Int j simul model 6 (2007) 4, 206-217
ISSN 1726-4529 Original scientific paper

AGENT-BASED SIMULATION OF A SHOP FLOOR CONTROLLER USING HYBRID COMMUNICATION PROTOCOLS

Komma, V. R.; Jain, P. K. & Mehta, N. K.
Department of Mechanical and Industrial Engineering,
Indian Institute of Technology Roorkee, Roorkee – 247667, India
E-Mail: vrkomma@gmail.com; pjainfme@iitr.ernet.in; mehtafme@iitr.ernet.in

Abstract
Agents are autonomous and computational entities that perceive their environment through sensors and act upon their environment through effectors. Application of agents in simulation of manufacturing system helps in modelling the concurrent behaving entities close to the real system. A manufacturing agent should be modelled to impart the required functionality to the manufacturing entity. As agents are autonomous, they run asynchronously and need to be synchronized with the simulation clock for simulating the manufacturing system. This paper deals with the synchronization of manufacturing agents, working of the agent-based simulator of a shop floor controller for a sample manufacturing system and the hybrid communication protocols involved during simulation. The manufacturing agents were built on JADE™ platform, which is a leading multi agent development framework that helps in developing FIPA compliant agent-based systems. Agent-based modelling and simulation of entities in the manufacturing system provides flexibility for real-time decision making.

(Received in June 2007, accepted in August 2007. This paper was with the authors 1 month for 1 revision.)

Key Words: Agent-Based Simulation, Agent-Based Manufacturing System (ABMS), Communication Protocols, Foundation for Intelligent Physical Agents (FIPA), Java Agent DEvelopment Framework (JADE™)

1. INTRODUCTION

Today's manufacturing system is facing great challenges from the customers in meeting the sudden changes in demand and increased product variety due to more personalized products. This market situation forces the manufacturing systems to be more flexible, more agile and more reliable. Discrete-Event Simulation (DES) is the commonly used tool in industries to study the performance of a manufacturing system. Most of the commercial software packages used in industries for simulation of manufacturing systems are implemented either with functional or object-oriented programming approaches [1].

Multi Agent System (MAS) is an important and relatively new specialization of Distributed Artificial Intelligence (DAI). Multi-agent technology has already been applied in many engineering applications, of which manufacturing is a leading one. Communication among agents in an agent-based system plays vital role in cooperativeness and decision making. In this paper, working of an agent-based simulator of a shop floor control is demonstrated and two important hybrid communication protocols are discussed, which have been used during the simulation. A brief introduction to agent-based technology and its relevance in meeting the challenges associated with the dynamic behaviour of the shop floor control is given below for better insight of the problem.

DOI:10.2507/IJSIMM06(4)2.096

1.1 Agent technology

Although many definitions for an agent are reported in literature, the most agreeable definition among them is: an agent is a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its design objectives [2]. In another similar definition, agents are defined as autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors [3]. From the above definitions, it follows that the common attributes that an agent should posses are autonomous behaviour, cooperative behaviour, reactive behaviour and proactive behaviour. On the basis of the characteristics of agent architectures, agents are classified as logic based, reactive, Belief-Desire-Intention (BDI) and layered agents [3].

In most of the previously reported works, Java™ has been reported as a programming language for MAS development due to its built-in support for multi threaded programming and its other useful features. Building agents from scratch is a time consuming and tedious task, therefore, researchers have developed several platforms for fast development of MASs, and most of them are freely available [4]. However, these platforms were highly specific to the application and rarely of use to others. Moreover, agents developed on one software platform were difficult to interoperate with agents built on another software platform. This forced the agent community to move towards preparing standards for agent interoperability and MAS development. One such effort is specifications from Foundation for Intelligent Physical Agents (FIPA) [5], a non-commercial Swiss based organization established in 1996. In course of time, FIPA specifications got wider acceptance from the agent community. FIPA promotes agent-based technology and the interoperability of its standards with other technologies. Very few of the available MAS development tools support FIPA specifications. Java Agent DEvelopment (JADE™) [6] framework, a non-commercial software developed at Telecom Italia Lab (TILab), Italy, is one of the best platforms that support FIPA specifications. Hence, JADE has been used in the present work for the development of manufacturing agents in the agent-based simulation model of a shop floor controller.

1.2 Shop floor control system

A dynamic market with high product variety demands more agility in the manufacturing system for which computer controlled machines are being employed on the shop floor with a flexible material handling system. Automated Guided Vehicle (AGV) is a widely used material transporting equipment in modern automated material handling systems due to its routing flexibility. Shop floor control involves control of both individual workstations and AGV system. Control of AGV system involves dispatching and routing of multi-AGVs while avoiding collisions and deadlocks along the guidepaths. As entities on the shop floor behave in a highly concurrent and dynamic fashion, distributed or decentralized approach is suitable for shop floor control. Thus, the problem of shop floor control should be decomposed in such a way that the decision-making is performed by many simple, autonomous and cooperative entities rather than following a predetermined plan. The distributed intelligent control with agent-based approach provides the advantages of adaptability, ease of upgradeability and maintenance and emergent behaviour. On the other hand, the disadvantages of this approach are that global optima cannot be guaranteed and predictions of the system's behaviour can only be made at the aggregate level [7].

