Sensible Agent Problem-Solving Simulation for Manufacturing Environments

K. S. Barber, E. White, A. Goel, D. Han, J. Kim, H. Li, T. H. Liu, C. E. Martin, R. McKay

The Laboratory for Intelligent Processes and Systems
Department of Electrical and Computer Engineering, ENS 240
University of Texas at Austin
Austin, TX 78712-1084
barber@lips.utexas.edu

Abstract

Agent-based technologies can be applied to many aspects of manufacturing. The need for responsive, flexible agents is pervasive in manufacturing environments due to the complex, dynamic nature of manufacturing problems. Two critical aspects of agent capabilities are the ability to: (1) classify agent behaviors according to autonomy level, and (2) adapt problem-solving roles to various problem-solving situations during system operation. This issue is addressed by research on Sensible Agents, capable of Dynamic Adaptive Autonomy. In Sensible Agent-based systems, levels of autonomy constitute descriptions of agent problem-solving roles. These roles are defined along a spectrum ranging from command-driven, to consensus, to locally autonomous, to master. Dynamic Adaptive Autonomy allows Sensible Agents to change autonomy levels during system operation to meet the needs of a particular problem-solving situation. This paper provides an overview of the Sensible Agent testbed, and provides examples showing how this testbed can be used to simulate agent-based problem solving in manufacturing environments.

Introduction

Manufacturing environments are inherently complex and dynamic. These characteristics create many challenges for the automation of manufacturing tasks such as planning and scheduling. The use of agent-based systems offers significant advantages to automated manufacturing including distribution of control and processing as well as adaptable automated or semi-automated problem-solving (Liu, 1996). However, simply applying the agent-based paradigm to manufacturing problems may not be enough to address the real-time demands of production systems. Agent-based systems operating in the manufacturing domain are subject to dynamic situational changes across many dimensions:

- *certainty of information* held by an agent or acquired by an agent about other agents in the system (e.g. inventory status, product status on factory floor, machine processing performance);
- *resource accessibility* for a particular agent (e.g. which tools are performing within tolerance);
- *goal constraints* for multiple goals (e.g. deadlines for goal completion, goal priorities); and
- *environmental states* (e.g. air quality in clean room).

Therefore, manufacturing requires agent-based problem solving to be flexible and tolerant of faulty information, equipment, and communication links. This research uses Sensible Agent-based systems to extend agent capabilities in dynamic and complex environments (Barber, 1996). Sensible Agents achieve these qualities by representing and manipulating the interaction frameworks in which they plan to achieve their goals. Agent interactions for planning can be defined along a spectrum of agent autonomy as shown in Figure 1. An agent's *level of autonomy* for a goal specifies the interaction framework in which that goal is planned. Although autonomy is traditionally interpreted as an agent's degree of freedom from human intervention, the concept of autonomy in this context refers to an agent's degree of freedom with respect to other agents.

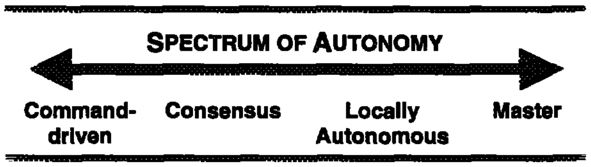


Figure 1: The Autonomy Spectrum

The autonomy spectrum is grounded in four discrete autonomy level categories:

Command-driven -- The agent does not plan and must obey orders given by another (master) agent.

Consensus -- The agent works as a team member, sharing planning tasks with other agents.

Locally Autonomous -- The agent plans alone, unconstrained by other agents.

Master -- The agent plans for itself and its followers who are command-driven.

Agents can be designed to operate at a single level of autonomy if: (1) the application is simple, (2) the designer correctly predicts the types of problem that agents will face, and (3) the environmental context and problem types

This research was supported in part by the Texas Higher Education Coordinating Board (#003658452) and a National Science Foundation Graduate Research Fellowship

remain constant. However, for complex applications in dynamic environments (i.e. most manufacturing problems), the appropriate level of autonomy may depend on the agent's current situation. An agent's optimal autonomy level may be affected by the agent's state, overall set of goals, and/or its environmental context. All of these characteristics may be dynamic and change during system operation. For example, scheduling agents in charge of directing parts to various process workstations to achieve maximum throughput may perform best by changing their problem-solving configurations as a result of load imbalance, the introduction of a new high-priority job, or machine failure. A Sensible Agent maintains solution quality in dynamic environments by using a technique that we call *Dynamic Adaptive Autonomy* (DAA). DAA is a capability that allows a Sensible Agent to modify its autonomy level for any goal during system operation. The process through which an agent chooses the most appropriate autonomy level for a given situation is called *autonomy reasoning*.

