Journal of Achievements in Materials and Manufacturing Engineering

VOLUME 40
ISSUE 1
May
2010

Dynamic scheduling for agent based manufacturing systems

J. Madejski*
Division of Materials Processing Technology, Management and Computer Techniques in Materials Science, Institute of Engineering Materials and Biomaterials, Silesian University of Technology, ul. Konarskiego 18a, 44-100 Gliwice, Poland
* Corresponding author: E-mail address: janusz.madejski@polsl.pl

Received 12.02.2010; published in revised form 01.05.2010

Analysis and modelling

ABSTRACT

Purpose: Development of the decision making architecture for the multi-agent societies with temporal restrictions. General ideas for the necessary architecture based on the blackboard one is presented.
Design/methodology/approach: Fuzzy logic approach that makes it possible to reach suboptimal solutions within the acceptable timeframe. Development of the relevant systems calls for compiling the experience gathered over the years in the system served by human 'agents'. Multiagent systems negotiation needs were analysed and cooperation issues in the form of clustering, cloning, and learning were analysed in search for the relevant tools.
Findings: Detailed review of the approach to development of the agent based Intelligent Manufacturing from the fundamental considerations to the latest hands-on developments.
Research limitations/implications: Many presented technologies call for detailed study before they can be implemented in practice.
Originality/value: Analysis of the local interactions among agents meeting the real-time reaction requirements.
Keywords: Artificial software agents; Multiagent systems; Scheduling; Negotiation

Reference to this paper should be given in the following way:
J. Madejski, Dynamic scheduling for agent based manufacturing systems, Journal of Achievements in Materials and Manufacturing Engineering 40/1 (2010) 66-69.

1. Introduction

The conventional scheduling approach, known as static scheduling may solve the problem and provide the - usually suboptimal - schedule, yet in real life conditions they turn out to be impractical because of their mostly unrealistic assumptions. This is because the real manufacturing systems are complex and dynamic with a big number of products and processes, with many production levels, and subject to random disturbances. One may name some of these disturbances like: new orders may come, those queued already may be cancelled, some jobs may become more or less important in time, technological equipment may fail, moreover some resources may temporarily become unavailable, as deliveries may be delayed, raw materials may be depleted, tools may not be available for a number of reasons (e.g., due to shorter service life due to poor quality), staff may get ill, etc. Therefore such dynamic entity needs dynamic scheduling. This boils down to real-time control, as all decisions have to be made based on the current state of the manufacturing system. The artificial software agents may take over manufacturing scheduling which consists in allocating and timing the manufacturing system resources in order to complete the queued jobs within the timeframe allowed using some desired criteria [1-4].

Efficient and timely collection and access to data describing the manufacturing system status feature an important issue in development of the decision making systems that will perform properly their tasks in the dynamic environment. The main problem then is how to acquire the right decision in these uncertain conditions. Moreover, making a decision does not mean that it will be an optimal one, as the time to obtain it is limited, and usually the decision making system will not have the power to develop the globally optimal modified work plans. A good approach to solve this problem is using the fuzzy logic approach that makes it possible to reach suboptimal solutions within the acceptable timeframe. Development of the relevant systems calls for compiling the experience gathered over the years in the system served by human 'agents'. Such knowledge base may be later used first to mimic the behaviour of the system controlled by human operators, and later - to populate it, sometimes modified, 'cleaned' and optimised as the decision making system prototypes for new manufacturing systems. The fuzzy models based on this knowledge are event oriented and represent single agents which - when needed - should be able to solve jointly problems exceeding the capacity of a single one. To this end negotiation skills are needed which lead either to delegation of a task to a single agent or in setting up an ad-hoc task group to handle the problem. To this end the following framework is proposed (Fig. 1).