2. RELATED WORK

Wide-ranging details of agent-based technology and MASs were addressed in [3, 8]. The development of agent-based technology in different applications and its past, present and future developments were reported in the roadmap of agent technology [4]. In the past one-and-half decade, researchers have applied agent technology to manufacturing enterprise integration and supply chain management, manufacturing planning, scheduling and execution control, materials handling and inventory management, and in developing new types of manufacturing systems such as Holonic Manufacturing Systems (HMS). A complete review of Agent-Based Manufacturing Systems (ABMS) can be found in [9]. Shop floor control mainly concerns with the control of the AGV system and the individual machines both. In [10], AI was used to develop an AGV controller for large complex guidepaths. The article reported that the agents were used as traffic managers, which facilitated the AGVs to access points and segments of the guidepath. An important simulation tool called Manufacturing Agent Simulation Tool (MAST) was developed for simulation of material handling system in [11]. A complete overview of agent-based manufacturing was presented in [12].

The preliminary steps in developing the agent-based simulation model for a shop floor controller are modelling of manufacturing agents and development of manufacturing domain-specific ontology. These steps are followed by synchronization of agents with the simulation clock and analyzing the working of the simulator for verification of the model. Modelling of manufacturing agents and ontology development for shop floor focusing on AGV system were reported in [13], which are adopted in this paper. A brief description of the agent modelling and ontology development issues relevant to the present paper is given below.

The manufacturing agents were built over the JADE framework by extending *jade.core.Agent* class. They were modelled with reactive and proactive characteristics. JADE FSMBehaviour and SimpleBehaviour (especially with multiple steps) were used in building the behaviours of most of the manufacturing agents. Behaviour objects can be dynamically added or removed to the agent's behaviour list to handle the messages from other agents. During communication among the manufacturing agents, information is exchanged for cooperativeness and decision making. For interpreting the content of the message by the receiver agent with the same meaning as that of the sender agent, they should share the common domain-specific ontology. For the manufacturing system, a semi-formal ontology for shop floor focusing on AGV system was developed on Protégé, a leading frame-based ontology and knowledge editor, available from Stanford University, USA. The developed ontology is translated into FIPA/ JADE compatible Java classes with the help of ontology '*Beangenerator'*, a plug-in for Protégé. JADE compliant ontology consists of the concepts, predicates (relations) and agent-actions. Concepts are the 'objects' in the domain which are represented as classes in Protégé editor. An 'object' carries some attributes which are represented as slots in the classes. Predicates represent the relations between the concepts in the domain, for example '*PartOnAgv'* represents "a part is loaded on AGV". When a predicate is used as a content of the message, the expected result from the receiver is either true or false. The agent-actions are special type of concepts that represent the actions that agents can perform, for example '*Transport'*, '*Operation'* etc.

From the above literature, it is identified that more focus is required in standardizing the agent-based technology in terms of development/standardization of its architecture, ontology, agent communication for realizing the agent-technology in the shop floor control of a factory. In this paper, synchronization of manufacturing agents during simulation is discussed which is important for agent-based simulation. Working of the agent-based simulator for a sample manufacturing system is demonstrated for Java code verification and two important hybrid communication protocols are discussed which are used in the agent-based simulator. The agent-based simulation provides a better opportunity for analyzing and decision making in the shop floor control.