Related Work

The organizational structure of agent-based systems, which provides the mechanism through which agents coordinate or cooperate to achieve system goals, has been the subject of much research over the past few decades (Chandrasekaran, 1981; Fox, 1981; Kirn, 1996; Nirenburg and Lesser, 1986; Singh, 1990; Smith, 1980; Werner and Demazeau, 1991; Wesson et al., 1981). One overall goal of multi-agent-systems research is adaptive self-configuration: allowing agents to reason about and change the organization of their coordination frameworks (Gasser, 1988). Most self-organizing systems rely on explicit, predefined differences in agent behavior and limit the reorganization primitives to a fixed number of predefined behaviors (Gasser and Ishida, 1991; Glance and Huberman, 1993; Ishida et al., 1992). Others are based on adapting application-specific roles that agents can play during problem solving (Glaser and Morignot, 1997). These systems do not explicitly represent the agent's problem-solving role. This limits the ability of these systems to reason about the appropriateness and potential modification of an agent's problem-solving interactions in an application-independent fashion.

Multi-agent researchers use simulation environments to test algorithms and representations in order to accurately measure the impact of new research. Existing simulation environments include: DVMT (now DRESUN) (Lesser, 1991), MACE (Gasser et al., 1989), and MICE (Durfee and Montgomery, 1990). Additionally, a number of single-use simulations have been used to measure or compare performance for many research topics including swarm behavior (Beslon et al., 1998), constraint planning (Liu, 1996), and hierarchical organizations (Glance and Huberman, 1993). Unfortunately, most of these testbeds do not support distributed heterogeneous computing environments. Additionally, these simulation environments are not designed to support third-party connections. Third-party connections allow researchers to compare the performance of different algorithms and representations within a common simulation environment.

Sensible Agent Architecture

As defined for this research, an "agent" is a system component that works with other system components to perform some function. Generally, agents have the ability to act and perceive at some level; they communicate with one another; they attempt to achieve particular goals and/or perform particular tasks; and they maintain an implicit or explicit model of their own state and the state of their world. In addition, Sensible Agents have many other capabilities supported by the Sensible Agent architecture shown in Figure 2.

Each Sensible Agent consists of five major modules (Martin et al., 1996):

The *Self Agent Modeler* contains the behavioral model of this agent (the *self-agent*) as well as declarative information specific to this agent. A system designer can specify the desired services, data, and interfaces of each system agent through the Self Agent Modeler. This module interprets internal or external events acting on the agent and changes its state accordingly. Other modules (within the self agent) can access this model for necessary state information.

The *External Agent Modeler* contains knowledge about other agents and the environment. This module maintains beliefs about states and events external to the self-agent and predicts the actions of other agents. Other modules within the self-agent can monitor this module for changes that

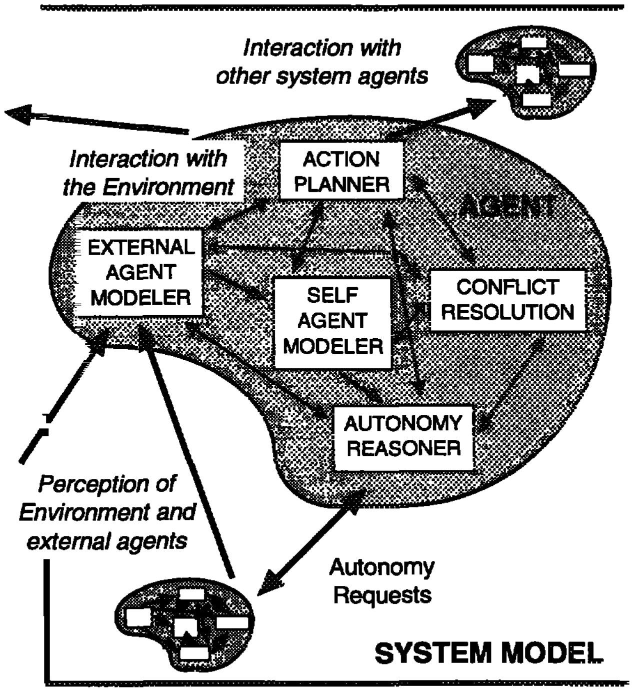


Figure 2: The Sensible Agent Architecture.

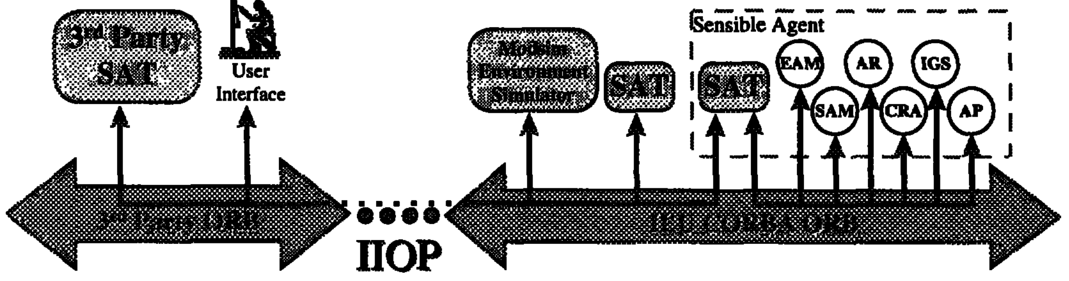


Figure 3: Functional View of Simulation Communication

affect their reasoning processes.

The *Action Planner* module solves domain problems and executes problem solutions. This module interacts with the environment and other agents in its system. Domain-specific problem solving information, strategies, and heuristics are placed inside this module. The Action Planner is able to interpret, plan to achieve, and execute solutions for domain-specific goals. This module draws information from all other modules in the self-agent.

