Investigating the Robustness of Re-Scheduling Policies with Multi-Agent System Simulation

Munir Merdan, Thomas Moser, Pavel Vrba, and Stefan Biffl

Increased complexity of current manufacturing systems together with dynamic conditions and permanent demands for flexible and robust functionality makes their management and control very difficult and challenging. Workflow simulation is an effective approach to investigate dynamic workflow scheduling policies and evaluate the overall manufacturing system performance. The results attained in simulation model can give directions how to maximize system output when selecting an appropriate scheduling practice for a real system. In this paper we investigate the abilities of multi-agent systems (MAS) in combination with dynamic dispatching rules and failure handling mechanisms to manage dynamic environment conditions (such as machine failures) for systems in the production automation domain. We measure system robustness by systematically assessing the total system performance (e.g., number of finished products) in a number of representative test cases. We use an agent-based simulation environment, MAST, which has been validated with real-world hardware to strengthen the external validity of the simulation results. We investigated the performance of a re-scheduling component which uses four different policies that define how to adjust the system schedule in case of machine disturbances/failures. In the context of the empirical study the *Complete Rerouting (CR)* re-scheduling policy outperformed all other policies.

Index Terms— System evaluation, re-scheduling policies, multi-agent systems, automated simulation, distributed control.

1. Introduction

Modern manufacturing systems have to face dynamic conditions during the production process: Machine breakdowns or disturbances, task priority changes, integration of new resources, order cancellations, unequal machine utilization rates, and product quality problems illustrate the many exceptions that can influence the system performance and typically occur unpredictably. If the system is able to handle remain reliable in improper or stressful environments, e.g., if it is able to operate correctly in the presence of invalid inputs., it is called robust [9]. Robust systems should exhibit proportional degradation of service (e.g., throughput) according to the class of problem that occurred. In this context, the robust scheduling tends to

Munir Merdan is with the Automation and Control Institute, Vienna University of Technology, Austria. (phone: +4315880137656; fax: +4315880137698; e-mail: merdan@acin.tuwien.ac.at).
Thomas Moser is with the Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, Vienna University of Technology, Austria. (e-mail: thomas.moser@tuwien.ac.at).
Pavel Vrba is with the Rockwell Automation Research Center Prague, Czech Republic (e-mail: pvrba@ra.rockwell.com).
Stefan Biffl is with the Institute for Software Technology and Interactive Systems, Vienna University of Technology, Austria. (e-mail: stefan.biffl@tuwien.ac.at).

keep the system performance high in the presence of disruptions [33]. Robustness implies that alternatives (e.g., a back-up machine) can be involved to compensate disturbances [42]. The scheduling of production resources is one of the key features in the current competitive and dynamic manufacturing environment. It determines the most suitable time slot to produce something. Its main objectives are the minimization of the production time of jobs, production costs, increased resource utilization, etc. A production schedule is considered as robust in case it performs well after a disruption (e.g. makespan, number of tardy jobs, etc.) [34][40]. There is a range of failure handling policies to respond to exceptions and thus to improve system robustness. Traditional centralized hierarchical manufacturing systems, due to their rigid structure and lack of flexibility, suffer from weak robustness and failure tolerance, i.e., they are not able to handle exceptions effectively and efficiently and therefore stop working or produce fewer products. Moreover, in these traditional systems all possible combinations of exceptions have to be predicted at design time, otherwise their occurrence at operation time can lead to scheduling errors and significant downtime losses [2]. Nevertheless, in complex systems the number of combinations of relevant exceptions grows exponentially, which makes system re-scheduling and modification very expensive or time consuming [1].

Production automation systems consist of many entities (like robots, machines, shuttles) that interact in a complex manner to provide the overall system functionality like product assembly. The role of scheduling within these systems is the synchronization of production needs with available resource capacities. The application of multi-agent systems (MAS) based on decentralized control architecture has been suggested as a promising approach for overcoming the deficiencies of inflexible centrally controlled systems [36]. MAS can help simulate the effects of production policies on coordination of the behavior of entities in a distributed production automation system [3]. Another major advantage is that the same MAS technologies can be used both for the software simulation as well as for the simulation using real miniature hardware (such as in the Odo Struger Lab at the Automation Control Institute at the Vienna University of Technology), allowing direct comparisons between software and miniature hardware simulation.

MAS can be defined as a network of autonomous and intelligent entities – agents – where each agent has individual goals, capabilities, and problem-solving behaviors. The agent supervises one or more manufacturing entities of either physical (e.g., machine, tool) or functional (e.g., order, product, task) character [4]. The agent is responsible for monitoring and controlling the functionality, respecting planning and scheduling aspects at the same time. Agents can also cooperate and coordinate their actions to recover from exceptional states and avoid a system shutdown. In general, the introduction of agent-based techniques can bring advantages such as heterogeneity, modularity, scalability, parallelism, flexibility, and robustness against failures; making them especially appropriate for building complex systems [5, 6, 7].

A. Related Work

A distributed control system is more robust than a centralized control system as this approach consists of redundant decision components, i.e., the loss of one decision component will not necessarily stop the system or cause a fatal failure in other decision components. Moreover, in case of the production reorganization, the same negotiation process continues to be executed, in spite of the presence of different machines and products, making the system robust to the change. Disturbances are treated by a re-scheduling component which uses re-scheduling policies without stopping or re-initializing the process that is currently running in the system [35]. Nevertheless, a blind application of the MAS approach to increase fault tolerance of a particular system can lead to the opposite result [1]. The lack of predictability concerning the outcome of the agents' interactions and the behaviour of the overall system due to the emergent behaviour of the system's components is the major drawbacks of multi-agent systems [41]. The handling policies give the agents some kind of global perspective and methodology how to act in the case of disruptions. Therefore, it is necessary to empirically investigate MAS performance under dynamic conditions, when agents use a range of promising behaviors and apply diverse handling policies in order to cope with system exceptions. The multi-agent paradigm enables describing and specifying the organization and the social behavior of the analyzed system by providing means to transform the system into simulation models and to allow testing the efficiency of related behaviors, like e.g., scheduling policies [38].

The simulation of the system design with realistic workshop scenarios is the foundation to ensure logical correctness of the system behavior before conducting expensive lab and field operations. A major advantage of simulation is the ability to speed up the simulated process compared to the real-time behavior in the real-world environment. This helps shorten the testing duration by up to an order of magnitude [20]; further, simulations may be run in parallel to increase the number of scenarios to be explored in a limited time frame. Consequently, statistical data analysis of the simulation results allows analyzing the impact of design and planning factors on system performance and answer fundamental questions on planning and coordination strategies in the production automation domain such as the impact of a variety of fundamental scheduling strategies. The results of these data analyses can then consequently be used to improve the engineering of production automation systems.

