Automatic Software Adaptation After Reconfiguring an Autonomous Manufacturing Systems

Daniël Telgen, Leo van Moergestel, Erik Puik, Laurens van den Brink, Tommas Bakker, Alexander Hustinx, John-Jules Meyer

[1] Department of Micro Systems Technology and Embedded Systems
HU University of Applied Sciences Utrecht
Nijenoord 1, 3552AS Utrecht, The Netherlands
Daniel.telgen@hu.nl
[2] Department of Information and Computing Sciences
Utrecht University,
Princetonplein 5, 3584 CC Utrecht, The Netherlands
J.J.C.Meyer@uu.nl

Abstract When using autonomous reconfigurable manufacturing system, that offers generic services, there is the possibility to dynamically manufacture a range of products using the same manufacturing equipment. Opportunities are created to optimally scale the production using reconfiguration means and automatically manufacture small amounts of unique or highly customizable products. Basically the result is a short time to market for new products. This paper discusses the problems that arise when manufacturing systems are reconfigured and the impact of this action on the entire system. The proposed software architecture and tooling makes it possible to quickly reconfigure a system without interference to other system, and shows how the reconfigured hardware can be controlled without the need to reprogram the software. Parameters that are required to control the new hardware can be added using a simple tool. As a result reconfiguration is simplified and can be achieved quickly by mechanics without reprogramming any systems. The impact is that time to market can be reduced and manufacturing systems can quickly be adapted to current real-time needs.

1. Introduction

Mass customization of products and a short time to market are important aspects of the changing manufacturing industry. These aspects of manufacturing are made possible by advances in technology and new business strategies. Agile Manufacturing, Lean Manufacturing, Holistic Manufacturing and Flexible manufacturing [1] are some examples of these strategies which consecutively focus on agility, efficiency, complexity, and flexibility. In technology additive (3D) printing and Reconfigurable Manufacturing Systems (RMS) are a driving force for changes in industry. This impact is especially large with the manufacturing of products that are made in low to medium quantities. This is for example common in microelectromechanical (MEMS) products like actuators and sensors, but also in other high-tech products. These changes combined are changing the market and enable the emergence of new markets for mass customization.

Reconfigurable Manufacturing Machines are an aspect of current research. This focuses on how to shorten the time to market and quickly scale manufacturing means to the demand. This includes research of distributed control systems, where every system can act autonomously. Cooperating autonomous systems have less impact on the overall system and are therefore less complex to reconfigure. To automatically adapt to these changes an automated translation system was designed that generated the correct hardware instructions based on an abstract product design [2]. However, while this design was able to translate abstract product descriptions to specific hardware, it did not fully support changes in the hardware, i.e., *reconfiguration*.

This paper focuses on the actual reconfiguration of the hardware and the changes that are made to the architecture to support reconfiguration processes. The paper will be outlined as follows. The next chapter will describe previous work and used methods. Section 3 will describe the three problems that will be addressed in this paper regarding reconfiguration of hardware and the impact it has on a Reconfigurable Manufacturing System. Section 4 describes the software architecture that is used to control all manufacturing systems. Section 5 describes how the architecture is used to enable simplified reconfiguration while discussing this in light of the problems as mentioned in section 3. Section 6 will give a general discussion of the subject, and section 7 mentions future work. Finally the paper will be concluded in section 8.

2. Background

a. Low cost, single purpose modular and reconfigurable platforms

To enable automatic reconfiguration a standardized platform has been introduced. This platform is called ‘equiplet’. An equiplet is defined as an autonomous system with modular hardware that can automatically be reconfigured. An equiplet is low cost and meant to perform one simple task. Hence, instead of creating one expensive machine with extensive capabilities an equiplet is substantially cheaper. Equiplets are designed to perform within a group to achieve the same efficiency and capabilities with a lower complexity and therefore less chance of failure. A group of equiplets has been named a grid [3]. A grid uses a flexible transportation system to transport products dynamically to any available equiplet. Equiplets make use of an automatic system to translate manufacturing actions, called product steps, automatically to specific hardware.

b. Automatic generation of control instructions

In related work a system was introduced that translated an abstract product design, containing manufacturing steps, towards specific instructions that could be send over an industrial bus to directly control the hardware. This system was originally designed to be performed in three steps and was implemented with the use of agent technology [4]. In this design each agent had specific knowledge that was required for the translation to the next, more specific, level. In the proposed architecture both manufacturing systems and products are autonomous. All systems are therefore depicted with a virtual counterpart, the agent.

