Int J Adv Manuf Technol (2017) 89:3123–3137
DOI 10.1007/s00170-016-9299-4

ORIGINAL ARTICLE

A dynamic multi-agent-based scheduling approach for SMEs

Ali Vatankhah Barenji[1] · **Reza Vatankhah Barenji**[2] · **Danial Roudi**[1] · **Majid Hashemipour**[1]

Received: 13 February 2016 / Accepted: 5 August 2016 / Published online: 15 August 2016

Abstract In modern manufacturing systems with computational complexities, decision-making with respect to dynamic rescheduling and reconfiguration in case of internal disturbances is an important issue. This paper introduces a multi-agent-based dynamic scheduling system for manufacturing flow lines (MFLs) using the Prometheus methodology (PM) considering the dynamic customer demands and internal disturbances. The PM is used for designing a decision-making system with the feature of simultaneous dynamic rescheduling. The developed system is implemented on a real MFL of a small- and medium-sized enterprise (unplasticized polyvinyl chloride (uPVC) door and window) where the dynamic customer demands and internal machine break downs are considered. The application has been completely modeled using a Prometheus design tool, which offers full support to the PM, and implemented in JACK agent-based systems. Each agent is autonomous and has an ability to cooperate and negotiate with other agents. The proposed decision-making system supports both static and dynamic scheduling. A simulation platform for testing the proposed multi-agent system (MAS) is developed, and two real scenarios are defined for evaluating the proposed system. The analysis takes into account the comparisons of the overall performances of the system models using the MAS scheduling and conventional scheduling approaches. The result of simulation indicates that the proposed MAS could increase the uptime productivity and the production rate of flexible flow-line manufacturing systems.

Keywords Multi-agent system · Dynamic scheduling · Flow-line manufacturing · Prometheus methodology · JACK platform

1 Introduction

Over the last decade, small- and medium-sized enterprises (SMEs) have gained importance, indicating the economic growth of a country. In addition, larger establishments have lost ground in terms of market share and employment [1]. Scheduling and control problems in the SMEs differ from those of the large-sized enterprises in three ways. First, in SMEs, an order is accepted based on the availability and capacity of the right type of equipment. Second, the number of job types is much higher than the large enterprises, and consequently, the amount of manufacturing data to be generated per unit of work is very high. Third, the demand is dynamic [2]. Therefore, the process planning and continuity of activities aiming at the flexible use of the manufacturing equipment and human resources are vital concerns [3, 4]. The conventional scheduling systems in SMEs have the following issues [5, 6]: (1) the systems are not reactive to parallel requests—the conventional systems are typically unable to manage a set of simultaneous events that must be addressed; (2) lack of distribution—the scheduling and control system uses a centralized decision support system, which is located on a host computer; and (3) weak response to reconfiguration in the case of disturbances even though a large number of internal and/or external disturbances may occur in the system.

A dynamic scheduling system (DSS), which allows enterprises to optimally match the desired customer demands with

✉ Ali Vatankhah Barenji
Ali.vatankhah@cc.emu.edu.tr

[1] Department of Mechanical Engineering, Eastern Mediterranean University, Famagusta, North Cyprus Via 10, Mersin, Turkey

[2] Department of Industrial Engineering, Hacettepe University, Beytepe Campus, 06800 Ankara, Turkey

their plans, is a time-dependent system [7]. In this system, decisions are taken based on the correctness of both logic and time, resulting in a considerably increased scheduling efficiency [8]. The logic correctness fulfills the constraints with respect to the resource capacity and order of operations, whereas, time correctness satisfies the time-based constraints such as interoperation and due dates [4]. DSSs are appropriate for systems with several internal and/or external disturbances (e.g., order changes and machine failures) [9]. In the literature, two different approaches are mainly used for solving the dynamic scheduling problems, namely, a dynamic planning-based approach and dynamic best effort approach [10]. In the dynamic planning-based approach, scheduling starts when a job arrives, and the job is accepted only if timeliness is guaranteed [11]. Whereas, in the dynamic best effort approach, the ability to schedule a job is not checked [12]. According to Yoon and Shen [13], DSSs can be categorized into hard and soft deadline systems. In the hard deadline systems, time correctness is crucial for all decisions, whereas in the soft deadline systems, time correctness is important but not crucial. Hence, dynamic scheduling for manufacturing flow lines (MFLs) is adequate for hard time-dependent systems and requires a dynamic planning-based approach [14]. A dynamic planning-based approach with dynamic customer demands, mainly reported in the computer science literature, is studied to allocate a central processing unit (CPU) and memory space, and a job typically requires only a single resource. For instance, Ramamritham et al. [15] proposed a scheduling algorithm for real-time multiprocessor systems with hard deadlines. The scheduling algorithm uses a search option to find a feasible schedule. Unlike the examples in the computer science literature, the resources in MFLs include machines and material handling systems (MHSs), and a job typically uses a subset or the entire set of resources. Recently, agent technologies have been applied for dynamic planning-based scheduling in manufacturing systems. For example, Yoon and Shen [16] constructed a multi-agent system (MAS) for scheduling a semiconductor manufacturing factory in which four types of agents were designed and developed. A scheduling agent determined an optimal scheduling plan by estimating a few possible scenarios.

The MAS provides a new method for solving distributed, dynamic scheduling problems. Extensive research literatures exist, which address many scheduling issues of modern manufacturing companies with agent technology [17, 18]. The MAS has often been employed with a contract-net negotiation protocol [19] for solving various problems of scheduling and failure handling in manufacturing tasks. For scheduling manufacturing tasks, Valckenaers and Van Brussel [20] have utilized an agent-based decentralized manufacturing execution system composed of exploring ant agents for providing a look ahead into dynamic resource scheduling problems. Kaplanoğlu [21] proposed a real-time scheduling system based on the multi-agent system, which is less sensitive to the fluctuations in demand or available vehicles than the traditional transportation planning heuristics (local control, serial scheduling) and provides flexibility by solving local problems. To set up dispatching rules, Chen et al. [22] implemented a distributed agent-based system by applying a multi-agent technique to a multi-section flexible manufacturing system, which assists the agents in choosing suitable dispatch rules pertaining to the dispatching region and resolves the entire dispatching problems of a manufacturing system by agent cooperation. In most of the existing literatures, a specific methodology is introduced to the design and development of the MAS, whereas for SMEs, it is not possible to employ a new methodology for the development of the MAS because of the limited budget of the SMEs. The best way to overcome this concern is to use a general-purpose design methodology [23]. An effort in this regard can be found in [24].

In an effort to highlight the effect of internal disturbance on the manufacturing control system, the authors in their past contributions designed and developed a multi-agent-based architecture for scheduling and control in the manufacturing industry where they employed a radio-frequency identification (RFID) technology (instead of a barcode system) for tracking and tracing the parts [25]. The developed scheduling and control system was tested on an educational manufacturing shop, and the extracted results were compared with those of the conventional centralized scheduling system [9, 25] for the manufacturing scenarios with and without internal disturbances (machine breakdown). The results highlight the potential of employing both the MAS and RFID technology as new paradigms for retrofitting the current manufacturing system. Two common ways to test the internal disturbances of a system and the performance of a control architecture are the use of small prototype shops and virtual platforms. The best results can be obtained by using a physical system; however, it is very expensive and time consuming. Recently, a simulation test platform has been developed by the authors for examining the scheduling process considering internal disturbances [26]. For modeling a shop, a process-oriented colored Petri net (CPN) modeling method is employed using a top-down approach. The result of the study justifies the applicability of the developed tool and introduces a generic and flexible test platform for examining any control architecture.

The literatures referred by the authors indicate that there are some valuable efforts in the design and development of DSSs using the MAS with general-purpose methodologies or specific methodologies [27, 28]. In a few of these reports in some ways, the dynamic customer demand and/or internal disturbances of the system are considered [29]. Since all these MASs are implemented just on autonomous decision-making platforms, the real response and breakdowns of the machines in the system are not well-thought-out. Therefore, this paper presents a multi-agent-based DSS for SMEs

considering both together the dynamic customer demands and internal disturbances of MFLs. A multi-agent DSS is developed based on the Prometheus methodology™ (PM) because this methodology is a general-purpose design methodology for developing software agent systems and is not tied to any specific model in the software platform [25]. For modeling the internal disturbances of the system, a simulation test platform [3] is linked to the developed multi-agent-based DSS. A case study is conducted on a company manufacturing doors and windows. In this study, the dynamic order behavior and the capability to reconfigure with respect to the internal disturbances of a system are considered for a wide range of products.