Considering the re-scheduling of production resources as one of the key features of production control, it is vital to examine the influence of re-scheduling strategies on production effectiveness, allowing better systems engineering by providing support for human decision making. Failure re-scheduling policies specify the overall tactics that define *when* and *how* the system has to cope with failure events. Several handling policies for task re-allocation carried out in case of extraordinary events (e.g., machine breakdowns) are explored in this study. In our previous work, we evaluated a range of workflow scheduling strategies based on multi-agent negotiation, where each resource agent performs local scheduling using dispatching rules for sequencing the tasks allocated to their machines [19]. Dispatching rules are extensively used in manufacturing due to their simplicity, efficiency, and ability to react almost instantaneously while requiring only local information for scheduling [8]. Considering their advantages, the usage of dispatching rules for sequencing re-scheduled tasks after a specific handling policy is applied can reduce effects of a particular exception and improve system performance [13]. Kutanoglu and Sabuncuoglu used this approach to study four reactive re-scheduling policies – no rerouting, queue rerouting, arrival rerouting, and all rerouting – developed for rerouting the jobs to alternative machines when their primary machine fails. The investigated policies are reactive in a sense that there is no provision of uncertainty in the planning/scheduling stage [10]. On the other hand, proactive-reactive scheduling/re-scheduling is implemented as a two-step process. First, a predictive schedule is generated in advance with the objective to optimize the shop floor performance over the time horizon considered. This schedule is then modified during execution in response to unexpected disruptions. The proactive-reactive scheduling is the most common dynamic scheduling approach used in manufacturing systems [13, 27, 32].

Bastos et al. [11] presented a MAS architecture capable to support dynamic resource allocation planning in production environments. This architecture also can manage disturbances in the production system in real time by applying two strategies – replacement and re-scheduling. A market-based theory coordinates agent behaviors. Guo and Nonaka propose a rescheduling method by means of which different adjusted schedules are obtained corresponding to the different lengths of downtime of a machine failure [39]. Wu et al. [12] presented an algorithm for automatic sequential resource (re)allocation among a group of agents in complex environments with limited shared resources and with uncertainties. Vieira et al. [13] extensively studied the effects of re-scheduling policies on the performance of a manufacturing system. They concluded that the use of different model types, such as a mathematical model of dynamic and stochastic manufacturing systems, queuing network model or discrete event simulation model, can give useful information to analysts. However, the mathematical model does not explicitly represent the production control policies that will be actually used to control the system. Further research is required to compare the performance of manufacturing systems under diverse dynamic conditions to explain the advantages and disadvantages of re-scheduling strategies in different problem settings. Moreover, considering robustness as a one of the key factors to preserve the stability of the manufacturing systems in the presence of uncertainties, more research is needed towards the development of more general efficient robustness measures and rescheduling strategies [18].

Besides, the issue of how to manage the occurrence of real-time events, termed dynamic scheduling, needs to be considered since re-scheduling decisions have to be made based on the current state of the manufacturing environment. Further problem is to identify, for every customer order, the optimal routing and sequencing decision that satisfies the due date at the minimal cost, within the structure of the given system [37]. Dynamic scheduling of such environment requires real-time scheduling algorithms and their effective integration with the distributed shop floor control structure [30]. The combination of MAS technology and dispatching rules brings significant advantages, while related agents dispatch jobs when needed and use accurate information available at the moment of dispatching to prioritize jobs. Further important point of MAS applications is their reactivity: agents can locally react to local changes faster than a centralized approach could [28]. The effectiveness of the agent-based scheduling approach to handle dynamic system changes (e.g. machine breakdowns, rush orders) becomes even more evident when compared with other non-agent-based approaches [23]. In a distributed environment agents need only to "freeze" the part of their schedule and recalculate tasks that are related to the broken resource bringing the system into a new evolved state due to their separated actions [25]. Extensive surveys of dynamic scheduling [24] in the manufacturing environment considering also agent-based systems were done by Ouelhadj and Petrovic [27], Shen et al. [29], and Babiceanu and Chen [22]. Nevertheless, it is not only important to select a good scheduling strategy but also to be able to effectively reorganize the shop floor production plan and repair or redo the production schedule in response to unexpected events [30], also sometimes from a more global perspective [26].

B. Our Contribution

In this paper, we simulate real-life scenarios to test system performance in a dynamic environment. We evaluate the robustness and failure tolerance of the four previously mentioned reactive re-scheduling policies from literature, where agents negotiate to coordinate their actions and apply dispatching rules for local scheduling. The paper presents the conceptual innovation of designing established rescheduling strategies with a multi-agent based control architecture. Furthermore, the evaluation concept is based on a simulation approach that has been validated with the real-world behavior of the system (hardware) [15], compared to previous evaluations with mathematical/statistical models that were not validated with real-world system data. We additionally investigate how specific production conditions such as different levels of machine efficiency as well as duration of machine failures influence the performance of a re-scheduling policy.
In order to strengthen the external validity of our research results, we use the real-world pallet transfer system at the Automation and Control Institute, Vienna University of Technology, as a reference model for our MAS architecture. Our simulation model completely mirrors the pallet transfer system configuration including placements of machines, possible duration of their failures as well as length of conveyors and related transportation times. The complexity of the scheduling problem comes from the number of investi-

gated parameters (re-scheduling policy, number of pallets, machine failure recovery time, level of workload, and number of product types used) as well as from the number of agents that have to coordinate their actions through negotiations.
The remainder of this paper is structured as follows: Section 2 describes the applied MAS simulation architecture. Section 3 provides an overview on re-scheduling policies to mitigate the impacts of machine failures on the overall system performance. Section 4 identifies the research issues and introduces the evaluation hypotheses. Section 5 describes the details of simulation experiments and defines significant parameters for evaluation. Section 6 presents and discusses the results of the empirical experiments. Finally, Section 7 concludes and suggests further work.

2. Multi-Agent Simulation Architecture

In our MAS architecture for manufacturing control of assembly workshop, each component of the pallet transfer system such as a robot, diverter, junction, conveyor, pallet, and docking station is represented and supervised by a corresponding *Resource Agent (RA)*. We also introduce the *Order Agent* (OA), which is responsible for receiving product orders from customers and controlling their accomplishment. The OA decomposes the product order into work orders, where each work order represents a single machine operation (welding, drilling, painting, etc.) that has to be done to finish the product. Each work order is forwarded to the corresponding Product Agent (PA) that negotiates with related RAs that provide particular operations. The standard communication Contract Net Protocol [14] is used to allocate an operation to the best suited machine resource. Each PA monitors and follows the execution of a sequence of work orders defined for a particular product. To test and simulate the transport and negotiation between the agents, we have developed an extension of the Manufacturing Agent Simulation Tool (see Section 3) called the Test Management System (MAST-TMS) [19,20]. Figure 1 presents an overview of the systems architecture and the evaluation framework.