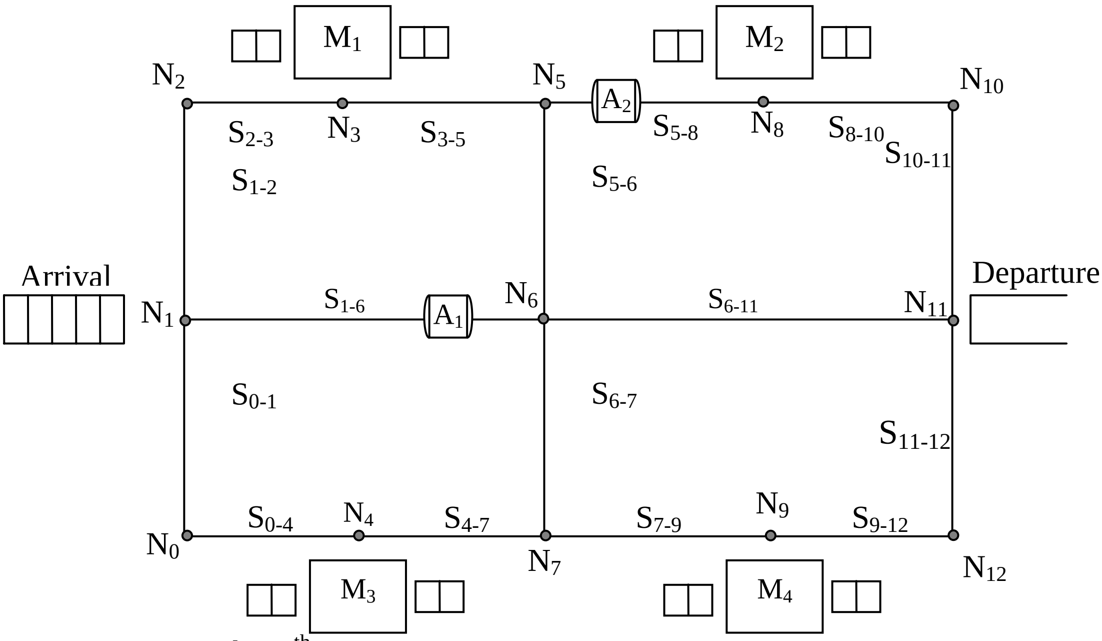
3. SYNCHRONIZATION OF AGENTS

In an agent-based system, agents are autonomous and run concurrently in an asynchronous manner. As agents are used as entities in simulation model, it leads to Parallel Discrete Event Simulation (PDES). Synchronization is an important aspect that must be dealt in developing PDES [12]. Historically, two principal approaches were suggested for advancing the simulation clock: next-event time advance and fixed-increment time advance. In the first approach, simulation clock and event list interact to determine which event will be processed next, the simulation clock is advanced to the time of this event, and the computer executes the event logic. In the second approach, the simulation clock is advanced in small time steps and all the events scheduled at that time are simulated [1]. The first approach is common in most of the simulation models.

For synchronizing the events that occur in several manufacturing agents, a JADE agent is developed that maintains shared variables of global virtual time (GVT) and a global event list. As this agent shares its attributes against the basic property of an agent, it is referred here as “Timer-Thread”. Each agent maintains a local virtual time (LVT) and a local event list. Maintaining the whole list of future events of agents in the global event list is inefficient as it can lead to a long list of events. The purpose of global event list is to provide synchronization to the agents while performing their activities but not to keep the whole list of events. GVT is advanced by fixed and equal increment of time with large time steps, which is similar to fixed-increment time advance mechanism. Within the time step (time between the start of two consecutive time steps), time is advanced as next-event time advance mechanism. In the present work, both the approaches of simulation were combined to achieve effective synchronization. Whenever GVT is advanced by Timer-Thread it notifies all the registered agents by sending a message to all of them and the agents update their LVT to GVT. Subsequently, the registered agents with the Timer-Thread post their individual events that must be executed before the upper limit of the time step into the global event list. Once the agents have posted their events in the global event list, the events are sorted according to their time stamp after which the agents are allowed to execute the events in sequence. The size of the time-step of GVT and the number of agents registered with the Timer-Thread are critical factors for maintaining the balance between the size of the global event list and the number of messages being exchanged within a specific time period of the simulation. If the time-step size is increased the size of the global event list increases, while reduced time-step size increases the number of messages being exchanged within a specific time period. While an agent is executing an event, it may interact with other agents to complete the event logic. This implies that the agents are not blocked from their execution process for the purpose of current event execution; they are free to participate in the communication and retain the nature of parallel simulation. The process of informing the registered agents on each time step increment resembles FIPA-Subscription-Protocol.

A behaviour class known as ‘*SimulatorBehaviour*’ was developed that helps an agent to post the agent’s events in the global event list and also retrieves them while synchronizing with time, which enables event-driven process. Each manufacturing agent extends the ‘*SimulatorBehaviour*’ to a ‘*TimerBehaviour*’ in which some of its functions (methods of the super class) are overridden to make the behaviour, specific to the manufacturing agent. In the following section, working of the simulation model is briefly explained with an example.

4. WORKING OF THE SIMULATION MODEL – A SAMPLE STUDY



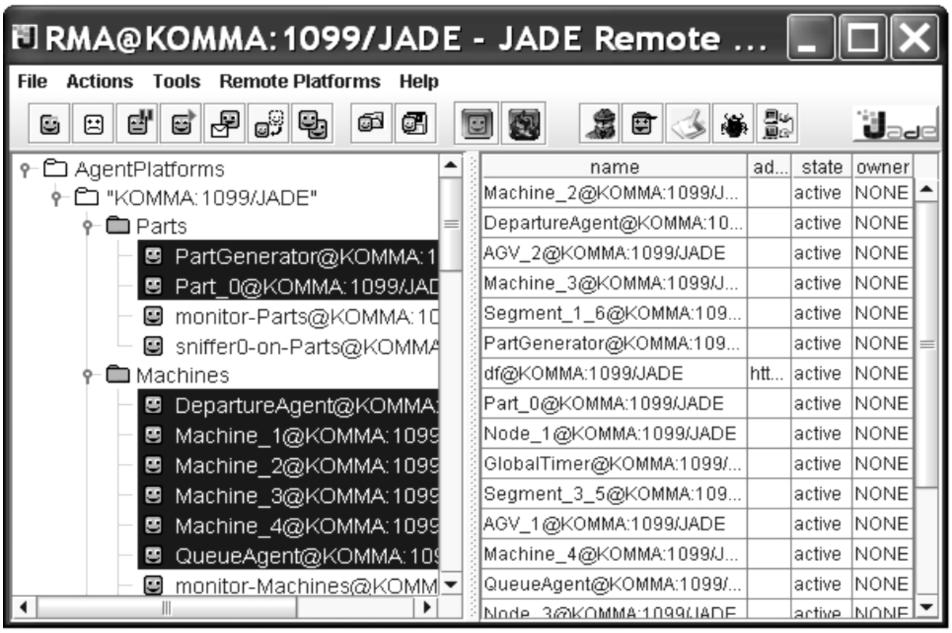
Legend: N_x → x^{th} Node
M_x → x^{th} Machine with its input and output buffers
A_x → x^{th} AGV
$S_{x\ y}$ → Segment connected node x to node y (x < y)

Figure 1: Sample model of a shop floor considered for demonstration.

