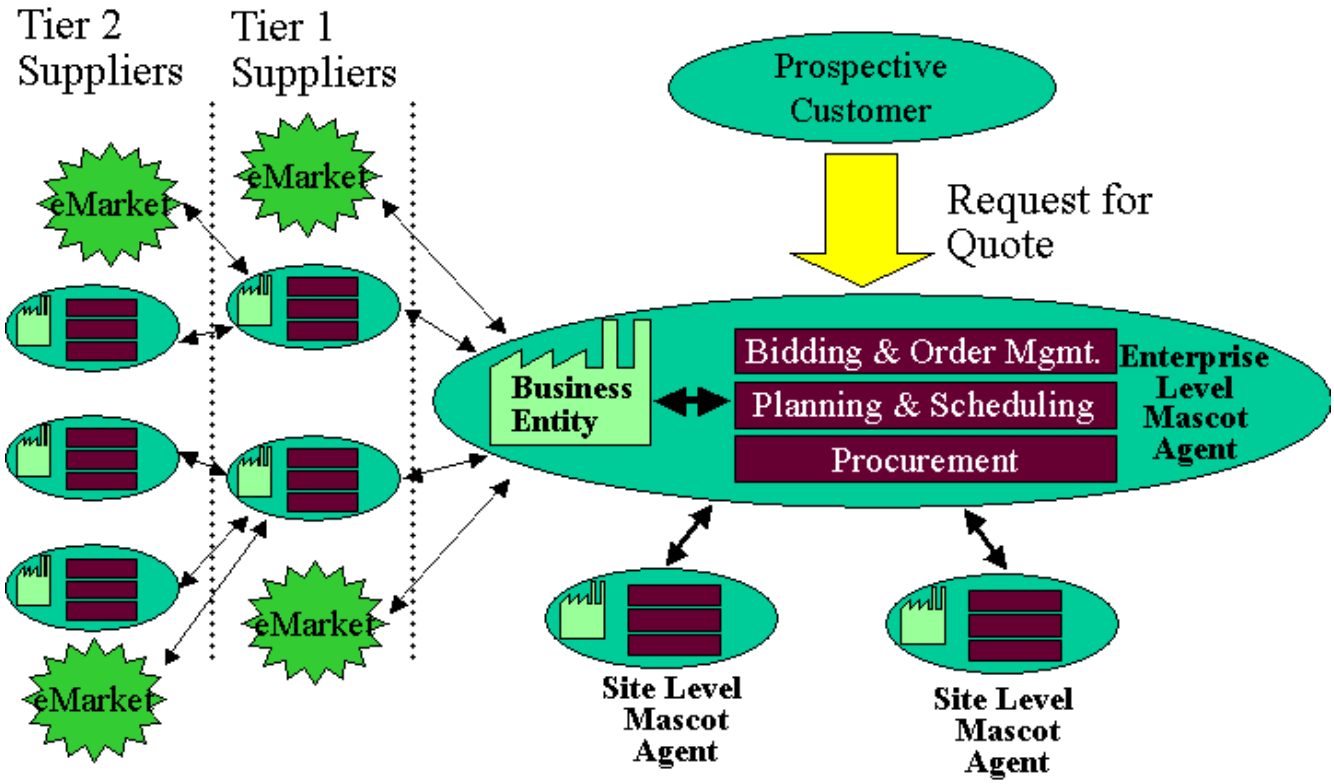

Figure 1: Overall MASCOT Architecture

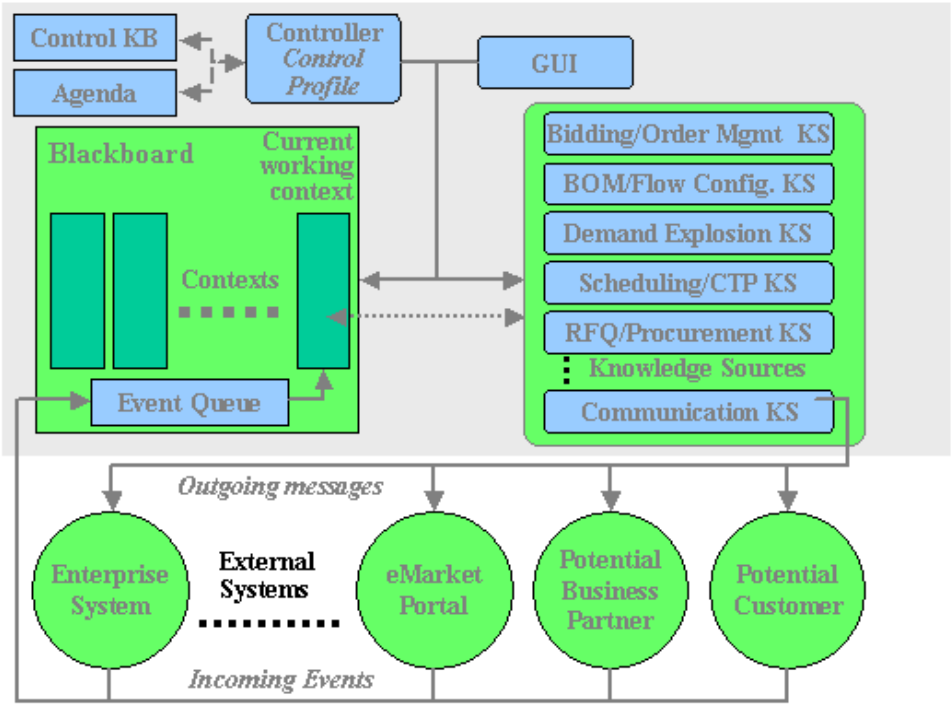

Figure 2: MASCOT Wrapper Agent Architecture

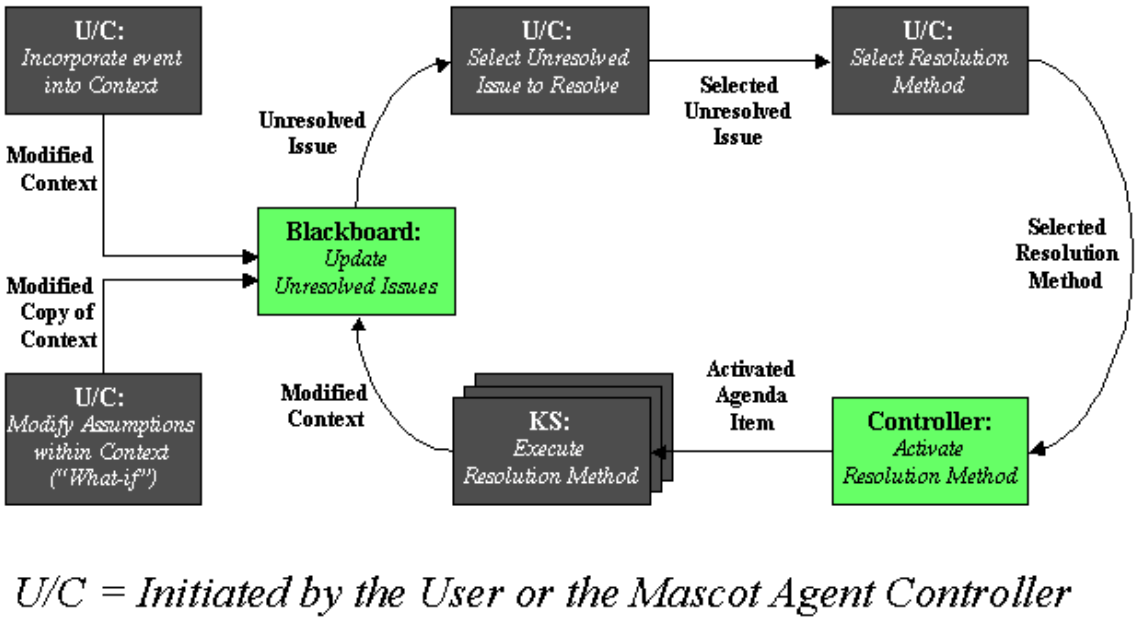