A. Manufacturing Agent Simulation Tool (MAST)

The Manufacturing Agent Simulation Tool (MAST) [16] [43] provides a unique combination of multi-agent based manufacturing control features and a simulator of the manufacturing environment used to verify the functionality of the agent-based control system.
The uniqueness of the solution is the application of a transparent control interface between the agent part and the simulation part enabling to switch over from simulation to real physical control. Thus, the agent control system developed for the simulation purposes is then reused completely for deployment to a real-

world manufacturing system. An important aspect of the proposed solution is the integration with the classical scan-based, real-time control executed on PLC (Programmable Logic Controller).

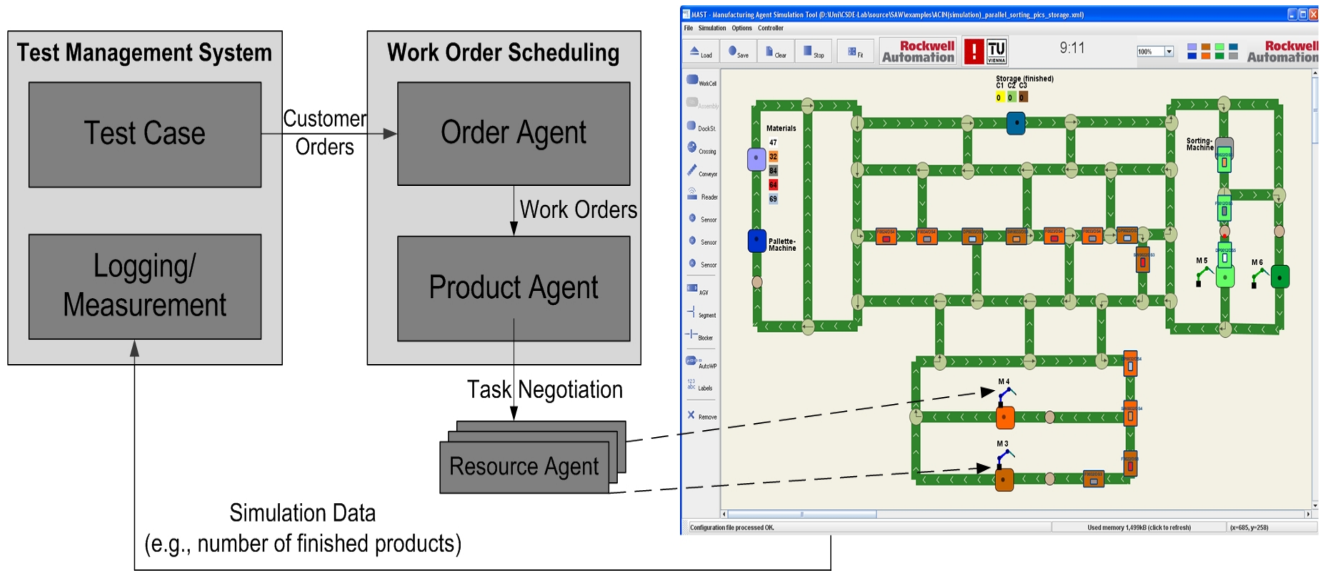

Figure 1: MAS Architecture and Evaluation Framework.

Such a solution simplifies the proliferation of this new multi-agent technology into the industrial automation domain because the existing, widely adopted industrial automation hardware and software architectures, like PLCs, operator panels and SCADA systems, can be reused.

The control interface is a central element of the MAST system architecture around which all the other parts are organized and through which they interact. It has been designed with the intent to be identical with a way how current PLC programs and visualization tools interact with the controlled manufacturing environment. Sensors and actuators are usually interconnected by an industrial network, like DeviceNet, and their respective values are transferred to the PLC memory via I/O cards. In the PLC memory, each I/O value as well as other additional parameters processed by the control programs are stored in variables called *tags*. Such a tag-based interface has been also deployed in the MAST system. It is used to exchange the I/O values between these main components: (i) the agent part, (ii) the simulation engine, and (iii) the visualization. In case of physical deployment when the simulation subsystem is disconnected the tag-based interface provides a link to the real I/O values of the physical manufacturing environment. Currently, it supports the data transfer from/to the commercial ControlLogix controller that, besides the I/O connectivity, executes the low-level control programs implemented in one of the IEC61131-3 PLC programming languages [15].

The agent subsystem provides the runtime environment for particular agents, where each agent instance represents and controls behavior of a specific manufacturing component. As the primary aim was the material handling tasks, the library of agent classes includes basic types of components like conveyor belt, diverter, docking station, etc. The essential characteristic is the cooperation of agents on various tasks, mainly routing of products through the conveyor network and dynamic reconfiguration of routing paths in case of machine break downs or modifications to the physical layout of the factory. There is no central decision making authority in the system – all the control logic and knowledge related to current state of the production process is fully distributed among the agents [16]. Each agent is aware only of its immediate neighbors and uses message sending as form of information exchange and cooperation. The programming language for agents as well as for the rest of MAST application is Java and the agent runtime environment is provided by the open source agent platform JADE.

The purpose of the simulation engine is to simulate the function and behavior of the physical system. Each simulated component contains virtual sensors that are activated by the movement of simulated products and virtual actuators that affect the actual product routing. For instance, when the product approaches the simulated diverter component, its input sensor is triggered by setting the appropriate tag value in the control interface to *true*. This automatically notifies the corresponding diverter agent that makes a decision about which direction to switch the product. The agent then sends a command to simulated actuator again by setting the value of the appropriate tag in the control interface.

The visualization module displays the status of the simulation in a GUI (see the right-hand side of Fig. 1). The user can watch the transportation of products and check the decisions of agents concerning the alternative routing in case of artificially introduced failures. The GUI also serves as a design tool for creating an arbitrary layout of a manufacturing system. The user selects particular manufacturing components from the graphical library, connects them together and sets specific parameters. Behind the scenes, with each component added to the system, corresponding agent is dynamically instantiated and specific tags for I/O interactions are added to the control interface.

The real-life control capabilities of MAST agents have been verified on the physical palette transfer system located in the Odo Struger Laboratory at the TU Vienna's ACIN institute. This laboratory environment consists of fourteen conveyor belts, twelve diverters, four docking stations and a number of sensors and stoppers. Raw materials and products are transported on top of palettes to reach a desired docking station. The palette is held in the docking station until particular machine finishes its operation; after that the palette is released and routed to another docking station where the next operation is scheduled. After building the simulation of the transport system in MAST and verifying the proper functionality in failure scenarios (selection of redundant paths), the control interface of MAST has been configured to access the real I/O values held in tags of a ControlLogix controller. Concurrently, the simulation part of MAST has been turned off. It has been verified that without any change required the agent control system was able to control the real-world manufacturing set-up and effectively cope with disturbances [15].

MAST is also designed as the development tool with a well-defined API that allows a developer to implement the new agent classes (and corresponding simulation components) with desired application specific behavior. This feature has been used to develop the MAST Test Management System (TMS) serving as a simulation platform for the purposes of the study presented in this paper.

