INTELLIGENT SOFTWARE AGENTS FOR NON-TRADITIONAL INFORMATION SHARING ENVIRONMENTS

(Extened Abstract)

Nina M. Berry and Soundar R.T. Kumara
Intelligent Design and Diagnostic Research Laboratory
Department of Industrial and Manufacturing Engineering
207 Hammond Building
The Pennsylvania State University
University Park, PA. 16802

ABSTRACT

This paper is primarily a discussion of the agent models and architecture needed to permit a collection of task-oriented intelligent software agents to resolve the objectives of an automated manufacturing domain. The major objectives involve maintaining the continuous flow of parts under production/transportation and the quick resolution of critical situations that could cause delays and bottlenecks in this domain. The agents used in this architecture represent the elementary tasks composed from the decomposition of objectives for the manufacturing domain. This architecture uniquely combines traditional software agents (e.g., planner) with agents that interface to physical entities (e.g., robots). These agents are integrated into an architecture to support individual and cooperative problem-solving schemes with an underlying blackboard system.

1 INTRODUCTION

The recent influx of agent-oriented systems for information transference has made the idea of using specialized agents to provide assistance to end users a reality. These specialized agents are also designed to represent a variety of specific task(s) for a given domain. The dynamic environment where many of these agent-oriented systems reside, naturally share the following three basic problem-solving characteristics: accurate coordination of the required tasks, flexibility to adapt to new tasks, and intelligence to react to unknown situations within a given domain. The design and implementation of an architecture based more on global information sharing and coordination, rather then traditional control issues is better at resolving the problem-solving and integration issues related to automated manufacturing.

The goal of this research is to address the fundamental problem of designing and developing a computational architecture with respect to an automated manufacturing domain. The resulting agent-based architecture requires the integration of centralized and cooperative

problem-solving along with replanning, to resolve the complex and dynamic issues. The manufacturing domain consists of four factory subsections responsible for the manipulation of pallets and crates. The pallets and crates are transported between each subsection by mobile robots capable of loading and unloading various machines or workstations in the domain. The major objectives of the manufacturing domain, are to ensure the continuous flow of parts and quick resolution of critical situations that could cause delays in the production process.

This paper presents the issues concerning the development of the agent architecture on top of the skeletal blackboard system, known as BB1[1]. The discussion is presented in two parts, the first contains issues of the architecture and the second with issues concerning the intelligent software agents used in the architecture. The first part presents the multi-agent architecture, communication, and problem-solving aspects of the agents. The second part provides an overview of aspects of agent identification, characteristics, and models.

2 MULTI-AGENT ARCHITECTURE

The architecture developed for this research combines aspects of problem-solving, blackboard architectures, and agent modeling (figure 1). This figure illustrates the integration of the eight intelligent agents designed in this research through the five internal blackboard panels, namely: execution, hypothesis, replanning, environmental, and learning. We treat the learning aspects as off-line and do not discuss them in this paper (these are indicated by darkened outlines). The line segments (those without arrows), connecting the agents to the four panels, permit the agents to communicate by using an event-based data structure. The agents cooperate by broadcasting event-based messages on the panels, where each event triggers other agents to respond with actions to accomplish the actions requested by the events. A detailed description of agent communication is given in the next section.

[1] BB1 originated at Stanford Univ. under Dr. Barbara Hayes-Roth.

