A Study on Transport and Load in a Grid-based Manufacturing System

Leo van Moergestel, Erik Puik, Daniël Telgen
Department of Computer science
HU Utrecht University of Applied Sciences
Utrecht, the Netherlands
Email: {leo.vanmoergestel, erik.puik, daniel.telgen}@hu.nl

John-Jules Meyer
Intelligent systems group
Utrecht University
Utrecht, the Netherlands
Email: J.J.C.Meyer@uu.nl

***Abstract*—Standard mass-production is a well-known manufacturing concept. To make small quantities or even single items of a product according to user specifications at an affordable price, alternative agile production paradigms should be investigated and developed. The system presented in this article is based on a grid of cheap reconfigurable production units, called equiplets. A grid of these equiplets is capable to produce a variety of different products in parallel at an affordable price. The underlying agent-based software for this system is responsible for the agile manufacturing. An important aspect of this type of manufacturing is the transport of the products along the available equiplets. This transport of the products from equiplet to equiplet is quite different from standard production. Every product can have its own unique path along the equiplets. In this article several topologies are discussed and investigated. Also, the planning and scheduling in relation to the transport constraints is subject of this study. Some possibilities of realization are discussed and simulations are used to generate results with the focus on efficiency and usability for different topologies and layouts of the grid and its internal transport system. Closely related with this problem is the scheduling of the production in the grid. A discussion about the maximum achievable load on the production grid and its relation with the transport system is also included.**

Keywords-Multiagent-based manufacturing; Flexible transport.

I. INTRODUCTION

In standard batch processing the movement of products is mostly based on a pipeline. Though batch processing is a very good solution for high volume production, it is not apt for agile manufacturing when different products at small quantities are to be produced by the production equipment. This article describes an agile and flexible production system, where the production machines are placed in a grid. Products are not following a single path, but different paths can be used in parallel, leading to parallel manufacturing of different products. The grid arrangement of production machines reduces the average path when products move along their own possibly unique paths within the grid during the production. To move the products around during production, the ways the production machines are interconnected should be investigated to find an affordable and good solution. An important aspect will also be the amount of products in the grid during production, because too many products will result in failures in the scheduling of the production. The investigation about transport and the amount of products in the grid, resulting in the total load of the grid, are the motivation and purpose of this article. The goal is to investigate the effect of different interconnection possibilities to the average production path and to see how the grid behaves under load. The work in this article is based on a paper presented at the Intelli 2014 conference [1] and other previous work. The design and implementation of the production platforms and the idea to build a production grid can be found in Puik [2]. In Moergestel [3] the idea of using agent technology as a software infrastructure is presented. Two types of agents play a major role in the production: a product agent, responsible for production of a product and an agent responsible for performing certain production steps on a production machine. Another publication by Moergestel [4] is dedicated to the production scheduling for the grid production system. The rest of this paper is organised as follows: In Section II of this article related work will be discussed. Section III will explain grid manufacturing in more detail, followed by Section IV about transport in the grid. Section V introduces the software tools built. The results are presented and discussed in Section VI. Finally, a conclusion where the results are summarized will end the article.

II. RELATED WORK

Using agent technology in industrial production is not new though still not widely accepted. Important work in this field has already been done. Paolucci and Sacile [5] give an extensive overview of what has been done in this field. Their work focuses on simulation as well as production scheduling and control [6]. The main purpose to use agents in [5] is agile production and making complex production tasks possible by using a multi-agent system. Agents are also introduced to deliver a flexible and scalable alternative for manufacturing execution systems (MES) for small production companies. The roles of the agents in this overview are quite diverse. In simulations agents play the role of active entities in the production. In production scheduling and control agents support or replace human operators. Agent technology is used in parts or subsystems of the manufacturing process. On the contrary, we based the manufacturing process as a whole on agent technology. In our case a co-design of hardware and software was the basis.

Bussmann and Jennings [7][8] used an approach that compares to our approach. The system they describe introduced three types of agents, a workpiece agent, a machine agent and

a switch agent. Some characteristics of their solution are:

- The production system is a production line that is built for a certain product. This design is based on redundant production machinery and focuses on production availability and a minimum of downtime in the production process. Our system is a grid and is capable to produce many different products in parallel;
- The roles of the agents in this approach are different from our approach. The workpiece agent sends an invitation to bid for its current task to all machine agents. The machine agents issue bids to the workpiece agent. The workpiece agent chooses the best bid or tries again. In our system, the negotiating is between the product agents, thus not disrupting the machine agents;
- They use a special infrastructure for the logistic subsystem, controlled by so called switch agents. Even though the practical implementation is akin to their solution, in our solution the service offered by the logistic subsystems can be considered as production steps offered by an equiplet and should be based on a more flexible transport mechanism.

However, there are important differences to our approach. The solution presented by Bussmann and Jenning has the characteristics of a production pipeline and is very useful as such, however, it is not meant to be an agile multi-parallel production system as presented here.

Other authors focus on using agent technology as a solution to a specific problem in a production environment. In [9] a multi-agent monitoring system is presented. This work focusses on monitoring a manufacturing plant. The approach we use monitors the production of every single product. The work of Xiang and Lee [10] presents a multiagent-based scheduling solution using swarm intelligence. Their work uses negotiating between job-agents and machine-agents for equal distribution of tasks among machines. The implementation and a simulation of the performance is discussed. In our approach the negotiating is between product agents and load balancing is possible by encouraging product agents to use equiplets with a low load. We did not focus on a specific part of the production but we developed a complete production paradigm based on agent technology in combination with a production grid. This model is based on two types of agents and focuses on agile multiparallel production. There is a much stronger role of the product agent and a product log is produced per product. This product agent can also play an important role in the life-cycle of the product [11].

In agent-based manufacturing the term holon is often used. While agent technology emerged from the field of computer science, the concept holon has its origin in computer integrated manufacturing (CIM) [12]. The concept was proposed by Koestler [13]. Parts of a system can be autonomous and stable on their own, but by cooperation they may form a bigger whole. This bigger whole could again be a part of an even bigger whole. A holon is both a part and a whole. A holon can represent a physical object or a logical activity. In the domain of manufacturing this can be a production machine, a production order or a human operator [14]. Agent technology can be used to implement a holon. This is where the two approaches, agent technology and the holon concept, meet. An important difference is that a holon can also be a passive entity like the aforementioned production order, while agents represent active autonomous entities. Fisher [15] uses a holonic approach for manufacturing planning and control. His work is based on the use of the Integration of Reactive behaviour and Rational Planning (InterRRap) agent architecture proposed by Müller [16]. Agents represent the holonic manufacturing components, forming a multiagent system. In our manufacturing model the holonic approach was not used, because a more simple multiagent system fitted our requirements.