B. MAST Test Management System (MAST-TMS)

The MAST Test Management System (MAST-TMS) [19,20] is an extension of the original MAST simulator providing means for automatic execution and evaluation of a predefined set of simulation experiments. Each experiment tests a specific scenario with a different set of input parameters – the number and type of products to be assembled, the workflow scheduling strategy to be applied, the number of pallets to be used, etc. The set of evaluation scenarios can also be automatically generated on the basis of user requirements – the user selects which parameters should be fixed for all tests and which should alter either within a specified range or according to a selected probability distribution. This feature has proven to be useful for evaluating the impact of the variation of one or more simulation parameters on the overall system performance
In order to execute the set of evaluation scenarios, the MAST system is reset into an initial state – the configuration of the workshop is loaded from a configuration file and the components of the workshop as well as their controlling agents are created. As a next step, the XML file containing the description of evaluation scenarios is loaded and the first scenario is injected into the system. The TMS uses the input parameters of the scenario to run the simulation experiment. All relevant events, like finalization of a product or occurred failure are logged to an output XML file. Once the experiment is finished, the agent system is again reset into the initial state and a next scenario is selected and corresponding simulation experiment conducted. This is done fully automatically without user intervention.

The simulation results are subsequently analyzed using a series of algorithms and methods, resulting in a number of derived key performance indicators (e.g., number of finished products or average machine utilization rate) that can be used for comprehensive statistical data analysis.

The MAST-TMS is a small graphical application, that manages running multiple tests in a batch with MAST. The prior solution for that task was a Java application that was executed within the same Java runtime environment as the simulation itself. Different requirements for the original MAST caused the application to focus on a single execution only, so after termination of the simulation some stale threads and their referenced memory keep occupying system resources. Also if the simulator crashed during the execution of a simulation, no more tests were executed until the issue was addressed manually (i.e. restart the simulator). Therefore, the MAST-TMS was designed as a standalone application which injects the test case data into MAST and terminates the Java Virtual Machine either after each the test case is finished or after a preset timeout. All resources that are occupied after termination of the simulation are freed when terminating the corresponding virtual machine. Also a failure of the simulator (e.g. freeze or run into unexpected program errors) can be detected and resolved automatically by an external application.

3. Re-scheduling Policies

The way how manufacturing systems treat exceptional events can significantly influence their performance. Using predefined schedules, such systems are doing well if everything is going well. However, it is of vital importance to define their reaction on unexpected events. The rescheduling component of the system uses a re-scheduling policy specifies what event triggers re-scheduling and what method will be applied for re-scheduling. Moreover, the re-scheduling policy also specifies the method applied for revision of the existing schedules [13]. Three policies related to the re-scheduling initiation events have been presented in literature [17, 18] – periodic, event-driven, and hybrid. A periodic policy is periodically initiated with a defined time period during which all available system information is collected and then used for deriving the re-scheduling setting. However, the effectiveness of this policy depends on the optimally adjusted length of the period, which might be hard to effectively anticipate. Moreover, this policy is not agile enough because critical events, which require prompt reaction, are not processed immediately, but wait at the end of the re-scheduling period. This is not the case for event-driven re-scheduling triggered immediately when the specific event (e.g., job arrival, machine failure) occurs. However, in a large system, where the number of such events happening simultaneously can be enormous, the application of this policy can lead to continuous re-scheduling and thus to lower stability and performance. A hybrid re-scheduling policy can be seen as combination of previous two approaches as the system reschedules periodically as well as when specific, user-defined events occur, synchronizing policy occurrence avoiding their overlapping (e.g. to have periodic and event-driven re-scheduling at the same time).
Having a smaller manufacturing system as a test case, we decided to apply an event-driven re-scheduling policy, considering machine failures as events that trigger re-scheduling. We implemented and tested four agent-based schedule repair methods corresponding to methods presented in [10]:

1. *Right-shift scheduling (RS):* when a machine breaks down this method postpones the job being currently processed as well as all other jobs that are waiting in the machine's queue until the machine is repaired. During the repair time period the resource agent (RA) that is in charge of this machine still responds to calls for bids from product agents (PA), offering its free capacity in the "after-repair" period.

2. *Agenda rerouting (AR):* after the machine fails all jobs from the machine's queue are rerouted to alternative machines. In contrast to previous case, the time that the jobs loose by waiting in the machine's queue for its repair can be saved. However, also in this case the RA bids on its services during the negotiations with the PA about new jobs.
3. *New jobs rerouting (NR):* in this case the RA keeps all jobs in the machine queue while refuses to bid on new arriving jobs. This policy tends to avoid the additional load to a failed machine by not accepting new job arrivals and to prevent system stress through the machine queue jobs rerouting.
4. *Complete rerouting (CR):* combines the AR and NR methods. The machine's RA addresses the PAs of jobs that are scheduled to the failed machine to reroute the jobs to alternative machines as well as does not participate in subsequent negotiations with PAs about new jobs.

Due to its simplicity, the RS policy is mostly applied by current manufacturing systems. Thus we consider it as a reference policy in our study. The AR and NR policies could be referred to as partial scheduling policies while the CR represents a regeneration policy.

As mentioned in the introduction section we additionally propose applying dispatching rules to handle dynamics of a manufacturing environment. These rules are used by RAs, which supervise functioning of machines for optimal sequencing of allocated jobs. The important advantage of dispatching rules is the ability to select the highest priority job from the machine's queue considering all reliable up-to-date information at the time of selection. We are using the Critical Ratio rule to prioritize jobs, expressed as:

$$CR = \frac{Rt}{Pt} \qquad \text{(eq. 1)},$$

where Rt is the remaining time from the current time to the job's due date and Pt is the remaining processing time of the product being currently processed. A task with lower critical ratio gets higher priority. We selected the critical ratio rule because it showed the best performance in comparison to others dispatching rules such as First Come First Served, Earliest Due Date and Shortest Processing Time [19].

4. Research Issues

Unforeseen uncertainties or exceptions (e.g., machine breakdowns, quality problems, traffic jams in the transportation system, arrivals of urgent jobs, etc.) during the production process require a fast reaction in order to avoid deviations from the initial production schedule. Taking into account that the occurrence of an exception is mostly unpredictable, the application of strategies which deal with the exception is necessary to avoid the stop of a manufacturing operation or reduction of the overall system performance whenever an exception occurs. A manufacturing control system needs to be able to effectively adjust the production scheduling plan and reorganize the production schedule accordingly.

In this paper we are applying diverse re-scheduling handling policies and empirically evaluate their performance under specific conditions, i.e., using different failure types and durations and diverse number of transport entities used. We also investigate how specific production conditions such as different types of failure classes (conveyor failures and machine failures), number of used pallets for transportation, as well as duration of failures influence the system performance.