The rest of the paper is organized as follows. Section 2 gives the detailed information about the stages of the PM and its capabilities. Section 3 describes the case study and highlights the drawbacks of MFLs. It also describes the design phases of the system for dynamic customer demands. Section 4 presents the proposed MASs and the employed simulation platform. Section 5 presents the conclusion and some future works.

2 Prometheus methodology

The PM is a general-purpose design methodology for developing software agent systems in which it is not tied to any specific model of the software platform [30]. The PM defines the detailed processes for specifying, designing, implementing, and testing/debugging agent-oriented software systems. In addition to the detailed processes (and several practical tips), it defines a range of artifacts produced during the processes. The PM consists of four steps, three of which deal with the design of the agent-oriented software and the last step deals with the implementation of the system. In this study, JACK is selected as a platform for implementing the proposed MAS. Figure 1 illustrates the design steps of the PM.

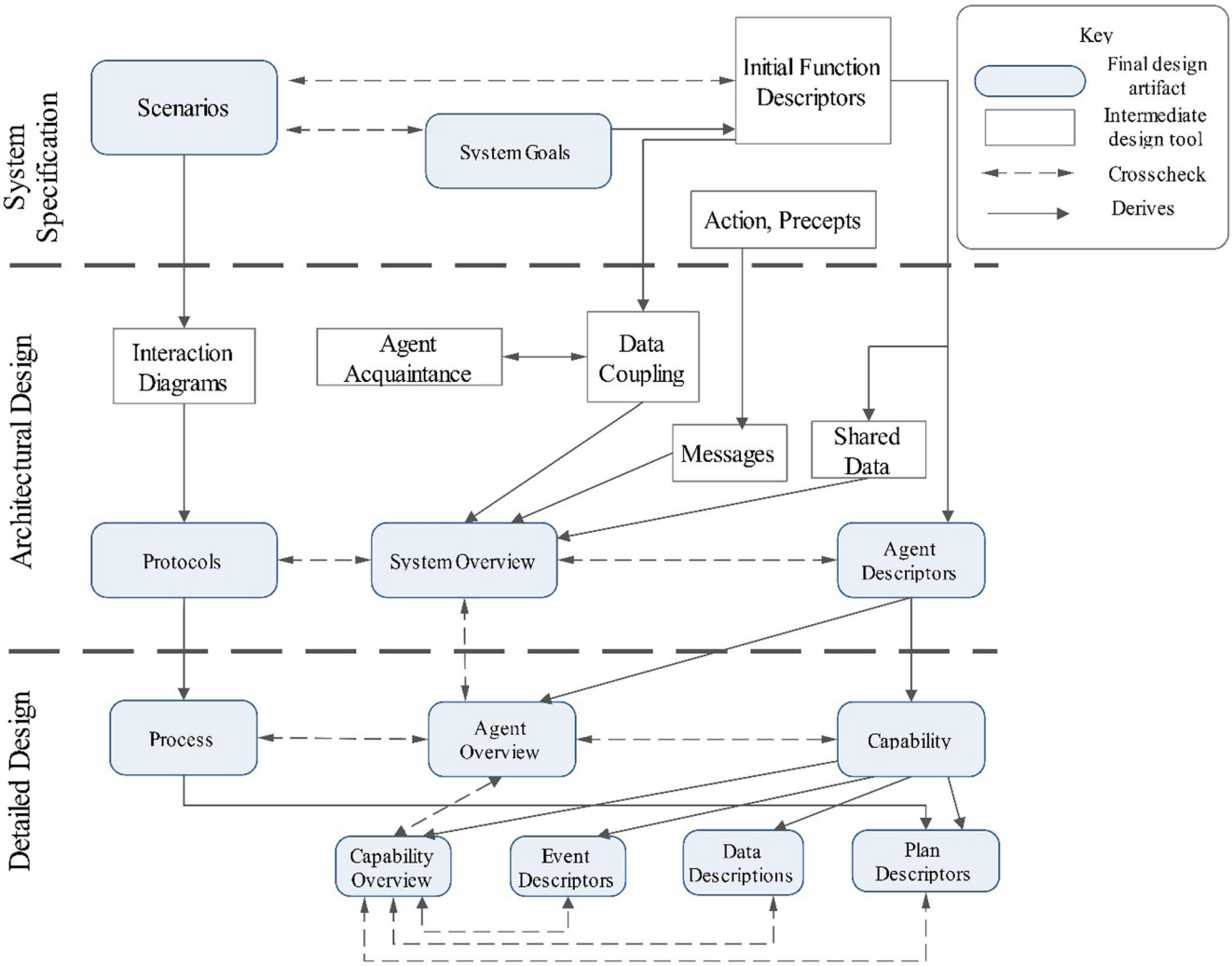

Fig. 1 Design steps of the PM [23]

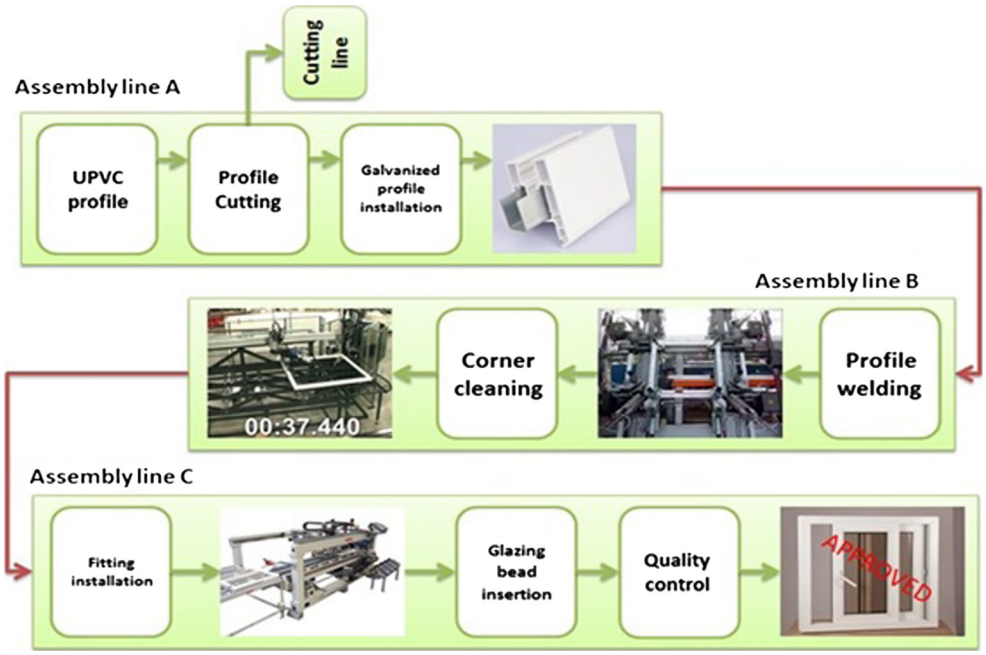

Fig. 2 Layout of the manufacturing MFL of Yaran Bahar Golestan

As shown in Fig. 1, the design steps of the PM are as follows [30].

The system specification phase focuses on the identification of the goals and basic functionalities of the system along with the inputs (precepts) and outputs (actions).

The architectural design phase uses the outputs of the previous phase to determine the types of agents in a system and their interaction.

The detailed design phase focuses on the internals of each agent and the ways to accomplish the tasks of the agents within the overall system.