Figure 3. Problem solving flow within MASCOT agents

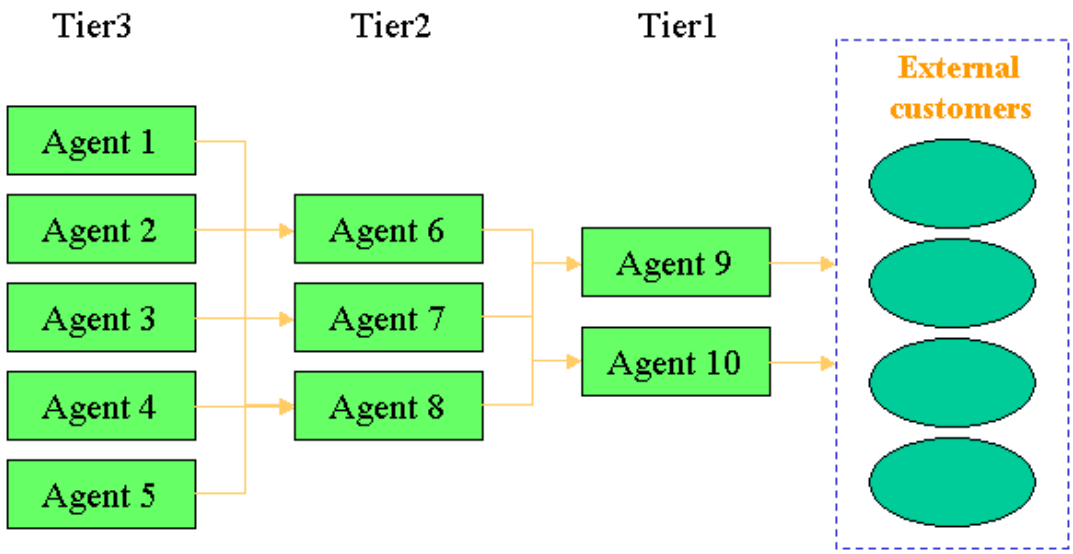

Figure 4. A Three-tier supply chain configuration

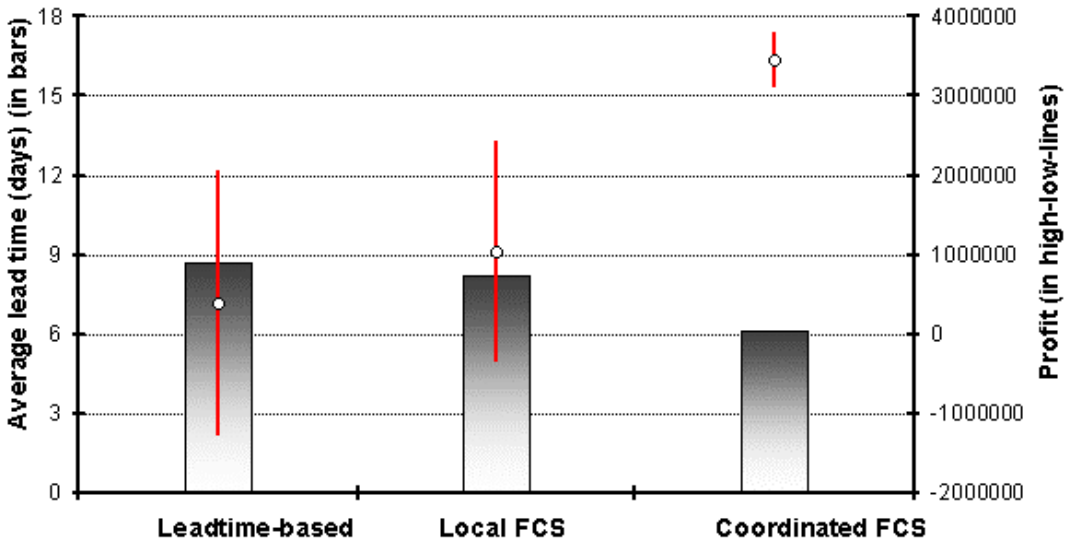

