Simulating the generic job shop as a multi-agent system

R. Zhou

Department of Mechanical Engineering,
National University of Singapore,
9 Engineering Drive 1,
Singapore 117576, Singapore
E-mail: Zhou_rong@nus.edu.sg

H.P. Lee*

Department of Mechanical Engineering,
National University of Singapore,
9 Engineering Drive 1,
Singapore 117576, Singapore

Institute of High Performance Computing,
1 Science Park Road,
#01-01 the Capricorn,
Singapore Science Park II,
Singapore 117528, Singapore
Fax: +65-6873-2934
E-mail: hplee@ihpc.a-star.edu.sg
*Corresponding author

A.Y.C. Nee

Department of Mechanical Engineering,
National University of Singapore,
9 Engineering Drive 1,
Singapore 117576, Singapore
E-mail: mpeneeyc@nus.edu.sg

Abstract: A model combining Discrete Event System (DES) and Multi-Agent System (MAS) is proposed to simulate a real-time job shop so that it can work as a test bed to systematically study the performance of control rules and algorithms in dynamic job shop scheduling. A definition of a generic job shop is given considering dynamic events, shop floor layout and the Material Handling System (MHS). It is first simulated as a DES and then implemented as an MAS so that data recording and analysis can be naturally distributed to the most related entities and events can be executed simultaneously at different locations. The state changes of agents and the communication among them are illustrated with state charts and sequential diagrams in Unified Modelling Language (UML). The proposed model is validated with a case study through statistical analysis and comparison with reported works.

Keywords: job shop; discrete event system; DES; multi-agent system; MAS; UML state chart; sequential diagram.

Reference to this paper should be made as follows: Zhou, R., Lee, H.P. and Nee, A.Y.C. (2008) 'Simulating the generic job shop as a multi-agent system', *Int. J. Intelligent Systems Technologies and Applications*, Vol. 4, Nos. 1/2, pp.5–33.

Biographical notes: R. Zhou is a PhD student in the Department of Mechanical Engineering, National University of Singapore. She received a BE in Mechanical Engineering in 1994 from the South China University of Technology. Her research interests are in the areas of production scheduling, artificial intelligence, multiagent systems, and simulation.

H.P. Lee is an Associate Professor in the Department of Mechanical Engineering, National University of Singapore as well as the Deputy Executive Director for Research, Institute of High Performance Computing. He received a BA and MA from Cambridge, ME from National University of Singapore as well as MS and PhD from Stanford University. His research interests are in biomechanics and nanomechanics.

A.Y.C. Nee is a Professor of Manufacturing Engineering, Department of Mechanical Engineering, National University of Singapore since 1989. He received a PhD and DE from Manchester and UMIST. His research interest is in computer applications to tool, die, fixture design and planning, distributed manufacturing systems, virtual and augmented reality applications in manufacturing. He is a Fellow of CIRP and an elected Fellow of the Society of Manufacturing Engineers (USA), both in 1990. He had held appointments as Head of the Department of Mechanical Engineering, Dean of Faculty of Engineering and currently, he is the Director of Research of NUS.

1 Introduction

The Job Shop Scheduling Problem (JSSP) has been studied extensively since the advent of computers in the 1950s and 1960s. Most instances of the JSSP have been identified as a NP-hard problem (Garey and Johnson, 1979), which implies that there is no polynomial time algorithm to solve them. As a result, many approximate methods have been explored to find near-optimal solutions with reasonable computation efforts, for example, genetic algorithms, Tabu search, simulated annealing, ant colony optimisation algorithm etc. The performance of those approaches on static problems has been examined extensively and many reported results on benchmark problems can be found in the OR-Library (www.ms.ic.ac.uk/info.html).

However, the performance of those approaches on dynamic/stochastic JSSPs has not been studied systematically and thoroughly. The main reason is that the formulations for dynamic scheduling problems are not as well defined as those for the static cases. Especially, both the problem composition and the dynamic/stochastic description of its components are generally over-simplified because the complexity rises exponentially as the number of components and dynamic events increases. Subsequently, many reported results of these algorithms can be applied only to certain particular applications and offer limited suggestions on their general performance in dynamic systems.

The objective of this paper is to build a common test bed to facilitate systematic examination of the performance of control policies and algorithms in a dynamic job shop environment. The test bed simulating a generic job shop should be able to provide the following functions:

1 it should have a realistic configuration of the job shop

2 it can generate dynamic or stochastic events like incoming jobs and machine breakdowns

3 it should have necessary scheduling algorithms or dispatching rules to guide processing and control rules to react to dynamic events

4 it can track job movement and the status of machine, workcentre and shop floor

5 it can provide statistical analysis for performance measures

6 it should have the ability to be extended structurally and functionally.

The test bed is implemented as a Multi-Agent System (MAS) because it is difficult to be represented mathematically with distributed entities and events which may even occur simultaneously. Besides, it is expected that the performance of the system can be improved through coordination and cooperation among communicating agents. Thus the proposed test bed as an MAS should not only be equipped with the above mentioned functions but also provide concurrent execution of distributed events as well as cooperation or coordination among entities, if necessary.

The above MAS can readily simulate a job shop and provide scenarios to examine the performance of tested algorithms or dispatching rules as most of manufacturing MASs do. However, it can only examine the short-term performance for the given problem. In order to study long-term performance of scheduling methods, this paper proposes to encapsulate the above MAS into the time frame provided by a Discrete Event System (DES), where simulated time advances along with the occurrence of dynamic events. The MAS reacts each time a dynamic event occurs. The detailed mechanism is explained in Section 5.3.

The structure of this paper is as follows: in Section 2 the related works on system modelling/test beds for reactive/dynamic scheduling are presented, in Section 3, the definition of a generic job shop is given, in Section 4, the generic job shop is modelled as a DES and in Section 5 a prototype of the job shop is implemented as an MAS. Section 6 is specially devoted to describe the communication of agents in the MAS and a case study is provided in Section 7. Finally, conclusions and ideas for future work are presented in Section 8.

2 Literature review

In order to build an up-to-date test bed for examining scheduling methods in dynamic/stochastic environments, main approaches as well as their test beds should be reviewed.

Traditionally, there are two types of strategies to react to dynamic/stochastic events, dynamic scheduling and predictive-reactive scheduling (Vieira et al., 2003). Dynamic scheduling is also called online scheduling or reactive scheduling (Li et al., 1993), where no schedules, except some dispatching rules, are used to guide the production.

The performance of dispatching rules can be tested through statistical analysis using discrete event simulation by modelling the production systems as a DES. Many works have been reported using this approach as reviewed by Ramasesh (1990). The test bed for this approach supports algorithmic scheduling and control rules. It is not necessary to be agent-based as there is no negotiation involved. For example, Law and Kelton (2000) applied simulation to find the best configuration of facilities using dispatching rules.

Predictive-reactive scheduling generates an initial schedule and refines it when dynamic events occur. There are two basic elements in this approach: scheduling algorithms and control policies. Scheduling algorithms are used to generate initial schedules and repair obsolete schedules while control policies are used to adjust the frequency of repairing a schedule. Control policies can be categorised as periodic, event-driven and hybrid (Vieira et al., 2003). The performance of proposed scheduling algorithms and control policies is generally tested on simulated manufacturing systems. In Sabuncuoglu and Bayiz (2000), the test bed is based on a simulation model coded in the C language with ten levels of frequency of scheduling and four types of problem instances. The down time distribution follows a Gamma distribution with a shape parameter of 1.4 and a mean of 40 min, the number of operations for one job is drawn from a discrete uniform distribution from 5 to 15 and processing times are generated from a discrete uniform distribution from 20 to 80. In Sabuncuoglu and Kizilisik (2003) six machines and three Automatic Guided Vehicles (AGVs) comprise the flexible manufacturing system. Job inter-arrival time is exponentially distributed with mean $\mu = 55$. Each job has either five or six operations with equal probability; operation times are drawn from a $2 - Erlang$ distribution with mean $\mu = 55$. The review shows that a test bed should have capability for statistical analysis.

The complex nature of the scheduling/rescheduling problem dictates that traditional simulation experiments can only be performed on small systems. Besides, the reasons to choose particular problem settings, such as the number of job types, machines or Material Handling Devices (MHDs), are generally not made clear in the literature. A good scheduling test bed should be able to facilitate the systematic selection of parameters and configurations as well as to support distributed computation, which can be realised through agent technology.

The intelligent agent has been proposed as a software design paradigm, which is different from the sequential, the structural and the object-oriented approaches by its autonomy in deciding when to invoke its action (Parunak, 1997). Manufacturing scheduling systems built as MAS are surveyed by Shen and Norrie (1999). The architecture of a multiagent scheduling system can be modelled to be heterarchical, hierarchical or hybrid. In a pure heterarchical MAS, the job allocation is through negotiation between job agents and machine agents and no schedule is prepared in advance. Decision making happens in distributed locations and the final sequence of operations is the emergent phenomenon of such coordination which can be based on the Contract-Net Protocol (CNP) (Smith, 1980), Market-Driven-Protocol or others. This approach has the advantages of robustness and flexibility, but its disadvantages are also apparent: firstly, the schedules may be suboptimal owing to the myopic nature of local decision making and secondly, results from experiments with the same parameters may vary due to the uncertainty in decision making of the negotiating agents.

In a purely hierarchical MAS, agents at higher levels can allocate tasks to their immediate lower level agents, which execute the tasks without any opinion. The system will produce schedules with good global performance since the agent at the higher level

can have a wider view of the system. However, this architecture lacks reactivity to dynamic events since events are first forwarded from the lowest level agents to the upper level agents and then the reaction decision is passed down from the upper level agents to the lowest level agents to be executed. The MAS may assume schedules to guide production in a similar manner performed in predictive scheduling. Cavalieri et al. (2000) experimentally benchmarked these two types of architectures and confirmed the above observation.