By adhering to the PM, the first step is to define the system specification. The system specification defines the actors participating in the system, describes the scenarios of participation by defining the initial functionality descriptors, and finally identifies the system goals. The actors are the entities using the system or interacting

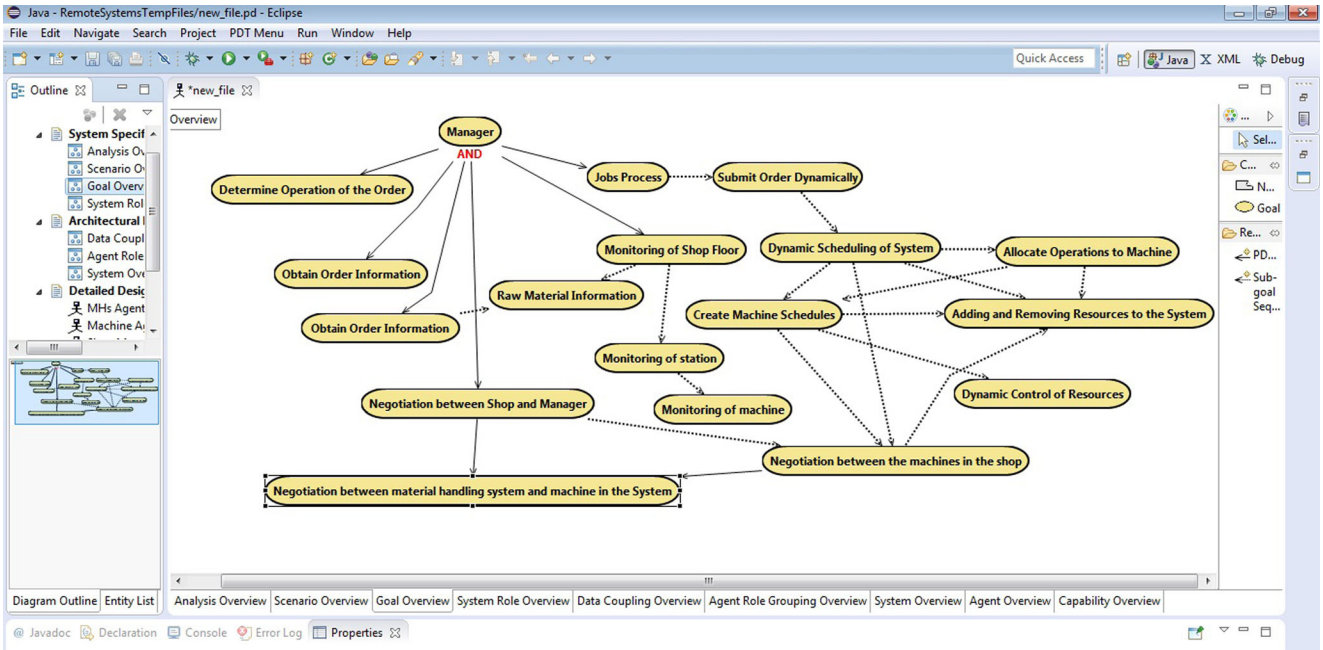

Fig. 3 Goal overview diagram of the system

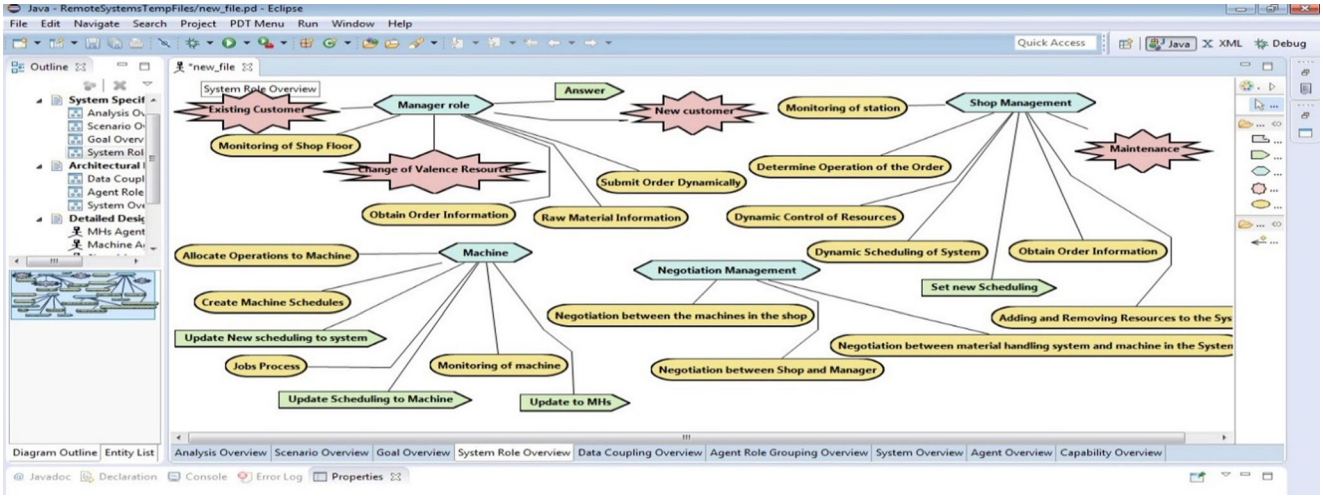

Fig. 4 System role overview

with the system in some way. The scenarios describe the occurrence of interactions [31]. The next step is to identify the tasks for each of these actors. The designer can identify the scenarios that each agent may act upon by elaborating the tasks assigned to each actor in the system. After identifying the scenarios, the designer can use them to determine the goals of the system. From these initial goals, the designer can determine the additional sub-goals. The goals are grouped into similar functions, and the duplicate goals are removed. The intention is to describe the functionality descriptors of the system. The next step is to identify the precepts in the system. The precepts are the types of information input to the system from the external environment. The designer can identify the precepts by studying the previous artifacts. The next stage in the system specification process is to describe the actions. The actions are defined by the information sent from the system to the external environment. The final stage of the system specification process is to develop the initial functionality descriptors. This groups the actions, precepts, and goals into a description that can be used in the future design. Once the system specification is defined, the architectural design of the system commences. The data coupling diagram can be produced from the initial functionality and data descriptors developed as a part of the system specification. The next step

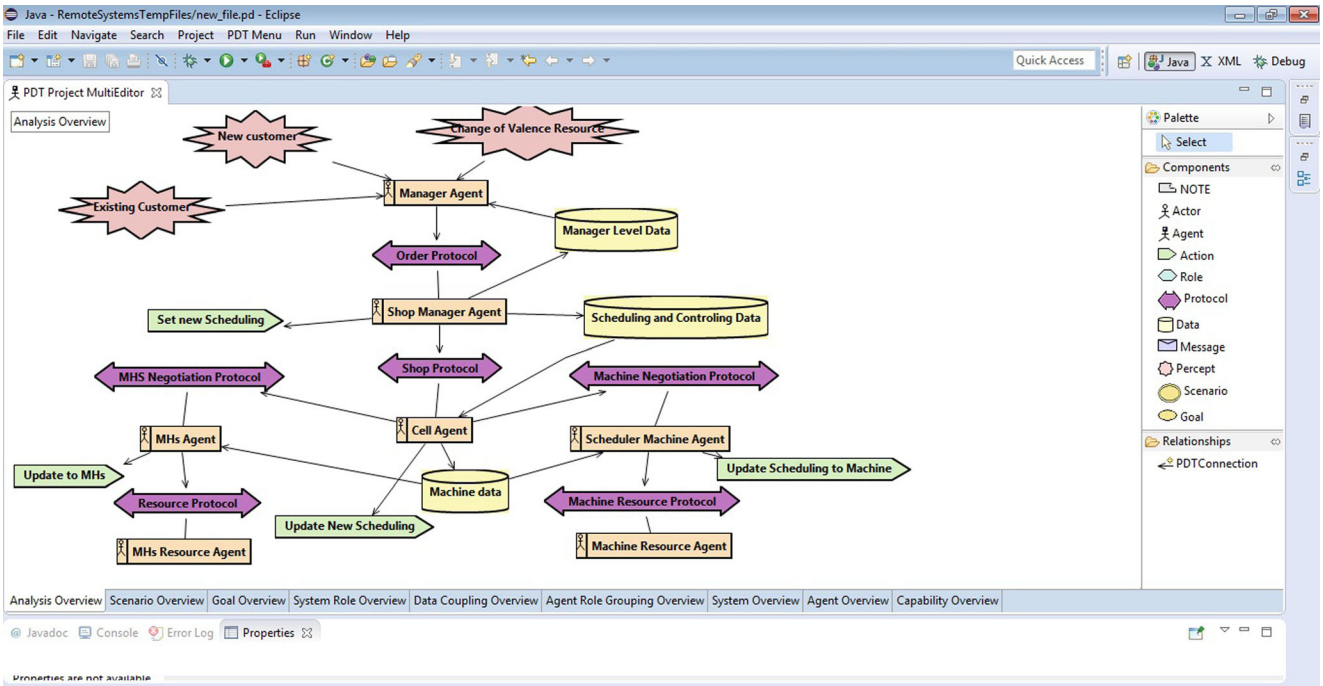

Fig. 5 System overview diagram in the architectural design stage

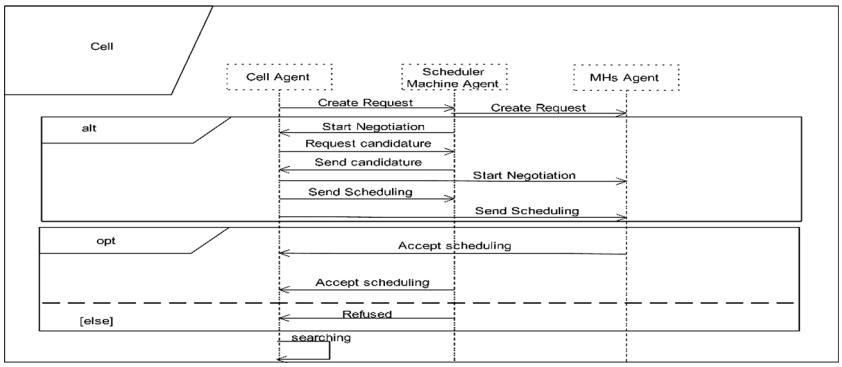

Fig. 6 Negotiation between cell agent and scheduler machine agent for updating new schedule

in the architectural design is to identify the agents existing within the system and create an agent acquaintance artifact. After defining the agents, in a detail design stage, the interactions between these agents can be defined. This is done with an interactive diagram and a protocol diagram. Additional interaction diagrams are produced for each scenario represented in the analysis overview diagram. The interaction and protocol diagrams describe the scheduling problem that must be coordinated among the participants in the process. It should be noted that the following artifacts produced using the PM were not produced in the first attempt [32]. Therefore, the artifacts and descriptions in this paper are the results of several iterations over the same problem; each iteration refines the design until an acceptable solution was obtained.

