Simulation of Multiagent-based Agile Manufacturing

Leo van Moergestel, Erik Puik, Daniël Telgen, Maarten Aalbers, Stefan Marchal, Frank van Rooden, Raymond Siudas and Wim Wakelkamp
Department of microsystem technology
Utrecht University of Applied Sciences
Utrecht, the Netherlands
{leo.vanmoergestel, erik.puik, daniel.telgen}@hu.nl

Abstract - At the Utrecht University of Applied Science we are developing an agent-based software infrastructure for agile industrial manufacturing of microdevices. This manufacturing is done on special devices called equiplets. A grid of these equiplets connected by a fast network is capable of producing an variety of different products in parallel. The multi-agent-based software infrastructure is responsible for the agile manufacturing. To test our concepts, a simulation was build based on the Jade framework, a Java-based framework for distributed multiagent systems.

Index Terms – Software agents, Multi-agent systems, Agile manufacturing, Simulation.

I. INTRODUCTION

The requirements of modern production systems are influenced by new demands like time to market and customer-specific small quantity production. In other words the transition time from product development to production should be minimal and small quantity production must be cheap. To fulfill these requirements new production methods should be developed. This new approach means new production hardware as well as co-designed software. At the Utrecht University of Applied Science we have developed special production platforms that are cheap, agile and easy configurable [1]. These platforms can operate in parallel. We call these platforms equiplets and a collection of these equiplets is called a production grid. The idea behind this concept is that we need a production system that is capable of producing a lot of different products in parallel. This is what we call multiparallel manufacturing.

The software infrastructure for such a production grid is highly responsible for this agile and diverse way of production. In ref. [2] an agent based software infrastructure is described. To test the concepts presented in that paper, a test environment should be built to see how such an agent-based production system will behave.

This paper describes the agent-based production system, the basics of the simulation environment as well as the first results.

II. AGENTS

Because our model is agent-based, this section will give a short introduction to agent technology. There are many definitions of what an agent is. We use here a commonly accepted definition by Wooldridge and Jennings [3]: An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.

A. Agent types

We will concentrate on some aspects that are important for our software architecture. Literature and papers about agents introduce among others, two types of agents:

1. reactive agents
2. reasoning agents

A reactive agent senses the environment and acts according to the information it gets from this sensing. There is no internal state involved. A reasoning agent also senses its environment but does have an internal state. Depending on the sensing input and the internal state it will search for an action to perform, one could say it will reason for the action to perform. The sensing input will also change the internal state. A special type of reasoning agent is the so-called belief-desire-intention-agent or BDI-agent [4].

- from the inputs of its sensors the agent builds a set of beliefs. Beliefs characterize what an agent imagines its environment state to be;
- desires describe agents preferences;
- intentions characterize the goals or desires the agent has selected to work on.

An agent is equipped with a set of plans. These plans have three components:

1. the postcondition or goal of the plan;
2. the precondition of the plan;
3. the course of action to carry out.

An agent will deliberately choose a plan to achieve its goals.

B. Multiagent systems
A multi-agent system (MAS) consists of two or more interacting autonomous agents. Such a system is designed to achieve some global goal. The agents in a multi-agent system should cooperate, coordinate and negotiate to achieve their objectives. When we consider the use of a multi-agent system, we should specify abstract concepts such as:

- **role:** what is the role of a certain agent in a multi-agent system. Perhaps an agent has more than one role;
- **permission:** what are the constraints the agent is tied to;
- **responsibility:** i.e. the responsibility an agent has in achieving the global goal;
- **interaction:** agents interact with each other and the environment.

The system described in the next section is based on a MAS. The actual operation of the system is due to the fact that agents have their roles in a complex environment and try to achieve its goals and are willing to assist other agents in achieving their goals.

### III. Multi-Agent Production System

The production system consists of a grid of so-called equiplets. An equiplet is a cheap basic production unit with an onboard computer system for control and communication. Equiplets get a front-end. This is part of the initial grid hardware configuration. A frontend is a production hardware add-on for the equiplet that enables it to perform production actions. The frontend will determine what kind of production actions, resulting in production steps, an equiplet can perform.

Let us assume that the grid offers a set \( S_{\text{grid}} \) of \( N \) production steps \( \sigma_1, \ldots, \sigma_N \). An equiplet with a certain frontend offers a set of production steps that is a subset of \( S_{\text{grid}} \).