III. GRID MANUFACTURING

In grid production, manufacturing machines are placed in a grid topology. Every manufacturing machine offers one or more production steps and by combining a certain set of production steps, a product can be made. This means that when a product requires a given set of production steps and the grid has these steps available, the product can be made [2]. The definition of a production step as used in this article is:

Definition[Production step] *A production step is an action or group of coordinated or coherent actions on a product, to bring the product a step further to its final realisation. The state of the product before and after the step are stable, meaning that the time it takes to do the next step is irrelevant and that the product can be transported or temporarily stored between two steps.*

The software infrastructure that has been used in our grid is agent-based. Agent technology opens the possibilities to let this grid operate and manufacture different kinds of products in parallel, provided that the required production steps are available [3]. The manufacturing machines that have been built in our research group are cheap and versatile. These machines are called equiplets and consist of a standardized frame and subsystem on which several different front-ends can be attached. The type of front-end specifies what production steps a certain equiplet can provide. This way every equiplet acts as a reconfigurable manufacturing system (RMS) [17]. An example of an equiplet front-end is a delta-robot. With this front-end, the equiplet is capable of pick and place actions. A computer vision system is part of the frontend. This way the equiplet can localise parts and check the final position they are put in. A picture of an equiplet with a delta-robot front-end is shown in Figure 1. For a product to be made a sequence of production steps has to be done. More complex products need a tree of sequences, where every sequence ends in a half-product or part, needed for the end product. The actual production starts at the branches of the tree and ends at the root. The equiplet is represented in software by a so-called equiplet agent. This agent advertises its capabilities as production steps on a blackboard that is available in a multiagent system where also the so-called product agents live. A product agent is responsible for the manufacturing of a single product and knows what to do, the equiplet agents knows how to do it. A product agent selects a set of equiplets based on the production steps it needs and tries to match these steps with the steps advertised by the equiplets. The planning and scheduling of a product is an atomic action, done by the product agent in cooperation with the equiplet agent and takes

Figure 1: An equiplet with a delta-robot frontend.

seven stages [4]. First, Let us assume that a single sequence of steps is needed:

1) From the list of production steps, build a set of equiplets offering these steps;
2) Ask equiplets about the feasibility and duration of the steps;
3) Indicate situations where consecutive steps on the same equiplet are possible;
4) Generate at most four paths along equiplets;
5) Calculate the paths along these equiplets;
6) Schedule the shortest product path using first-fit (take the first opportunity in time for a production step) and a scheduling scheme known as earliest deadline first (EDF) [4];
7) If the schedule fails, try the next shortest path.

For more complex products, consisting of a tree of sequences, the product agent spawns child agents, which are each responsible for a sequence. The parent agent is in control of its children and acts as a supervisor. It is also responsible for the last single sequence of the product. In Figure 2, the first two halfproducts are made using step sequences $< \sigma_1, \sigma_2 >$ and $< \sigma_3, \sigma_4 >$. These sequences are taken care of by child agents, while the parent agent will complete the product by performing the step sequence $< \sigma_4, \sigma_7, \sigma_2, \sigma_1 >$.

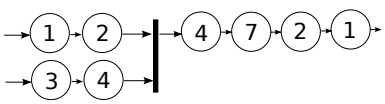

Figure 2: Manufacturing of a product consisting of two half-products.

Every product agent is responsible for only one product to be made. The requests for products arrive at random. In the implementation we have made, a webinterface helps the end-user to design or specify his or her specific product. At the moment, all features are selected a product agent will be created. During manufacturing a product is guided by the product agent from equiplet to equiplet. In Figure 3,

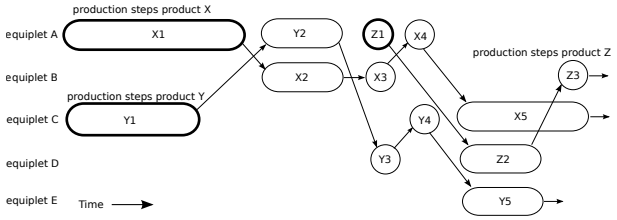

Figure 3: Three products in production.

the situation is shown for three products X, Y and Z using five different equiplets (A, B, C, D, E). Production step i of product X is denoted by Xi. From Figure 3 the following properties become clear:

- Production steps can differ in length as opposed to batch processing, where every step in the production line should normally take the same amount of time;
- Production can start at random;
- It will be unlikely that all equiplets are used at 100%;
- A failing production step on a product will not block the whole manufacturing process of other products as in a production pipeline used in mass production;
- Products have their own unique paths along the equiplets.

The path the product has to follow during manufacturing will in general be a random walk along the equiplets. Figure 4 gives an impression of such a random walk.

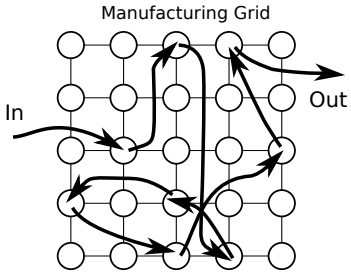

Figure 4: Random walk of a product in the grid.

This random walk is more efficient when the equiplets are in a grid arrangement against a line arrangement as used in batch processing. Some calculations on the average number of hops has been done for a random path between nodes on a line, on a circle and in a grid. In Figure 5, the number of hops is plotted against $\sqrt{N}$, where N is the number of nodes among the line, the circle or in the grid. The increase of the average path length (number of hops) is the highest for nodes put on a line. So a random walk along a line is behaving bad, when the number of nodes increases.

Figure 6 shows the global system architecture. The multiagent system is a distributed system consisting of computers belonging to the equiplet hardware where the equilet agents (EqA) live and some general computer platforms. The general computer platforms contain the product agents (PA) and blackboards that are used for sharing information that

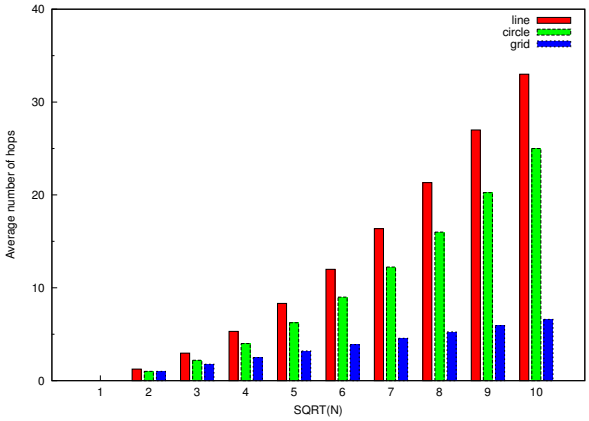

Figure 5: Number of hops for different configurations of N nodes.