To cope with the disadvantages and combine the advantages of the previous two types of MAS, some hybrid architectures have been proposed, such as Holonic Manufacturing System (HMS) (Bongaerts, 1998; Bongaerts et al., 2000; Wyns, 1999), Biological Manufacturing System (BMS) (Okino, 1993) and Fractal Manufacturing System (FrMS) (Ryu and Jung, 2003; Warnecke, 1993). Basically, agents in those systems have the autonomy to promptly react to dynamic changes and simultaneously be guided by those agents with global views. Valckenaers et al. (1994) compared the above three architectures and found that the hybrid one performed well in a wider range of situations. This is reasonable since it actually embodies the benefits of a predictive-reactive scheduling approach in a multiagent environment.

It is also worth noting that dynamic/stochastic scheduling is closely related to the work of Shop Floor Control (SFC), which moves and allocates jobs to machines according to a given schedule and makes adjustment promptly if deviation from the schedule occurs. For example, the SFC rule in Bongaerts (1998) is to continue allocating jobs according to old schedule while the scheduler is working for the new one at the time of disturbance. The proposed test bed also needs to be able to support adopting proper SFC rules, especially when predictive schedules are used.

Generally, the overall performance of an MAS is assessed through experimental comparison with other approaches in particular applications. This approach raises a question of whether the studied problem has some underlying features which favour certain approaches. Systematic evaluation based on statistical analysis on different dynamical levels, problem configurations and performance measures may avoid such bias. Long term performance based on statistical analysis can be carried out when the generic job shop is simulated as a discrete-event system.

Simulation is the most common method for constructing models that include the temporal dynamics of manufacturing systems, many of which can be modelled as DES (Askin and Standridge, 1993). Combining MAS with DES enables proposed test bed not only to meet the requirements in Section 1 but also to have advantages as follows:

1 simultaneous execution of events on distributed locations

2 distribution of event generation, state keeping, event-list managing and data recording/analysing

3 possible performance improvement through agent coordination or negotiation

4 examination of long-term performance

5 scalability of the MAS to support further extension of the test bed

6 a common test bed that could use the similar structure and logic between simulation and actual control of the job shop.

3 The generic job shop

A generic job shop in this paper refers to a generalised representation of real life job shops considering not only the configuration of their floor layout and Material Handling System (MHS) but also performance measures and dynamic events which are generated by jobs, machines and MHDs. The configuration and performance measures of a generic job shop are discussed in Sections 3.1 and 3.2, respectively, while dynamic events are described in Section 4.3.

3.1 Configuration

A generic job shop can be physically made up of several workcentres, a receiving/shipping station and material transportation devices as shown in Figure 1. A *workcentre*, shown in Figure 2, processes one type of operation using several similar machines. It has a queue to buffer incoming jobs when all machines are not available and another queue for completed jobs to wait for transportation. The *receiving/shipping station* receives new jobs and ships out all completed jobs. All workcentres and the receiving/shipping station are located in the shop floor according to certain layouts. The distances between every two of them can be given in a layout matrix. The MHDs transport jobs between workcentres.

Figure 1 The components of a job shop

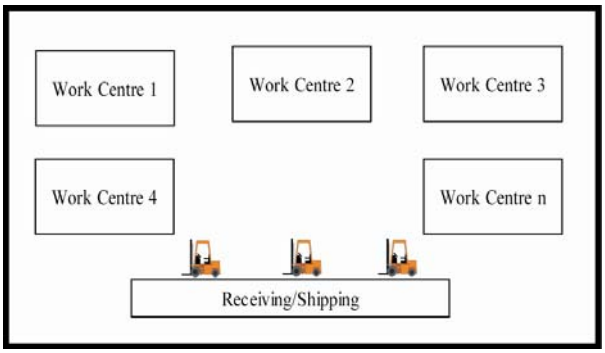
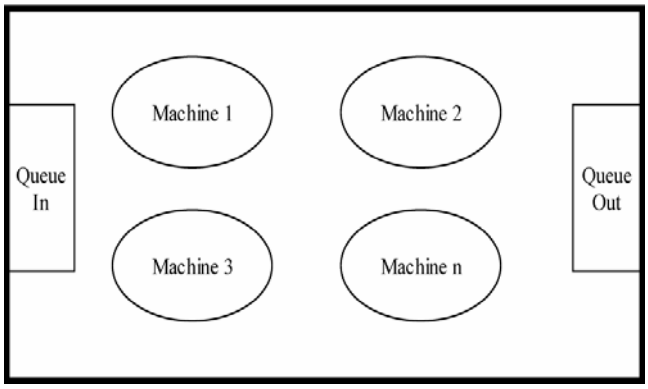


Figure 2 The components of a workcentre



3.2 Performance measures

The performance measures, which are of interest in manufacturing systems, include mean flow time, Work-In-Process (WIP), buffer size and resource utilisation. The *mean flow time* refers to the average time taken from the release of a job at the beginning of the routing to its arrival at an inventory point at the end of the routing (here it is the shipping station). WIP represents the inventory in the shop floor between the start and end points of a product routing a *buffer size* refers to the size of a workcentre queue *resource utilisation* is the fraction of time a resource is not idle for lack of jobs (Hopp and Spearman, 2000).

4 Discrete event simulation model

Three basic elements in discrete event simulation include the state of the system, event actions and event lists. The state of a generic job shop system is decided by the status of machines and jobs in it where machines are located in different workcentres and jobs are distributed either in workcentres or on travelling devices. According to Koestler (1967), architectures of manufacturing systems are inherently hierarchical. In Section 4.1, entities are organised hierarchically so that the global state can be monitored by distributed entities at different levels. In Section 4.2, the possible states for each type of entities are described. In Section 4.3, the dynamic events and their actions are presented and finally, in Section 4.4, the mechanism to maintain distributed event lists is explained.

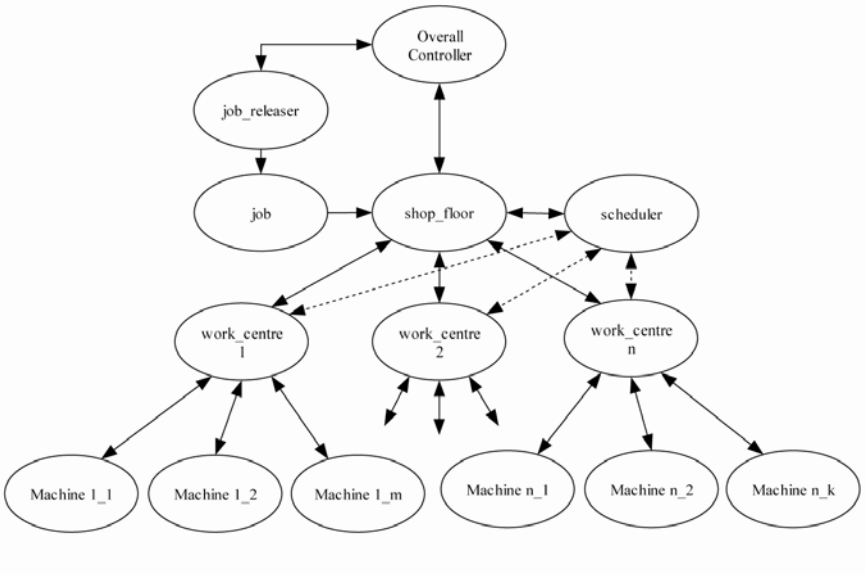
4.1 Decomposition of the global state

The hierarchical relationship in a generic job shop is illustrated in Figure 3. Entities like machines or jobs can be grouped and monitored by an upper level entity which, in turn, forms another group with its similar entities and is monitored by an even upper level supervisor. For example, a group of machines are monitored by their workcentre manager and the state of a workcentre is monitored by the shop floor monitor. A job can be monitored by either a workcentre manager or the shop floor depending on whether it stays in a workcentre or travels on a MHD. In this way, the global state of the job shop can be tracked through monitoring workcentres and travelling jobs.

There are seven types of entities in the simulated job shop: machines, workcentres, a shop floor, jobs, a scheduler, a job releaser and a controller. Basically, the higher level in the hierarchy is an entity, a wider system view would have this. A *machine*, a *job* or *a scheduler* can only monitor its own states while a *workcentre* has a wider scope by monitoring jobs, machines and buffers. Similarly, a *shop floor* can have an even wider view of monitoring workcentres, travelling jobs and MHDs. The state of an entity in a higher level does not contain the detailed state information of its supervised entities. For example, the state of the job shop does not contain the information of the buffer status in its supervised workcentres. This approach facilitates distributing data as well as their analysis to their most related locations.

The *job releaser* and the *controller* are not parts of a job shop but responsible for generating new jobs and advancing the simulation time, respectively.

Figure 3 The hierarchical relationship among all types of components in a simulated job shop



4.2 States of entities

A job can be in a state of waiting-for-process, in-process or on-travelling, assuming that only one buffer is needed in a workcentre and finished jobs can be sent to the next stage immediately. It is in *waiting-for-process* when it waits at the buffer of a workcentre for being processed, it is *in-process* when processed on a machine and it is *on-travelling* when it is travelling between workcentres.

A machine can be in a state of busy, idle or down. It is *busy* when it processes a job and *idle* when it waits for a job. The *down* state refers to the period from machine breakdown to its recovery.

A workcentre can be in any of the four states: idle, partial, full or buffered. It is *idle* when there is no job in it and all available machines are idle. It is *partial* when machines are only partially used. It is *full* when all machines are busy and there is no waiting job. Finally, it is *buffered* when all machines are busy and there are waiting jobs.

The shop floor can be in any of the three states according to the number of jobs in it: *idle* (no job on the floor and all workcentres are idle), WIP (at least one job is on the floor) and *completed* (simulation completed and analysis can be carried out).