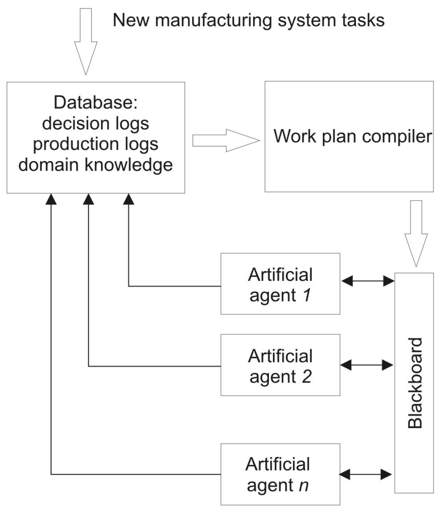

Fig. 1. Task planning and execution framework

The proposed agent architecture may incorporate three types of agents:

- Type A - representing the physical system entities, like a workpiece, a machine or production line/cell, humans, the shop floor subsystem, or the entire plant in a supply chain, part-oriented scheduling, and even the whole scheduling process - here represented as the Database in Fig.1
- Type B - agents either existing already or created on the fly to resolve a scheduling conflict
- Type C agent, as a high-level supervisory entity focused the overall manufacturing goals - the Work Plan Compiler in Fig. 1.

In general all system elements - agents - function in a layered architecture; using different mechanisms at different levels. The agents used include the functional agents, usually designed using the BDI approach, employing the voting protocol for communication and Contract Net negotiation protocol to reach final decisions and complete task allocation problems [7-18].

The Database, developed in My SQL, in addition to the domain knowledge contains relations allowing it to store the production and resource data. The Work Plan Compiler accesses the necessary information whenever an event occurs which needs its intervention. Such even may be either a new task to be allocated to an agent, a machine break down, or, e.g., a tool setup request. The minimum relevant database entity types required to store the data needed for system status logging and task planning are as follows:

- Resource status: data on the particular resources' service history, including, but not limited to its current status, maintenance schedule, down times, expected time to repair, mean time between failures, resources recommended for tending, maintenance, and repairs, current physical location, availability, etc.
- Domain knowledge: in the form of the precompiled fuzzy projects, specifying the optimal course of actions to be taken in case of any events, like allocation of a new task, machine breakdown, maintenance required, etc.
- Decisions: this entity type is used for storing the history of all decisions made - and realised, along with the relevant statistics describing the efficiency of the action taken, which will be used in improvement of the plan development strategy for the future decisions.
- Production: necessary for recording the manufacturing system core tasks flow, like the production rate, time requirements, resources used, and all events that affected the efficient work flow, like a list of broken machines, including machine name, machine age, degree and cause of breakdown.

2. Decision making architecture proposal

To guarantee that the agent finds the relevant solutions within a time limit, as is in the case of many agents, shown in Fig. 1 who – learning from the Blackboard about the new events - have to react to them, either committing to carry out the new task, or looking for assistance, or simply getting involved in negotiations with others which one of them would do the job, the architecture is needed that will ensure this goal. Meeting this requirement calls for a two level approach, as reaching a solution in real time does not allow lengthy deliberation process, so - as mentioned above - calls for a number of pre-compiled procedures, contingency plans to be carried out whenever is needed. The most important issue remaining how to select them and evaluate which will be the most relevant at a given system state [5-9].

To this end the task execution levels should use the following approaches:

- Real-time scheduler, the fixed-priority one, ensuring the timely reaction with the reasonable system operation quality and letting it function safely until a better solution is not worked out. As many agents processes run on the same CPU, it means that some minimum amount of time should be guaranteed for each of them to eventually take over the tasks and execute it in an immediate reaction to an event, so that there is no delay in waiting for the allocated CPU time slot for the agent.

- The second level scheduler (deliberative one) improves the solutions quality while there is enough time for that. Results of this planning are used for modifications of the pre-compiled procedures used by the first level scheduler, and therefore, affect agents' plans from the moment when the priorities are updated. This deliberative scheduler can carry out global optimisation of the work plans, including, e.g., earlier maintenance, when the workload is lower.

This architecture ensures an agent reaction that meets the requirements of the real-time systems, being also capable of adapting the system behaviour to the dynamic environment conditions.

3. Local interactions in the multi-agent environment

The architectures proposed in the literature for agent-based manufacturing systems fall into three approaches: the Hierarchical approach, the Federation one, and the Autonomous Agent one.
Any modern manufacturing enterprise is composed of many, most often distributed physically, semi-autonomous units, all having a certain degree of control over local resources or having varying information requirements. In such real situations, a certain number of agent-based industrial applications still use the hierarchical architecture.