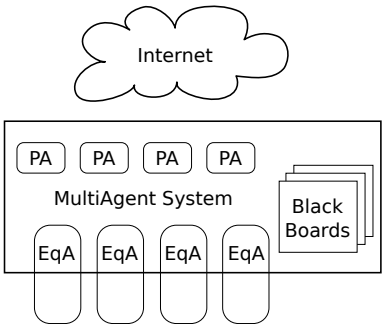

Figure 6: Global architecture of the software infrastructure.

should be available for all agents. The platforms are connected using standard Ethernet. New request for products to be built are received from the internet. Such a request will spawn a new product agent, that will plan its production path and schedule its production using the information available on the blackboards as described earlier in this section.

IV. Transport in the Grid

In the production grid, there is at least the stream of products to be made. Another stream might be the stream of raw material, components or half-products used as components. We will refer to this stream as the stream of components. These components could be stored inside the equiplets, but in that case there is still a stream of supply needed in case the locally stored components run short. This increases the logistic complexity of the grid model. In the next subsections, models will be introduced that alleviate the complexity by combining the stream of products with the stream of components.

A. Building box model

In the building box model, a tray is loaded with all the components to create the product. To maintain agility, this set of components can be different for every single product. Before entering the grid, the tray is filled by passing through a pipeline with devices providing the components. In this phase a building box is created that will be used by the grid to assemble the product. The equiplets in the grid are only used for assembling purposes. Figure 7 shows the setup.

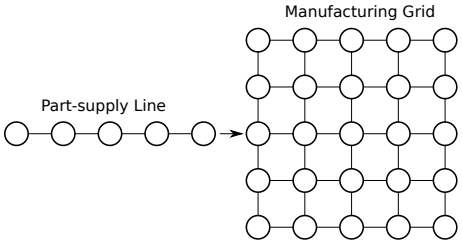

Figure 7: Production system with supply pipeline.

A problem with the previous setup is the fact that more complex products should be built by combining subparts that should be constructed first. In the previously presented setup all parts needed for the construction of the subparts should be collected in the building box, making the assembling process more complicated. Another disadvantage of putting all components for all subparts together in a building box is that this slows down the production time, because normally subparts can be made in parallel. A solution is shown in the setup of Figure 8. Subparts can be made in parallel and are input to the supply-line that eventually could be combined with the original supply-line.

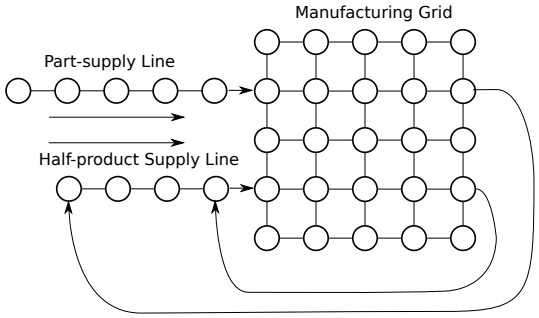

Figure 8: Production system with loops.

The next refinement of the system is presented in Figure 9. Here a set of special testnodes has been added to the system. These nodes are actually also equiplets, but these equiplets have a front-end that makes them suited for testing and inspecting final products as well as subparts that should be used for more complicated products.

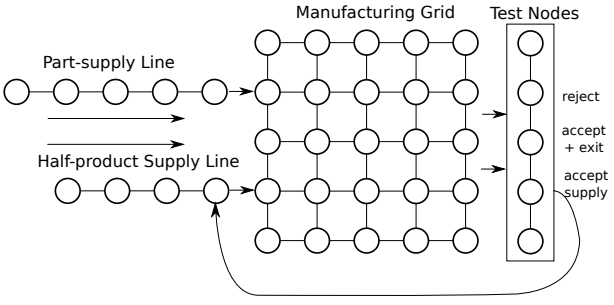

Figure 9: Production system with tests and loops.

A test can also result in a reject and this will also inform the product agent about the failure. If the product agent is a child

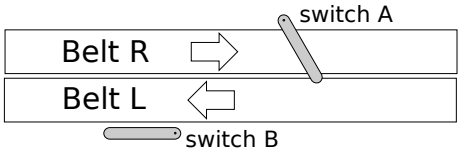

Figure 10: Bidirectional conveyor belt with switches.

agent constructing a subpart, it should consult the parent agent if a retry should be done. In case it is the product agent for the final product, it should ask its maker what to do.

B. Conveyor belt-based systems

A conveyor belt is a common device to transport material. Several types are in use in the industry. Without going into detail, some kind of classification will be presented here:

- belts for continuous transport in one direction;
- belts with stepwise transport from station to station. These types of belts can be used in batch environments, where every step takes the same amount of time and the object should be at rest when a production step is executed;
- belts with transport is two directions. This can also be realised by using two one direction belts, working in opposite direction.

In Bussmann [7], an agent-based production system is built using transport belts in two directions where a switch mechanism can move a product from one belt to another. A special switch-agent is controlling the switches and thus controlling the flow of a product along the production machines. In Figure 10 this solution is shown. Switch A is activated and will shift products from belt R to the belt L that will move it to the left. This concept fits well in the system developed by Bussmann, because that system is actually a batch-oriented system. In a grid the use of conveyor belts might be considered, but for agile transport several problems arise, giving rise to complicated solutions:

- should the direction in the grid consist of one-way paths or should be chosen for bidirectional transport?
- a product should be removed from the moving belt during the execution of a production step. A stepwise transport is inadequate, because of the fact that production steps can have different execution times in our agile model. This removal could be done by a switch mechanism as used by Bussmann, but every equiplet should also have it own switch-unit to move the product back to the belt.
- because the grid does not have a line structure for reasons explained in the first part of this paper, a lot of crossings should be implemented. These crossings can also be realised with conveyor belt techniques, but it will make the transport system as a whole expensive and perhaps more error-prone.