4.3 Events and their actions

The global state of a job shop system is changed by the actions of any dynamic event concerning jobs and stochastic events in the shop floor. The dynamic events related to a job entity include its arriving at or leaving resources like machines, workcentres, the shop floor and MHDs. The stochastic events include dynamic incoming of job orders, machine breakdowns/ups and processing time variation. However, it is not necessary to model all the aforementioned events. Only five types of essential events are identified according to the previous assumptions and they can be categorised as job-related or machine-related events.

4.3.1 Job-related events

Job-related events are initiated by jobs and there are two types of them:

- *New job event*: a new job event represents a new job order which is released to the shop floor by the job releaser according to certain distribution functions. The event action for this event is illustrated in Figure 4. The event is registered to the shop floor which then increases the size of its WIP and confirms the registry. The job then heads to the next workcentre from the receiving station travelling on a MHD. The time required to travel to next workcentre is decided by the speed of its MHD and the distance between the two workcentres. When this event is fired, another event, the incoming job event is generated immediately.
- *Incoming jobs*: an incoming job event represents the arrival of a travelling job to a workcentre. When it is fired, the job enters a workcentre from the shop floor and requests service; the workcentre then allocates the job according to its state and control rules. If the job cannot be processed immediately, it will be put into the workcentre buffer. Otherwise, it will be sent to one of the machines and another event, namely a 'leaving job' event (from the machine), will be generated. The event actions and state changes on related entities are represented in Figure 5.

Figure 4 Event actions upon a new job event

4.3.2 Machine-related events

Machine-related events are initiated by a machine and may be of three types:

1 leaving jobs

2 machine breakdowns

3 machine ups.

a *Leaving job*: a leaving job event represents the completion of an operation by a machine. When this event is fired, the completed job leaves its machine and workcentre, travels to the next workcentre and then generates another incoming job event. Meanwhile, the newly freed machine is available for the next job. If it is allocated with another job, a new leaving job event will be generated, otherwise it will be idle. Finally, the workcentre reduces the size of its WIP by one. The event actions and the state changes of the related entities are presented in Figure 6.

b The locations of the previous events in a job shop are illustrated in Figure 7. The leaving job event causes a job to leave both its machine and workcentre simultaneously, assuming that the finished job can be immediately transported to the next workcentre. The relationship between job events is shown as an event diagram in Figure 8.

c *Machine breakdowns/ups*: a machine breakdown event is assumed in this work to occur only when a machine is busy processing jobs (Law and Kelton, 2000). The machine will change its state to *down* on a machine breakdown event and immediately create a machine-up event to represent the time that it will take to be repaired. Meanwhile, the interrupted job is sent to another available machine generating another incoming job event or it is sent to the buffer. Similarly, when a machine-up event occurs, the machine is ready to process operations. If a job is allocated to it, the machine will go to the state of busy and a new leaving job event will be generated. Otherwise, it remains idle. The action and state changes for both events are illustrated in Figures 9 and 10, and their relationship is given in the event diagram of Figure 11.

Figure 5 Actions and state changes in the shop floor, workcentres and machines upon incoming job event

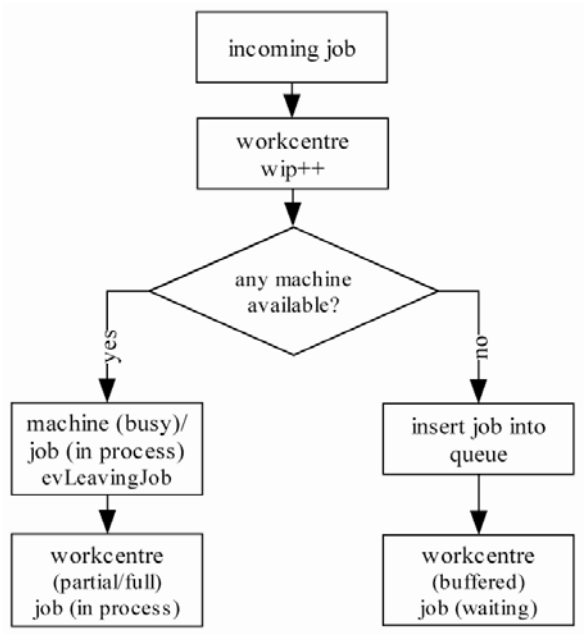


Figure 6 Event actions and state changes in the workcentre and on the machine for a leaving job event

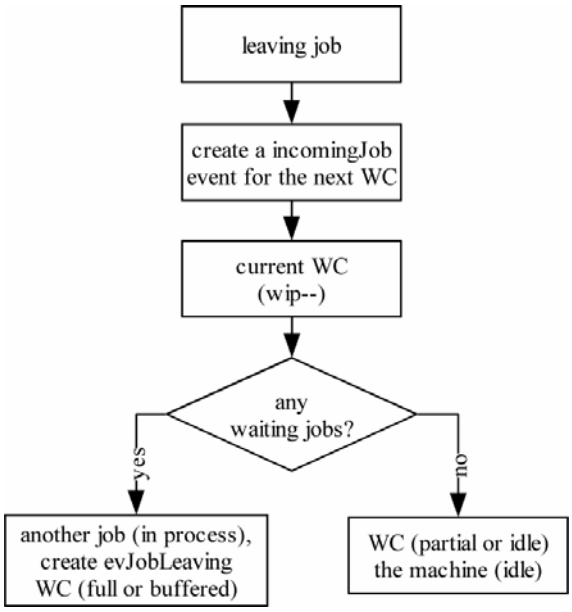


Figure 7 The dynamic events incurred by a routing job

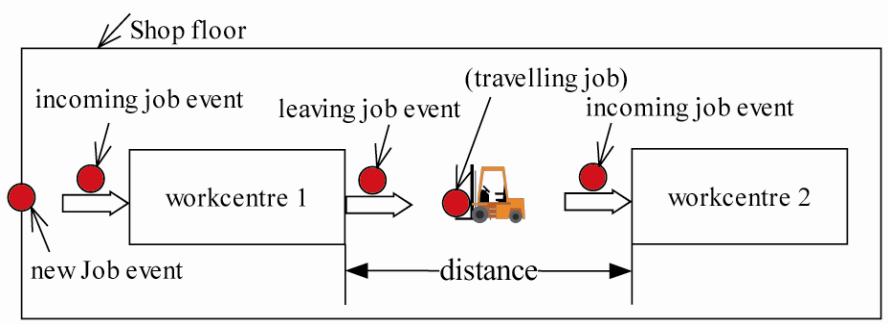


Figure 8 Event graph of job related events

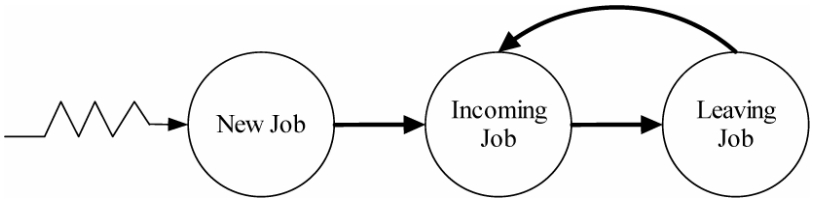


Figure 9 Actions and state changes in the workcentre and at a machine upon a machine breakdown event

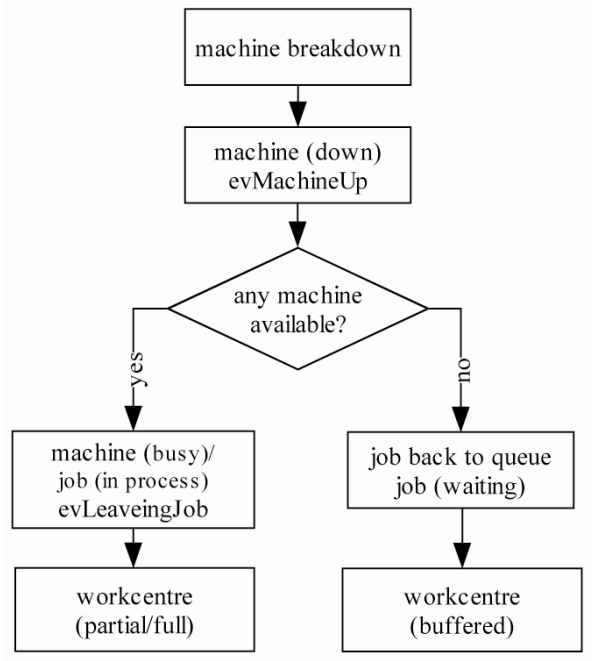


Figure 10 Actions and state changes in a workcentre and at a machine upon a machine up event

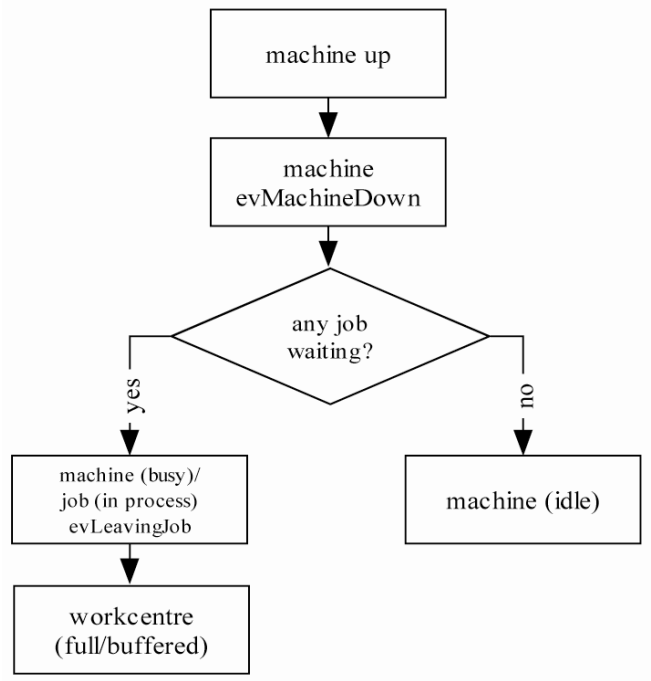
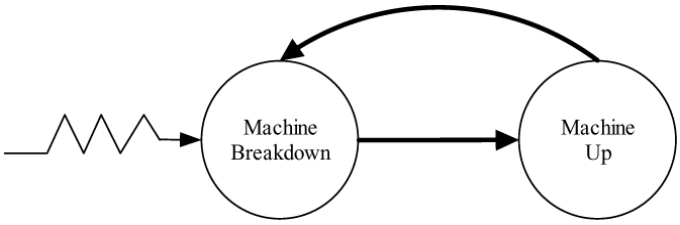


Figure 11 Event graph of machine breakdown and up



4.4 Event lists

The global state is maintained as one entity and all events are sorted in one event list according to their occurring times in conventional approaches. However, in a system where the global state is decomposed and monitored by many distributed entities, the global event list also has to be decomposed and monitored by the respective entities. This approach can reduce the size of the list and thus the sorting time. However, the correct simulation time should be maintained carefully since the event lists are distributed and the execution of one event may cause event changes at different entities. The analysis of the event list in each component is given in Section 4.4.1 and the mechanism to maintain correct simulation time is presented in Section 4.4.2.

