Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Computer-Integrated Manufacturing 20 (2004) 473–483
Robotics and Computer-Integrated Manufacturing
www.elsevier.com/locate/rcim

A multi-agent architecture for control of AGV systems

Pooya Farahvash, Thomas O. Boucher*

Industrial & Systems Engineering, Rutgers University, 96 Frelinghuysen Road Piscataway, NJ, 08854, USA

Abstract

Agent is an autonomous, computational entity that can be viewed as perceiving its environment and acting upon it. Agents are event-driven objects that can be integrated in automated manufacturing environments to control certain tasks. In this paper a set of agents (a multi-agent system) is introduced to control an automated manufacturing environment. The architecture includes functions at the manufacturing cell level, materials handling and transport level, and factory scheduling level. Communication between these agents is accomplished by using a relational database (blackboard system). The relational database also integrates the requirements of a manufacturing execution system within the multi-agent task structure, which is unique to this architecture. Manufacturing cell and scheduling agents have been previously described in the literature. Here we focus our attention on the functions of the agents of the transport system, which is composed of a set of AGVs.

Keywords: Multi-agent systems; Automated guided vehicles; Flexible manufacturing systems

1. Introduction

Generally speaking an agent is a software (or hardware) object that is capable of doing some specific tasks autonomously. Parunak [1] defines an agent as a "proactive object". That is, a software entity is an agent if it has the data and code encapsulation of a software object, its own thread of control (making it an active object) and the ability to execute autonomously without being invoked externally (thus proactive rather that reactive). Wooldridge [2] has another definition: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." The definition of agent that has been followed in this paper is that which Weiss introduced [3]: "Agents are autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors." This definition is related to Wooldridge's. Sensors and effectors can be either physical (field devices) or software (files or data streams). From these definitions the following characteristics can be introduced for agents:

- *Autonomous*: The agent can work independently. Independent in the sense that it can perform its assigned jobs without the need of continuous interference of human intelligence.
- *Interacting*: Agents may affect other agents or may be affected by other agents or even humans. Communication is one of the major characteristics that must be considered in the design of multi-agent systems. Also they can interact with their environment through sensors and actuators.
- *Intelligent*: For the purpose of pursuing and executing tasks, some level of intelligence must be present in the design of the agent. The agent operates rationality in a variety of environmental circumstances.
- *Flexible*: Agent design and architecture should consider different environmental states and act properly in different situations.

A *multi-agent system (MAS)* is a loosely coupled network of problem-solver entities that work together to

*Corresponding author. Tel.: +1-732-445-3657; fax: +1-732-445-5467.
E-mail address: tboucher@rci.rutgers.edu (T.O. Boucher).

0736-5845/$ - see front matter
doi:10.1016/j.rcim.2004.07.005

find the answer to problems that are beyond the individual capabilities or knowledge of each entity [4]. In other words a MAS is a set of agents that either cooperate or compete to solve a problem. A MAS is usually used in cases where the problem is complex, data is decentralized, and computation is asynchronous. In such cases, it is preferred to distribute tasks over a number of agents and let them work autonomously and also interact with each other.

A major subject of research in multi-agent systems is architecture design. The central issues in architecture design are the responsibilities and behavior of individual agents, the coordination of activities among agents, and the communication protocols to be employed. Some examples applied to flexible manufacturing systems (FMS) exist in the literature. MetaMorph [5] is an agent-based, mediator–centric architecture. In MetaMorph, mediators assume the role of system coordinators among intelligent agents, such as machine agents and tool agents. Communication is accomplished using Knowledge Query and Manipulation Language (KQML) protocol [6]. A completely decentralized approach is introduced in [7]. Here, workpieces and workstations are represented by agents. As workpieces are introduced into the FMS, an instance of a workpiece agent is created. The workpiece agent negotiates with workstation agents for the operations necessary to manufacture the parts. In [8], a conceptual architecture of an adaptive production control system is introduced. Workpieces and resources are represented by distributed agents that negotiate for work through a bidding process. If imbalances arise in the work allocation, a supervisory agent can be introduced to assess the current status of a group of resources and optimize the workload across the resources within its span of control. In [9], an architecture is proposed that incorporates error recovery when system failure occurs. In addition to agents for planning and control, recovery agents are designed to diagnose error states and provide recovery plans. Recovery agents are coordinated with cell, workstation, and equipment agents using mediator agents, whose purpose is to facilitate communication.

Some researchers have focused on architectures for shop floor material handling systems. In [10], AGVs are represented by intelligent agents. They bid against each other for transporting loads between pick up and delivery points and determine the best paths for accomplishing the task. Communication among agents and pick up/delivery stations is accomplished using a blackboard system. In [11], colored Petri nets and simulation are used for performance analysis of a transport system with an agent architecture consisting of part agents and AGV agents. The architecture is completely distributed. Parts initiate a bidding process by broadcasting a request, to which AGVs respond. However, the emphasis here is on evaluating the performance of different physical transport system designs under the control of the proposed agent architecture. A hybrid architecture that includes AGV agents and other warehouse transporter agents is introduced in [12,13]. This architecture includes both distributed agents and hierarchical control agents. Lower-level agents are allowed to negotiate with one another within the boundaries set by the higher-level agents. Communication among agents is accomplished using a blackboard system. The researchers show how the proposed architecture is applied to warehouse picking operations using gantry robots and to a shop floor environment consisting of part agents, Cell Agents, and material handling agents, including AGVs.

In the current proposal we introduce an architecture that integrates shop floor agents for scheduling, cell control, transportation, and material management. Agents are designed at the supervisory control layer. These agents operate autonomously and interact with each other using a blackboard system that is unique in that it integrates database requirements for the factory manufacturing execution system (MES) within the agent framework. The remainder of this paper is structured as follows. In Section 2 we describe the agent architecture adopted in this research. In Section 3 we discuss the architecture of the multi-agent system. In Section 4 we describe agent communication. In Section 5 we discuss a typical example of agents coordinating their activities in the completion of a task. This is followed by a description of methods used by the agents in controlling the AGV system. An experimental case study is discussed in Section 7. Some concluding remarks are given in Section 8.

2. Agent architecture

Agent architecture is a map of the internals of an agent and its data structure, the operations that may be performed on these data structures, and the control flow between these data structures. Agents that we designed in this project are reactive agents. Wooldridge defines reactive agents as agents in which decision-making is implemented in some form of direct mapping from situation to action. He introduces a method to formalize the architecture of reactive agents. We follow this formalization, but introduce new components to this architecture.

An agent will typically sense its environment (either by physical or software sensors) and, after performing some computations, issue action(s) through its communication process or through its actuators; see Fig. 1. We assume all states of an agent's environment can be shown by a set $S : S = \{s_1, s_2, s_3, \ldots\}$. At any given instant, the environment is assumed to be in one of these states. All actions that can be performed by