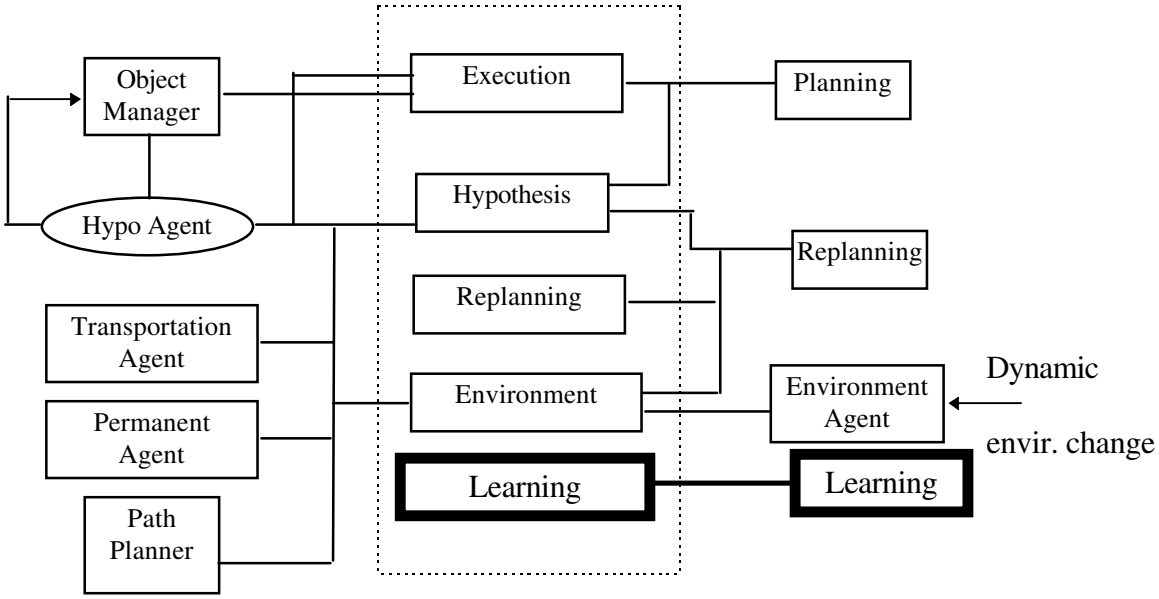

Figure 1: Multi-agent architecture with intelligent agents

3 AGENT COMMUNICATION STRUCTURE

The data structures used by the agents to communicate their desires (requests) are based upon the BB1 objects structures. BB1 provides an underlying object-oriented environment which all applications are built upon [Wolverton *et al.*, 94]. The BB1 architecture creates an event-object to represent all changes to the blackboard(s) by the application. The BB1 event-objects reflect typical object-oriented structures with sets of attributes and links [Ibid]. The links are used to inherit attribute values that are not directly associated with the object, but can be found through the link relationship to other object(s).

The hierarchical structure shown in figure 2(a), illustrates the inheritance process of the communication events-objects which are represented as *instances*. The event-object is an *instance* of one of the four *individual* categories of communication (cooperation, bid, data, and interrupt). The attributes for the four communication objects are found within the concept or *class* object called communication. This hierarchical design is used to accommodate the different attributes associated with the event *instances* of the individual categories of communication.