Let us consider a hypothetical discrete part manufacturing environment that consists of four machines, two AGVs, one arrival and one departure location as shown in Fig. 1. Parts arrive at arrival queue (system input queue) in unit quantity at discrete points of time. Three part types were considered with a specific proportion of part mix. The arbitrarily chosen layout of the shop floor consists of 13 nodes (N_0 to N_{12}) and 16 segments. Nodes are regularly placed at intervals of 2 length units and all the segments are assumed as bidirectional. For simplicity, it is assumed that the segments are orthogonally connected and a junction node can have a maximum of four connected segments around the node (for example, N_6). The layout of the flowpath has been represented as an adjacency matrix ($A_{Ni,Nj}$, i and j range from 0 to 12) and the distances between two adjacent nodes are represented as a distance matrix ($D_{Ni,Nj}$, i ranges from 0 to 12 and j varies from 1 to 4). All possible paths from a source node (i.e., current position of AGV) to all other nodes in the layout are effectively determined by a recursive algorithm based on the well-known Dijkstra's shortest-path finding algorithm. These paths are sorted in ascending order of their path lengths. The recursive algorithm is repeatedly applied for all pairs of nodes in the network and all the paths are stored in the form of tables. These tables are look-up tables that reduce the computational time of on-line controller. On failure of any segments, the affected tables can be selectively updated. Parts' information such as number of part types, proportion of part mix, number of stages required to process each part type, sequence of machines and processing times for completing each part type are provided in an input file to the simulation model. It is assumed that at the start of the simulation, both the AGVs are at the centre of the layout i.e. at N_6. The speed of the AGVs is assumed as 1 length unit/time unit.

The manufacturing system consists of many agent types such as part-generator agent, system input queue agent, part agent, machine agent, AGV agent, node agent and segment agents. The part-generator agent imitates the arrival of the parts by generating parts at different points of time. System input queue agent simulates the input queue of the system where parts arrive onto the shop floor. A part agent is responsible for the part activities on the shop floor. A machine agent is responsible for a workstation that consists of a machine and its input (IB) and output (OB) buffers. Demonstration of the working of the agent-based simulation model with the traced events and activities that occurred during a trial run of simulation for a sample time period is given below.

JADE platform was started with its default main-container (container is a platform in which several agents can reside) which consists of AMS (Agent Management Services), DF (Directory Facilitator) and RMA agents. AMS agent provides white page services to agents on JADE platform while DF agent provides yellow page services. RMA agent is a JADE tool agent that provides graphical user interface (GUI) to the agent platform. JADE provides several useful tool agents such as Sniffer, Introspector for debugging the agents. A '*BootABMSAgent*' is started that reads the configuration of the manufacturing system from an input text file and starts several containers and agents in them. All machine agents are started in '*Machines*' container; all AGV agents are started in '*AGVs*' container and so on. After launching all the agents '*BootABMSAgent*' is terminated from the agent platform. Screenshot of the JADE RMA agent GUI at simulation time of 10 time units is shown in Fig. 2 for illustration of the agent platform. Left side pane in the figure shows the loaded containers and agents while right side pane shows the states of the selected agents.



```
AgentPlatforms
  "KOMMA:1099/JADE"
    Parts
      PartGenerator@KOMMA:1
      Part_0@KOMMA:1099/JAD
      monitor-Parts@KOMMA:1C
      sniffer0-on-Parts@KOMMA
    Machines
      DepartureAgent@KOMMA:
      Machine_1@KOMMA:1099
      Machine_2@KOMMA:1099
      Machine_3@KOMMA:1099
      Machine_4@KOMMA:1099
      QueueAgent@KOMMA:10
      monitor-Machines@KOMM
```

name	ad...	state	owner
Machine_2@KOMMA:1099/J...		active	NONE
DepartureAgent@KOMMA:10...		active	NONE
AGV_2@KOMMA:1099/JADE		active	NONE
Machine_3@KOMMA:1099/J...		active	NONE
Segment_1_6@KOMMA:109...		active	NONE
PartGenerator@KOMMA:109...		active	NONE
df@KOMMA:1099/JADE	htt...	active	NONE
Part_0@KOMMA:1099/JADE		active	NONE
Node_1@KOMMA:1099/JADE		active	NONE
GlobalTimer@KOMMA:1099/...		active	NONE
Segment_3_5@KOMMA:109...		active	NONE
AGV_1@KOMMA:1099/JADE		active	NONE
Machine_4@KOMMA:1099/J...		active	NONE
QueueAgent@KOMMA:1099/...		active	NONE
Node_3@KOMMA:1099/JADE		active	NONE

Figure 2: Screenshot of the JADE RMA agent GUI at simulation time of 10.