In the context of the empirical study, we define system performance, equivalent with the production effectiveness function ***E***, as the average number of finished products within a given shift. As argument we consider following parameters: level of workload, number of pallets, re-scheduling policy, and type of failure. We define the following null hypotheses to be falsified by our experiments:

RI1: Influence of number of pallets on system performance *E*. Let us assume that a higher number of pallets will result in a higher overall system output. *IF n* and *m* are the number of pallets introduced, and *E* is the production effectiveness function, *THEN* we define the following null hypothesis:

$$\boldsymbol{H_0\text{-}1}: \{\exists \mathrm{m} > \mathrm{n} \mid E(\mathrm{m}) \leq E(\mathrm{n})\} \quad \text{(eq. 2)}$$

RI2: Impact of re-scheduling policy on system performance *E*. Let us assume that the use of different re-scheduling strategies will result in a diversity of resulting system outputs, because some re-scheduling policies include the machine agents representing broken machines into the negotiation process, while others do not. Furthermore, the different handling of already queued or future awarded jobs to broken machines could also have a significant impact on the overall system performance. *IF α* represents the AR re-scheduling policy, *β* represents the NR re-scheduling policy and *γ* represents the CR re-scheduling policy, *THEN* we can define the following null hypothesis:

$$\boldsymbol{H_0\text{-}2}: \{\forall \alpha, \beta, \gamma \mid E(\gamma) \leq \min(E(\alpha) \vee E(\beta))\} \quad \text{(eq. 3)}$$

RI3: Impact of failure type on system performance *E*. Let us assume that the duration of machine's unavailability (two different periods for machine failures and machine disturbances) has a direct influence on the overall system performance. *IF f* is a machine disturbance (**f**aster recovery) and *s* is a machine failure (**s**lower recovery), *THEN* we propose the following null hypothesis:

$$\boldsymbol{H_0\text{-}3}: \{\forall \mathrm{f}, \mathrm{s} \mid E(\mathrm{f}) \leq E(\mathrm{s})\} \quad \text{(eq. 4)}$$

RI4: Impact of workload level on system performance *E*. Let us assume that the number of orders directly influences the relative system performance, i.e., the number of finished products compared to the number of ordered products. *IF l* represents a lower number of received orders and *h* represents a higher number, *THEN* we identify the following null hypothesis:

$$\boldsymbol{H_0\text{-}4:}\ \{\forall h > l \mid \frac{E(l)}{l} \leq \frac{E(h)}{h}\} \quad \text{(eq. 5)}$$

5. Simulation Approach

This section describes the details of the simulation experiment carried out in MAST-TMS environment. A total of sixty four evaluation scenarios have been tested. In each test case, the Critical Ratio (CR) dispatching rule has been applied and different values of following parameters set: the re-scheduling policy (RS, AR, NR and CR), the number of pallets providing transportation (10, 15, 20, or 25), the machine failure recovery time (f - faster recovery or s - slower recovery) and the level of workload (l – low: 2,880 or h – high: 5,760 number of orders). An order consists of a product type to be produced and randomly generated due date (in seconds). For simplicity, three pre-defined product types were used – simple, medium and complex – that differ in the number of machine operations and raw materials/semi-products needed to assemble the final product. So called product trees depicting both required raw materials (ovals) and assembly operations (rectangles) for these three product types are shown in Figure 2 (simple, medium and complex products from right to left).

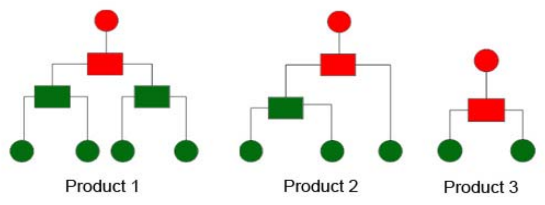

Figure 2: Product Trees of the used Product Types.

Machine operation times and transportation times were considered as fixed for all evaluation scenarios. The shift time for an evaluation scenario was set to one full day (24 hours). This implies that the overall time needed for producing all workload items is larger than the set shift time, resulting in less items being produced than actually ordered. Otherwise, if all ordered products would have been produced, we would not have been able to measure the effectiveness of the chosen simulation parameters by using the number of finished products as a reliable parameter.

The machine failure specification consists of the machine identifier and start/end time points of the failure. We classified the risk of failing a machine using two different failure classes: machine disturbances (f – faster recovery), which can be repaired in approximately 0.1% of the overall shift time, and machine failures (s – slower recovery), which requires a longer repair time, in our case about 10% of the overall shift time. For effective comparison of the robustness of the re-scheduling policies, failures with the same specifications were used for all evaluation scenarios. In order to be able to easily identify the re-scheduling decisions and the measured results, we used a workshop layout consisting of four machines, each capable of performing between 2 and 4 different machine operations. The machines were configured in a redundant way, i.e., there were 2 machine sets, each consisting of 2 machines, which offer the same functionality, resulting in better load-balancing and also failover capabilities. Moreover, the chosen workshop layout imitates the existing pallet transfer system[1] at the Institute for Automation and Control, Vienna University of Technology. In addition, the layout contains two storages, one for the raw materials and one for the final products.

The sixty four evaluation scenarios were split down into four batches, each containing sixteen test cases and particular re-scheduling policy applied. These batches were run in parallel on four high-performance mainframe servers, taking it approximately eight hours to finish a single batch. After all tests had been finished, the resulting data were collected from the mainframe servers and analyzed. The effect of disruptions on shop floor activities and robustness of the different re-scheduling policies are measured by the difference in a number of finished products, when diverse dynamic environment changes occur. We used descriptive statistic and analyzed the results of finished product with three different variables: the number of pallets, the machine failure recovery time, and the level of workload. We performed a correlation analysis by means of Spearman Rank to check the significance correlation between the selected variables, with respect to different selected re-scheduling policies. Statistical data analysis of the simulation results helps analyzing the impact of design and planning factors and consequently answers fundamental questions on planning and coordination strategies in the production automation domain such as the impact of a variety of fundamental scheduling strategies in a given environment.

6. Experimental Results and Discussion

In our previous research [20] we investigated how transport system failures can influence system performance. In this study we explore the impact of additional factors such as failures, disturbances, overloads, etc. on the system output. Especially, we judge the manufacturing system ability to absorb a machine failure using predefined failure tolerance policies. As important factors we consider different duration of machines downtime, diverse levels of production workload as well as various number of transportation units (pallets).

[1] http://www.acin.tuwien.ac.at

A. Number of finished products considering all factors

Similarly to Kutanoglu and Sabuncuoglu in [21], but applying the decentralized multi-agent control and using Contract Net Protocol (CNP) [31] for workload balancing, we reached the same conclusion – the superiority of CR (complete rerouting) policy over the other policies.