3 Case study and design of the proposed multi-agent DSS

Yaran Bahar Golestan (YBG) is a small enterprise that produces make-to-order unplasticized polyvinyl chloride (uPVC) doors and windows by using automated machines. YBG is located in the north of Iran and provides doors and windows mainly to the internal market orders. The company has two main departments: the manufacturing support and management department located in a downtown and an MFL located a few kilometers away in the industrial area. The manufacturing support and management department is in charge of the design, production planning and scheduling, and marketing of the products. Moreover, finance and administrative sections are included in this department. The production process of the MFL involves the production of the frames of windows/doors and several assemblies in addition to the test and quality control phases. Figure 2 shows the layout of the uPVC part of the MFL.

The window components, such as fittings, profiles, and glasses, are provided by partner companies, according to the window/door design specifications. The window frames are manufactured in the MFL. Nearly, 15 models of doors and windows are under production: tilt and turn windows, slide hung, top light, sliding–folding, center hinge/pivot, etc.

The problems with the current scheduling and control architecture, which can be potentially improved by using multi-agent-based dynamic decision-making, are as follows:

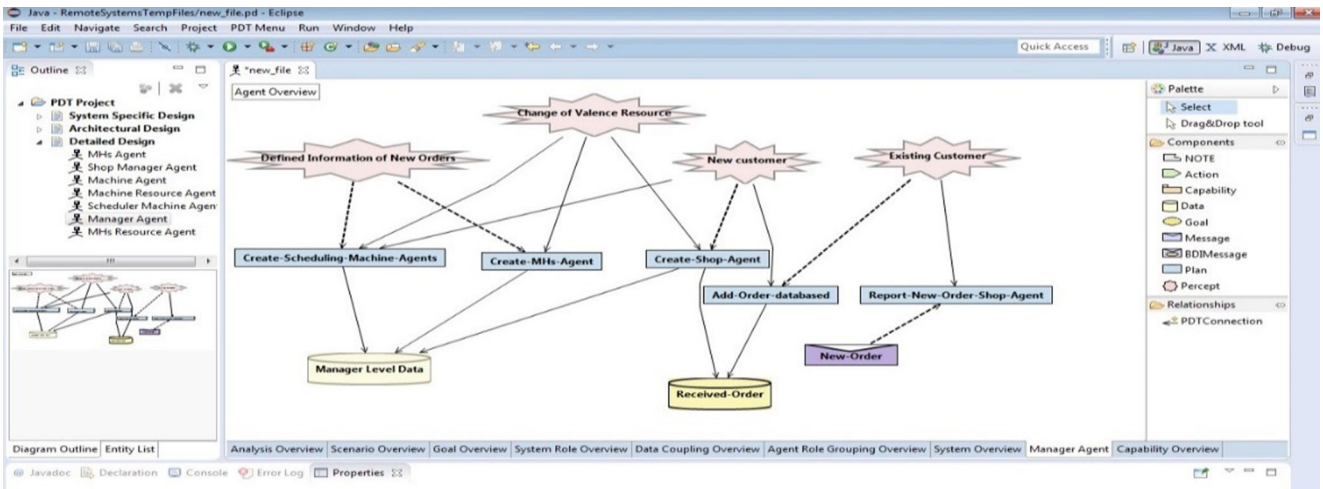

Fig. 7 Manager agent architecture

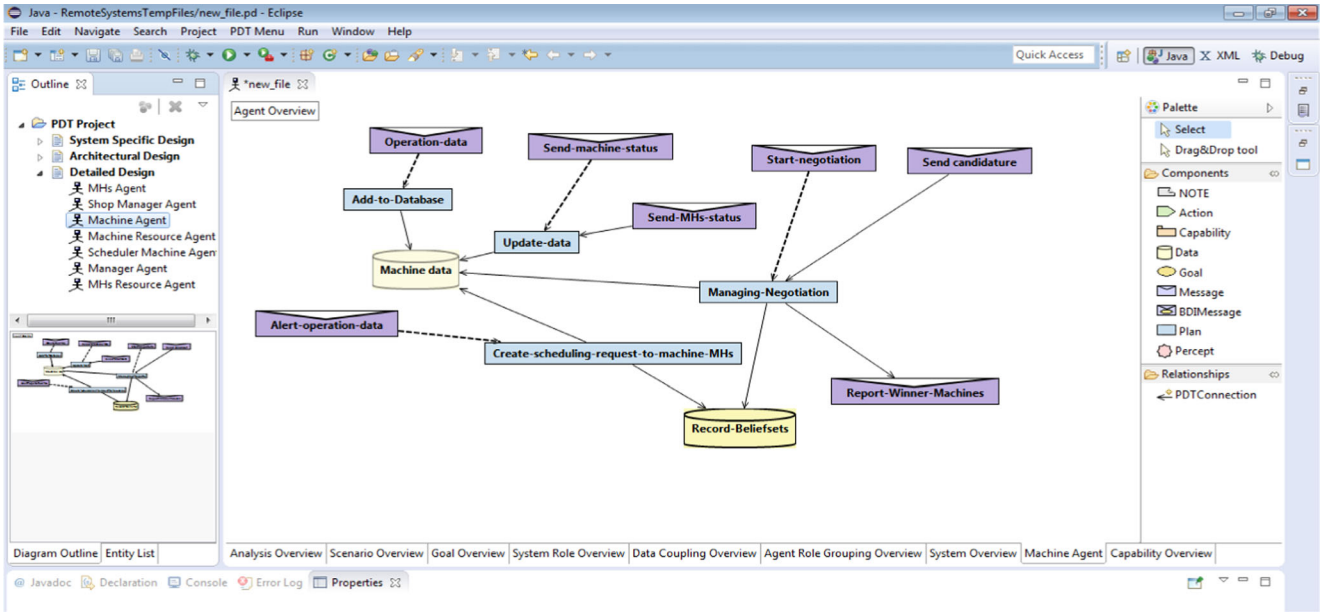

Fig. 8 Detailed design of cell agent

- The manufacturing system is scheduled by using static scheduling in the manufacturing support and management department; thus, all the decisions are taken by this unit.
- The stations (machines) have no autonomous scheduling unit for their operations.
- The system lacks real-time scheduling and is not flexible in the case of dynamic customer demands.
- The scheduling system is not reactive to internal disturbances of the system.

The development of a multi-agent-based DSS to address these problems is justified as follows:

- When the dynamic customer demands accrue, the dynamic decision-making system can schedule the system in a dynamic manner.
- The development of a multi-agent-based dynamic decision-making system can be found optimal when scheduling during a machine fails disturbance.