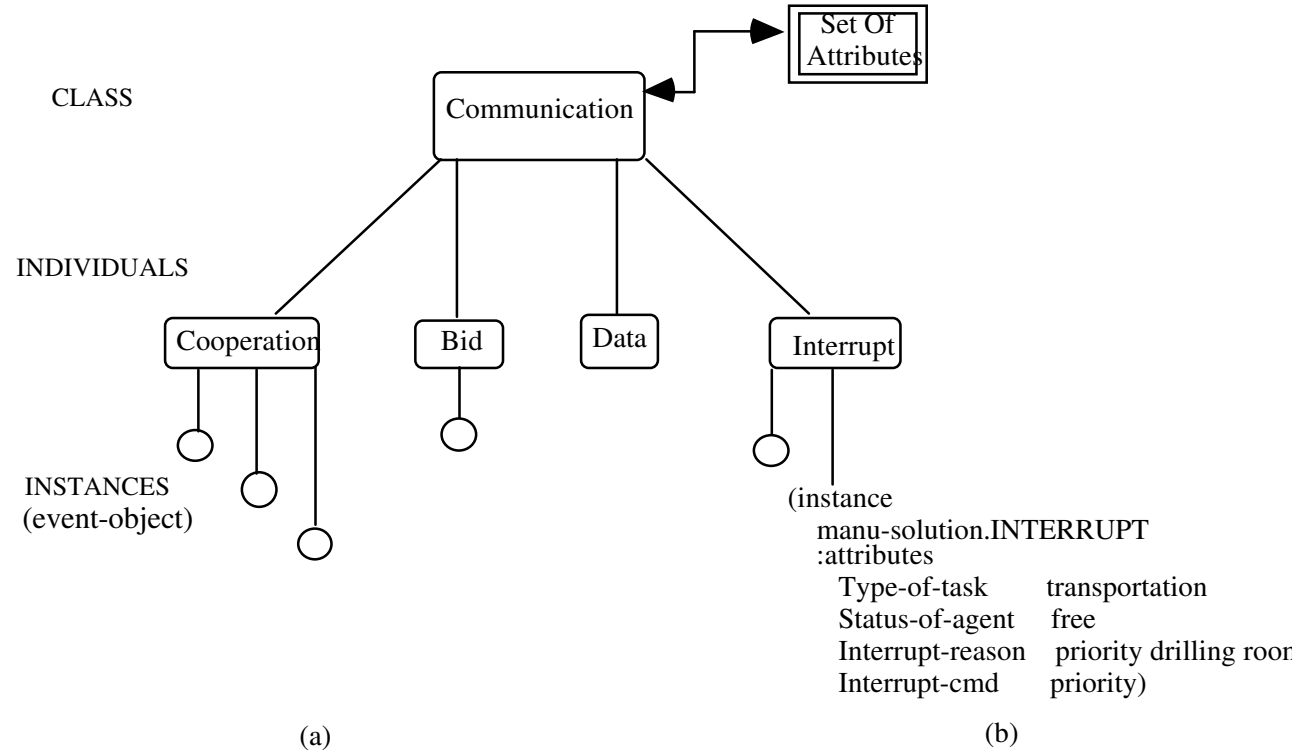

Figure 2: (a) Basic event object inheritance hierarchy and (b) example of an event-object

An example of an event-object for the *instance* of the INTERRUPT *individual* is shown in figure 2(b). This particular interrupt event will trigger an agent whose task is associated with transportation and the status for this agent must be free. The first line of the event establishes that the event-object within BB1 is of type *instance* and automatically establishes and instantiates the link for this object. The manu-solution.INTERRUPT signifies the first half of the name of the object and BB1 will append as identification number to the end of this name for differentiating this object from other INTERRUPT *instances*. The third entry related to the attributes with their values to the right of the attribute name. All of the attributes on this list originated in the *class* communication object at the top of the hierarchy. This general approach reduces the need to develop unique event objects for each possible event *instance*. Hence, another valid instance of an interrupt event could be created using a different set of attributes. Furthermore, this hierarchical structure supports the problem-solving schemes that are discussed in the next section.

4 PROBLEM-SOLVING

Another important aspect of the architecture involves the type of problem-solving being supported by the agent architecture. Both centralized and cooperative problem-solving are

directly established in this architecture. Definitions and examples of these two forms of problem-solving within our architecture are given below:

- Centralized
  - Definition:
    - Centralized problem-solving is the process of a central agent planning or coordinating the actions of the agents.
  - Example:
    - The *planner agent* creating ordered plans for the pallets and crates as they enter the factory domain. These plans depict the actions the other agents will directly perform on the pallets and crates.
- Cooperative
  - Definition:
    - Cooperative problem-solving refers to the agents ability to determine when they need some form of assistance to complete a task. The categories of assistance include information transference, agent reactions, and task interruptions.
  - Example:
    - *Transportation/Permanent agents* requesting current status of a pallet or crate is a type of information transference. These same agents requesting a path plan is a type of agent reaction request.