As presented in Figure 3 as well as in Table 1 showing the mean numbers of finished products depending on number of palettes, the CR policy outperforms RS (right-shift scheduling) policy approximately by 12% to 16%. This is an expectable result because the immediate exclusion of the failed machine from production and full re-scheduling of tasks in its queue to other available machines significantly compensates the failure state when no action would be taken in case of RS policy (the tasks in machine's queue have to wait for machine repair). This is of course true only if suitable alternative resources are available.

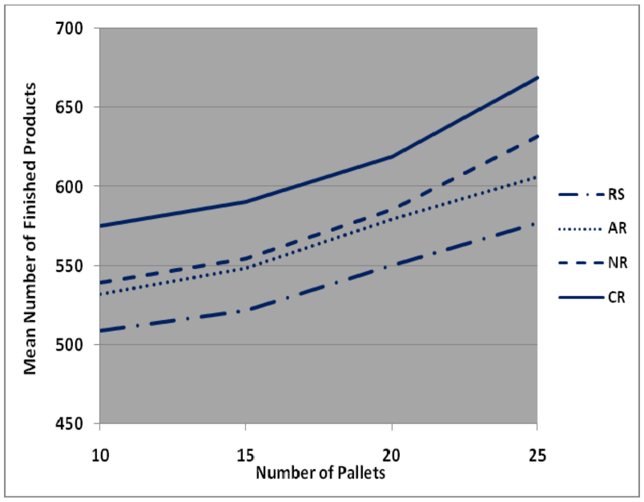

Figure 3. Mean number of finished products of a shift by policy.

Table 1: Finished products by number of pallets and re-scheduling policy.

Policy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD[2]	Mean	STD	Mean	STD	Mean	STD
RS	509	64	522	68	551	71	577	75
AR	532	71	549	71	580	75	606	78
NR	539	69	555	72	586	77	632	58
CR	575	74	591	77	619	86	669	66

Notable is a good performance of the NR (new jobs rerouting) policy, which proves that the failed machine should be urgently excluded from further scheduling until repaired. However, due the fact that the NR policy keeps already awarded jobs in machine's queue, its performance is approximately by 5.6% to 6.6%

[2] STD - Standard Deviation

weaker than of the CR policy. Interesting fact is also the influence of number of pallets on production performance. The 150% increase or pallets results only in 15% gain in production output. This can be justified by increased number of traffic jam situations caused by a limited space for palettes in a conveyor-based transportation system.

B. Number of finished products considering failure duration

Extensive experiments have also been done to identify how the efficient failure recovery impacts the system performance. As mentioned earlier, in a general overview, the CR-(f - faster recovery or s - slower recovery) policy wins the race, but relation between the NR-(f,s) and AR-(f,s) policies deserve deeper study (consult Figure 4 and Table 2).

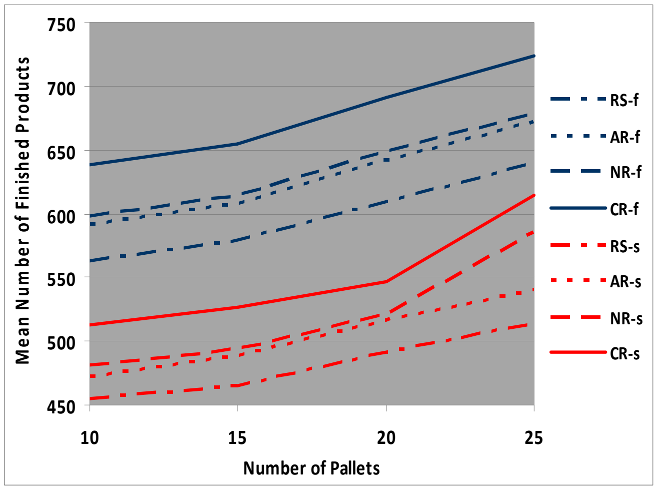

Figure 4. Influence of failure duration (f - faster recovery or s - slower recovery).

Table 2: Finished products by defined failure duration.

Policy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RS-f	564	71	579	75	610	79	640	83
AR-f	592	79	608	79	642	83	672	86
NR-f	598	76	615	80	650	85	679	62
CR-f	638	82	655	85	692	96	724	71
RS-s	455	57	466	61	491	63	514	67
AR-s	472	63	489	63	517	67	541	70
NR-s	481	62	495	64	521	69	585	54
CR-s	512	66	527	69	547	76	615	61

While the difference between the NR-f and AR-f policies, when short disturbances periodically occur, stays almost constantly around 1% in favour of the NR-f policy, this ratio notably grows when the failure duration prolongs (NR-s and AR-s curves respectively) and the number of pallets rises. The fact that the NR-s policy performs by 8% better than AR-s when using 25 pallets can by explained such as this policy is handicapped just by those products which are already awarded to the machine while all others incoming after the failure are rebalanced to other available machines. On the other hand, the AR-s policy profited by re-scheduling of the queued jobs, but only to such a point where the number of palettes in the system caused traffic jams and consequently increased the number of unfinished products that were on their way to alternative machines instead of waiting in the machine's queue for repair. This finally resulted in weaker performance of the AR-s policy.

C. Number of finished products considering workload

The introduction of two levels of workload (l – low: 2,880 or h – high: 5,760 orders respectively) has again confirmed the supremacy of the CR-(l, h) policy (see Figure 5 and Table 3).

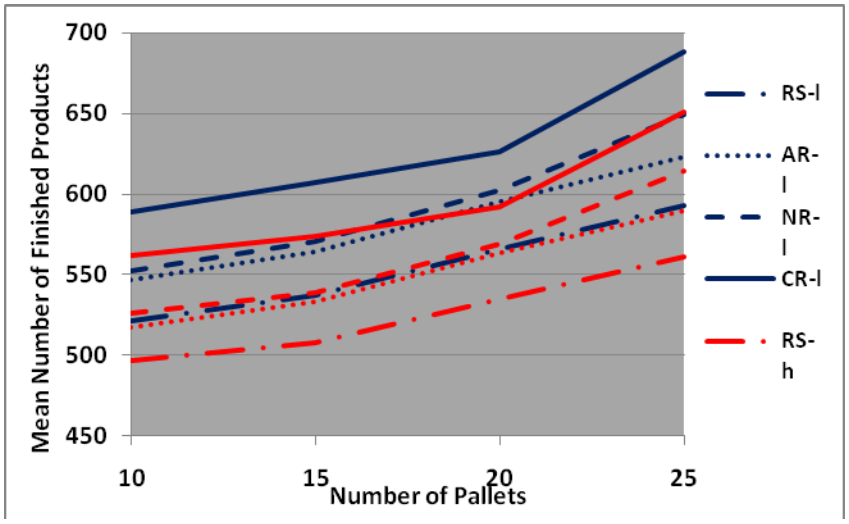

Figure 5. Influence of different levels of workload (l – low: 2,880 or h – high: 5,760 orders).

Table 3: Finished products by level of workload.