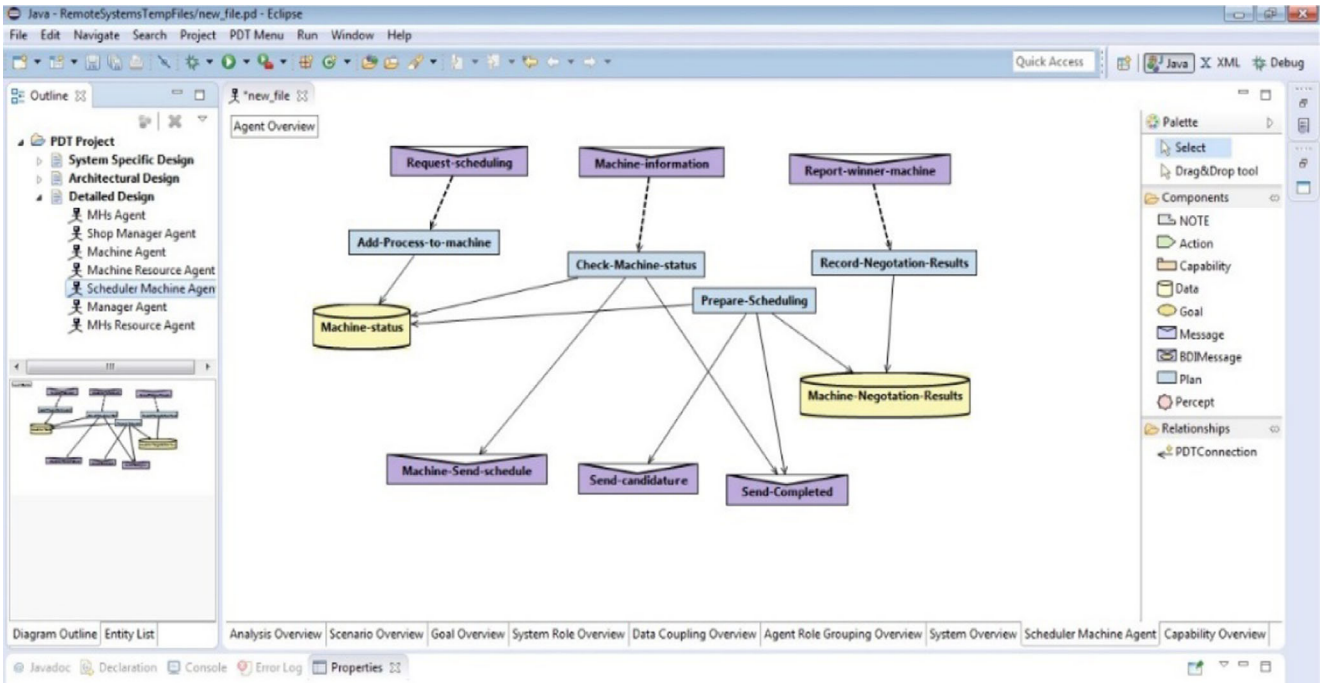

Fig. 9 Detailed design of scheduler machine agent

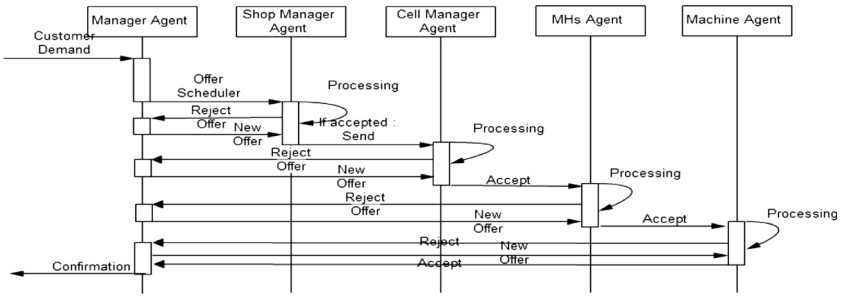

Fig. 10 Sequence diagram of decision-making mechanism

- The proposed system makes autonomous station level scheduling.
- The proposed MAS communicates with the system in real time.

3.1 System specification design

The system specification phase is the first part of the PM. The system specification design phase consists of four sub-phases: analysis overview, scenario overview, goal overview, and system role overview. System goals are specified in the goal overview diagram, resulting in a list of goals and sub-goals with associated descriptors. This phase is responsible for the identification of system goals, development of a set of scenarios that have adequate coverage of the goals, identification of functionalities linked to one or more goals, negotiation among the types of agents, and determination of the scenarios of the system. Figure 3 shows the goal overview diagram of the system.

The scenario overview phase was developed by a set of scenarios having an adequate coverage of the goals, providing a process-oriented view of the system to be developed. The system role overview defines a set of functionalities linked to one or more goals and captures a piece of the system behavior. Figure 4 shows the system role overview in which there are four main roles: manager role, shop management role, cell role, and negotiation management role.

The sub-goals are also designed in the system specification stage. For example, four sub-goals of machine scheduling after the arrival of unpredictable orders are defined: the machine is busy and has a task, the machine is free and has a task, the machine is free and has no task, and the machine is loaded and has no task.

3.2 Architecture design

This stage identifies the types of agents according to the PM in which the roles of the agents in the system are determined. This phase consists of three parts: data coupling overview, agent role grouping overview, and system overview. The negotiation protocols for the agents are designed in this phase. A system overview diagram is illustrated in Fig. 5. All the agents are defined in this stage: manager agent, shop manager agent, cell agent, MHS agent, scheduler machine agent, MHS resource agent, and machine resource agent. The last two agents are interface agents, and the other five agents are software agents used for the dynamic scheduling decision-making system. The proposed system follows a top-down approach by considering the real-time negotiation between all types of agents. The negotiation protocols of the agents are shown in Fig. 5 using the arrows. Protocols consist of an order protocol, shop protocol, MHS negotiation protocol, machine negotiation protocol, resource protocol, and machine resource protocol.

In order to describe all the interaction protocols, we developed the interaction protocols depicted by using the agent UML (AUML). Figure 6 shows an example of the negotiation protocol corresponding to the cell agent, scheduler machine agent, and MHS agent. This shows the negotiation between the cell agent and the scheduler machine agent for updating a new schedule in the machine and the concurrent communication with the MHS for transferring the material to the machine. This communication between the machine agent and the MHS agent is initiated by an MHS negotiation protocol. These negotiation protocols are coded in the Prometheus™ software.

3.3 Detailed design

In this stage, a detailed design is developed for each type of agent. The agents receive messages from the main platform event of their environment or other agents, which operate on their plans; thus, they act according to the records in their database.

For example, the manager agent manages the customers and updates the new orders to the system. The manager agent

Table 1 Mapping Prometheus modeling concepts into JACK concepts [34]

Prometheus entity	JACK concept
Agent	Agent
Capability	Capability
Precept	Event
Plan	Plan
Data	Belief Set
Action	–

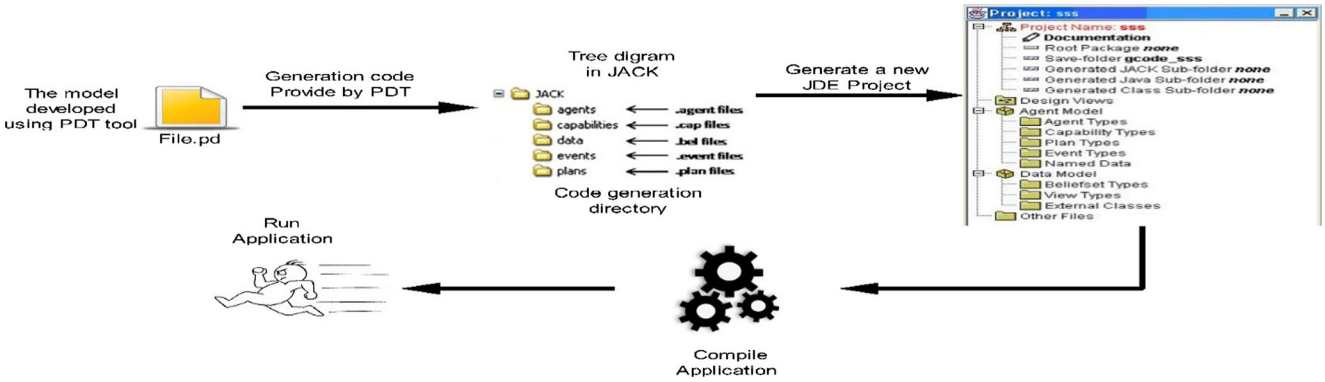

Fig. 11 Code generation process

uses its belief sets, plans, and message events to accomplish this task. The architecture of the manager agent is shown in Fig. 7 in the form of a Prometheus™ design view.

The cell agent manages and controls the cell level of a factory, and this agent consists of two subagents namely the MHS agent and the scheduler machine agent. The detailed design of this agent is shown in Fig. 8.

The other agent playing an important role in the rescheduling and dynamic scheduling of the cell level is the scheduler machine agent. This agent consists of two databases: machine status and machine negotiation results. The detailed design of this agent is illustrated in Fig. 9.