C. Transport by automatic guided vehicles

An alternative for conveyor belts is the use of automatic guided vehicles (AGV). An AGV is a mobile robot that follows certain given routes on the floor or uses vision, ultrasonic sonar or lasers to navigate. These AGVs are already used in industry mostly for transport, but they are also used as moving assembly platforms. This last application is just what is needed in the agile manufacturing grid. The AGV solution used to be expensive compared to conveyor belts but some remarks should be made about that:

- The AGVs offer a very flexible way for transport that fits better in non-pipeline situations;
- Low cost AGV platforms are now available;
- From the product agent view, an AGV is like an equiplet, offering the possibility to move from A to B.
- A conveyor-belt solution that fits the requirements needed in grid productions will turn out to be a complicated and expensive system due to the requirements for flexible transport.

In the grid, a set of these AGVs will transport the product between equiplets and will be directed to the next destination by product agents.

1) AGV system components: An AGV itself is a driverless mobile robot platform or vehicle. This AGV is mostly a battery-powered system. To use an AGV, a travel path should be available. When more than one AGV is used on the travel path, a control system should manage the traffic and prevent collisions between the AGVs or prevent deadlock situations. The control system can be centralised or decentralised.

2) AGV navigation: There are plenty ways in which navigation of AGVs has been implemented. The first division in techniques can be made, based on whether the travel path itself is specially prepared to be used by AGVs. This can be done by:

- putting wires in the path the AGV can sense and follow;
- using magnetic tape along the path to guide the AGV;
- using coloured paths, by using adhesive tape on the path to direct the AGV;
- using transponders, so the AGV can localise itself.

The second type of AGV does not require a specially prepared path. In that case navigation is done by using:

- laser range-finders
- ultrasonic distance sensors
- vision systems

Though it might look as if the decision for using AGVs has already been made, further research should be done to see what the efficiency will be for several implementations. This will be the subject of the next two sections.

V. SOFTWARE TOOLS

Two simulation software packages have been built. A simulation of the scheduling for production and a simulation for the path planning. The path planning tool will be used to calculate the efficiency for different transport interconnections. The scheduling tool will be used to calculate the number of active product agents within the grid. This number is important, because it will tell how many products should be temporally stored, waiting for the next production step to be executed.

A. Path planning simulation software

A path planning tool has been built, to calculate a path a certain product has to follow along the equiplets. The Dijkstra path algorithm has been used [18]. The tool can work on different grid transport patterns. This tool will be used to study several possible grid topologies. A screen-shot of the graphical user interface of the tool is shown in Figure 11. Several different topologies and interconnections can be chosen by clicking the appropriate fields in the GUI. The average transport path for all nodes is one of the results of this simulation.

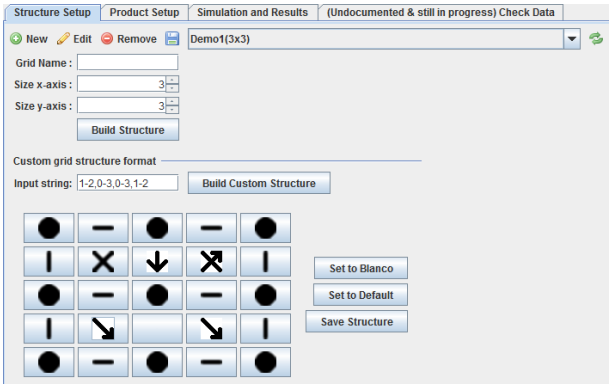

Figure 11: Path planning GUI.

B. Scheduling simulation software

The software for the scheduling simulations consists of two parts. One part is a command-line tool that is driven by a production scenario of a collection of product agents, each having their own release time, deadline and set of production steps. This production scenario is a human readable XML-file. The second part is a GUI for visualisation of the scheduling system. In Figure 12, a screen shot of this visualisation tool is shown.

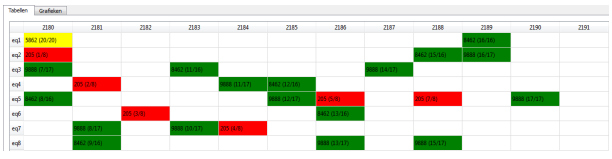

Figure 12: GUI of the scheduling simulator.

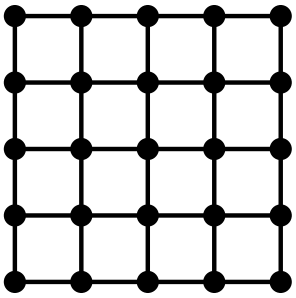

Figure 13: Standard fully connected grid.

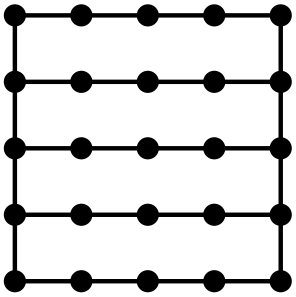

Figure 14: Grid with bidirectional lanes and bidirectional backbone lanes.

VI. RESULTS

This section shows two types of results. First, the transport possibilities are investigated using the path planning tool. Next, the results of the scheduling simulations are discussed.

A. Transport possibilities

To calculate the average pathlength in the grid for different paths, several structures have been investigated. Some of these structures were chosen to fit conveyor belt solutions of some type. All structures will also fit within the AGV-based solution.

1) A fully connected grid. where all paths are bidirectional paths as in Figure 13.
2) A grid where all paths are bidirectional, but this design has removed the crossings as in Figure 14. This structure could be implemented by conveyor belts in combination with switches;
3) A structure with five unidirectional paths and two bidirectional paths as in Figure 15. This structure is also a possible implementation with conveyor belts;
4) A structure with bidirectional paths combined in a single backbone as in Figure 16;
5) A structure with five bidirectional paths and two unidirectional paths as in Figure 17;
6) A fully connected grid, but now with half of the paths unidirectional as in Figure 18.

For all these structures the average path is the result from a simulation of 1000 product agents, all having a random walk within the grid. Each product agent has an also random set of equiplets it has to visit ranging from 2 to 50 equiplets per product agent. Every path or hop between adjacent nodes is considered to be one unit length. If the paths have no crossings, a conveyor belt might be used, because crossing belts will

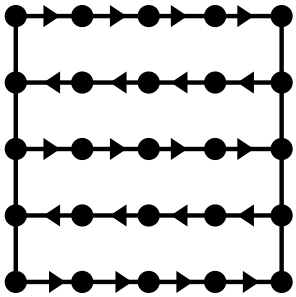

Figure 15: Grid with unidirectional lanes and bidirectional backbone lanes.