Policy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RS-l	522	66	537	70	566	73	593	77
AR-l	547	73	564	73	596	77	623	80
NR-l	552	71	571	74	602	79	650	60
CR-l	589	76	608	79	635	87	688	68
RS-h	497	62	508	66	535	69	561	73
AR-h	518	69	533	69	564	73	590	76
NR-h	527	67	539	70	569	75	615	56
CR-h	562	72	574	75	603	82	651	64

It is interesting to note here that the system with higher workload, especially due to the overloading and traffic jam, has shown a 4.7% to 6% weaker performance. Nevertheless, even then the CR-h policy with a higher workload was able to outperform all remaining policies.

D. Empirical Results

Analyzing the empirical results, we derive the following implications for the impact factor analysis regarding the re-scheduling policies and the four defined null hypotheses:

Influence of number of pallets on system performance *E*. The data analysis shows that an increase of the number of pallets will increase the overall system output. As shown in Table 1 the number of pallets has a significant impact on the overall system performance (in terms of number of finished products) with a p-value < 0.01. In the study context, an increase of the number of used pallets (i.e., from *n* to *m*) always increased the number of finished products. Therefore we can state that

$$\{\exists \mathrm{m} > \mathrm{n} \mid E(\mathrm{m}) > E(\mathrm{n})\} \ ,$$

and so we can reject null hypothesis ***H_0-1***. In addition however, we have to state that we increased the number of pallets in a way that the overall system cannot be saturated or at least close to this state. Therefore, we reject the null hypothesis ***H_0-1*** just for a reasonable range of *n* and *m*.

Impact of re-scheduling policy on system performance *E*. As shown in Figure 3, the Complete Rerouting (CR) policy outperforms all other rerouting policies (p-value < 0.01). Therefore, we conclude that the overall system performance of the CR policy (γ) is higher than the system performance of all other rerouting policies, namely the NR (β) and AR (α). We can state that

$$\{\forall \alpha, \beta, \gamma \mid E(\gamma) > \max(E(\alpha) \vee E(\beta))\} \ ,$$

and hence we can reject null hypothesis ***H_0-2***. Again, we have to state that we are well aware of cases for which the CR policy does not outperform all other policies. It depends mainly on the timing factor. In the present example, there are slow and fast recovery types of failures which take 0.1% and 10% of the overall shift time duration for recovery, but this is not a complete dataset. If the time to repair is similar to time to reroute then CR policy can be worse. Also when the system is closed to saturation state, it can be better not to start rerouting since the system can become saturated and the performance can drop significantly.

Impact of failure type on system performance *E*. Figure 4 shows that the failure recovery time has a direct impact on the resulting overall system performance. A faster failure recovery (f) will increase the production effectiveness, while slower failure recovery (s) decreases the production effectiveness with a p-value < 0.01. In the study context, a decrease of the failure recovery time always led to an increased number of finished products. Therefore we can state that

$$\{\forall \mathrm{f}, \mathrm{s} \mid E(\mathrm{f}) > E(\mathrm{s})\}$$

and reject null hypothesis ***H_0-3***. There may be cases different to our experiment setup, when the system offers so many alternatives so that not all of them are utilized, so that it is not important if the failure of one machine takes short or long.

Impact of workload level on system performance *E*. As shown in Figure 5, the number of orders directly influences the relative system performance. As observed in the experiment, an increase of the number of orders (i.e., from *l* to *h*) always lead to a decreased overall system performance. Therefore we can state that

$$\{\forall h > l \mid \frac{E(l)}{l} > \frac{E(h)}{h}\}$$

and reject null hypothesis ***H_0-4***.

7. Conclusion and Future Work

In this paper we present the use of MAS to handle dynamic production environments (e.g., machine disturbances/failures) in order to improve the engineering of production automation systems. Agents are used to supervise machines and automatically reschedule work orders in case of machine unavailability. We evaluated a re-scheduling system component which uses four different re-scheduling policies: Right-shift scheduling (RS), Agenda rerouting (AR), New jobs rerouting (NR), and Complete rerouting (CR). The evaluation in the context of an empirical study found that the Complete Rerouting (CR) re-scheduling policy outperformed all other re-scheduling policies. Analogous conclusions have been also made by other researchers that investigated similar topics. In addition, we investigated the influence of the number of pallets, the duration of machine failures, and the level of workload on the overall system output. We show that a multi-agent approach combined with dynamic dispatching rules and different re-scheduling policies (especially the Complete Rerouting re-scheduling policy that had the best performance) performs well in terms of robustness minimizing the effects of disruptions on the system performance. The experiments also show that a higher number of pallets always increased the number of finished products, but with diminishing marginal gains caused by traffic jams. Further on, longer failure durations have a certain impact on the overall system performance, since the unavailability of the failed machine as well as its participation in the job allocation process seriously influence system performance. Finally, we found in the study context a lower size of workload to lead to higher system output, because of the lower system utilization and fewer occurrences of traffic jams.

Further work will focus on improving the engineering of production automation systems by implementing proactive-reactive scheduling/re-scheduling policies and evaluating of their performance compared to reactive re-scheduling policies. Additionally, more complex workshop layouts with more machines and more complex routes will be used to evaluate the performance of the presented re-scheduling policies. Besides, Besides, we are planning to investigate the system performances when all agents are not subject to the same rescheduling policy or dispatching rule at the same time and apply them according to the current system conditions. We are also going to extend the spectrum of used dispatching rules. As previous research focused on the impact of transport system failures and this research considered the influence of machine failures, the further work will investigate the combination of both failure types. The implementation project of this approach on the miniature hardware simulation system located at the Odo Struger Laboratory of the Automation Control Institute (ACIN) is already under way at the time of this writing.

Acknowledgments

The authors would like to acknowledge the financial support by the Austrian Science Fund (FWF), Project No. FWFP19644-N13, as well as by the FIT–IT: Semantic Systems program, an initiative of the Austrian federal ministry of transport, innovation, and technology (bm:vit) under contract FFG 815132. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. This work has been partially funded by the Vienna University of Technology, in the Complex Systems Design and Engineering Lab.

References

[1] Tichy P., Slechta P., Staron R. J., Maturana F. P., and Hall K. H.. Multiagent Technology for Fault Tolerance and Flexible Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Volume 36, Issue 5, 2006, pp. 700-704.

[2] Heragu S., Graves R., Kim Byung-In, and St Onge A., “Intelligent agent based framework for manufacturing systems control,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 32, 2002, pp. 560-573.

[3] Jennings, N.R., and Wooldridge, M.J. Agent technology: foundations, applications, and markets. Springer-Verlag, 1998.

[4] Shen, W. and Norrie, D.H. Agent based systems for intelligent manufacturing: a state of the art survey. *International Journal of Knowledge and Information Systems*, 1(2), 1999, pp. 129-156.