3.4 Decision-making mechanism or rescheduling

An algorithm for rescheduling the system for dynamic customer demands is proposed in this section. Figure 10 illustrates the sequence of the decision-making mechanism in the proposed MAS. The manager agent informs a new or unpredictable order to the shop manager agent. The shop manager agent sends the related questions to the cell agent, and this agent sends the questions to the scheduler machine agent and the MHS agent. The scheduler machine agent communicates with the machine resource agent in real time and sends the related information to the cell agent. This agent by

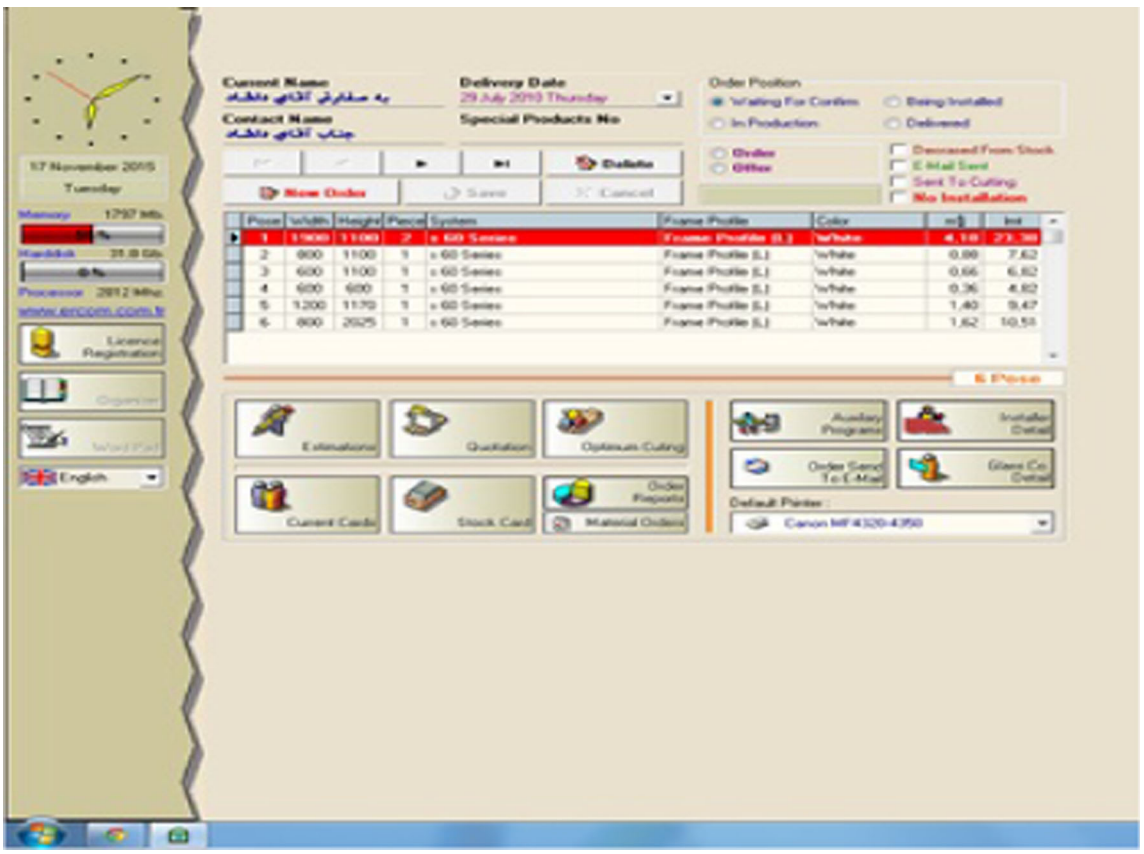

Fig. 12 Manager Agent

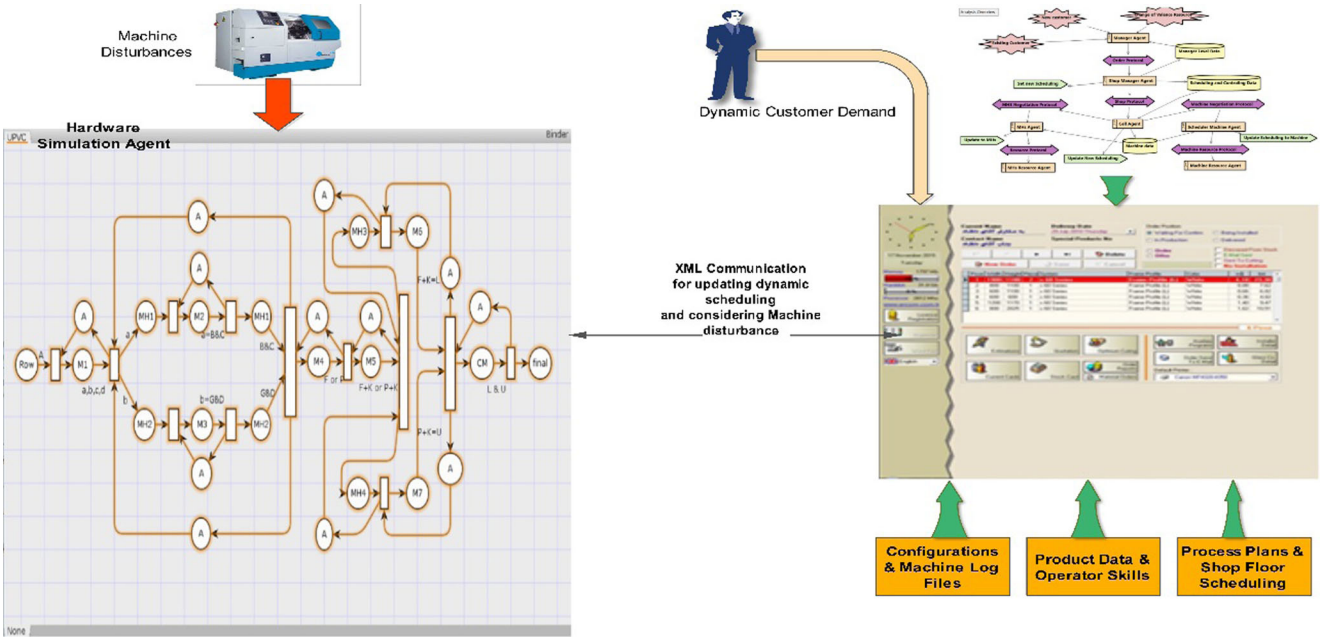

Fig. 13 Simulation test platform [26, 36]

considering the information from the scheduler machine agent answers the questions posed by the shop manager agent. The shop manager agent by considering the information from the cell level takes a decision and informs to the manager agent. If the manager agent confirms this decision, it will send the related information to the shop manager agent. The shop manager agent creates a new schedule and a new subagent and sends them to the cell agent and the MHS agent. The cell agent sends the new data to the scheduler machine agent, and this agent updates the new schedule to the machine.

4 Implementation

The generation code and implementation software were started manually from the design stage. This makes it possible to diverge the design and implementation stages [33] and generates a gap between them [34]. To bridge the gap, a methodology introducing refined design models that can be directly implemented in an available programming language should be used. The PM follows this approach, which is an advantage of this methodology. The last stage (the detailed design phase) of this methodology offers the models sufficiently close to the concepts used in a specific agent-oriented programming language named JACK [35]. Hence, the entities obtained during the design can be directly transformed into the concepts used in JACK. Table 1 shows the Prometheus entities being translated into their equivalent JACK concepts. It should be noted that some entities (actor, goal, protocol, role, and scenario) are not transformed into JACK concepts. The action concept is not transformed into a JACK-specific concept, but it can be implemented in the associated agent as a method.

Figure 11 illustrates a systematic method for the code generation process. This process generates a code by using the Prometheus design tool (PDT) and converts this code into a JACK concept. The user can press the generate button in the code era catalog (JACK) to generate a JACK folder, which contains several subfolders (agents, capabilities, data, events, and plans), automatically. The same occurs for the capability, data, message, and plan entities created in the model except for the file extension and folder, which are sorted as depicted in the tree diagram of JACK in Fig. 11.