During the initialization of different manufacturing agents, they register different ontologies with their content managers. Manufacturing agents register AGV system / shop floor ontology for communication among manufacturing agents. '*GlobalTimer*' agent is the Timer-Thread which is started in the simulator container. With the help of their '*TimerBehaviour*', most of the manufacturing agents register with the '*GlobalTimer*'. GVT is set to advance its time in steps of 4 time units by the '*GlobalTimer*'. A time unit can be a minute or a fraction of minute. The step size is an arbitrarily selected value, which is slightly smaller than inter-arrival time and operation times at different machines. Suppose the first part is scheduled to arrive at 5 time units, which is the first event in the system and it will occur in '*PartGenerator*' agent. At the start of the simulation, i.e. at time 0, no events are available that are to be executed within the upper limit of the first time step. Therefore, '*GlobalTimer*' increments the GVT to 4 and sends a message to all the registered agents. On receiving the time increment, all agents check their local event lists in search of any events that must be executed within the upper limit of the current time step (i.e. before time 8) and add those events to the global event list. Each agent confirms '*GlobalTimer*' that it has posted its events to the global event list. The last agent that posts its events to the global event list, sorts the events in ascending order, comparing the time stamps of the events. After the events are sorted, the agents are allowed to execute the events in the same order as they appear in the global event list.

As only 'PART-ARRIVAL' event was added to the global event list by the '*PartGenerator*' agent, it starts executing the event and updates the current GVT to 5. In the execution of event logic, '*PartGenerator*' agent decides the part type that will arrive and sends a request to AMS to start a new part agent with the specific information. Let the part arrived be of part type-I requiring five stages to obtain finished part. The new part is named as 'Part_0' and the name is unique for the part. The number '0' (zero) in the part name represents the part number. Part type-I requires the processing stages of Machine_1 (4) – Machine_2 (20) – Machine_3 (8) – Machine_4 (15) – Machine_2 (9) to complete the operations. The values in parentheses are the operation times required for processing of the part type at the corresponding machines.

Event logic of an event is executed by the agent in between the 'start' and 'end' of the event. To complete an event, communication among the agents may also be required. During execution of an event, simulation time remains constant. On completion of the event, it will be removed from the global event list so that the next event in the list can be executed by the corresponding agent. While executing PART-ARRIVAL event, '*PartGenerator*' agent sends a request to AMS agent to launch a new part agent. If the content and all of the FIPA-ACL (Agent Communication Language) message slots are properly filled then AMS starts a new part agent in 'Parts' container and informs the same to the '*PartGenerator*' agent. At this stage, '*PartGenerator*' agent sends a Query-If message to the newly launched part agent to know whether the part agent has executed its basic activities in the system. '*PartGenerator*' agent waits till it receives reply from Part_0 agent. Filtered and formatted traced output of the simulation with the help of Java logging mechanism for a sample time period is shown in Table I. For ease of understanding, trace statements of different agents at different steps are separated in rows of the table.

During the above simulation, only one part is allowed to enter the shop floor for ease of understanding the conversations. A sample ACL message is shown in Table II, which is a subscription message that was sent by Part_0 to Machine_1. In response to this message, the machine informs the part whenever the status of the part is changed at the machine. The ACL message representation provides the understanding of the ACL message slots. The content of the message is represented in FIPA-SL (Semantic Language) format. In the following section, two of the important hybrid communication protocols, used during the simulation, are briefly discussed.

Table I: Filtered and formatted traced output of the simulator for a sample time period.

GVTime = 4
PART-ARRIVAL-EVENT
'*PartGenerator*' added event at time 4 5_PartGenerator_PART-ARRIVAL-EVENT_0 (Note: event is formatted as '*eventTimeStamp_agentLocalName_eventName_eventNumber*')
Total number of registered time listeners is 38.