Figure 5. Performance comparison of 3 lateral coordination policies

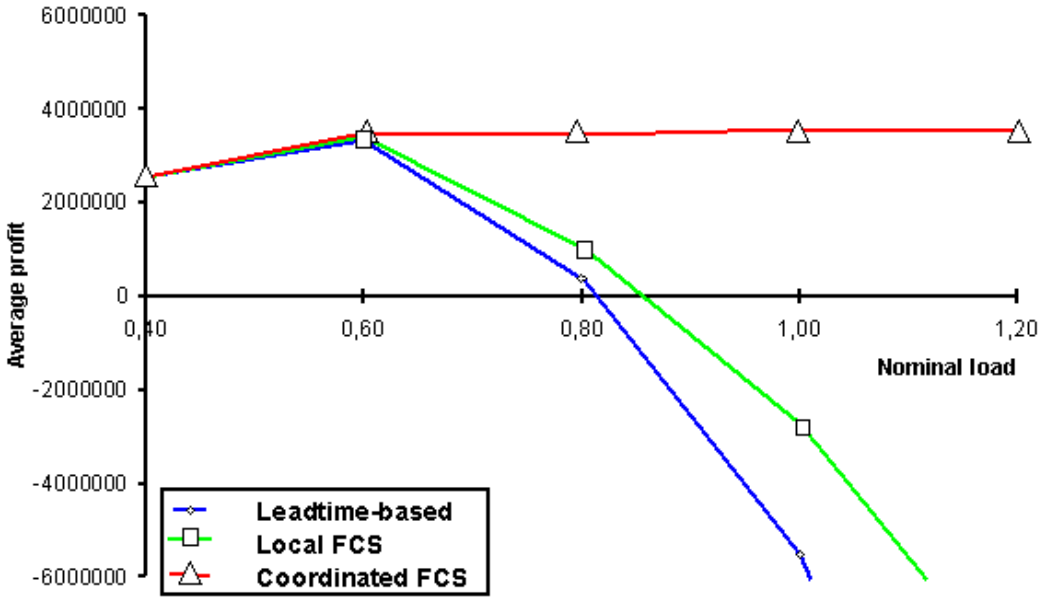

Figure 6: Sensitivity to the average external demand as a percentage of the supply chain's nominal capacity