The *Conflict Resolution Advisor* identifies, classifies, and generates possible solutions for conflicts occurring between the self-agent and other agents. This module monitors the Action Planner, Self-Agent Modeler, and External Agent Modeler to identify conflicts. It also offers suggestions to the Action Planner or Autonomy Reasoner so these modules may resolve the conflict.

The *Autonomy Reasoner* determines the appropriate autonomy level for each goal, assigns an autonomy level to each goal, and reports autonomy level constraints to other modules in the agent. It also handles all autonomy level transitions and requests for transition made by other agents.

Testbed Design

Accurate simulation of complex domains requires an environment that handles complex modeling issues and produces reasonable visual and numerical output of the current world state. The Sensible Agent simulation testbed will provide us with an environment for running repeatable experiments in which Sensible Agent functionality can be evaluated. The wide-ranging functionality and implementations of each module requires us to support a multi-platform and multi-language research environment. Our current implementation allows us to integrate C++, Java, Lisp and ModSIM implementations on Solaris, WindowsNT and Linux platforms.

Figure 3 illustrates the current Sensible Agent simulation architecture. The different modules and the environment simulator are implemented as CORBA objects that can communicate through the Xerox Inter-Language Unification (ILU) distributed object environment (Xerox, 1998). ILU provides the necessary cross-language and distributed object support for our research environment.

ILU's support of the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) standards allows us to use the OMG Interface Definition Language (IDL) to formally define the interactions among agents and their modules. The use of IDL permits continued Sensible Agent evolution in an implementation-language-independent manner and facilitates parallel research initiatives within the framework of an integrated distributed object system. Additionally, ILU's support of the CORBA Internet Inter-Orb Protocol (IIOP) standard will allow us to connect the simulation with external ORBs (OMG, 1998) -- and other applications that support IIOP – to further enhance the Sensible Agent Testbed's extensibility. The use of IIOP combined with the OMG's CORBAServices Object Naming Services (COSNaming) allows us to present a public interface to our Sensible Agent Testbed, allowing external entities to connect with our testbed for experimentation and collaboration. We are currently in the process of integrating the different modules and testing the IIOP communication channel.

The far right of Figure 3 shows how the different modules within a Sensible Agent can communicate with each other without loading the inter-agent communication medium with unnecessary message traffic. Figure 4 shows a logical view of these communication pathways. The Sensible Agent Template (SAT) provides a single handle which other Sensible Agents and the environment can use for the messaging and naming of an individual agent. The separation of the user interface from other agent functionality (the far left of Figure 3) allows us to collect data and view the simulation state without local access to the heavy computing power needed by the individual agents and modules.

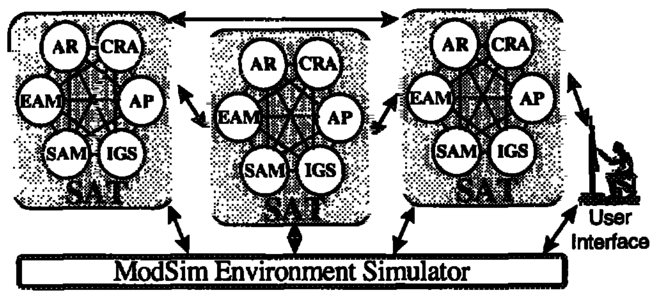
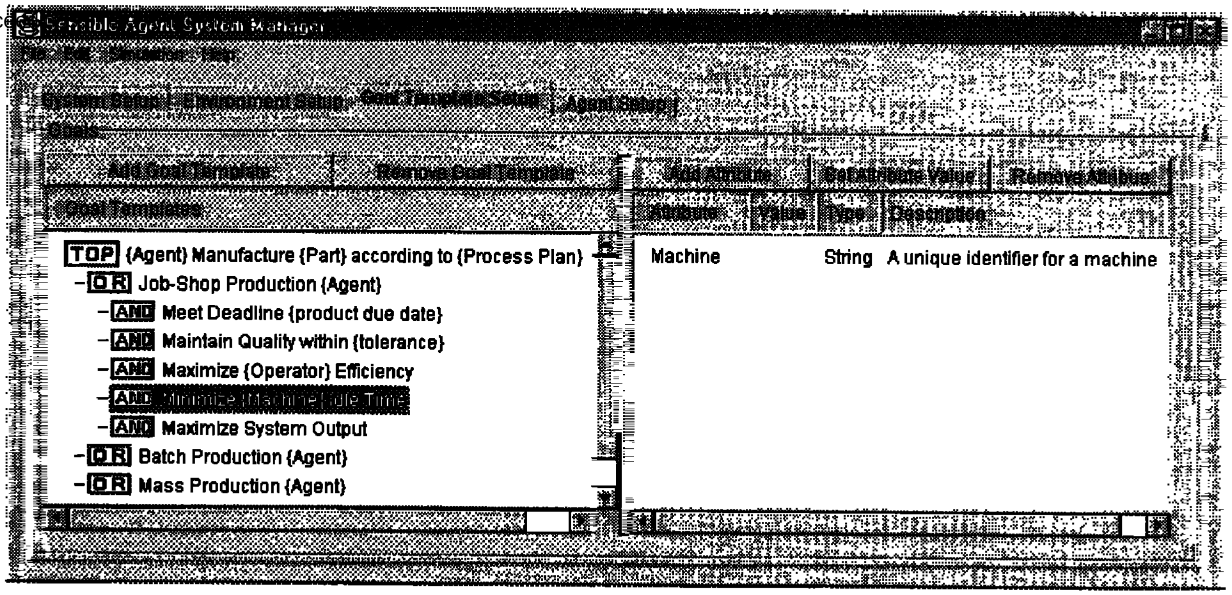