To make a product, a certain set of production steps should be available. This means that the set of needed production steps for a product is also a subset of \( S_{\text{grid}} \). Because for a product the order of production steps is important, the product is characterized in its simplest form by a tuple of production steps: i.e. \( \langle \sigma_4, \sigma_7, \sigma_2, \sigma_1 \rangle \).

The control of an equiplet is done by an equiplet agent. This agent is responsible for a certain equiplet with its frontend. It interacts with the production hardware, other agents in the grid and possibly, in a semiautomated environment, with a human equiplet operator. The goal of an equiplet agent is to offer and perform production steps. To meet this goal, the equiplet hardware should be in good condition and raw material for production should be available. An equiplet agent in the grid will publish its production steps in a global by other agents accessible place. Now the so-called product agent comes into the picture. This agent is responsible for the product to be made. Its goal is a finished product.

A product agent chooses an equiplet based on the set of production steps published by the equiplet. It will direct the still unfinished product to the equiplet and log the production data resulting from the production step or steps; thus creating a production log. When the product agent needs steps that are unavailable on the current equiplet it will search for another equiplet and direct the product to that equiplet and so on until the final production step is completed and the product will be removed from the grid. In this way we can produce different products in parallel as long as the set of production steps offered by the grid fits to the needs of the products to be made.

The scheduling system for the grid should take these constraints into account:

- equiplet agents offer production steps. They can only perform one production step at a time;
- product agents need production steps, but these steps can only be executed by equiplets;
- the product agent will indirectly be bound to the equiplet agent, because it is the production step that is the resource the product agent is interested in;
- when an equiplet agent is executing a production step, all other possible production step for that equiplet are unavailable at that moment;
- a production step can have parameters;
- item the time of a production step is variable or fixed and may depend on the production step parameters.

Fig. 1 shows the way scheduling is accomplished by the participating agents. As explained earlier, equiplet agents publish their set of production steps. This global accessible information is called a blackboard. The product agents choose the right equiplet agents to build the product they represent.

A product agent carries in its knowledge base all the production steps that are needed to complete the product together with the path (order) through these steps. XML is apt to express such a step overview. At the moment a product agents come to life, it will try to plan its production path among the equiplets. The reason why it plans the whole path is that the product agent wants to know in advance if the production is within the production deadline. If the production path is planned and feasible, the corresponding equiplets will be claimed by the product agent. If the deadline is not feasible, the product agent will negotiate with other product agents to see if they are willing to adjust their scheduling to give away a claim on a certain equiplet.
IV. Simulation

The simulation software uses Jade [5] as a platform. The reasons for choosing Jade are:

- the simulation is a multiagent-based system. Jade provides most the requirements we need for our application like platform independence and inter agent communication;
- Jade is Java-based. Java is a versatile and powerful programming language;
- because Jade is Java-based it also has a low learning curve for Java programmers;
- in this simulation at least the equiplet agents are not that intelligent that we need special multi-agent environments. The product agents should be capable to negotiate to reach their goals. But the Jade platform can easily be upgraded to an environment that is special designed for BDI agents like 2APL [5] or Jadex [5]. Both 2APL as well as Jadex have a more steep learning curve for Java developers.

To supply simulation parameters to our software, XML is used as a structured language. An XML file, that drives the simulation, is called a scenario.

The results of the simulation will be displayed on an monitoring screen and stored in a file that can be used for further examination in a spread-sheet application.

To realize the simulation software for our production grid, we used a stepwise approach based on increments.

- The first increment was a distributed but still simple system where no negotiation is used and where the grid production can be monitored and evaluated.
- The next increment focused on the planning and scheduling part, especially optimizing;
- In next increment deadlines and deadline awareness were introduced in the system.
- In the final system (so far) negotiating between the participating product agents was subject of research.

A. Basic distributed system

In all increments a distributed system of desktop and laptop computers has been used. The simulation, driven by an XML-based scenario, starts on one system and expands over the connected systems. First the equiplets are generated, then the real simulation starts by generating product agents at certain time intervals according to the scenario. These product agent establish their production path among the equiplets and travel over the network to the equiplet they need at a certain moment.