The inclusion of cooperative problem-solving indirectly relates to the autonomous existence of the agent. The autonomous agents reflect the rational and intentional aspects of decentralized problem-solving. The autonomous nature of the agents promotes a form of independent existence which is not governed or justified by the existence of other agents [Demazeau *et al.*, 90]. This independence permits the agent to accomplish its task even when detached from the original architecture. Furthermore, the autonomous agents do not directly send messages to other agents, since their existence in the architecture is not known to the other agents. The autonomous behavior of the agents permit them to work independently to solve their own problem/task, with respect to other agents.

5 AGENT IDENTIFICATION

The intelligent agents in this research represent an elementary task decomposed from the automated manufacturing domain. The decomposition process generates a set of explicit and implicit agents, defined as common internal task (CIT) agents. The decomposition of the internal tasks reflect the actions required to accomplish any problems within a given domain. One source of internal tasks are found in the predefined plans for the pallet and crate parts in the automated manufacturing domain considered. These individual action steps are extracted and grouped into categories of conceptual generalization of the steps. An example of steps extracted from several

predefined plans is shown in figure 3(a). A generalization of these steps into a many-to-one correlation between the extracted actions and the generalizable tasks is shown in figure 3(b).

Extracted steps from predefined plans

Pallet# ready for pickup at drilling machine# in drilling room .
Pallet# ready for pickup at pallet filling station queue_loc# in drilling room.
Crate# ready for pickup at nailing station#_queue_loc# in crating room.
Crate# ready for pickup at labeling station queue_loc# in crating room.

(a)

Generalized action

Signal a part is ready for pickup

(b)

Figure 3: (a) Extracted predefined plan steps and (b) generalization of the predefined actions

The agents are based on the results of task generalization (explicit agents), architecture requirements (implicit agents), and other domain requirements (explicit agents). The explicit agents include:

- transportation/permanent agent—represents the physical robots
- environmental agent—represents sensor information from the environment
- planning agent—generates the plans for pallets and crates

The implicit agents include:

- replanning agent—reacts to dynamic actions
- path planner agent—represents globalized perspective of the domain paths
- object manager agent—monitors the distribution of the plan steps
- learning agent—creates new rules for the planning and replanning agents
- hypo agent—represents the individual steps of the pallets and crates plans.

These agents must be capable of accomplishing all of the defined tasks for this domain either alone or through cooperative behaviors. These two requirement are primary in ascertaining the functionality of the agents which will discussed in the next section on agent models.

6 AGENT MODELS

The general model for the CIT agents in our multi-agent architecture is shown in figure 4 which consists of control, problem-solving, and execution components. The trigger reasoner module determines what task the agent is capable of handling and represents the main module in

the control component. Once the agent has been triggered successfully the flow of control is transferred through the problem-solving module to the execution component.