A JACK developer environment (JDE) was used to import the code generated by the PDT. In this process, according to [34], five steps are followed: (a) the compiler utility submenu available in the tool menu is chosen in the JDE; (b) the Convert Non-JDE JACK is selected for converting the existing JACK code; (c) the folder that contains the code generated by the PDT is introduced into a content list; and (d) after

Table 2 Impact factor for calculating number of squares

Product	Impact factor
Window type A, B, C, D 0 m^2 < size < 2.2 m^2	1
Door type A, B 0 m^2 < size < 2 m^2	1.2
Door type C 0 m^2 < size < 2 m^2	1.3
Window type A size > 2.2 m^2	1.15
Window type B size > 2.2 m2	1.25
Window type C, D size > 2.2 m^2	1.45
Door type A size > 2.2 m^2	1.35
Door type B size > 2.2 m^2	1.45
Door type C size > 2.2 m^2	1.65

Table 3 Customer demand for July in YBG Company

Day	Window	Door	Window A, B, C, D 0 < size < 2.2	Window A size > 2.2	Window B size > 2.2	Window C, D size > 2.2	Door A, B 0 < Size < 2.2	Door C, 0 < size < 2.2	Door A size > 2.2	Door B size > 2.2	Door C size > 2.2	Number of squares
1	220	260	100	60		60	50	50	60	50	50	617.00
2	130	20	50	40		40		20				180.00
3	170	30			100	70			30			267.00
4	0	0										0.00
5	400	150	200		200		50	50	50			642.50
6	0	0										0.00
7	220	40	220				40					268.00
8	230	150		80	50	100				50	100	537.00
9	220	130	100	120			130					394.00
10	430	70	230	200						70		561.50
11	0	0										0.00
12	560	0	100	240	220							651.00
13	480	50		120	100	260	50					700.00
14	440	160	150	150		140			160			741.50
15	460	40	60	200	200		40					588.00
16	290	280	20	20	50	200	150	130				744.50
17	0	0										0.00
18	470	200	10	100	100	260	50	150				882.00
19	0	730					500	230				899.00
20	70	320	70				320					454.00
21	140	440			100	40			400	40		781.00
22	150	350	50	100				100		200	50	667.50
23	170	400	70		100			100	100	100	100	770.00
24	0	0										0.00
25	0	340						100	140	100		464.00
26	0	340						140	100	100		462.00
27	140	160	40	100			100	60				353.00
28	0	410					410					492.00
29	140	440	80	10		50	440					692.00
30	70	43	70					100	100	100	130	694.50
31	0	0										0.00
									Total	14,503.00		

Table 4 Result of simulation and conventional system for accepted parts

Performance measures	Conventional DSS	Multi-agent-based DSS
Total acceptance rate for dynamic demand (%)	70.3	79.8
Makespan C_{max}	204.53	182.16

defining the address and the folder name, the generate button is pressed, and the new JDE project will be obtained.

Presently, the inside structure of the documents and their augmentations are distinctive with a specific end goal readable by the JDE. Lastly, after generating a few Java classes and finishing the generated code, the JACK program can be easily transferred into Java using the facilities provided by the JDE and executed. Figure 12 illustrates the manager agent that sends jobs to other agents. By using the customer demands, the manager agent creates a list of operations. In addition, the available schedule is chosen from the list. When a schedule is created, the selected order information is sent to the shop manager agent. After the shop manager agent accepts this schedule, a new order is sent to the cell manager agent. The cell manager agent cooperates with the required equipment, which provides transportation, and raw materials and communicates with the machine agent and the MHS agent, which help in the completion of the work during execution.

4.1 Simulation platform

In this section, the simulation test platform for evaluating the proposed MAS is linked to the developed MAS. This effort helps find a result very close to the real implementation because the hardware and software are considered simultaneously. The linked system is illustrated in Fig. 13. This platform contains two main modules:

(a) Hardware simulation agent module—this belongs to the CPN model of the system, and it is used to analyze the behavior of the company.
(b) The proposed MASs—it is related to the scheduling and control architecture of the system.

The hardware simulation characterized the physical actions that occur in the manufacturing environment such as machines, robots, and MHSs. The hardware was modeled using CPN. The communication module was developed based on an XML tool. The multi-agent-based scheduling and control system was used to send and receive information using a CPN tool [36] via the communication module. In the simulation platform, the manager agent was used to create dynamic customer demands and send them to the system for evaluation, thus, solving the problem by considering the hardware simulation agent. A machine disturbance was created in the simulation hardware agent, and the information regarding this disturbance was sent to the MAS via the XML-based communication module. The DSS was used to create and propose a new schedule and send it to the hardware simulation agent.

5 Test scenarios and result

This section discusses two scenarios for evaluating the proposed MAS in the MFL. The first scenario evaluates the response of the system in terms of the dynamic customer demands, and the second scenario focuses on the quantitative indicators based on various production performance measures, such as lead time and throughput, for internal disturbances. To compare the current state and the future state, the approach introduced in [37] is used to quantify the degree of improvements. The number of units produced is considered using the number of squares. Generally, the performance of the manufacturing systems is measured in units/hour. As per this convention, the performance in the case of an MFL should be measured in terms of the number of windows/doors produced per hour or shift. However, in this case, the size and complexity of windows/doors differ considerably. Hence, this unit of measurement may not fairly reflect the performance of the MFL. To overcome this problem and to establish a normalized production rate and productivity, the number of squares is used. Table 2 presents the impact factor for calculating the number of squares of the products. As examples, a door type B with 1.8 m^2 size is considered as 2.25 m^2 and a window type A with 3.4 m^2 size is considered as 3.91 m^2.

Table 5 Summary of lead-time experiment results

Lead time						
Scenario	Type	SD	Mean	LCL 95 %	UCL 95 %	CV%
2.1	Conventional	0.85	10.32	9.45	11.19	8.2
	MAS	0.56	9.45	8.64	10.26	5.9
2.2	Conventional	3.457	17.26	15.894	18.626	20.02
	MAS	2.24	13.756	12.49	15.022	16.2

Table 6 Summary of throughput experiment results

Throughput						
Scenario	Type	SD	Mean	LCL 95 %	UCL 95 %	CV%
2.1	Conventional	57.4	1211	1191	1231	4.73
	MAS	48.48	1259	1251	1267	3.85
2.2	Conventional	109.7	803	711	895	13.66
	MAS	89.6	980	908	1052	9.14

In the simulation, the setup time was not considered, and it was assumed that a failure of negotiation would never occur. A robot and a conveyor were used for transportation, and the orders were queued in the order of arrival. Each transport action required 1 min, and the average processing time was equal to 20 min. The tests considered the data corresponding to the month of July. The adopted unit of time was 1/50th of a minute as in the standard time data. Using the Welch method, a warm-up period of 20 h was used to fill the machine queues and obtain steady state results.

In the first scenario, the company categorized the customers into two groups. Group one includes the customers who order a high volume of products with long due dates, whereas group two includes the customers who order a small list of products with short due dates. The total daily production capacity of the company is 22,300 squares. The competitive strategy of the company for the month of July is providing at least 10,000 squares for group one of the customers. This article focuses on providing decisions for the second group of customers, which deals with the dynamic demands. For sketching this, the data related to July are used and the results of both the current and newly developed systems are compared. Normally every day, 5–10 requests with different due dates are received from clients. The company uses the due dates and the number of squares in deciding whether to accept or reject the requests. Table 3 lists the customer demands corresponding to the month of July and the number of squares.

In this work, the comparison between the scheduling approaches considered number of squares in a batch of orders in July, aiming at reducing the makespan (C_{max}). After running the test platform for the proposed MAS by considering the warm-up time, the dynamic demand was sent to the scheduling system. Table 3 summarizes the total accepted number of squares for this month and makespan achieved for each scheduling system. We have to mention that both the systems producing windows have priority to produce doors. Therefore, the system tries to finish the windows before the door. It starts producing doors after finishing windows.

Based on the experimental results of the first scenario, the best C_{max} was founded by the proposed DSS. The rate of acceptance of the multi-agent-based DSS is equal to 0.798, and the rate of acceptance of the conventional system in July is equal to 0.703. This rate shows that more products can be produced by using the MAS than the conventional system because the MAS can reschedule in the dynamic model (Table 4).

The second scenario focuses on the quantitative indicators, namely, lead time, throughput, and resource utilization, for the case of internal disturbances. For evaluating the performance of the system, two sub-scenarios are considered: a well-functioning system with no disturbances and a 20 % probability of a failure occurring in the profile-welding machine. The probability distribution for the failure time was regarded as an exponential distribution with a rate of 25. A Weibull distribution with a total availability of 85 % was used for the repair time with the mean repair time as 60 min.

The experience gained from the simulation debugging and testing allowed us to draw some conclusions concerning the operation of the proposed MAS in the MFL. The system was found to function robustly in accordance with the specifications for both the normal operation and in the presence of disturbances. Furthermore, the reconfigurability of the system was demonstrated by its accurate reactions to the introduction, removal, and modification of manufacturing components. The average values of standard deviations (SD) and the coefficient of variation (CV) for each subtest scenario are presented in Tables 5, 6, and 7.