4.4.1 Analysis of event lists

Each entity maintains an event list although only machines, jobs and the job releaser are the initiators of events. Other types of components only receive events from their supervised entities and keep only the earliest ones in their own event lists.

The event list of a machine can contain at most three possible types of events: machine breakdowns, machine ups and leaving jobs. Its size can be at most two since machine breakdown and machine up events cannot coexist. The event list of a job entity is a one-item list containing one incoming job event. Similarly, the job releaser also has a one-item list containing one new job event.

A workcentre entity keeps only the earliest events from its supervised machines; the job shop entity in turn keeps only the earliest events from all workcentres and travelling jobs. The controller is at the top of the hierarchy and it decides the earliest event time for the next simulation round.

4.4.2 Mechanism to maintain correct simulation times

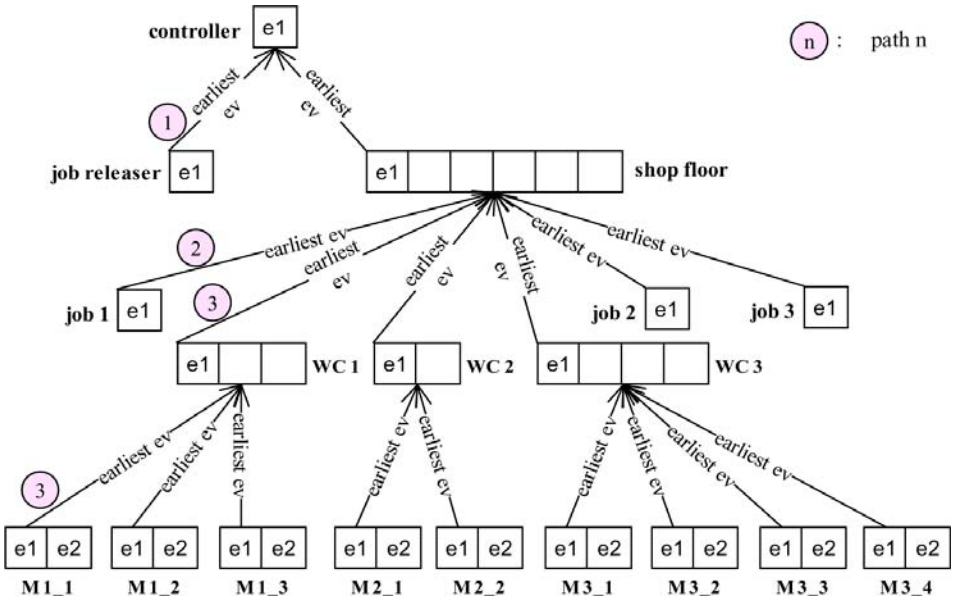
The mechanism to maintain correct simulation times is illustrated in Figure 12. Each entity forwards its earliest events up to its respective monitor and finally the earliest events reach the controller. There are three propagating paths: the first one is from the job releaser directly to the controller, the second one starts from the travelling jobs to the shop floor and then to the controller, and the third one starts from machines, goes through workcentres and the shop floor and finally ends at the controller.

An example for maintaining the event lists in a typical simulation round is given as follows. Workcentre 1 (WC1) has three machines, m1_1, m1_2 and m1_3. Each of them forwards its earliest event to WC1. WC1 compares the three events, finds out the earliest

one and keeps it in its event list. The same procedure happens concurrently at workcentres 2 and 3. The three workcentres forward their earliest events to the shop floor, which at the same time, also keeps the events of travelling jobs. So the earliest event that will occur on the shop floor can be found and further forwarded to the controller, which also receives the event of generating the next job from job releaser. The controller then finds the earliest event and announces the occurring time as the next simulation time to both the job releaser and the shop floor. The shop floor forwards the new time to all workcentres, which pass down to their machines. Each entity checks its own event list upon receiving the new time and starts to act if there are any events due. If not, it takes no action. It is obvious that there could be many concurrent events occurring at the different locations. The detailed messages for coordinating those single or concurrent events are illustrated in Section 6.

It can also be seen that the size of the job shop event list is bound to the sum of both the sizes of workcentres and the travelling jobs. In addition, the size of a workcentre event list is related to the size of its machines. An overall long event list is thus avoided and list-sorting time is reduced.

Figure 12 The hierarchy of event lists



5 Implementing the simulated generic job shop as a MAS

The implementation of agent-based simulation essentially includes two steps:

1 identifying behaviours of each individual agent

2 coordinating the communication among agents.

The behaviours of agent and the change of its status can be expressed clearly in state charts while the coordination of communication can be illustrated in the sequential diagrams of Unified Modelling Language (UML, http://www.omg.org/technology/

documents/formal/uml.htm). There are also many packages for implementing MAS such as Java Agent Development Framework (JADE, http://jade.tilab.com/), Cougaar (Cougaar Software, Inc., http://www.cougaar.org/), AnyLogic (AnyLogic™, http://www.xjtek.com/anylogic/), etc. The JADE is used in the current work.

All entities of the generic job shop are modelled as autonomous agents pursuing their own interests with unique functions. The possible stable states for the main agents have been identified in Section 4.2 and the transition between them in real time is described using UML state charts. Some transient states or actions such as data recording, list sorting and message sending are also included in the state charts for a better illustration and the stable states are shaded. It may be noted that in the given state charts event (ev), message (mg) and symbol C refers to a conditional gate. Finally, the mechanism of how the MAS fits to the time frame decided by DES is described.

5.1 Main agents

The state chart of a job agent is illustrated in Figure 13 which shows three stable states and four transient states. The stable states are:

1 *waiting for process*

2 *in processing*

3 *on travelling* and the transient states are:

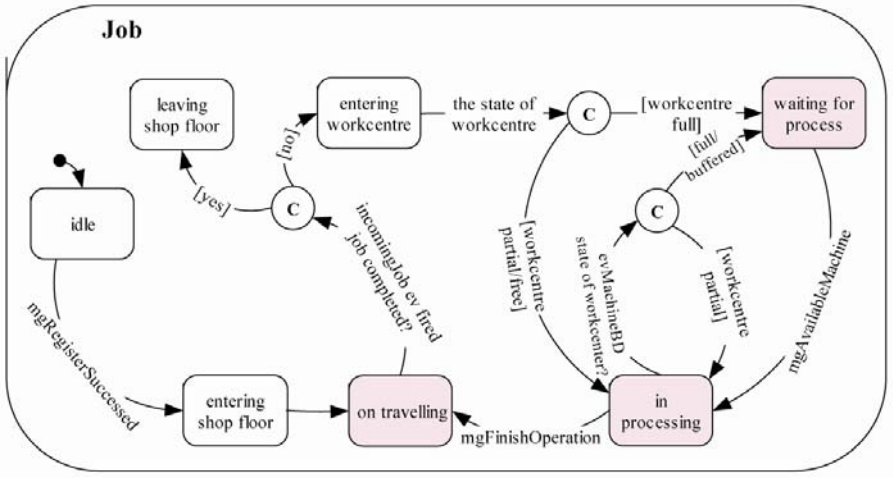
   a *idle*

   b *entering shop floor*

   c *entering workcentre*

   d *leaving shop floor.*

Figure 13 State chart of a job agent



The life cycle of a job agent involves the stable and the transient states. It starts in the *idle* state and changes to *on travelling* state after entering the shop floor. It turns to either the waiting *for process* state or the *in process* state after entering a workcentre depending on the current state of the workcentre. If it is in the *waiting for process* state and receives

an 'available machine' message from its workcentre, the job will be processed on the appointed machine and its new state will be *in process*. It remains in this state until it receives either a 'machine breakdown' or a 'finish operation' message. It will go back to the *waiting for process* or the *in process* state if the former is received. Otherwise, its operation will be finished. It then turns to the on travelling state and moves to the next workcentre or the shipping bay if all operations are completed.

The flow time, waiting time and actual processing time of a job can be tracked by its job agent, which can record the time it reaches or exits the shop floor, workcentre buffers or machines. The state charts of the machine agent, workcentre agent and shop floor agent are illustrated in Figures 14–16.