In any multi-agent system their coordination efficiency is a key factor. As mentioned above, the agent may carry out tasks alone or jointly with the others, when it would otherwise not be able to cope with, thus increasing its status with the award function, profiting from the actions of other agents.

Negotiations leading to cooperation among agents should not only be aimed at benefitting the individual agents but, first of all, at improving the overall system performance. In the multi-agent systems, there are the following relationships types:

- Order - when one agent - a 'supervisor' one delegates some task to another agent, usually after negotiations, so that the best option is selected (e.g., the task will be carried out as soon as possible, taking into account the current workloads of the available agents)
- Cooperation - two or more agents may join forces to carry a task together (e.g., lift and transport an element which is too heavy or too big for one of them, or doing a task together may cut the operation time)
- Non-cooperation - when agents act in a 'selfish' way disregarding other agents' plans and invitations to cooperation negotiations.

Analysing cooperation among agents one should take into account their relative location at the time. The agents may be located close to each other or stay far away from each other. The issue of 'closeness' is relative as the cooperating software agents running on different distributed CPUs may still act as if they were close, albeit the physical distance among the computers may be significant.

Anyway, according to literature [3] the agents tend to choose partners close to their own localisation. Moreover they tend to mimic other adjacent agents approach which leads to two additional classes of interactions among the agents:

- coordinated agents - the agents that currently are executing some task together, and
- local interacted agents - developing some collective behaviour common to a group of adjacent agents.

This approach assumes [3,4] that every agent gets into interactions frequently with only a limited number of the agent group, which may be called its 'neighborhood' and, as the studies show [10-16] agents interact with other agents in more or less stable way, so that the connections between them do not change much.

Agents may be of various types, and thus their society may be split - in federation architectures - to the following approaches have been used: Facilitators, Brokers and Mediators. Facilitators are several related agents which are combined into a group. A facilitator is a communication interface between agents. Every facilitator is responsible for ensuring communication between a local collection of agents and remote agents, by: routing outgoing messages to their destinations, translating incoming messages for its agents.

Brokers resemble the facilitators having two additional functions such as monitoring and notification. The difference between a facilitator and a broker is that a facilitator is responsible only for a given group of agents, whereas any agent may contact any broker in the same system for finding service agents to complete a special task.

In addition to the functions of a facilitator and a broker, a mediator assumes the role of system coordinator by promoting cooperation among intelligent agents and learning from the agents' behavior. The Federation multi-agent architectures can to coordinate multi-agent activity via facilitation as a means of reducing overheads, ensuring stability, and providing scalability.

The Autonomous Agent approach is different. The autonomous agent should have the following characteristics at least: it is not controlled or managed by any other software agents or human beings; it can communicate/interact directly with any other agents in the system and also with other external systems; it has knowledge about other agents and its environment; it has its own goals and an associated set of motivations. The Autonomous Agent approach is well suited for developing distributed intelligent design systems where the system consists of a small number of agents and for developing autonomous multiple robotic systems.

4. Temporal restrictions

As mentioned above the problem solving task has requires splitting it into some smaller entities, which makes it easier to use the domain knowledge in a modular way, i.e., for simple tasks there are always good procedures, while there are non for the complex ones. Moreover, splitting the big problem into smaller chunks makes it easier to allocate the sub-tasks to the particular agents, and these sub-tasks may be carried out in parallel at times. In addition, this approach makes it possible to launch the relevant -fixed priority - reactions in real time.

Sharing of the solutions of sub-problems - extending the Database with the Domain Knowledge (Fig. 1.) helps the other agents to benefit in future from the 'experience' gathered by the other agents.

Development of the efficiently operating agent-based manufacturing systems, including the real-time ones, is usually carried out so far using such programming languages like C++, Java, Lisp, Prolog, Objective C and SmallTalk.

5. Conclusions

Efficient and timely collection and access to data describing the manufacturing system status feature an important issue in development of the decision making systems that will perform properly their tasks in the dynamic environment. Meeting the real-time reaction requirement calls for a two level approach, as reaching a solution in real time does not allow lengthy deliberation process, so - as mentioned above - calls for a number of pre-compiled procedures, contingency plans to be carried out whenever is needed. There are two different approaches to agent design: the physical decomposition approach and the functional decomposition one. In the physical decomposition approach, agents represent physical entities, like workers, machine tools, tools, fixtures, or products, etc. On the other hand, in the functional decomposition approach, there is no relationship between agents and physical entities, but agents are assigned to some functions like product distribution, , transport management, order acquisition, scheduling, material handling, etc. Development of multi-agent systems requires taking into account the specific features of these two abovementioned approaches.