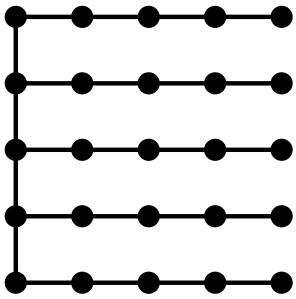

Figure 16: E-shaped connection, with bidirectional lanes.

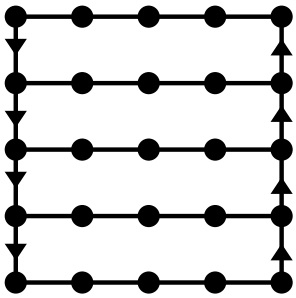

Figure 17: Bidirectional lanes with unidirectional backbones.

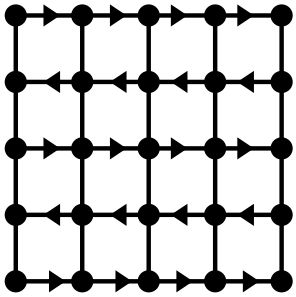

Figure 18: Fully connected grid with unidirectional lanes.

result in a more complex system. All structures can also be implemented with AGVs. For some structures the average path can also easily be calculated and the results of these exact calculations are comparable with the simulation results.

The results of the simulation are given in a table and also plotted as a histogram in Figure 19. In Table I, a second outcome from the simulation is also shown. This is the percentage of agents that could find an alternative path of

TABLE I: Results of the simulation.

Structure	1	2	3	4	5	6
Average path	3.2	3.9	6.4	5.1	6.0	3.6
% Alternatives	60	16.7	8.4	0	0	27

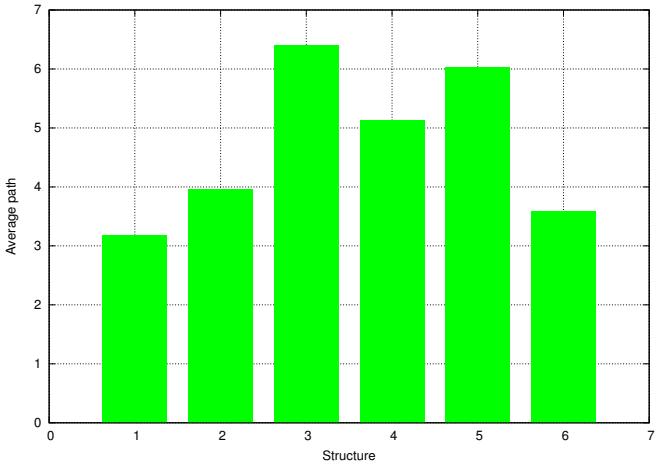

Figure 19: Simulation results for different structures.

the same length. This result is of interest when in a traffic control implementation, alternative paths become important.

As could be expected, the best result is achieved in the fully connected grid with bidirectional paths. Changing the grid to an almost identical structure of Figure 18 with unidirectional paths, results in only a small penalty. This structure could also be useful in an AGV-based transport system, reducing collision problems because of the one-way paths used. Both structures also offer a relative high percentage of alternative paths, that could also be useful in an AGV-based system. The structures that fit a conveyor belt solution show a path length that is considerably higher.

When AGVs are used, the architecture of the production system slightly changes. The solution fits well in the software infrastructure. For the product agent, the AGV can be seen as just another equiplet, but instead of providing production steps, transport in the grid is offered. Another difference is that the product agent will be tied to an AGV during the whole production. The resulting architecture is shown in Figure 20. The AVGs are represented by transport agents (TA). During the execution of a production step, the equiplet agent can also cooperate with the transport agent to put the product on the AGV in the right position. This might come handy when a product is too large for the equiplet to handle by itself.

B. Scheduling

The next results were generated using the scheduling tool. This tool was used in earlier research [4] to discover the scheduling approach to be used. The scheduling is based on timeslots, having a certain duration. Such a timeslot is the minimal allocatable unit of time. The methods used for scheduling were derived from real time scheduling schemes adapted to the multiagent environment. To explain these schemes, some symbols used in expressions should be defined:

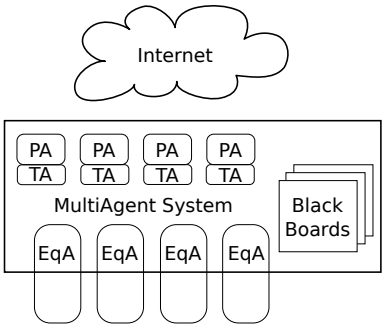

Figure 20: Global architecture with transport.

P is the product set. A single product is denoted as P_i.

r_i is the first timeslot after release of product P_i

d_i is the timeslot for the deadline of product P_i

τ is the current timeslot

$s_i(\tau)$ is the number of timeslots of product P_i that is left to be done.

Five well-known scheduling schemes are as follows.

1) Fixed priority, FP. Every task is assigned a priority depending on the task type. The highest priority tasks are completed before the lower priorities are run.
2) Earliest deadline first, EDF. The task with the first deadline to come gets the highest priority and is handled first.
3) Least slack first, LSF. The task with the minimum slack gets the highest priority and is handled first. Slack is defined as the total time available until the deadline minus the time to complete the task. The slack for product P_i at timeslot τ can be written as $(d_i - \tau) - s_i(\tau)$.
4) Smallest critical ratio first, CR. The critical ratio is defined as the total number of timeslots available divided by the number needed. For a product P_i at timeslot τ: $(d_i - \tau)/s_i(\tau)$. If this number turns out to be 1, all timeslots should be used. If it is lower than 1, the scheduling is infeasible. A high number shows that many slots are available for a relative small number of needed timeslots.
5) Shortest process first, SPF. The task with the shortest time to complete gets the highest priority.

All these types can be used in conjunction with what is called preemption. By this is meant that when a higher priority task arrives, another already running task is paused (preempted) to make way for the higher priority task. After completion of the higher priority task, the preempted task is resumed. Because all agents are equal, fixed priority is inadequate as a scheduling scheme for the production grid. Both EDF and LSF are considered optimal in the sense of feasibility: this means that if there exists a feasible schedule, EDF or LSF will find it [19]. However, this is only true for the situation where a single resource is scheduled among requesters, as is the case in a single processor computer system, where the processor time is scheduled among different tasks.