B. Scheduling

To schedule its production path, a product agent builds a production matrix based on the information on the blackboard given by the equiplet agents. Let us consider a simplified situation with 3 equiplets and 3 production steps. Consider a product that has steps \(< \sigma_5, \sigma_2, \sigma_4 >\). This results in a production matrix as in:

<table>
<thead>
<tr>
<th></th>
<th>E₁</th>
<th>E₂</th>
<th>E₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ₅</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>σ₂</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>σ₄</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Matrix element \(a_{ij}\) gives the relation between equiplet \(E_i\) and production step \(\sigma_j\). If the step is supported then \(a_{ij} = 1\). Not supported steps result in \(a_{ij} = 0\).

Optimization should result in a new matrix where \(a_{ij}\) has a slightly different meaning and can be bigger than 1, giving the product agent a clue for its choice. The product agent will choose the equiplet corresponding with the highest value of \(a_{ij}\). A very useful optimization is minimizing the transitions for a product from equiplet to equiplet. The optimizing algorithm will search for columns with sequences of \(a_{ij} \geq 1\) and increment these values by one. The matrix transforms to:

<table>
<thead>
<tr>
<th></th>
<th>E₁</th>
<th>E₂</th>
<th>E₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ₅</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>σ₂</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>σ₄</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Based on this matrix, the product agent will choose equiplet E₃ for steps \(\sigma_4\) and \(\sigma_2\). The advantage of this approach is that we can simply put different kind of optimizations in a sequence. All optimizations have their impact on the production matrix.

C. Deadline awareness

Every product has its release time, but there also exists a deadline for the product to be completed. All product agents released within a given time step try to schedule their path of production steps over the available equiplets. They use a global accessible timetable where they publish their claims on equiplets during certain time intervals.

D. Negotiating between product-agents

When a product agents does not succeed to find a production path before its deadline, it will enter this sequence of states:

- find the bottleneck in its production step path;
- based on this bottleneck it will search for product agents that can solve the problem by giving away their claim on a certain equiplet on a certain time;
- start negotiating with this list of agents, one after one, to see if a solution can be found;
- start producing the product if the deadline is feasible else return failure.

The negotiating is implemented by assigning different roles to a product agent. The agents that is planning its production path enters the role of negotiator if it cannot solve the
scheduling on its own. Another agent willing to negotiate with this negotiator because it still has some spare time before its deadline, will start a reschedule, but does not give up its already claimed path to have a backup in case of failure.

V. RESULTS

This section describes some results of the simulation.

A. Monitoring the grid production

In our simulation one can observe many parameters of the simulation including:

- load of the individual equiplets;
- production time of a product;
- role switching of the product agents to see if there are negotiating agents;
- total load of the production grid.

B. Negotiating

In fig. 2 the reservation of equiplets for time slices is depicted along with the reservations for product P1 with deadline a and P2 with deadline b.

Product agent P2 discovers that its scheduling fails. It will search for a solution and discovers that the longest gap in time is between step1 and step2. Next it searches for a product agent to negotiate with. In our case this is product agent P1. It asks P1 if it wants to negotiate to make room for three time slices in the time interval from T06 to T09 and P2 cancels its claims for step2 and step3. P1 agrees and reschedules its production path and thus makes room for P2 in the requested interval. P2 reschedules its path in the new situation and this results in the planning given in fig. 3, where both products meet their deadline requirements.

C. making a batch of a single product

When a simulation setup was made to produce a quantity of just a single type of product, just as in the case of batch processing, the production turned in a pipelined production system. When there is redundancy, so that every step is doubly available, it results in two parallel pipelines. This shows that our approach is consistent with efficient batch processing as applied in many production situations [6].

VI. RELATED WORK

Agent-based simulation in manufacturing is described by Paolucci an Sacile [7], but in these simulations, agents are used to simulate missing hardware subsystems or human operators. In our simulation we are close to the real situation without introducing extra agents, because agent technology is the pivot on which everything hinges in our approach. A manufacturing approach that has similarities with our approach can be found in [8], though there are important differences like the goals and roles of the agents [2].

VII. CONCLUSION

The multi-agent system can realize a multiparallel production paradigm. The biggest problem so far seems to be a fair use of the equiplets. When the load of the grid increases, the possibilities for new different products decrease. This results in an under-use of some equiplets, but as stated earlier, these equiplets are cheap, so it is not a big problem if their use in under 100%. However to improve this situation we are currently investigating the possibilities to (semi)automatically replace a frontend with unused production steps in favor of a frontend supporting overused production steps. We expect this approach to improve the throughput of the production process.

REFERENCES