References

[1] L. Hernández, V. Botti, A. García-Fornes, A deliberative scheduling technique for a real-time agent architecture, Engineering Applications of Artificial Intelligence 19/5 (2006) 521-534.

[2] Kun-Yung Lu, Chun-Chin Sy, A real-time decision-making of maintenance using fuzzy agent, Expert Systems with Applications 36/2 (2009) 2691-2698.

[3] Y. Jiang, T. Ishida, A model for collective strategy diffusion in agent social law evolution, Proceedings of the 20th International Joint Conference on "Artificial Intelligence IJCAI'07", Hyderabad, India, 2007, 125-132.

[4] Yichuan Jiang, Toru Ishida, Local interaction and non-local coordination in agent social law diffusion, Expert Systems with Applications 34/1 (2008) 87-95.

[5] K.J. Wróblewski, R. Krawczyński, A. Kosieradzka, S. Kasprzyk, Priority rules in production flow control, WNT, Warsaw, 1984 (in Polish).

[6] N. Jennings, M. Wooldridge, Applications of Intelligent Agents, in: N. Jennings, M. Wooldridge (Eds.), Agent Technology. Foundations, Applications, and Markets, Springer-Verlag, 1998, 3-28.

[7] K .J. Wróblewski, Fundamentals of production flow control, WNT, Warsaw, 1993 (in Polish).

[8] J. Madejski, Agents as building blocks of responsibility-based manufacturing systems, Elsevier, (ref no PROTEC 4585 - 27 June 2000).

[9] L. Interrante, S. Goldsmith, Emergent Agent-Based Scheduling of Manufacturing Systems, in: Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, 1998.

[10] B. Burmeister, S. Bussmann, A. Haddadi, K. Sundermeyer, Agent-Oriented Techniques for Traffic and Manufacturing Applications: Progress Report, in: N. Jennings, M. Wooldridge (Eds.), Agent Technology. Foundations, Applications, and Markets, Springer-Verlag, 1998, 161-174.

[11] S. Ossovski, Co-ordination in Artificial Agent Societies. Social Structure and Its Implications for Autonomous Problem-Solving Agents, Springer-Verlag, 1999.

[12] C. Meloni, Autonomous agents architectures and algorithms in flexible manufacturing systems, in: IIE Transactions 32/10 (2000) 941-951.

[13] L. Kerschberg, Knowledge Rovers: Cooperative Intelligent Agent Support for Enterprise Information Architectures, in: P. Kandzia, M. Klusch (Eds.), Cooperative Information Agents, Proceedings of the 1st International Workshop, CIA'97, Kiel, Germany, Springer-Verlag, 1997, 79-100.

[14] T. Ohko, K. Hiraki, Y. Anzai, Addressee Learning and Message Interception for Communication Load Reduction in Multiple Robot Environments, G. Weiss (Ed.) Distributed Artificial Intelligence Meets Machine Learning. Learning in Multi-Agent Environments, ECAI'96 Workshop LDAIS, Budapest, Hungary, 1996 and ICMAS'96 Workshop LIOME, Kyoto, Japan, 1996, Springer-Verlag 1997, 242-258.

[15] J. Madejski, Survey of the Agent-Based Aproach to Intelligent Manufacturing, Journal of Achievements in Materials and Manufacturing Engineering 21/1 (2007) 67-70.

[16] R.C Arkin, T. Balch, Cooperative Multiagent Robotic Systems, in D. Kortenkamp, R.P. Bonasso, R. Murphy (Eds.), Artificial Intelligence and Mobile Robots, AAAI Press / The MIT Press, 1998, 277-296.

[17] T. Schael, Workflow Management Systems for Process Organisations, Springer-Verlag, 1998.

[18] J. Madejski , Modelling of the Manufacturing System Objects Interactions, Journal of Achievements in Materials and Manufacturing Engineering 24/2 (2007) 167-170.