‘PartGenerator’ leaving state Wait-For-Next-Arrival and entered in state Request-AMS. ‘*PartGenerator’* leaving state Request-AMS and entered in state Query-Part
Part_0 is part type-1 and it has 5 stages viz. Machine1- Machine2- Machine3- Machine4- Machine2 Part_0 arrived at time 5 Part_0 entered in state Add-To-Queue
‘QueueAgent’ entered in ‘*AddRemoveResponder’* *‘QueueAgent’* received a request from Part_0 to ‘*AddToQueue’*
Part_0 received inform message from ‘*QueueAgent’* Position of the part in system input queue is 0 Part_0 leaving state Add-To-Queue Part_0 is in stage 0 and waiting for an AGV Part_0 is entering behaviour ‘*CNPWithMachines’* (Note: It is a contract net protocol with machines)
Machine_1 entered in behaviour ‘*PartCNResponder’*
Part_0 entered in behaviour ‘*CNPWithAGVs’* (Note: It is a contract net protocol with AGVs) Part_0 sent Accept message to AGV_1
AGV_1 added event at 5 5_AGV_1_START-OF-NEW-TASK_1
Machine_1 entered in behaviour Handle-Accept-Proposal
Part_0 leaving behaviour ‘*CNPWithAGVs’* Part_0 leaving behaviour ‘*NegotiateWithAGVs’* Part_0 entered in behaviour ‘*ReplyToPartGenerator’*
‘PartGenerator’ leaving state ‘Query-Part’ Maximum number of Parts have arrived in ‘*PartGenerator’* Removing PART-ARRIVAL-EVENT from global events list at by ‘*PartGenerator’*
START-OF-NEW-TASK Event
Start of START-OF-NEW-TASK event in AGV_1 Status of AGV has been changed from EMPTY-IDLE to EMPTY-WAIT AGV_1 entered in behaviour ‘*CanIGo’* (Note: It is a request behaviour to get permission to move ahead)
Segment_1_6 entered in behaviour ‘*RespondToNode’*
Segment_1_6 leaving state UNOCCUPIED and entered in state OCCUPIED
AGV_1 entered in behaviour ‘*LeaveSidingOrParking’*
Node_6 entered in behaviour ‘*RespondToLeaveParking’* and removing the AGV_1 identity from Parking_6.
AGV_1 leaving state EMPTY-WAIT and entered in state EMPTY-MOVE AGV_1 entered on segment Segment_1_6
End of 5_AGV_1_START-OF-NEW-TASK_1 event in AGV_1 and removing it from global event list at time 5
GVTime = 8
ENTER-A-PARKING Event
AGV_1 added event at 8 9_AGV_1_ENTER-A-PARKING_2
Start of ENTER-A-PARKING event in AGV_1
AGV_1 entered in behaviour ‘*LeaveSegment’*
Segment_1_6 entered in behaviour ‘*RespondToLeavingAGV’*
Segment_1_6 leaving state OCCUPIED and entered in state UNOCCUPIED
current step of path of AGV_1 is increased to 1 AGV_1 entered in parking_1 at Node 1 and node significance is PICKUP_NODE Status of AGV has been changed from EMPTY-MOVE to PICKUP AGV_1 entered in behaviour ‘*RequestToGetReady’*
Part_0 entered in behaviour ‘*Respond2AGVRequest’* and came to know the arrival of AGV_1 at resource ‘*QueueAgent’*
Part_0 leaving state Wait-For-Negotiation and entering Remove-From-Queue
‘QueueAgent’ entered in ‘*AddRemoveResponder’* Part_0 removed from system queue from position 0
Part_0 leaving state Remove-From-Queue and entered in state Load-On-AGV
AGV_1 entered in behaviour LOADING-BEHAVIOUR End of ENTER-A-PARKING event in AGV_1 and removing it from global event list at time 9

PICKUP-END Event
AGV_1 added event at 9 10_AGV_1_PICKUP-END_3
Start of PICKUP-END event in AGV_1
AGV_1 entered in behaviour ‘*CanIGo’*
Part_0 loaded on AGV_1 and Part_0 is in state ‘*TravelOnAGV’*
Segment_1_2 leaving state UNOCCUPIED and entered in state OCCUPIED
Status of AGV has been changed from PICKEDUP-WAIT to PICKEDUP-MOVE
AGV_1 entered in behaviour ‘*LeaveSidingOrParking’*
AGV_1 leaving behaviour ‘*CanIGo’*
Node_1 entered in behaviour ‘*RespondToLeaveParking’*
AGV_1 leaving state ‘*LeaveSidingOrParking’*
AGV_1 entered on segment Segment_1_2
End of 10_AGV_1_PICKUP-END_3 event in AGV_1 and removing it from global event list at time 10
GVTime = 12
....

Table II: SUBSCRIBE ACL message from Part_0 to Machine_1.

(SUBSCRIBE
:sender (agent-identifier :name Part_0@KOMMA:1099/JADE :addresses (sequence http://KOMMA:7778/acc) :X-JADE-agent-class-name *abms.parts.PartAgent*)
:receiver (set (agent-identifier :name Machine_1@KOMMA:1099/JADE :addresses (sequence http://KOMMA:7778/acc) :X-JADE-agent-class-name *abms.machines.MachineAgent*))
:content " ((iota ?x (PositionAtMachine (agent-identifier :name Machine_1@KOMMA:1099/JADE :addresses (sequence http://KOMMA:7778/acc)) ?x (agent-identifier :name Part_0@KOMMA:1099/JADE :addresses (sequence http://KOMMA:7778/acc)))))"
:reply-with R1177221087250_0
:language fipa-sl
:ontology agvs
:protocol fipa-subscribe
:conversation-id C11297322_1177221087250)

5. COMMUNICATION PROTOCOLS

Most of the interaction protocols, used in the communication, are the basic FIPA-Interaction-Protocols (IPs), while a few are hybrid protocols built from the combination of the basic FIPA-Interaction-Protocols. Communication protocols among Part-Machine-AGV agents and AGV-Node-Segment agents are two important hybrid protocols.

5.1 Hybrid communication protocol among Part-Machine-AGV agents

When a part reaches the first position of the system input queue or is processed on the machine of stage *i*, then it is eligible to move to a machine of stage *i*+1 of the processing cycle of the given part. At this state of the part two cases are possible. In the first case, when only a single part processing route is available, the part would like to know whether space is available or not in IB of the target machine before moving ahead. In the second case, when alternate routes are available for the part, meaning thereby that more than one machine can perform the required operation, the part should select a machine among the available alternate machines on which space is available. In case there is more than one such machine, the final selection will be governed by a bidding process.