Figure 14 State chart of a machine agent

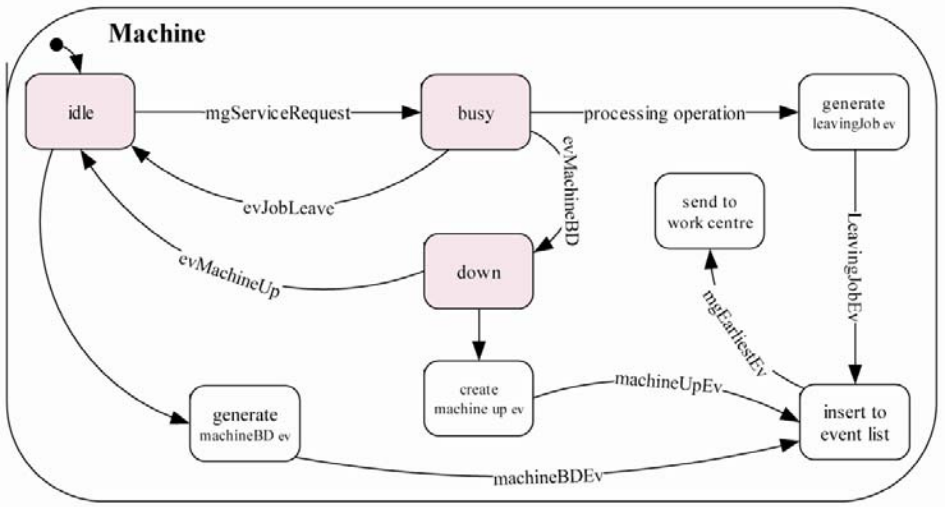


Figure 15 State chart of a workcentre agent

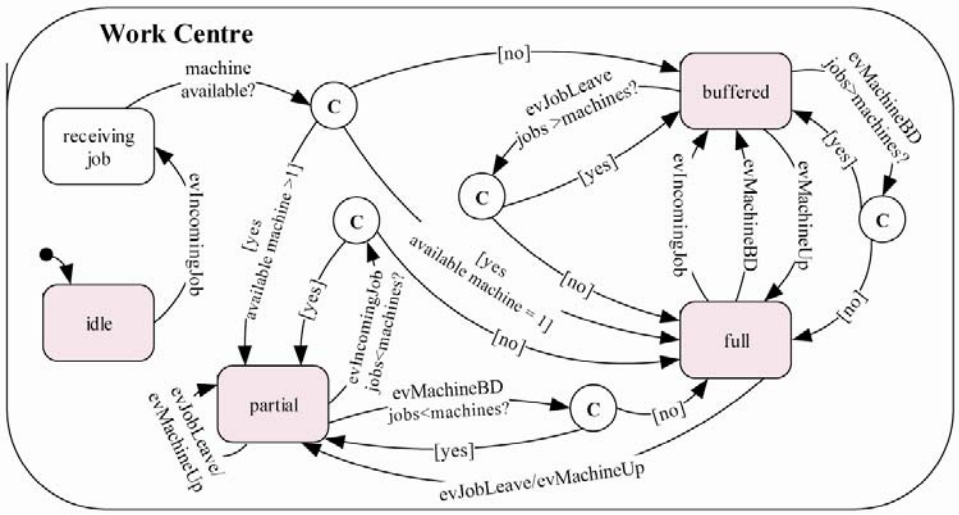
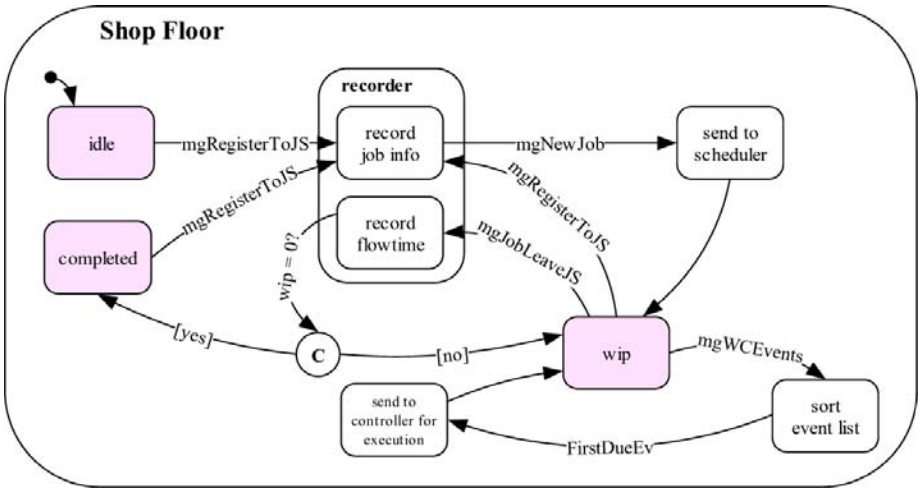


Figure 16 State chart of a job shop agent



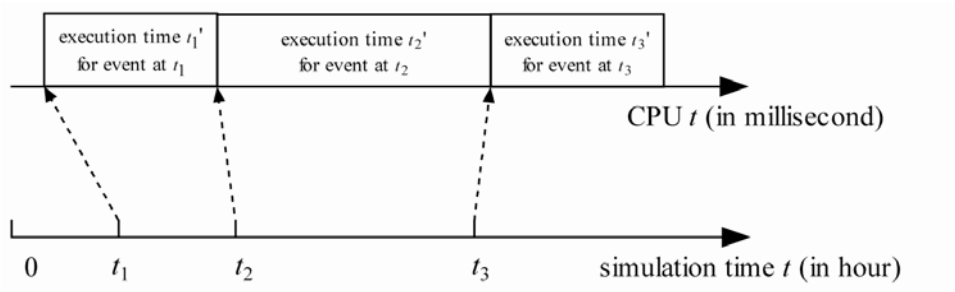
5.2 *Other agents*

The controller and the job releaser in Figure 3 are also utilised as agents. The controller works to initiate the whole system and maintain the simulation clock and the job releaser generates new jobs with particular information concerning technical sequence, processing times, starting and due time, etc.

5.3 *Fitting the MAS into the time frame of DES*

There are two types of time in the system: simulation time and execution time. The simulation time is a clock time when an event starts to be executed. It is decided by DES. The execution time refers to the period of CPU time MAS takes for event execution. Their relationship is illustrated in Figure 17.

Figure 17 The relationship between simulation time and execution time



The event occurred at time t_1 causes MAS to execute taking t_1' CPU time. Then next simulation time t_2, maybe hours after, is decided at the end of execution time t_1'. Another period of event execution then starts. The simulation proceeds in this way from t_1 to t_2 and t_3 till a predefined termination time is reached while the events are executed one after another by the agents.

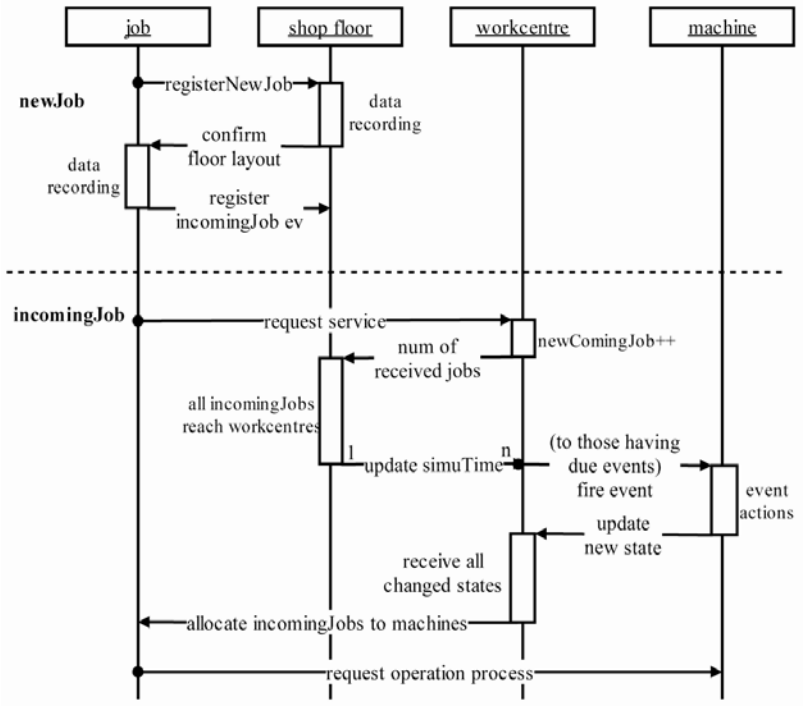
6 Communication in the MAS

All communications in an MAS are realised through message passing. The holding of an event always produces a string of messages, spreading to other agents, who may react to the messages by further sending the messages to other agents. Messages may be passed concurrently in many distributed locations and it is crucial to coordinate them so that all event lists can be updated in a consistent manner and the correct simulation time can be maintained. Message passing for a single event is analysed in Section 6.1 and that for concurrent events in a single agent is described in Section 6.2. The mechanism to coordinate all agents is given in Section 6.3.

6.1 Message passing for a single event

- *Message passing for job-related events*: the message passing for two job-related events, that is, the new job event and the incoming job event, is illustrated in Figure 18.
- *Message passing for machine-related events*: the message passing for three machine-related events that is, the leaving job, the machine breakdown and the machine up event is illustrated in Figure 19.