Table 7 Summary of resource utilization experiment results

Resource utilization						
Scenario	Type	SD	Mean	LCL 95 %	UCL 95 %	CV%
2.1	Conventional	0.039	0.871	0.801	0.941	4.47
	MAS	0.031	0.962	0.92	1.004	3.22
2.2	Conventional	0.054	0.69	0.774	0.848	7.82
	MAS	0.044	0.921	0.895	0.974	4.77

The first sub-scenario of scenario two is the system operated predictably and has no disturbance. In this scenario, the proposed MAS presents smaller values of manufacturing lead time (9.45), higher values of throughput (1259), and higher values of resource utilization (0.962) than the conventional system (10.32, 1211, and 0.871, respectively). The better performance of those systems is a result of the cooperation of the autonomous entities.

The second sub-scenario of scenario two is the experimental test considering the occurrence of unexpected disturbances in the welding machine. It is obvious that the performance indicators degrade in the presence of disturbances. From the analysis of lead time, throughput, and resource utilization, it can be verified that the proposed MAS offers a better performance than the conventional system.

6 Conclusion and future research

The dynamic rescheduling method is widely used in the modern production plants. This study attempts to solve the scheduling problems of the MFL by using a multi-agent-based DSS. The proposed system is designed and developed in order to solve scheduling complexities during a dynamic order change and occurrence of internal disturbances in the MFL. The design uses the capabilities of the MAS in order to solve real-time scheduling complexities. Feasible and effective schedules are expected from negotiation/bidding mechanisms between agents. In this study, we tried to clarify the problems of an MFL and how the MAS can be helpful for SMEs. The MAS scheduling and control system is designed based on the PM and implemented in the JACK platform. The simulation platform based on a hybrid agent is used for simulation and testing. This platform considers both the software (MAS) and hardware of the system. A real case study was used for simulation, and the results indicate that the proposed multi-agent scheduling system outperforms the conventional approach as well as the dispatching-based production control approach used in practice. Furthermore, the proposed system performs better in terms of the running time because the MAS scheduling system can take immediate actions to reschedule tasks in the event of high failures. The developed method offers three advantages: (a) the dynamic order behavior and the capability to reconfigure the system with respect to the internal disturbances are considered simultaneously; (b) the system is developed based on general-purpose design methodology (i.e., Prometheus methodology™) and is not tied to any specific model in the software platform; and (c) for modeling the internal disturbances of the system, a simulation test platform is linked to the developed multi-agent-based DSS. The use of the developed system needs moderate knowledge on modeling manufacturing systems by Petri nets; this concern might be considered as disadvantage of this approach.

References

1. Young RE, Vesterager J (1990) An approach to implementing CIM in small and medium size companies. International Journal of NIST Special Publication 785:63–79
2. Levy M, Powell P (2000) Information systems strategy for small and medium sized enterprises: an organisational perspective. J Strateg Inf Syst 9:63–84
3. Bai D, Zhang Z-H, Zhang Q (2016) Flexible open shop scheduling problem to minimize makespan. Comput Oper Res 67:207–215
4. Vatankhah Barenji R, Hashemipour M, Guerra-Zubiaga DA (2015) A framework for modelling enterprise competencies: from theory to practice in enterprise architecture. Int J Comput Integr Manuf 28: 791–810
5. Monostori L, Váncza J, Kumara SR (2006) Agent-based systems for manufacturing. CIRP Ann Manuf Techn 55:697–720
6. M. Paolucci and R. Sacile (2016) Agent-based manufacturing and control systems: new agile manufacturing solutions for achieving peak performance: CRC Press
7. Lu SH, Kumar P (1991) Distributed scheduling based on due dates and buffer priorities. IEEE Trans Autom Control 36:1406–1416
8. A. V. Barenji, R. V. Barenji, and M. Hashemipour (2013) Structural modeling of a RFID-enabled reconfigurable architecture for a flexible manufacturing system, in *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2013 European Conference on*, pp. 1–10
9. Barenji AV, Barenji RV, Hashemipour M (2014) A frameworks for structural modelling of an RFID-enabled intelligent distributed manufacturing control system. S Afr J Ind Eng 25:48–66
10. Shen W, Hao Q, Yoon HJ, Norrie DH (2006) Applications of agent-based systems in intelligent manufacturing: an updated review. Adv Eng Inform 20:415–431
11. Zhong RY, Huang GQ, Lan S, Dai Q, Zhang T, Xu C (2015) A two-level advanced production planning and scheduling model for RFID-enabled ubiquitous manufacturing. Adv Eng Inform 29: 799–812
12. Kerzner HR (2013) Project management: a systems approach to planning, scheduling, and controlling. John Wiley & Sons, Hoboken
13. Yoon HJ, Shen W (2008) A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints. IEEE Trans Semicond Manuf 21:83–91
14. M. Pinedo (2015) Scheduling: Springer. doi: 10.1007/978-3-319-26580-3
15. Ramamritham K, Stankovic JA (1994) Scheduling algorithms and operating systems support for real-time systems. Proc IEEE 82:55–67
16. H. J. Yoon and W. Shen 2005 Agent-based scheduling mechanism for semiconductor manufacturing systems with temporal constraints, in *IEEE International Conference Mechatronics and Automation*, pp. 1123–1128
17. Caridi M, Cavalieri S (2004) Multi-agent systems in production planning and control: an overview. Prod Plan Control 15:106–118
18. Gibson MR, Ohlmann JW, Fry MJ (2010) An agent-based stochastic ruler approach for a stochastic knapsack problem with sequential competition. Comput Oper Res 37:598–609
19. R. Smith, The contract net protocol: highlevel communication and control in a distributed problem solver, 1980, *IEEE Trans. on Computers, C*, 29, 12
20. Valckenaers P, Van Brussel H (2005) Holonic manufacturing execution systems. CIRP Ann Manuf Techn 54:427–432
21. Kaplanoğlu V (2014) Multi-agent based approach for single machine scheduling with sequence-dependent setup times and machine maintenance. Appl Soft Comput 23:165–179

22. Chen K-Y, Chen C-J (2010) Applying multi-agent technique in multi-section flexible manufacturing system. Expert Syst Appl 37: 7310–7318
23. L. Padgham and M. Winikoff (2002) Prometheus: a methodology for developing intelligent agents, in *International Workshop on Agent-Oriented Software Engineering*, pp. 174–185
24. L. Padgham and M. Winikoff (2002) Prometheus: a pragmatic methodology for engineering intelligent agents, in *Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, pp. 97–108
25. Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. Int J Adv Manuf Technol 71:1773–1791
26. Barenji AV, Barenji RV, Hashemipour M (2016) Flexible testing platform for employment of RFID-enabled multi-agent system on flexible assembly line. Adv Eng Softw 91:1–11
27. Baykasoglu A, Gorkemli L (2016) Dynamic virtual cellular manufacturing through agent-based modelling. Int J Comput Integr Manuf:1–16
28. Sahin C, Demirtas M, Erol R, Baykasoğlu A, Kaplanoğlu V (2015) A multi-agent based approach to dynamic scheduling with flexible processing capabilities. J Intell Manuf:1–19
29. Renna P (2010) Job shop scheduling by pheromone approach in a dynamic environment. Int J Comput Integr Manuf 23(5):412–424
30. Padgham L, Winikoff M (2005) Prometheus: a practical agent-oriented methodology. Agent-oriented methodologies:107–135
31. L. Padgham, J. Thangarajah, and M. Winikoff (2007) The Prometheus design tool—a conference management system case study, in *International Workshop on Agent-Oriented Software Engineering*, pp. 197–211
32. L. Padgham and M. Winikoff (2005) *Developing intelligent agent systems: a practical guide* vol. 13: John Wiley & Sons
33. R. H. Bordini, M. Dastani, and M. Winikoff (2006) Current issues in multi-agent systems development, in *International Workshop on Engineering Societies in the Agents World*, pp. 38–61
34. Gascueña JM, Fernández-Caballero A (2011) Agent-oriented modeling and development of a person-following mobile robot. Expert Syst Appl 38:4280–4290
35. M. Winikoff (2005) JACK™ intelligent agents: an industrial strength platform, in *Multi-Agent Programming*, ed: Springer, pp. 175–193
36. A. Shaygan and R. V. Barenji (2016) Simulation platform for multi agent based manufacturing control system based on the hybrid agent, *arXiv preprint arXiv:1603.07766*
37. Gurumurthy A, Kodali R (2011) Design of lean manufacturing systems using value stream mapping with simulation: a case study. J Manuf Technol Manag 22:444–473