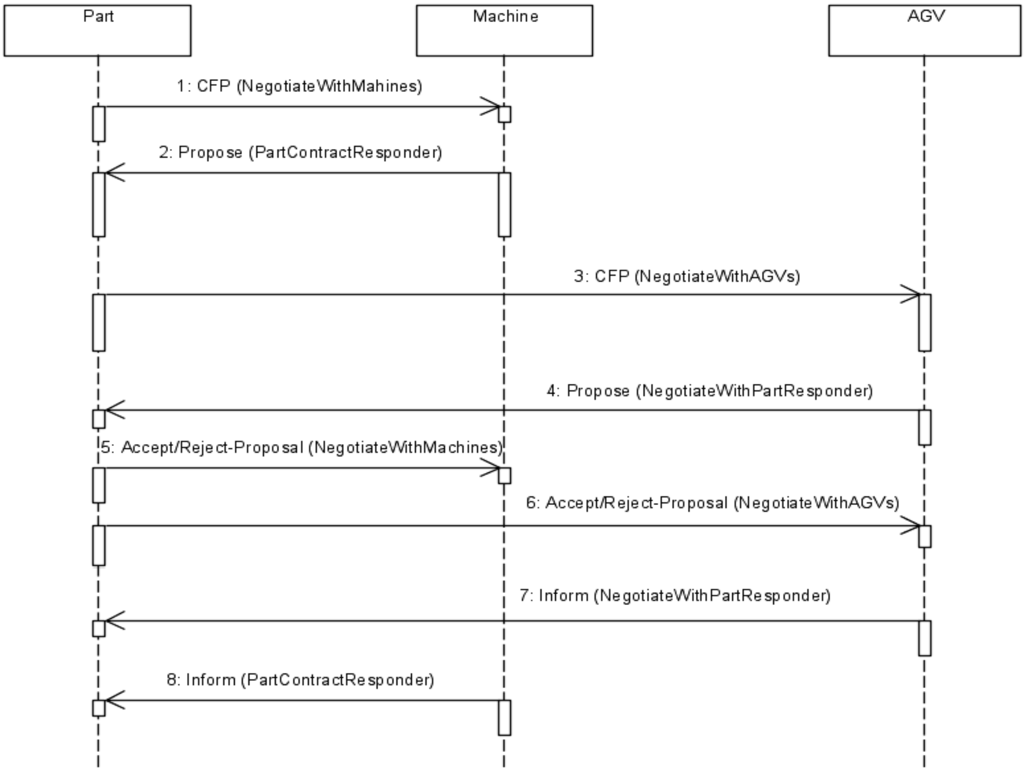

Part
Machine
AGV
1: CFP (NegotiateWithMahines)
2: Propose (PartContractResponder)
3: CFP (NegotiateWithAGVs)
4: Propose (NegotiateWithPartResponder)
5: Accept/Reject-Proposal (NegotiateWithMachines)
6: Accept/Reject-Proposal (NegotiateWithAGVs)
7: Inform (NegotiateWithPartResponder)
8: Inform (PartContractResponder)

Figure 3: Conversation among Part-Machine-AGV represented in UML sequence diagram.

In the considered model, processing routes of each part type are different but one part has only one processing route. The sequence of messages in the hybrid communication protocol among Part-Machine-AGV is represented in UML-Sequence Diagram in Fig. 3, where the name in parenthesis on the arrows is the behaviour name in the agent that is responsible for the communicative-act. It is to be noted that the hybrid communication protocol is generalized and is also applicable to the case where parts have alternate routes. The hybrid protocol is developed as a combination of two FIPA-Contract-Net protocols (CNP), one between Part-Machine and the other between Part-AGV. In a FIPA-CNP, an initiating agent sends a call for proposal (CFP) to agents that can provide the required service. The participating agents may submit proposal with 'Propose' Communicative-act or 'Refuse' to submit the proposal. Generally, the initiator evaluates the bids and accepts the proposal of one of the participants. The agent which gets the 'Accept-Proposal' from the initiating agent performs the proposed task and informs its result back to the initiator. In the proposed hybrid protocol, part initiates a CFP with one or more machines based on their capability of performing the next stage operation. In response, part receives proposals or refusals for performing the required operation. For each received proposal, the part initiates a CNP by calling a CFP to potential AGVs for transporting the part from its current position to the proposed machine. Based on part transporting time (which is directly proportional to the distance of the proposed machine from the current position of the part) and availability of AGVs, a machine-AGV combination is selected. Part sends ‘Accept-Proposal’ to the selected machine and AGV. All other machines and AGVs receive ‘Reject-Proposal’.

5.2 Hybrid communication protocol among AGV-Node-Segment agents

Let an AGV be in siding buffer of a segment or in parking at a node and it wants to travel along one of the segments that is connected to the node. The AGV asks the node at which it is currently standing whether it is allowed to move to a particular node, which is directly connected to the current node. The node forwards the request to the segment that connects the two nodes. Based on the intended direction of the AGV and the current status of the segment, the segment may accept or reject the forwarded request of the node. The same information will be communicated to the AGV through the node. The sequence of messages in the hybrid communication protocol among AGV-Node-Segment is represented in UML-Sequence Diagram in Fig. 4. This protocol is built from two FIPA-Request protocols, one between the AGV and Node and the other between the Node and Segment. The proposed hybrid protocol is also suitable for a more generalized case of multi-lane paths where more than one parallel segment connects the two nodes.