Figure 4: Logical View of Simulation Communication



TOP {Agent} Manufacture {Part} according to {Process Plan}
- OR Job-Shop Production {Agent}
  - AND Meet Deadline {product due date}
  - AND Maintain Quality within {tolerance}
  - AND Maximize {Operator} Efficiency
  - AND Minimize {Machine} Idle Time
  - AND Maximize System Output
- OR Batch Production {Agent}
- OR Mass Production {Agent}

Machine String A unique identifier for a machine

Figure 5: Sensible Agent System User Interface

Manufacturing Example

As applied to the manufacturing domain, the domain-specific planning responsibilities assigned to each agent result from object-oriented analysis and design for the application problem. Decisions regarding levels of abstraction per agent functional responsibility are made in the system analysis and design phase. For example, a process-planning agent may be defined which encapsulates resource selection, process selection, and costing services. Alternatively, each of these services may be assigned to individual agents (e.g. process selection agent).

A complete specification for agents comprising a particular Sensible Agent-based system in the manufacturing domain is beyond the scope of this paper. However, for the purposes of the following discussions, we assume a Sensible Agent-based system for job-shop scheduling. Resources on the factory floor (e.g. material handling devices, robots, machine operators) can be assigned to respective agents. Each of these agents controls access to production lines (to perform load balancing between lines) and job scheduling between machines on different lines (to reduce idle times for any one machine). The agent makes these decisions by monitoring resource availability and status, execution, and goal achievement. The following sections provide examples describing the Sensible Agent Testbed as it might apply to a manufacturing simulation. Individual module prototypes are provided. We are currently in the process of fully integrating these modules.

System Designer Interface

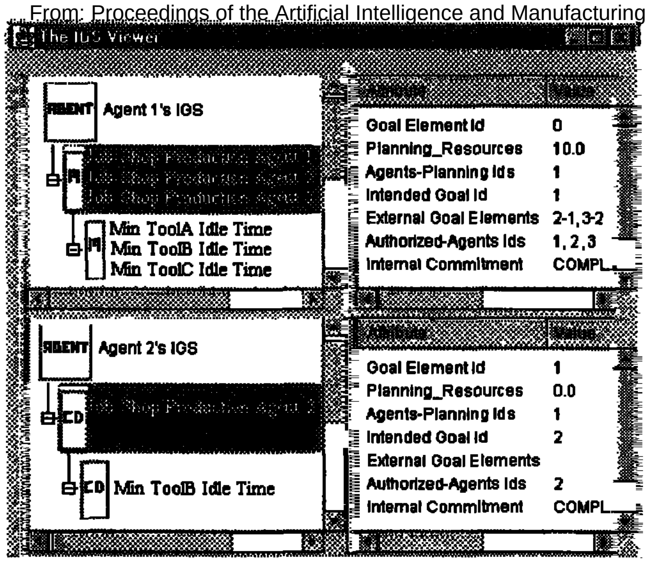
The Sensible Agent System User Interface, shown in Figure 5, provides facilities to initialize the system and view behavior of the agents during the simulation. Initialization consists of defining the agents in the system, assigning the initial goals to the agents, and defining the environmental stimuli for the simulation. The definition of each agent includes declarative knowledge, behavioral knowledge, and the goal tree from which the agent plans. During execution, the behavior of each agent can be displayed, including the internal workings of each of the modules, interactions among modules, and interactions among agents.

Figure 5 shows an example of the definition of a goal tree. Agents plan from AND/OR goal trees which represent the agents' alternatives (Jennings, 1993; Lesser, 1991). The goal/subgoal relationship reflects a task-reduction planning paradigm. Autonomy levels can be assigned at a single level or at each step of task reduction (to each goal, its subgoals, and any further subgoals these subgoals may have).

The goal tree displayed in Figure 5 defines how an agent would manufacture a part. The top-level goal displayed is an example of a goal template that would then be instantiated when placed in an agent's Intended Goals Structure. The items in curly braces, "{ }", identify those variables which must be instantiated in order to begin planning for a goal. For example, the top-level goal has {Agent}, {Part}, and {Process Plan} as variables that must be defined for an agent to accomplish that goal.