Figure 18 Message passing for job related events



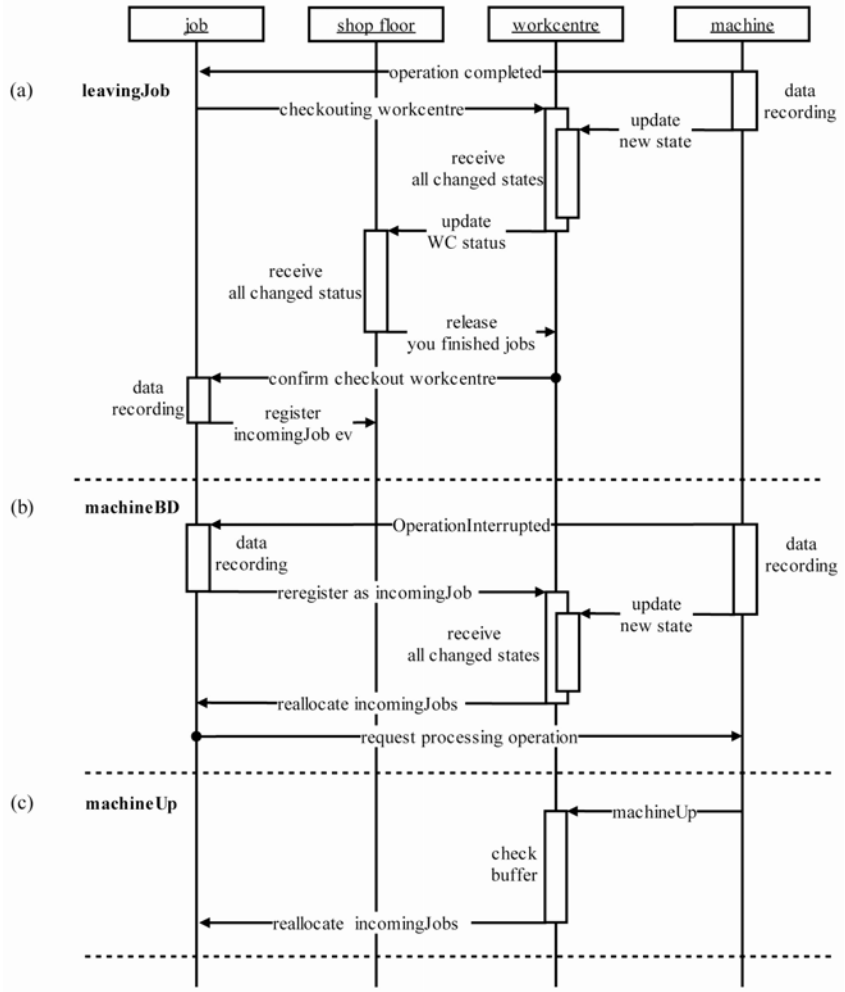
The machine agent sends a message to the job agent representing the job processed on the machine when a leaving job event is fired and notifies it on completion of its operation. The job agent then requests to 'checkout' from the workcentre while the

machine updates the workcentre about its new state. The workcentre checks whether there are waiting jobs or interrupted jobs to be reallocated accordingly. Finally, it permits the job agent to leave the workcentre. The job agent in turn registers to the shop floor agent with another incoming job event as shown in Figure19(a).

The broken down machine sends a message to its job agent announcing an interruption when a machine breakdown event is fired. The job then reregisters itself to the workcentre as a waiting job while the machine updates the workcentre about its new state. Finally, the workcentre reallocates the interrupted job according to its new state and sends the job agent a 'waiting' or a 'processing' message. The job then acts accordingly as described above in Figure 19(b).

The machine will notify the workcentre about its new state when a machine up event is fired and the workcentre will check its buffer to see whether there are jobs waiting. If there are, an allocation message will be sent to the appropriate jobs from the workcentre, otherwise no more messages will be generated. This procedure is shown in Figure 19(c).

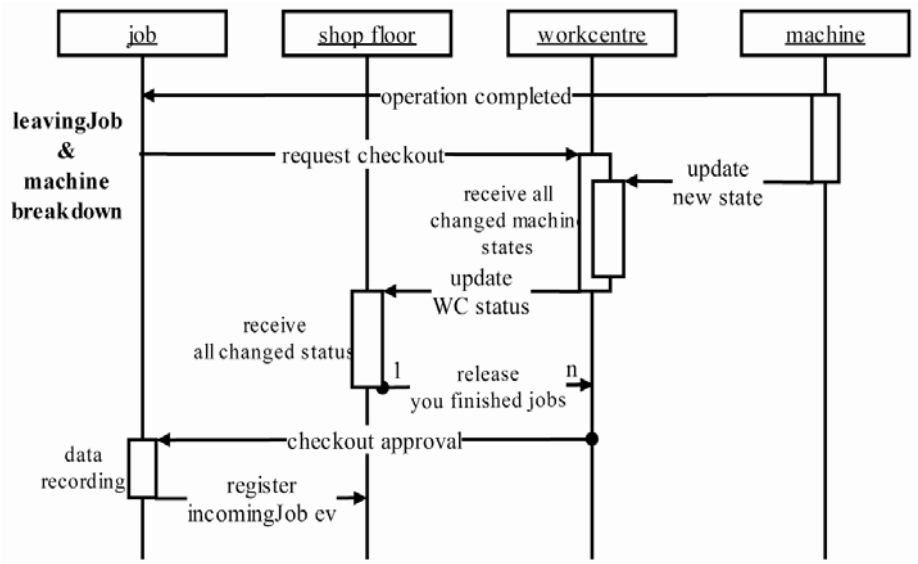
Figure 19 Message passing for machine-related events



6.2 Message passing upon concurrent events in a single agent

It is possible that there could be several events fired at the same moment within one agent. The message passing for possible concurrent events is described as follows: A job agent can only have an incoming job event in its event list and thus has no concurrent events. A machine agent can have at most two concurrent events: a leaving job and a machine breakdown event. Its state turns to be down while the finished job leaves the workcentre the machine belongs to. The messages incurred are shown in Figure 20.

Figure 20 Message passing upon concurrent events of machine breakdown and leaving job in a machine agent



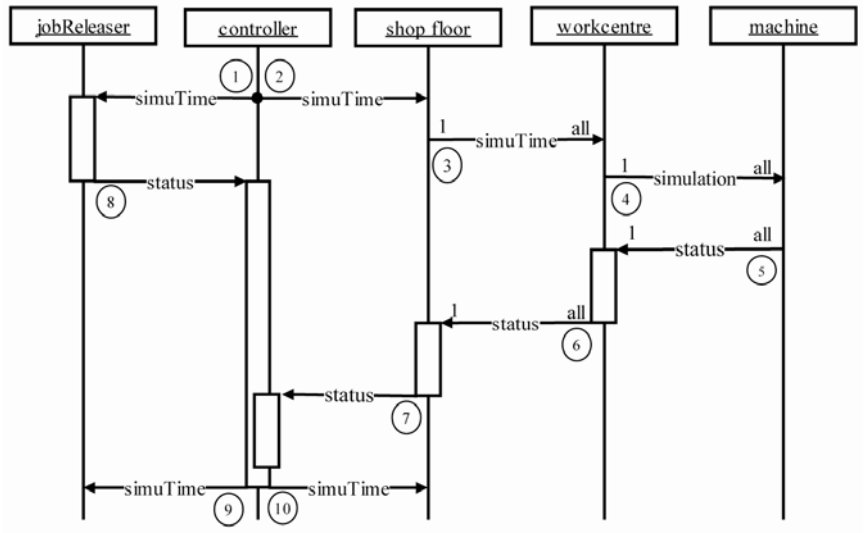
The workcentre agents, the shop floor agent and the controller agent do not initiate or execute any events by themselves, but they monitor the status of agents and coordinate messages passing in their domains.

6.3 Agent coordination

The basic information flow in a simulation loop is illustrated in Figure 21. It starts from sending all agents the current simulation time by messages 1–4. Agents from the lowest level then update their supervisors of their new status, after event actions, by messages 5–8. The messages contain the information about the time of their respective next events. Finally, the controller updates the simulation time to the earliest event time and the next loop starts through messages 9 and 10.

A workcentre supervises several machine agents and is responsible for coordinating their messages to assure correct status updating. Thus, there are coordinating messages of a workcentre agent between messages 3 and 6. Similarly, a shop floor agent is responsible to coordinate workcentres through messages 2–7 in Figure 21.

Figure 21 The basic information flow in a simulation loop



6.3.1 Coordination work of a workcentre

The workcentre receives a time message from the shop floor agent and then begins to coordinate all the actions in it. The most complex situation is when a workcentre has to receive new incoming jobs and all of its machines have simultaneous due events. The goals of a workcentre are thus to make sure that:

1 new incoming jobs are properly allocated

2 interrupted jobs are reallocated

3 waiting jobs are allocated when machines are available

4 all machines update their new status

5 completed jobs leave the workcentre.

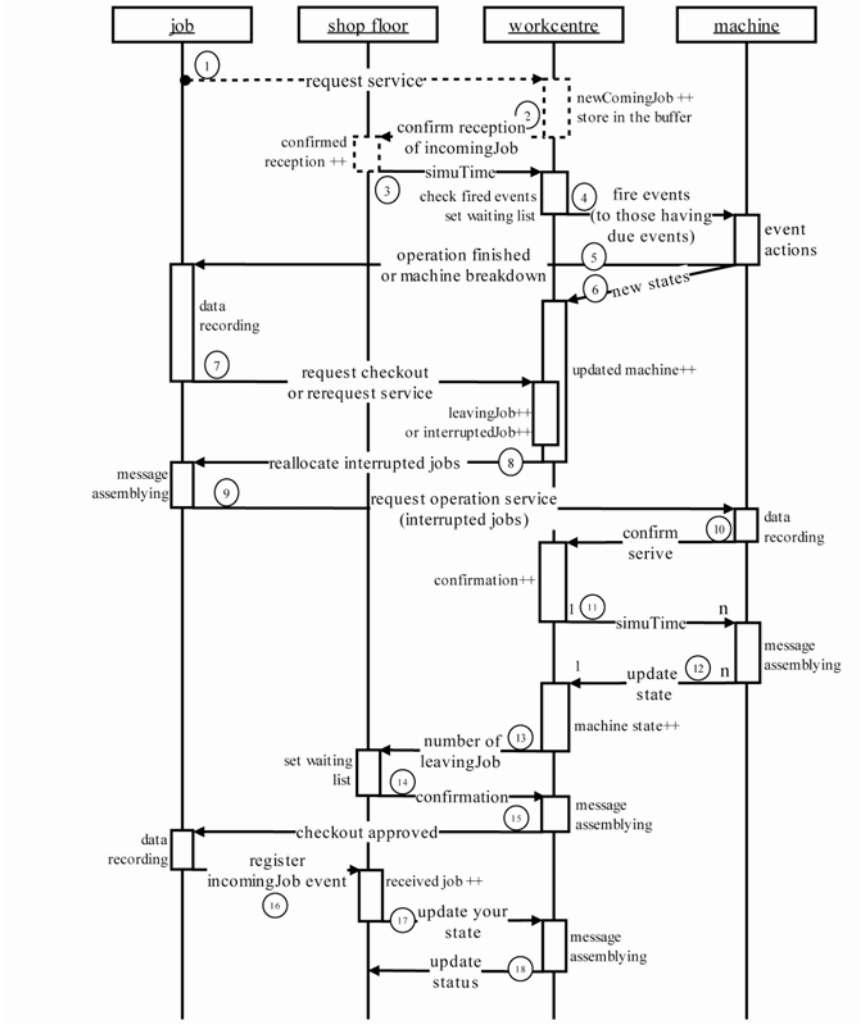
A workcentre can only update its new status to its supervisor after all the above goals are realised. The coordination messages are illustrated in Figure 22 based on the most complex situation mentioned above.