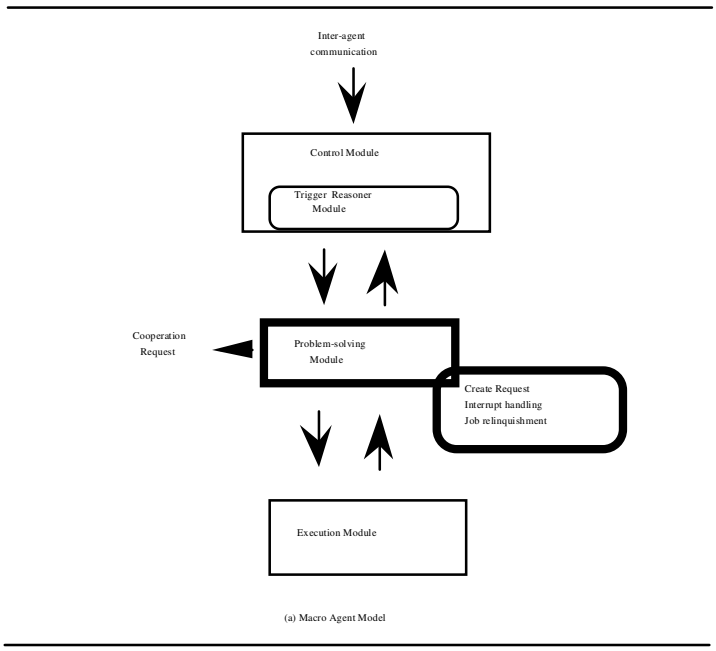

Figure 4: Models of CIT agents

The problem-solving module is specifically used to handle situations where the normal task reasoning in the control and execution modules cannot be evaluated correctly. This problem-solving component differs, depending on the classification of the agent. A problem-solving component for the *transportation/permanent agents*, would require an interrupt handling due to the reactive and reflexive characteristic requirements of these agents. Based on these characteristics, the agents should have the capabilities of handling dynamic situations in a stimulus-response manner.

The execution module represents the intentions or actions the agent will execute to accomplish the single task. The execution module represents a loop of execution that differs based on each agent. If an interrupt occurs within this loop, control is transferred back to the problem-solving module for evaluation. Once the agent has completed the execution loop control is returned to the controller module, where the agent watches for another task to accomplish.

7 SIGNIFICANTS OF AGENT ARCHITECTURE

Today, the importance of intelligent software agent systems are aggressively being explored in information intense domains. However, the application of these types of agents to

automation of manufacturing remains a somewhat un-chartered area of application. The area of automated intelligent manufacturing provides a unique challenge for intelligent agent systems to replace, the traditional control issues used to resolve the problem-solving and integration with approaches related to global information sharing and coordination. This architecture uniquely addresses some interesting issues and provides some fascinating approaches to agent integration and coordination:

- Represent both software programs and physical objects as intelligent software agents, then integrate them into the same architecture.

- The agents are unaware of the explicit existence of other agents; instead the architecture and agents were designed to function on a task request level instead. Therefore, the assumption is that an agent(s) exist that can respond to an agent's request.

- Agents achieve their desires/actions based on their localized beliefs of the environments and without considering or negotiation actions with the other agents.

- The agents are naturally greedy in their needs to achieve their own desires; however, this also makes the coordination and cooperation significant if the agent is going to establish accurate beliefs about the domain.

The significants of this architecture to the development of intelligent manufacturing systems would be three-fold:

1. The conversion of current computer integrated manufacturing systems to intelligent agents could be easily accomplished through the upgrading of controller code to autonomous agent code.

2. The resulting plug n' run integrated architecture will permit for easier upgrading and modifications to the overall system.

3. The agent based architecture could be generalized by addressing scalability and portability issues.

8 CONCLUSIONS

In this paper we have proposed a framework for the application of intelligent agents to areas outside of the traditional information processing arena. The aspects of the architecture built on the BB1 system are discussed. The items presented included agent communication, problem-solving, agent identification, agent characteristics, and agent models. Furthermore, the concepts discovered in this research will benefit further study into agent resolution and adaptation to inherently complicated issues surrounding dynamic environments.

9 ACKNOWLEDGMENTS

Ms. Nina M. Berry wishes to acknowledge the financial support from the NSF-GEE Fellowship #EID-9116806 and PSU Academic Computing Fellowship programs.

10 REFERENCES

[Demazeau *et al.*, 90]
Y. Demazeau and J.-P. Mueller. Decentralized AI, North-Holland, New York, New York, 1990.

[Martial, 91]
F. von Marital. Coordinating Plans of Autonomous Agents, Springer-Verlag, New York, New York, 1991.

[Wolverton *et al.*, 94]
M. Wolverton and L. Brownston. *BB1 v3.2 Manual*, Knowledge Systems Laboratory Report No. KSL 94-XX, Department of Computer Science, Stanford University, 1994.