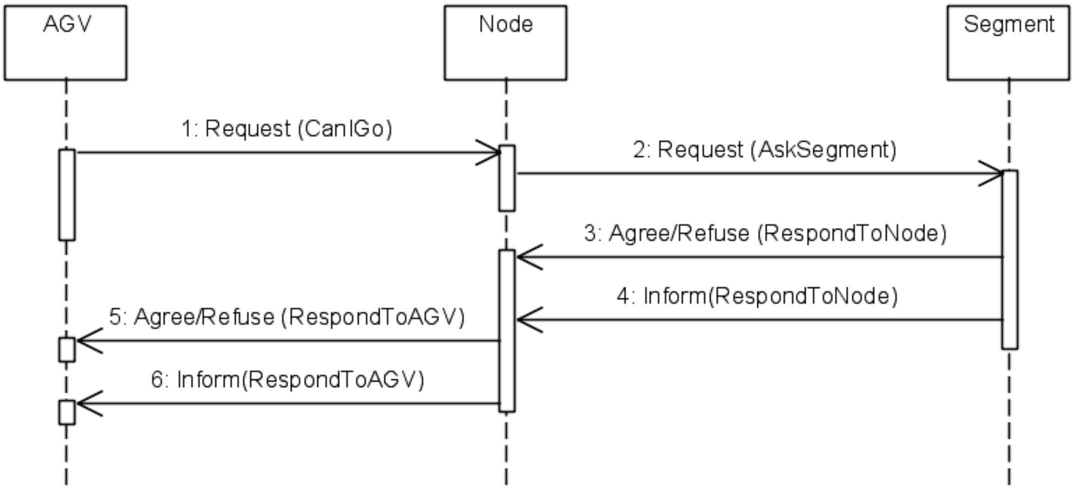


Figure 4: Conversation among AGV-Node-Segment represented in UML sequence diagram

Here, the node is analogous to the traffic signal at a junction. This is due to the reason that for some of the segments that are connected to the node, the node may respond with green signal for the proposed movement and for other segments the node may respond with red signal to oppose the proposed movement of an AGV. Similarly, a segment is analogous to a railway track between two stations. This is because trains do not cross each other on the same track and also do not overtake each other. When a train is moving in one direction, no other train is allowed to move in the opposite direction till it reaches the destination station. At the destination station, the train is put on a parallel track which spans for a short length to the main track, similar to the siding on a segment at the node.

6. CONCLUSIONS

Agent-based modelling and simulation of manufacturing system provides higher flexibility for real-time decision making due to the autonomous nature of agents in the system. The procedure developed for synchronization of agents is useful to build more generalized agent-based simulation models in a manufacturing environment. The hybrid communication protocol among the Part-Machine-AGV agents is useful to manage load on machines and AGVs for parts having multiple routings. The hybrid protocol among AGV-Node-Segment agents is helpful in coordinating distributed AGVs along the segments and is also valuable in modelling complex layouts having multi-lane paths between two successive nodes. These two protocols are built from the basic FIPA-Interaction Protocols. A simulator has been developed based on agent-based approach and its working is explained with a sample study. More research is required in standardizing the building blocks of the agent-based simulation of manufacturing system for its wider use in industries.

REFERENCES

[1] Law, A. M.; Kelton, D. W. (1991). *Simulation Modelling and Analysis*, McGraw-Hill Inc., Singapore

[2] Wooldridge, M.; Jennings, N. R. (1995). Intelligent agents: theory and practice, *The Knowledge Engineering Review*, Vol. 10, No. 2, 115-152

[3] Weiss, G. (Ed.) (1999). *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Massachusetts, USA

[4] Luck, M.; McBurney, P.; Preist, C. (2003). Agent technology: enabling next generation computing - a roadmap for agent based computing, *AgentLink II*, from *http://www.agentlink.org/admin/docs/2003/2003-48.pdf*, accessed on 01-05-2007

[5] FIPA (2006). *Foundation for Intelligent Physical Agents*, from *http://www.fipa.org*, accessed on 01-05-2007

[6] JADE (2006). *Java Agent DEvelopment Framework*, from *http://jade.tilab.com*, accessed on 01-05-2007

[7] Deen, S. M. (Ed.) (2003). *Agent Based Manufacturing – Advances in the Holonic Approach*, Springer- Verlag, Heidelberg, Germany

[8] Wooldridge, M. (2001). *An Introduction to Multiagent Systems*, John Wiley & Sons Ltd., London, England

[9] Shen, W.; Norrie, D. H. (1999). Agent based systems for intelligent manufacturing: a state-of-the-art survey, *Knowledge and Information Systems, an International Journal*, Vol. 1, No. 2, 129-156

[10] Wallace, A. (2001). Application of AI to AGV control – agent control of AGVs, *International Journal of Production Research*, Vol. 39, No. 4, 709-726

[11] Vrba, P. (2003). MAST: manufacturing agent simulation tool, *Proceedings of Emerging Technologies and Factory Automation (ETFA '03)*, IEEE Conference, 282-287

[12] Paolucci, M.; Sacile, R. (2005). *Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance*, CRC Press, Florida, USA

[13] Komma, V. R.; Jain, P. K.; Mehta, N. K. (2006). Agent-based simulation model of automated guided vehicle system, *Proceedings of First International and 22nd AIMTDR Conference*, 21st-23rd December 2006, Indian Institute of Technology Roorkee, Roorkee, India, 303-308