The adjustments that has been made to adapt the scheduling schemes to the situation of the production grid had to do with preemption and the way a feasible scheduling and a failing scheduling are treated. For scheduling in the grid the following list of objectives has been worked out:

1) It should offer a best effort to schedule products that will arrive at random times.
2) It should schedule products at high grid loads.
3) It should be fast and reliable. The scheduling should take a small amount of time.
4) It should introduce only a small intercommunication overhead. This will mean that the amount of inter-agent messages should be kept low.
5) It should be fair. When a product is scheduled for production with a feasible scheduling, meaning the product will be completed before the deadline, the scheduling system should be designed such that it will guarantee that the feasible scheduling will not be changed to an infeasible scheduling at a later time by the scheduling system.

In [4], two approaches for preemption are introduced. To describe the differences EDF will be used as an example. If a product arrives at time t with deadline T_d, two scenarios are possible:

1) All products with a later deadline will temporarily give up their schedules starting from t to make way for the newly arrived product. This product will be scheduled and the products that gave up their scheduling try to reschedule within the constraints for their deadlines according to the EDF scheme. However, if one of these reschedules fails, objective 5 of the scheduling system is not achieved. This will result in reporting a scheduling failure for the newly arrived product and a restore of the schedules of the already active products. This approach is called the strong approach.
2) The newly arrived product will first try to schedule its production without disturbing the other products. Only if it fails it will follow the schedule and reschedule approach mentioned in the first scenario. This approach is called the weak approach.

The implementation of these two approaches is done by sending broadcast messages as well as agent to agent messages. In case of an infeasible schedule for the newly arrived agent, this agent will broadcast its deadline to all active product agents. In reply to this broadcast, agents with a later deadline will send their claimed production steps to the new agent and this will try to schedule its production assuming these claimed steps are now available. If the scheduling succeeds it will try to reschedule all other active agents. In case of success it will adjust the scheduling information on the blackboard involved with the new scheduling and send the new schedules to the participating agents. By locking the access to the blackboard by other agents during (re)scheduling, this scheduling action is atomic.

In [4], it is shown that the approaches weak and strong result in almost the same rate of successful schedules, having an average difference in results below 0.5%. Both approaches

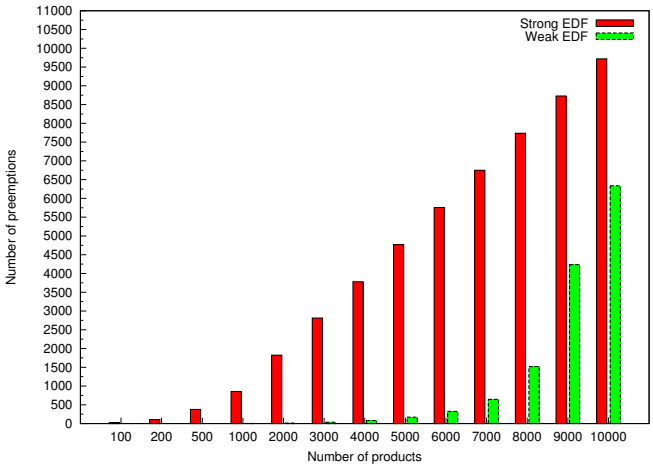

Figure 21: Number of preemptions for strong and weak EDF.

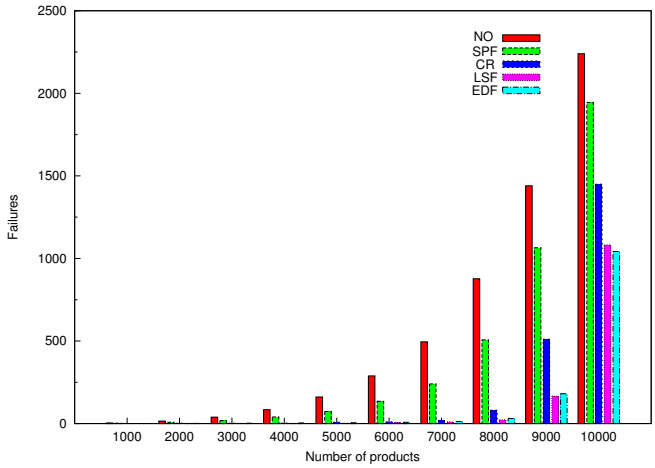

Figure 23: Failure-count for different scheduling algorithms.

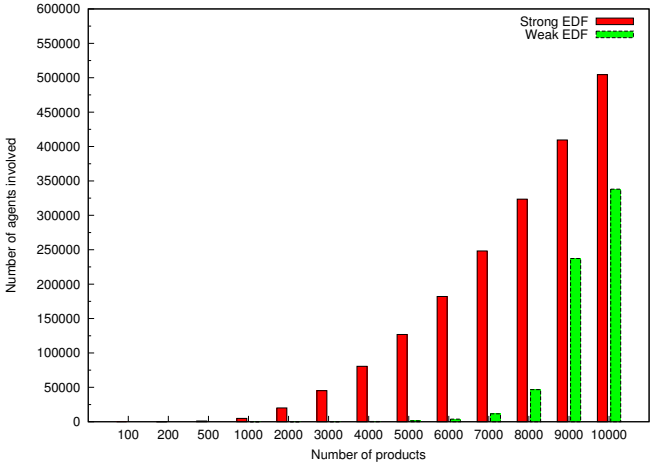

Figure 22: Number of agents involved for strong and weak EDF.

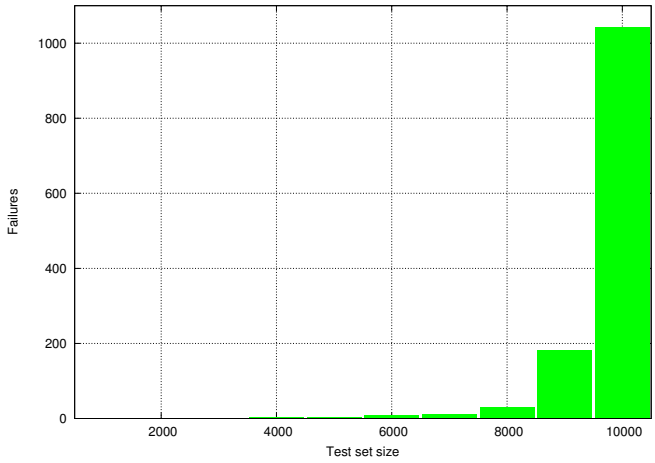

Figure 24: Simulation Results for different sizes op product sets.

fulfill objectives 1 and 2. In Figure 21, the number of preemptions for both strong and weak versions of EDF are plotted for different sizes of test sets. Another important value that gives and impression of the inter-agent communication overhead is shown in Figure 22. From both figures it becomes clear that the amount of overhead in inter-agent communication and rescheduling calculations used by the strong version is much higher than the weak version. Objectives 3 and 4 are more feasible using the weak version. Both versions were already by design compliant with objective 5. When we consider the different scheduling algorithms mentioned before, leaving out fixed priority, and using the weak approach the resulting number of failures for different numbers of products are plotted in Figure 23. Earliest deadline first (EDF) turned out to be a good choice. The success rate is comparable to LSF, but the advantage of EDF over LSF is that the deadline is a constant value while the slack changes over time and has to be recalculated, introducing an extra small overhead.