Intended Goals Structure

A Sensible Agent represents the goals it has chosen to pursue in an Intended Goals Structure (IGS). The IGS differs from a goal tree. The goal tree contains templates for candidate goals, which have not been accepted by any agent. On the other hand, the IGS is an agent's representation of the actual goals it will attempt to achieve



Agent 1's IGS

Min ToolA Idle Time
Min ToolB Idle Time
Min ToolC Idle Time

Goal Element Id	0
Planning_Resources	10.0
Agents-Planning Ids	1
Intended Goal Id	1
External Goal Elements	2-1,3-2
Authorized-Agents Ids	1,2,3
Internal Commitment	COMPL

Agent 2's IGS

Min ToolB Idle Time

Goal Element Id	1
Planning_Resources	0.0
Agents-Planning Ids	1
Intended Goal Id	2
External Goal Elements	
Authorized-Agents Ids	2
Internal Commitment	COMPL

Figure 6: The IGS Viewer

(its own intended goals) as well as any additional goals for which it must plan (external goals, intended by other agents). For a formal definition of agent intention, see (Cohen and Levesque, 1990). The IGS contains AND-only compositions of these goals and therefore does not represent alternative solutions or future planning strategies as goal trees do. Instead, the IGS represents what an agent has decided to do up to this point. Each agent maintains a unique IGS as shown in Figure 6.

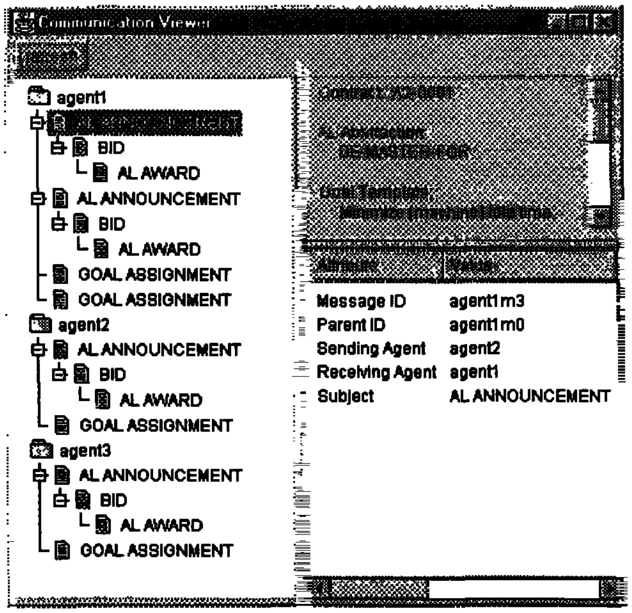
The example IGS Viewer in Figure 6 shows that autonomy levels are assigned to each goal in each agent's IGS. For example, the information displayed for Agent 1's goal "Job-Shop Production Agent 1, Job-Shop Production Agent 2, Job-Shop Production Agent 3" shows that Agent 1 is a master agent for this goal. Note that Agent 1, as master, must consider its own goal as well as goals intended by Agents 2 and 3 when it is planning for job-shop production. This goal is actually a combination of three intended goals, one for each agent in the planning framework. The combination of one or more goals is referred to as a goal element. Each constituent intended goal in this case is derived from the goal template "Job-Shop Production {Agent}," where "{Agent}" is a variable as described in Figure 5. Agent 2 is command-driven to Agent 1 for its corresponding goal "Job-Shop Production Agent 2." The model of Agent 3's IGS is not shown.

The details of Agent 1's autonomy assignment for the selected goal are described in the window to the right of its IGS display. For a formal discussion of autonomy levels and their representations, see (Martin, 1997). These details reflect that Agent 1 is planning alone for this goal. Agent 1 has therefore allocated some amount (10%) of its planning-resources toward this goal. As part of this overall goal, Agent 1 is considering its own intended goal (id=1) as well as two external goals intended by Agent 2 (id=2-1, Agent 2-Goal Element Id 1) and Agent 3 (id=3-2). Agent 1 is authorized to allocate subgoals to Agents 1, 2, and 3, and Agent 1 is completely committed to achieving this goal. The details of the complementary assignment for Agent 2, as a command-driven agent, are shown to the right of Agent 2's IGS window. These details reflect that Agent 2 knows Agent 1 is planning and has the authority to allocate subgoals to Agent 2. Notice that Agent 2 is not planning and therefore allocates no planning-resources to the task.

Agent Communication

Sensible Agents use their communication abilities for allocation of goals, negotiating Autonomy Level Agreements (ALA), and general information passing. Sensible Agents currently use a protocol based on the Contract Net Protocol (Smith, 1980; Smith and Randall, 1988) for allocating goals and for allocating planning responsibility through ALAs. A contract negotiation begins with a contract announcement, followed by bids from interested agents. Finally the announcing agent makes a decision and awards the contract to one of the bidders.