[5] Jennings, N.R., and Bussman, S. Agent-Based Control Systems: Why are They Suited to Engineering Complex Systems? *IEEE Control Systems Magazine*, 23(3), 2003, pp. 61-73.

[6] Sycara K. Multiagent systems. *AI Magazine*, 19(2), 1998, pp. 79- 92.

[7] Pechouček M. und Marík V., “Industrial deployment of multi-agent technologies: review and selected case studies,” *Autonomous Agents and Multi-Agent Systems*, vol. 17, Dez. 2008, S. 397-431.

[8] Holthaus, O. Design of efficient job shop scheduling rules. *Computers and Industrial Engineering*, 33(1–2), 1997, pp. 249–252.

[9] J. Schönberger and H. Kopfer, “A General Approach to Robustness in Logistics – Basic Concepts, Quantification Approaches and Experimental Evaluations,” *Logistik Management*, 2009, pp. 299-323.

[10] Kutanoglu, E., and Sabuncuoglu, I. Routing-based Reactive Scheduling Policies for Machine Failures in Job Shops. *Int. J of Production Research*, Vol. 39, 2001, pp. 3141-3158.

[11] Bastos, R., Oliveira, F., and Oliveira, J. Autonomic computing approach for resource allocation. *Expert Systems with Applications* 28, Elsevier Ltd, 2005, pp. 9-19.

[12] Wu, J., and Durfee, E.H. Sequential resource allocation in multi-agent systems with uncertainties. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, 2007.

[13] Vieira, G. E., Hermann, J.W., and Lin, E. Re-scheduling manufacturing systems: a framework of strategies, policies and methods. Journal of Scheduling, 6 (1), 2003, pp. 36-92.

[14] Smith R.G. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, Vol. 29, 1980, pp. 1104-1113.

[15] Vrba, P., Mařík, V., Merdan, M. Physical Deployment of Agent-based Industrial Control Solutions: MAST Story. In Proceedings of IEEE International Conference on Distributed Human-Machine Systems, Athens, 2008, pp. 133-139.

[16] Vrba, P. MAST: Manufacturing Agent Simulation Tool. In Proceedings of IEEE Conference on Emerging Technologies and Factory Automation, September 2003, Lisbon, Portugal, Volume 1, pp. 282-287.

[17] Sabuncuoglu, I. and Bayiz, M. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126 (3), 2000, pp. 567-586.

[18] Ouelhadj, D., and Petrovic, S. Survey of Dynamic Scheduling in Manufacturing Systems. *Journal of Scheduling*, October 2008, http://www.springerlink.com/content/gq81525hm27872up/

[19] Merdan, M., Moser, T., Wahyudin, D., Biffl, S., and Vrba, P. Simulation of Workflow Scheduling Strategies Using the MAST Test Management System. The 10th IEEE International Conference on Control, Automation, Robotics and Vision, ICARCV 2008.

[20] Merdan, M., Moser, T, Wahyudin, D., and Biffl, S. Performance Evaluation of Workflow Scheduling Strategies Considering Transportation Times and Conveyor Failures. In Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management, Singapore, 2008.

[21] Kutanoglu, E., and Sabuncuoglu, I. An investigation of reactive scheduling policies under machine breakdowns. In Proceedings of the 4th Industrial Engineering Research Conference, 1995, pp. 904-913.

[22] Babiceanu, R. & Chen, F., 2006. Development and Applications of Holonic Manufacturing Systems: A Survey. *Journal of Intelligent Manufacturing*, 17(1), 111-131.

[23] Boccalatte, A. et al., 2004. A multi-agent system for dynamic just-in-time manufacturing production scheduling. In Systems, Man and Cybernetics, 2004 IEEE International Conference on. pp. 5548-5553 vol.6.

[24] Kocjan, W., 2002. Dynamic scheduling: State of the art report. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.7135.

[25] M. Merdan, 2009. Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain. PhD Thesis. Vienna University of Technology. Available at: http://www.ub.tuwien.ac.at/diss/AC05040230.pdf.

[26] Sadeh N., 1991. Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling. PhD Thesis. Carnegie Mellon University, Pittsburgh, PA,

[27] Ouelhadj, D. & Petrovic, S., 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 12(4), 417-431.

[28] Sandholm, T.W., 2000. Automated contracting in distributed manufacturing among independent companies. *Journal of Intelligent Manufacturing*, 11(3), 271-283.

[29] Shen, W., Wang, L. & Hao, Q., 2006. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(4), 563-577.

[30] Wang, C., Ghenniwa, H. & Shen, W., 2008. Real time distributed shop floor scheduling using an agent-based service-oriented architecture. *International Journal of Production Research*, 46(9), 2433-2452.

[31] Smith R. G., "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, pp. 1104-1113, 1980.

[32] Aytug H., Lawley M.A., McKay K., Mohan S., and Uzsoy R., "Executing production schedules in the face of uncertainties: A review and some future directions," *European Journal of Operational Research*, vol. 161, Feb. 2005, pp. 86-110.

[33] Leon V.J., Wu S.D., and Storer R.H., "Robustness measures and robust scheduling for job shops," *IIE - Transactions*, vol. 26, 1994, p. 32.

[34] Pfeiffer A., Kadar B., and Monostori L., "Stability-oriented evaluation of rescheduling strategies, by using simulation," Computers in Industry, vol. 58, Sep. 2007, pp. 630-643.

[35] P. Leitao, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, Oct. 2009, pp. 979-991.

[36] Bussman, S., N.R. Jenning, and M. Wooldridge. Multiagent System for Manufacturing Control. A Design Methodology. Springer Series on Agent Technology, 2004.

[37] D. Zhang, A. Anosike, and Ming Kim Lim, "Dynamically Integrated Manufacturing Systems (DIMS)—A Multiagent Approach," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 37, 2007, pp. 824-850.

[38] Cauvin A., Ferrarini A., and Tranvouez E., "Disruption management in distributed enterprises: A multi-agent modelling and simulation of cooperative recovery behaviours," *International Journal of Production Economics*, vol. 122, Nov. 2009, pp. 429-439.

[39] Guo B. and Nonaka Y., "Rescheduling and optimization of schedules considering machine failures," *International Journal of Production Economics*, vol. 60, 1999, pp. 503-513.

[40] Sabuncuoglu I. and Goren S., "Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research," *International Journal of Computer Integrated Manufacturing*, vol. 22, 2009, p. 138.

[41] Ferber J., Gutknecht O., and Michel F., "From Agents to Organizations: An Organizational View of Multi-agent Systems," *Agent-Oriented Software Engineering* IV, 2004, pp. 443-459.

[42] Van Dyke Parunak H., "A Practitioners' Review of Industrial Agent Applications," *Autonomous Agents and Multi-Agent Systems*, vol. 3, Dez. 2000, S. 389-407.

[43] Vrba, P., Mařík, V. Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, Vol. 40, Issue 1, 2010, pp. 1-11.