For the previous and coming simulations, the following conditions were used:

- Every product agent has a random number of equiplets to visit ranging from 1 to 20 with an average of 10 equiplets.
- The number of timesteps in a simulation run is 10000 and the duration of a production step is 1 timestep.
- The time window between release time and deadline of a product is random between 1 to 20 times the total production time of a product. This total production time in timesteps is in this case equal to to number of equiplets the product agent has to visit.
- Each equiplet is offering a single unique production step.

For a set with 10000 product agents, each having an average number of 10 production steps, the amount of production steps needed is 100000. During 10000 timesteps the grid consisting of 10 equiplets is actually offering a maximum of 100000 production steps. Figure 24 shows that in the given situation of 10000 product agents the number of scheduling failures is over 1000. In Figure 25, the average number of product agents in a manufacturing grid consisting of 10 equiplets is shown for different sizes of test sets. This information shows

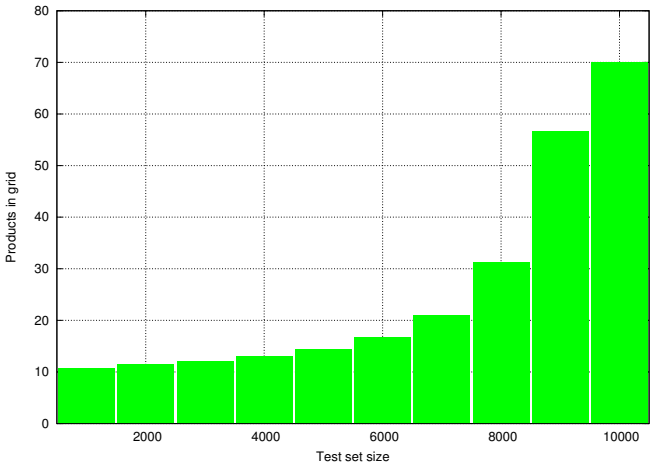

Figure 25: Simulation Results for different sizes op product sets.

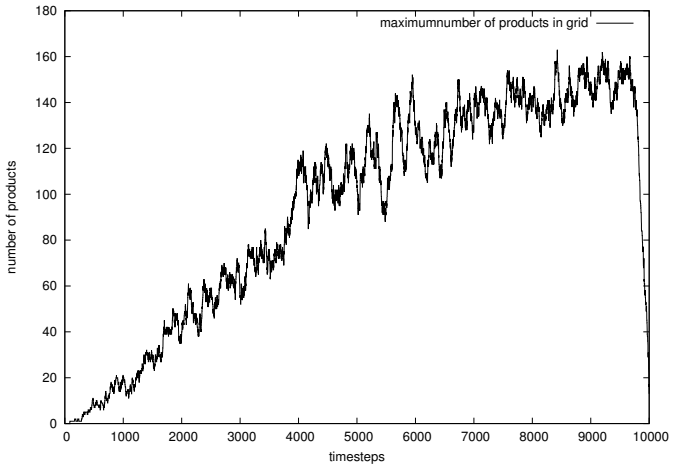

Figure 27: Maximum number of active products in the grid.

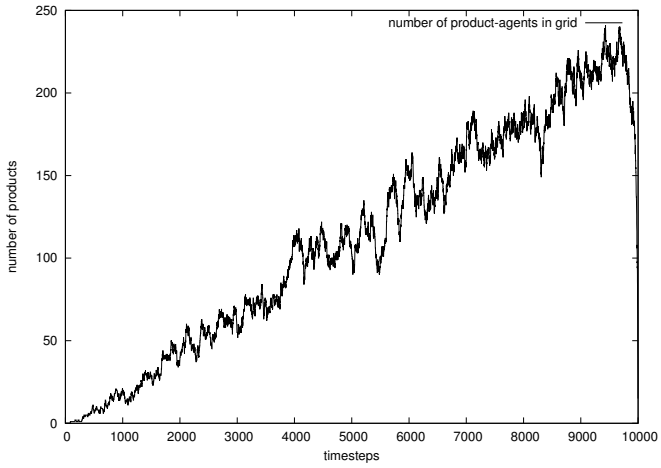

Figure 26: Increasing number of active products in the grid.

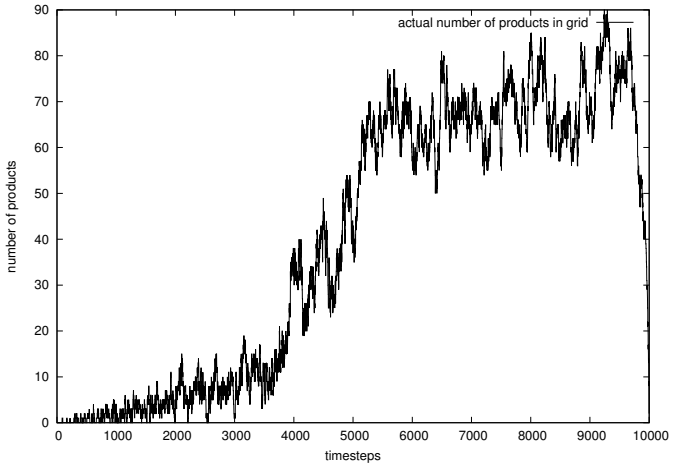

Figure 28: Actual number of products in the grid.

that the number of products exceeds the number of equiplets by far above 8000 products. Given the aforementioned test conditions, a number of 8000 products compares to a load of 80%. At a load of 80% the number of products, which are not handled by equiplets and have to wait, is two times the number of equiplets, resulting in an average of two AGVs waiting for service by an equiplet.