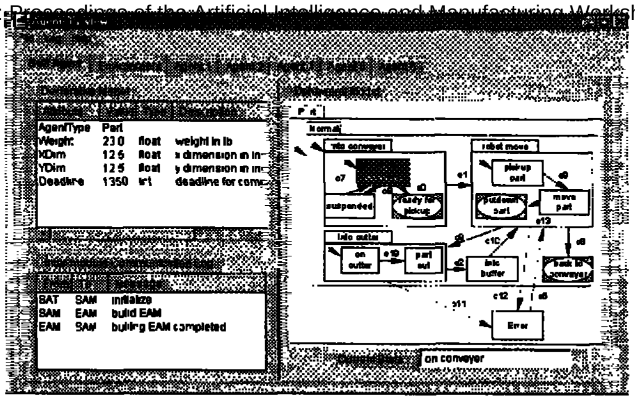
Figure 7 demonstrates this process. An initial announcement message is highlighted. Agent 2 is proposing to enter an ALA with Agent 1. The body of the message describes the proposed AL: Agent 2 has a goal, Minimize {machine} Idle Time, and would like Agent 1 to be its master for that goal. We can see that Agent 1 bid and was awarded the contract. Both sent and received messages are displayed as threaded conversations under both sending and receiving agents. A similar conversation occurs between Agents 1 and 3. Subsequent threads show that Agent 1 uses the GOAL ASSIGNMENT message to assign goals to Agents 2 and 3 under the authority granted by the Master/Command-driven Autonomy Level.



```
agent1
  [illegible]
    BID
      AL AWARD
  AL ANNOUNCEMENT
    BID
      AL AWARD
  GOAL ASSIGNMENT
  GOAL ASSIGNMENT
agent2
  AL ANNOUNCEMENT
    BID
      AL AWARD
  GOAL ASSIGNMENT
agent3
  AL ANNOUNCEMENT
    BID
      AL AWARD
  GOAL ASSIGNMENT
```

Message ID	agent1m3
Parent ID	agent1m0
Sending Agent	agent2
Receiving Agent	agent1
Subject	AL ANNOUNCEMENT

Figure 7: Communication Viewer



AgentType	Part		
Weight	23.0	float	weight in lb
XDim	12.5	float	x dimension in in
YDim	12.5	float	y dimension in in
Deadline	1350	int	deadline for comp

BAT	SAM	initialize
SAM	EAM	build EAM
EAM	SAM	building EAM completed

Figure 8: SAM/EAM Graphical User Interface

Self and External Agent Modelers

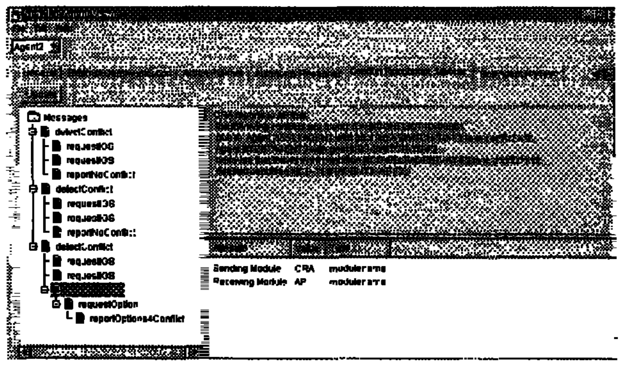
An agent's behaviors are described by its internal states and external events as well as system and local goals. An agent's understanding of the external world is built from its interpretations of the states, events, and goals of other agents and the environment. A Sensible Agent must be able to readily perceive, appreciate, understand, and demonstrate cognizant behavior. Behavioral and declarative knowledge are equally important for an agent to understand itself, other agents, and the environment.

Extended Statecharts (ESCs) (Suraj et al., 1997), a derivative of Statecharts (Harel, 1987) with respect to temporal and hierarchical extensions, have been developed as a comprehensive mechanism for the behavioral modeling of Sensible Agents. ESCs allow for the explicit representation of declarable, problem-specific failures. Exit-safe states represent states wherein a sub-state is in a stable state to process the transition out of a parent state. That is, events causing a transition between higher level (parent) states are handled only if the sub-state of the parents' state is exit-safe. This extension explicitly allows the developer to specify certain non-interruptible critical operations. Non exit-safe states do not allow for higher-level state transitions. For example, a part cannot make the transition directly to the robot-move state from the on-conveyer sub-state; the part must be in the ready-for-pickup sub-state before the part can be moved to the robot agent. Figure 8 shows ESC graphical notations and an example of the behavioral model for an agent. The declarative knowledge in the SAM is a set of attributes for the self agent and in the EAM is a set of attributes for other agents. The upper left of Figure 8 is a representative set of declarative knowledge for this domain. Each tab represents agent 3's beliefs about the other agents and the environment. If a new agent joins the system, a new tab is dynamically added to the view.

Action Planner and Conflict Resolution Advisor

Agents must interact intelligently, cooperating towards the completion of goals. The Action Planer (AP) uses a two staged planning methodology to (1) improve coordination among agents through the use of autonomy levels, and (2) leverage previous work in the planning field as applied to various domains (Barber and Han, 1998). The first stage consists of a hierarchical planner, providing the ability to differentiate between those activities that are critical to success and those that are replaceable details. This stage builds the IGS through the selection of subgoals and formation of ALAs. ALAs are formed among agents through the use of KQML in regards to the selected subgoals. The second stage provides an interface to plug in a domain specific planner. For each primitive goal in the IGS, this stage creates a plan by selecting the appropriate actions to execute. Autonomy levels are used as a selection aid for both stages (Barber and Han, 1998). During the planning process, conflicts may occur, including goal, plan, and belief conflicts. Both the IGS and the plan can be passed to the Conflict Resolution Advisor(CRA) to check for inconsistencies among agents.