A workcentre receives the new incoming jobs through messages 1 and 2 before it receives a time message by message 3. It then checks the event list to determine the number and the types of fired events and a waiting list can be drawn accordingly. For example, the workcentre will expect to receive both a ‘checkout’ request from the job and a new state updating it from the machine if the event for finishing a job is fired. The workcentre then fires all the events by message 5.

The workcentre is contacted by all the expected machines and jobs through messages 6 and 7 after the event actions are finished. The workcentre may be unbalanced at this time with newly available machines and waiting jobs in the buffer. It then allocates the waiting jobs or reallocates the interrupted jobs to the machines through messages 8–10. The simulation time is forwarded through message 11 to all the machines, which

immediately update their status by message 12. Finally, the completed jobs are approved to leave the workcentre by messages 13–16, and a workcentre can update its new status by message 18.

Figure 22 Coordination work of a workcentre agent



6.3.2 Coordination work of the shop floor

The shop floor prepares to monitor all the dynamics at the moment it receives a time message from the controller agent. The most complex situation for the shop floor agent to coordinate is when the following dynamics occur simultaneously:

1 new jobs come to the shop floor

2 some travelling jobs arrive at their workcentres

3 some jobs in workcentres complete their operation and are ready to travel to the next stage.

The shop floor needs to make sure that

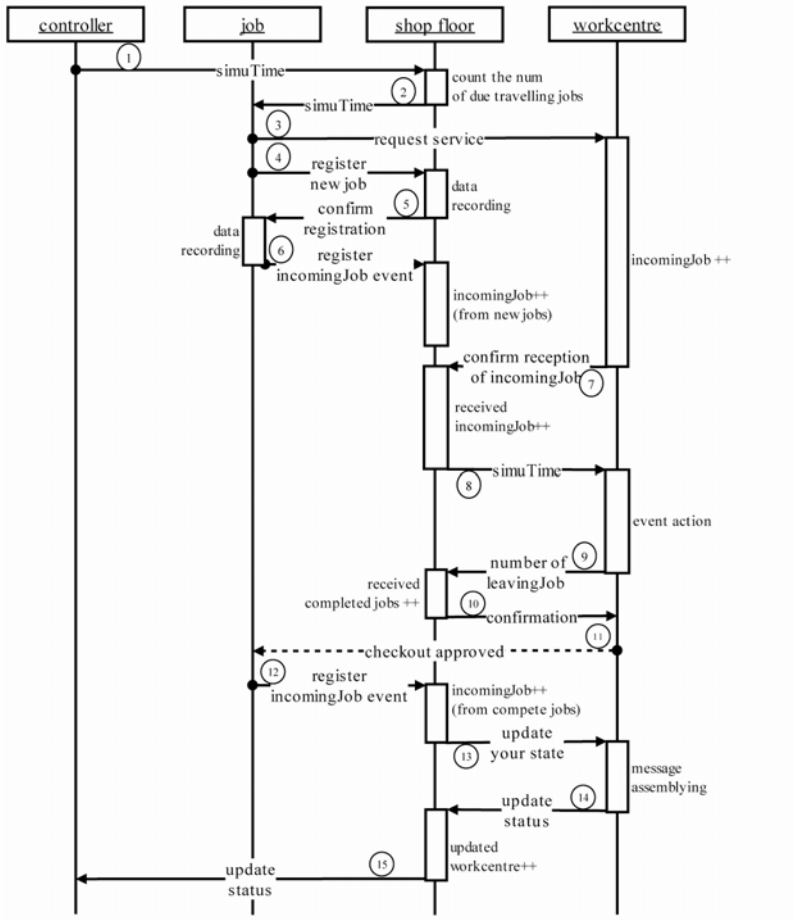
1 all new jobs are registered

2 travelling jobs are received by their workcentres

3 jobs leaving their workcentres reach the shop floor.

Only after all the above dynamics have been handled can the shop floor agent update its new status to the controller agent. The coordination messages are given in Figure 23.

Figure 23 Coordination work in the job shop agent



The shop floor agent also receives a time message through message 1 and initially makes sure that all the travelling jobs are received by their workcentres by messages 2, 3 and 7. It then updates all workcentres with the new simulation time by message 8. Meanwhile, there may be some new jobs entering the shop floor: they are handled through messages 4–6. The shop floor agent is then notified the number of leaving jobs by the workcentres through message 9 and starts to collect all the expected leaving jobs through message 12. It notifies all the workcentres to update their new states by message 13 after all the leaving jobs are collected. Finally, it updates its new status to the controller agent by message 15.

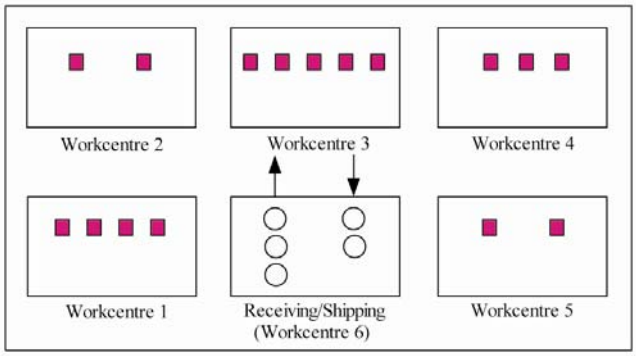
7 Case study

The case study pursued here adopts the data from the example on pages 684–695 of Law and Kelton (2000). The MAS model is built on JADE and runs on an AMD Opteron Linux Cluster with 26 nodes (2.2 GHz, 4 GB RAM) + 8 nodes (2.4 GHz, 32 GB RAM) in the Institute of High Performance Computing (IHPC, http://www.ihpc.a-star.edu.sg). The random number generator used in simulation is proposed by L'Ecuyer (1999).

7.1 Inputs

The studied job shop is shown in Figure 24 with five workcentres and one Receiving/Shipping station. The machines in a particular workcentre are identical, but the machines in different stations are dissimilar. The distances (in feet) between the six workcentres are given in Table 1. Jobs are transported between workcentres by MHDs assuming that enough of them are available and the time spent on the trip is proportional to the distance between the two locations.

Figure 24 Layout of the manufacturing system



Jobs arrive at the Receiving/Shipping station (workcentre 6) with inter-arrival times that are independent exponential random variables with a mean of 1/15 hr. There are three types of jobs: 1–3, with respective probabilities 0.3, 0.5 and 0.2. Job types 1–3 require 4, 3 and 5 operations to be done, respectively, and each operation must be done at a specified workcentre in a prescribed routing, which is given in Table 2. Each job enters the shop floor at the Receiving/Shipping station, travels to the workcentres on its routine and then leaves the system at the Receiving/Shipping station. All MHDs move at a constant speed of 5 feet per second.

Table 1 Distances between workcentres

Workcentre	*1*	*2*	*3*	*4*	*5*	*6*
1	0	150	213	336	300	150
2	150	0	150	300	336	213
3	213	150	0	150	213	150
4	336	300	150	0	150	213
5	300	336	213	150	0	150
6	150	213	150	213	150	0

Table 2 Technical routes of jobs

Job type	*Work stations in routing*
1	3, 1, 2, 5
2	4, 1, 3
3	2, 5, 1, 4, 3

A job joins a single FIFO buffer if all the machines in the workcentres are busy. The time to perform an operation at a particular machine is given in Table 3.

Table 3 Processing times of all operations

Job type	*Mean service time for successive operations (hours)*
1	0.25, 0.15, 0.10, 0.30
2	0.15, 0.20, 0.30
3	0.15, 0.10, 0.35, 0.20, 0.20

7.2 Simulation results

The simulation ran 10 replications of length 920 hr, which equals to 115 8-hr days. The results for different performance measures are listed in Table 4. The first row shows the configuration of the job shop, which is comprised of five workcentres with four, two, five, three and two machines, respectively. All performance measures except maximum number in queue and maximal size of working-in-process are the average values of ten experiments.