Another simulation has been set up to study the behaviour of the grid under increasing load. This simulation is based on a scenario with a linear increasing amount of product agents in time as shown in Figure 26. In this situation, a product is considered active in the grid between its release time and its deadline. Because of the fact that at the end of the graph, the grid is actually overloaded, the maximum number of active agents in the grid will not increase due to the fact that more and more products will have an infeasible scheduling and will not contribute to the number of products in the grid. This effect is shown in Figure 27. The actual number of products in the grid is shown in Figure 28. This graph is not based on release time and deadline, but on release time and time of completion. When we look at the actual number of active products in the grid, the resulting graph shows an remarkable shape. In the beginning, the actual number is even less than the number plotted in the graph of Figure 26. This is due to the fact that in a grid that is only used by a small amount of product agents, every product will be finished far before its deadline. A finished product is now not considered active in the grid any more. However, at a certain point there is a steep increase in the number of products and the graph saturates at the same level of 70 products as shown in Figure 25 for a test set of 10000. The number of rejected products due to a failing scheduling will increase. This also means that overloading the grid will generate many active products that should be stored somewhere, because in this given situation, only 10 products can be handled by an equiplet.

VII. Conclusion

For the agile grid-based and agent-based manufacturing the buildingbox as well as the AGV-based system offer advantages:

- By using a building box, the transport of parts to the assembling machines (equiplets) is combined with the transport of the product to be made. It will not happen that a part is not available during manufacturing;

- Because the product as well as it parts use one particular AGV during the production, there is never a competition for AGV during the manufacturing process;
- An AGV can use the full possibility and advantage of the grid-based system being a compact design resulting in short average paths;
- The product agent knows which equiplets it should visit and thus can use the AGV in the same way as an equiplet. The product agent can instruct the AGV agent to bring it to the next equiplet in the same way as it can instruct an equiplet agent to perform a production step;
- An AGV can bring the production platform exact to the right position for the equiplet and can even add extra movement in the X-Y plane or make a rotation around the Z-axis;
- If an AGV fails during production the problem can be isolated and other AGVs can continue to work. In a conveyor belt system a failing conveyor might block the whole production process.

There are also some disadvantages:

- There should be a provision for charging the battery of the AGV.
- Simulations show that the amount of agents in the grid shows a strong increase in a grid that is loaded over 80%. This will result in a lot of AGVs in the grid leading to traffic jam;
- Only products that fit within the building box manufacturing model can be made.

Agent-based grid manufacturing is a feasible solution for agile manufacturing. Some important aspects of this manufacturing paradigm have been discussed here. Transport can be AGV-based provided that the load of the grid should be kept under 80% to overcome the temporary storage requirements and problems with the communication and rescheduling overhead.

ACKNOWLEDGEMENT

The authors would like to thank Mathijs Kuijl, Bas Alblas, Jaap Koelewijn, Pascal Muller, Lars Stolwijk, Roy de Kok and Martijn Beek for their effort and contributions in developing the software tools.

REFERENCES

[1] L. v. Moergestel, E. Puik, D. Telgen, M. Kuijl, B. Alblas, J. Koelewijn, and J.-J. Meyer, "A simulation model for transport in a grid-based manufacturing system," Proceedings of the Third International Conference on Intelligent Systems and Applications (INTELLI 2014), Sevilla, Spain, 2014, pp. 1–7.

[2] E. Puik and L. v. Moergestel, "Agile multi-parallel micro manufacturing using a grid of equiplets," Proceedings of the International Precision Assembly Seminar (IPAS 2010), Chamonix, France, 2010, pp. 271–282. Springer ISBN-13: 978-3642115974.

[3] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing," Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2011) Kobe, Japan, 2011, pp. 281–288.

[4] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Production scheduling in an agile agent-based production grid," Proceedings of the Intelligent Agent Technology (IAT 2012), Macau, 2012, pp. 293–298.

[5] M. Paolucci and R. Sacile, Agent-based manufacturing and control systems: new agile manufacturing solutions for achieving peak performance. Boca Raton, Fla.: CRC Press, ISBN-13: 978-1574443363, 2005.

[6] E. Montaldo, R. Sacile, M. Coccoli, M. Paolucci, and A. Boccalatte, "Agent-based enhanced workflow in manufacturing information systems: the makeit approach," J. Computing Inf. Technol., vol. 10, no. 4, 2002, pp. 303–316.

[7] S. Bussmann, N. Jennings, and M. Wooldridge, Multiagent Systems for Manufacturing Control. Berlin Heidelberg: Springer-Verlag, ISBN-13: 978-3642058905, 2004.

[8] N. Jennings and S. Bussmann, "Agent-based control system," IEEE Control Systems Magazine, vol. 23, no. 3, 2003, pp. 61–74.

[9] D. Ouelhadj, C. Hanachi, and B. Bouzouia, "Multi-agent architecture for distributed monitoring in flexible manufacturing systems (fms)," Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, CA, USA, 2000, pp. 2416–2421.

[10] W. Xiang and H. Lee, "Ant colony intelligence in multi-agent dynamic manufacturing scheduling," Engineering Applications of Artificial Intelligence, vol. 16, no. 4, 2008, pp. 335–348.

[11] L. van Moergestel, E. Puik, D. Telgen, H. Folmer, M. Grünbauer, R. Proost, H. Veringa, and J.-J. Meyer, Enhancing Products by Embedding Agents: Adding an Agent to a Robot for Monitoring, Maintenance and Disaster Prevention, ser. Communications in Computer and Information Science, J. Filipe and A. Fred, Eds. Springer Berlin Heidelberg, 2014, vol. 449, ISBN-13: 978-3662444399.

[12] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," Journal on Engineering Applications of Artificial Intelligence, vol. 22, issue 7, pp. 979–991, Pergamon Press, Inc. Tarrytown, NY, USA, 2009.

[13] A. Koestler, The Ghost in the Machine. Arkana Books, London, 1969, Reprinted 1990, Penguin Books, ISBN-13: 978-0140191929.

[14] S. Bussmann and D. McFarlane, "Rationales for holonic manufacturing control," Proceedings of the second international workshop on intelligent manufacturing systems, 1999, pp. 177–184.

[15] K. Fisher, "Agent-based design of holonic manufacturing systems," Robotics and Autonomous Systems, vol. 27, no. 1-2, 1999, pp. 3–13.

[16] J. Müller, The design of intelligent agents: a layered approach. Springer, 1996, ISBN 978-3540620037.

[17] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," International Journal of Production Research, vol. 46, no. 4, 2008, pp. 599–620.

[18] M. Sniedovich, "Dijkstras algorithm revisited: the dynamic programming connexion," Control and Cybernetics, vol. 35, no. 3, 2006, pp. 599–620.

[19] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri, Scheduling in Real-Time Systems. Chichester, West Sussex: John Wiley and sons, 2002, ISBN-13: 978-0470847664.