A Sensible Agent can dynamically select a suitable conflict resolution strategy according to: (1) the nature of conflict (i.e. goal conflict, plan conflict, belief conflict), (2) the agent's social roles (represented by its autonomy levels), and (3) its solution preferences. *Utility* is used for decision making, which is the total weighted utility value of a specific solution for its attributes (see (Goel et al., 1996) for details of definition and calculation of utility based on AL). In addition to using utility to evaluate potential



Messages
detectConflict
requestIOG
requestIGS
reportNoConflict
detectConflict
requestIGS
requestIGS
reportNoConflict
detectConflict
requestIGS
requestIGS
requestOption
reportOptions4Conflict

Sending Module	CRA	modulename
Receiving Module	AP	modulename

Figure 9: User Interface for CRA

solutions, agents should also use certain indices (e.g. effectiveness, performance, agent properties, and system properties) to evaluate available CR strategies. Finally the agent must conduct some trade-off reasoning between solutions and CR strategies (Liu and Barber, 1998).

There are three kinds of decision-making styles that agents can make by tuning the weight factors:

1. CR strategies are selected after preferred potential solutions -- preferred solutions are the most important targets,
2. CR strategies are selected before preferred potential solutions -- the solutions will be the natural results

of applying specific CR strategies, and

3. Balanced consideration between preferred solutions and CR strategies.

In our preliminary experiments (Liu et al., 1997) we followed the second style to design a CR strategy selection algorithm for robot path planning. The results show that conflicts can be successfully resolved and the system can endure highly dynamic and uncertain environments.

The decision making approach proposed here can also be applied to conflict resolution problems in multi-agent systems which require the flexibility of applying multiple CR strategies. Figure 9 shows the user interface of the CRA for one Sensible Agent. The tree structure shows all messages this CRA has sent and received, threaded causally. The two panels on the right display detailed information about the selected message. In the scenario shown, the Action Planner (AP) has requested that the CRA detect conflicts. After checking its own IGS and those of external agents, the CRA reports on detected conflicts and explains the reason in the displayed message. CRA also recommends solutions and CR strategies to AP which executes CR strategies to eliminate conflicts. Execution of strategies may involve re-planning at either of the stages of the AP or adjusting the beliefs held in the SAM and EAM.

CONCLUSIONS AND FUTURE WORK

The need for responsive, flexible, and *sensible* agents is pervasive in manufacturing environments due to the complex, dynamic nature of manufacturing problems. Sensible Agents can be applied to many aspects of manufacturing from incorporation of manufacturing knowledge in design to shop floor control, and the capability of DAA may prove critical to the implementation of agile manufacturing via multi-agent systems. System designers cannot predict every combination of run-time conditions on the factory floor. Tool status, operator availability, raw material quality and accessibility, unscheduled maintenance, and machine wear can all introduce unexpected problems in the process plan. In order to maintain both high productivity and market responsiveness, manufacturing systems must be adaptive and flexible. Dynamic Adaptive Autonomy can provide agent systems which are capable of being both.

Although the Sensible Agent testbed currently provides much of the functionality required for testing Sensible Agents in manufacturing applications, much work remains. Future work for the AP module includes the construction of methods to facilitate the extension of planners or planning methods with DAA through incorporation into SA-based systems. In addition, automation of the task of autonomy reasoning (selecting the optimal planning framework in which to plan) is critical to the future development of Sensible Agent-based systems. We are investigating techniques of reinforcement learning and case-based reasoning for this task. These techniques will also be introduced in the CRA. We are currently working on a series of conflict resolution experiments for the CRA. Future work for the SAM and EAM modules includes incorporating more advanced reasoning capabilities and continuation of work on ESC knowledge representation to model uncertainty and time. The communication capabilities of Sensible Agents will soon be extended to include a variety of negotiation techniques. Sensible Agents will also be equipped to reason about which technique to use in various situations. The simulation environment is currently being extended to include support for inter-ORB communication and is undergoing testing to ensure that all internal and external agent messages are delivered appropriately.

References

Barber, K. S. 1996. The Architecture for Sensible Agents. In *Proceedings of the International Multidisciplinary Conference, Intelligent Systems: A Semiotic Perspective*, 49-54. Gaithersburg, MD: National Institute of Standards and Technology.

Barber, K. S. and Han, D. C. 1998. Multi-Agent Planning under Dynamic Adaptive Autonomy. In *Proceedings of the Accepted by 1998 IEEE International Conference on Systems, Man, and Cybernetics*. San Diego, CA. Forthcoming.

Beslon, G., Biennier, F., and Hirsbrunner, B. 1998. Multi-Robot Path-Planning Based on Implicit Cooperation in a Robotic Swarm. In *Proceedings of the Second International Conference on Autonomous Agents*, 39-46. Minneapolis/St. Paul, MN: ACM Press.

Chandrasekaran, B. 1981. Natural and Social System Metaphors for Distributed Problem Solving: Introduction to the Issue. *IEEE Transactions on Systems, Man, and Cybernetics* 11(1): 1-5.

Cohen, P. R. and Levesque, H. J. 1990. Intention is Choice with Commitment. *Artificial Intelligence* 42: 213-261.

Durfee, E. H. and Montgomery, T. A. 1990. Using MICE to Study Intelligent Dynamic Coordination. In *Proceedings of the IEEE Conference on Tools for AI*.

Fox, M. S. 1981. An Organizational View of Distributed Systems. *IEEE Transactions on Systems, Man, and Cybernetics* 11(1): 70-80.