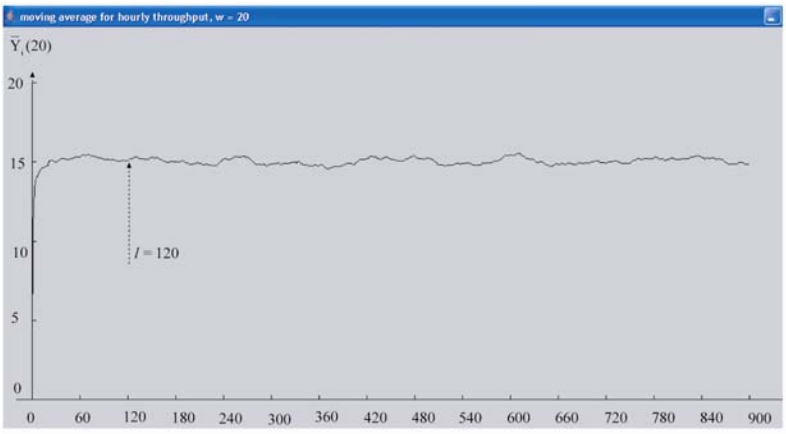
7.3 Statistical calculation

First a warming up period is obtained using Welch's procedure (Law and Kelton, 2000) on 920 hourly throughputs in each of the 10 replications. The moving average $\overline{Y}_i(20)$ uses a window of 20, and is plotted in Figure 25. A warming up period of l = 120 hr is obtained.

Table 4 Simulation results

Number of machines: 4, 2, 5, 3, 2					
Number of forklifts: enough					
Machine efficiency: 1					
Performance measure	*1*	*2*	*3*	*4*	*5*
Proportion machines busy (workcentre)	0.806	0.450	0.795	0.570	0.825
Average number in queue (workcentre)	1.662	0.137	0.653	0.276	0.790
Maximum number in queue (workcentre)	35	9	12	10	17
Average daily throughput (shop floor)			120.075		
Average time in system (shop floor)			1.067		
Average total time in queues (shop floor)			0.240		
Maximal size of working-in-process (shop floor)			56		

Figure 25 Moving average ($w = 20$) of hourly throughputs



Then a 90% confidence interval for the steady-state mean daily throughput is constructed as follows.

$$120.075 \pm t_{9,0.95}\sqrt{\frac{0.54}{10}} \text{ or } 120.075 \pm 0.23$$

which contains 120.

7.4 Result analysis

The expected daily throughput is 120 jobs per 8-hr day, which is the maximum possible (because the inter-arrival times of jobs are independent exponential random variable with a mean of 1/15 hr). The 90% confidence interval built in the previous section demonstrates that the system can reach a daily throughput of 120 jobs.

The simulation results of a similar system built by Law and Kelton (2000) are listed in the Table 5. The system has more constraints such as limited MHD and machine efficiencies while the case study in this paper assumes enough MHD and no machine breakdown. Both systems achieve 120 expected daily throughputs. This can be explained by the fact that Law and Kelton's system achieves the same level of proportion of busy machines despite limited resources. However, the limited resources cause both the average and the maximum number in the queues of Law and Kelton's system much greater than those in our system. Subsequently, the average time of a job in the system is longer in Law and Kelton's system than in our system. Thus, both the statistical analysis and the comparison to the existing report validate that the proposed DES-MAS system correctly simulate a dynamic job shop.

The case study also demonstrates other advantages, including distributed data collection and calculation. All the data have been collected and maintained by their most related agents. For example, the average/maximum numbers of jobs in queues are collected by five workcentre agents and the shop floor agent keeps those data that are out of the scope of the other agents. Those data are average daily throughput, average time in system, average total time in queues and the size WIP etc. Of course, the information of machine utilisation can be properly maintained by each machine itself. A workcentre can

request them to provide such information when it needs to calculate the proportion of busy machines under it supervision. Thus the burden of computation can be naturally distributed to the different computation nodes.

Table 5 Simulation results from Law and Kelton (2002)

Number of machines: 4, 2, 5, 3, 2					
Number of fork lifts: 2					
Machine in workcentre 1 and 5 have efficiencies of 0.9					
performance measure	*1*	*2*	*3*	*4*	*5*
Proportion machines busy (machines)	0.81	0.45	0.8	0.58	0.83
Average number in queue (workcentre)	16.55	0.25	2.15	0.49	46.73
Maximum number in queue (workcentre)	111.00	11.00	32.00	14.00	262.00
Average daily throughput (shop floor)	119.88				
Average time in system (shop floor)	5.31				
Average total time in queues (shop floor)	4.37				
Maximal size of working-in-process (shop floor)	–				

The CPU time from one replica ranges from 1393 to 4689 min, which varies as the load of the cluster changes. It seems long because:

1 the simulation time is advanced in a time step of 0.01 hr instead of 1 hr for a simulated period of 920.00 hr

2 more than 12,000 jobs in average are tested in each simulation

3 the communication overhead is high

4 all the experiments have been executed in one cluster and agents are actually not distributed to different CPUs, which implies that the advantage of concurrent computation is not realised.

The computing time would decrease if agents are distributed to different computing nodes in a stable network.

Further work includes testing the scheduling policies on more stochastic scenarios, examining scheduling algorithms in dynamic environments and, more interestingly, exploring the intelligence of agents such as negotiation, coordination and cooperation to work with scheduler in improving overall performance of a system. The test bed could also be extended to include a non-deterministic transportation system.

8 Conclusion

An MAS simulating a real-life job shop is built in order to provide a test bed for studying approaches in dynamic job shop environment. The essential architecture of a job shop manufacturing system is first identified and then it is built as a DES which can examine the performance of a system for a long time. The DES is implemented as an MAS so that the intelligent agents can be used to realise distributed computation and prompt reaction to dynamic events.

To the best of the authors' knowledge, this is the first implementation of MAS with DES for job shop systems. This approach requires careful coordination among event lists which are distributed among different agents in order to keep correct simulation time. The coordination involves much communication among agents. The agents in this model do not necessarily lose their autonomy. The discrete events set the time steps and the agents are autonomous within their event execution periods. In this way, a long term performance of an MAS can be examined.

All the communication and state changes are clearly illustrated using UML sequential diagrams and state charts. A case study demonstrates the advantage of distributed data collection and analysis. It also validates the proposed system by statistical analysis and comparison to existing simulation results on a similar test case.

References

Askin, R.G. and Standridge, C.R. (1993) *Modelling and Analysis of Manufacturing Systems*, John Wiley & Sons, Inc.

Bongaerts, L. (1998) 'Integration of scheduling and control in holonic manufacturing system', PhD Thesis, Katholieke University, Leuven.

Bongaerts, L., Monostori, L., McFarlane, D. and Kadar, B. (2000) 'Hierarchy in distributed shop floor control', *Computers in Industry*, Vol. 43, pp.123–137.

Cavalieri, S., Garetti, M., Macchi, M. and Taisch, M. (2000) 'An experimental benchmarking of two multi-agent architectures for production scheduling and control', *Computers in Industry*, Vol. 43, pp.139–152.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: an Introduction to the Theory of NP-Completeness*, San Francisco: W.H. Freeman.

Hopp, W.J. and Spearman, M.L. (2000) *Factory Physics – Foundations of Manufacturing Management*, 2nd edition, Irwin: McGraw-Hill.

Koestler, A. (1967) *The Ghost in the Machine*, London: Arkana Books.

L'Ecuyer, P. (1999) 'Good parameters and implementations for combined multiple recursive random number generators', *Operations Research*, Vol. 47, No. 1, pp.159–164.

Law, A.M. and Kelton, W.D. (2000) *Simulation Modelling and Analysis*, McGraw Hill.

Li, R.K., Shyu, Y.T. and Adiga, S. (1993) 'A heuristic rescheduling algorithm for computer-based production scheduling systems', *International Journal of Production Research*, Vol. 31, pp.1815–1826.

Okino, N. (1993) 'Bionic manufacturing systems', in J. Peklenik (Ed). *Proceedings of the CIRP Seminar on Flexible Manufacturing Systems Past-Present-Future*, Bled, Slovenia, pp.73–95.

Parunak, H.V.D. (1997) '"Go to the Ant": engineering principles from natural multi-agent systems', *Annals of Operations Research*, Vol. 75, pp.69–101 (Special Issue on Artificial Intelligence and Management Science).

Ramasesh, R. (1990) 'Dynamic job shop scheduling: a survey of simulation research', *OMEGA International Journal of Management Science*, Vol. 18, No. 1, pp.43–57.

Ryu, K. and Jung, M. (2003) 'Agent-based fractal architecture and modelling for developing distributed manufacturing systems', *International Journal of Production Research*, Vol. 41, No. 17, pp.4233–4255.

Sabuncuoglu, I. and Bayiz, M. (2000) 'Analysis of reactive scheduling problems in a job shop environment', *European Journal of Operational Research*, Vol. 126, pp.567–586.

Sabuncuoglu, I. and Kizilisik, O.B. (2003) 'Reactive scheduling in a dynamic and stochastic FMS environment', *International Journal of Production Research*, Vol. 41, No. 17, pp.4211–4231.

Shen, W. and Norrie, D.H. (1999) 'Agent-based systems for intelligent manufacturing: a state-of-the-art survey', *International Journal Knowledge and Information Systems*, Vol. 1, pp.129–156.

Smith, R.G. (1980) 'The contract net protocol: high-level communication and control in a distributed problem solver', *IEEE Transactions on Computers*, Vol. C-29, No. 12, pp.1104–1113.

Valckenaers, P., Bonneville, F., Brussel, H.V. and Wyns, J. (1994) 'Results of the holonic system benchmark at KU Leuven', *Proceedings of the Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, 10–12 October, Troy, New York, pp.128–133.

Vieira, G.E., Herrmann, J.W. and Lin, E. (2003) 'Rescheduling manufacturing systems: a framework of strategies, policies, and methods', *Journal of Scheduling*, Vol. 6, No. 1, pp.39–62.

Warnecke, H.J. (1993) *The Fractal Company, a Revolution in Corporate Culture*, Berlin: Springer-Verlag.

Wyns, J. (1999) 'Reference architecture for holonic manufacturing systems: the key to support evolution and reconfiguration', PhD thesis, Katholieke University, Leuven.