Gasser, L. 1988. Distribution and Coordination of Tasks Among Intelligent Agents. In *Proceedings of the Scandinavian Conference on Artificial Intelligence, 1988 (SCAI 88)*, 189-204.

Gasser, L. and Ishida, T. 1991. A Dynamic Organizational Architecture for Adaptive Problem Solving. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 185-190. American Assosciation for Artificial Intelligence.

Gasser, L., Rouquette, N. F., Hill, R. W., and Lieb, J. 1989. Representing and Using Organizational Knowledge in DAI Systems. In *Distributed Artificial Intelligence*, vol. 2, Gasser, L. and Huhns, M. N., Eds. London: Pitman/Morgan Kaufman, 55-78.

Glance, N. S. and Huberman, B. A. 1993. Organizational Fluidity and Sustainable Cooperation. In *Proceedings of the 5th Annual Workshop on Modelling Autonomous Agents in a Multi-Agent World*, 89-103. Neuchatel, Switzerland.

Glaser, N. and Morignot, P. 1997. The Reorganization of Societies of Autonomous Agents. In *Multi-Agent Rationality: Proceedings of the Eigth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Boman, M. and van de Velde, W., Eds. New York: Springer, 98-111.

Goel, A., Liu, T. H., and Barber, K. S. 1996. Conflict Resolution in Sensible Agents. In *Proceedings of the International Multidisciplinary Conference on Intelligent Systems: A Semiotic Perspective*, 80-85. Gaithersburg, MD: National Institute of Standards and Technology.

Harel, D. 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* 8: 231-274.

Ishida, T., Gasser, L., and Yokoo, M. 1992. Organization Self-Design of Distributed Production Systems. *IEEE Transactions on Knowledge and Data Engineering* 4(2): 123-134.

Jennings, N. R. 1993. Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems. *The Knowledge Engineering Review* 8(3): 223-250.

Kirn, S. 1996. Organizational Intelligence and Distributed Artificial Intelligence. In *Foundations of Distributed Artificial Intelligence, Sixth-Generation Computer Technology Series*, O'Hare, G. M. P. and Jennings, N. R., Eds. New York: John Wiley & Sons, Inc., 505-526.

Lesser, V. R. 1991. A Retrospective View of FA/C Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6): 1347-1362.

Liu, J. S. 1996. Coordination of Multiple Agents in Distributed Manufacturing Scheduling. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University.

Liu, T. H. and Barber, K. S. 1998. Selection of Conflict Resolution Strategies in Dynamically Organized Sensible Agent-based Systems. Forthcoming.

Liu, T. H., Chuter, C., and Barber, K. S. 1997. Virtual Environment Simulation for Visualizing Conflict Resolution Strategies in Multiple Robot Systems. In *Proceedings of the 5th IASTED International Conference, Robotics and Manufacturing*, 154-158. Cancun, Mexico: IASTED Press.

Martin, C. E. 1997. Representing Autonomy in Sensible Agent-based Systems. Master's Thesis, Department of Electrical and Computer Engineering, University of Texas at Austin.

Martin, C. E., Macfadzean, R. H., and Barber, K. S. 1996. Supporting Dynamic Adaptive Autonomy for Agent-based Systems. In *Proceedings of the 1996 Artificial Intelligence and Manufacturing Research Planning Workshop*, 112-120. Albuquerque, NM: AAAI Press.

Nirenburg, S. and Lesser, V. R. 1986. Providing Intelligent Assistance in Distributed Office Environments. In *Proceedings of the ACM Conference on Office Information Systems*, 104-112.

OMG. (1998) "OMG Home Page." *http://www.omg.org/*, Current as of: June 25, 1998.

Singh, M. P. 1990. Group Ability and Structure. In *Decentralized A.I. 2, Proceedings of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Demazeau, Y. and Muller, J.-P., Eds. Saint-Quentin en Yvelines, France: Elsevier Science, 127-145.

Smith, R. 1980. The Contract Net Protocol: High-level Communication and Control in a Distributed Problem-Solver. *IEEE Transactions on Computers* 29(12): 1104-1113.

Smith, R. G. and Randall, D. 1988. Frameworks for Cooperation in Distributed Problem Solving. In *Readings in Distributed Artificial Intelligence*, Bond, A. H., Ed. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 61-70.

Suraj, A., Ramaswamy, S., and Barber, K. S. 1997. Extended State Charts for the Modeling and Specification of Manufacturing Control Software. *International Journal of Computer Integrated Manufacturing*.

Werner, E. and Demazeau, Y. 1991. The Design of Multi-Agent Systems. In *Proceedings of the Proceedings of the 3rd European workshop on Modelling Autonomous Agents in a Multi-Agent World*, 3-28. Kaiserslautern, Germany: Elsevier Science Publishers.

Wesson, R., Hayes-Roth, F., Burge, J. W., Staaz, C., and Sunshine, C. A. 1981. Network Structures for Distributed Situation Assessment. *IEEE Transactions on Systems, Man, and Cybernetics* 11(1): 5-23.

Xerox. (1998) "Inter-Language Unification." *ftp://ftp.parc.xerox.com/pub/ilu/ilu.html*, Current as of: June 25, 1998.