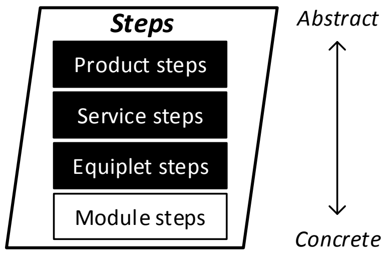


Figure 1: Translating several levels of manufacturing steps

As seen in Figure 1, a product is first described with abstract product steps. A product agent (not shown in the image) dynamically schedules a step at an autonomous manufacturing machine, the equiplet. The equiplet agent represents this machine and negotiates with the product agent if it has the correct capabilities to take the next step within the requested timeframe. If they agree, the product step will be scheduled at the equiplet; details of the scheduling itself have been published by van Moergestel et al [5]. The translations are performed in several steps that become more explicit for a specific equiplet. First the product design steps are translated to the standardized services that refer to the capabilities that the equiplets have. This is done by using a set of possible product steps and the capabilities of all the equiplets to determine the correct translation. After this step is complete the results are put on the service steps blackboard. The standardized service steps are then finally translated to specific instructions for the hardware that will perform the required actions. On the hardware some information does still need to be added by specific hardware modules. These are called the module steps. These add extra specific parameters on the hardware level, i.e., position translations for required kinematics systems. This is similar to systems like finetuning parameters, that was also introduced by Arkin [6].

This translation system will need to be adapted when systems are reconfigured. Both translations and capabilities will change if an equiplet is reconfigured. This paper discusses the parameters that need to be changed and shows some changes to the architecture to enable reconfiguration of manufacturing systems.

3. Problem Description

Reconfigurable machines classically take a lot of time to reprogram. When a system is reconfigured all control systems for the modules have to be reprogrammed, which is classically done by software engineers that add new functionality to the system. In the proposed system this process is simplified. Using the automatic translation system some steps of this reconfiguration process can be automated since it will automatically translate the instructions to the hardware to the new reconfigured hardware. This eliminates the need to manually reprogram the control systems.

Some of the translation steps require specific system parameters. Most manufacturing actions use actuators to perform a specific action at a location in space. This requires motion planning to move tools to the correct position. Kinematic models are used to calculate the required movement, i.e., position, velocity and acceleration, to get to the correct position. Equiplets as shown in Figure 2 use cameras with computer vision abilities to find the objects. As such a translation has to be made between both the position found by the camera and the actions the actuators have to perform to get a tool to the product to perform the next manufacturing step.

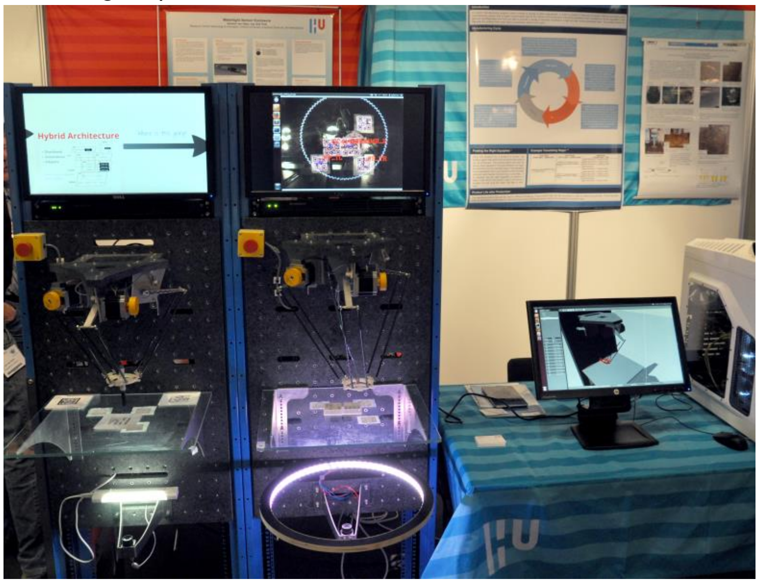


Figure 2: two equiplets with deltarobots and a 6 axis simulator on the right

To be able to create a correct kinematics model and acquire the right position it is essential to have specific parameters of the hardware modules. Including calibration of the camera system, positioning of the hardware, measurements of the tools, etc. Hence, reconfiguration disrupts the possibility to automatically generate instructions for the hardware. In classic systems reconfiguration would require the change of software and new coding to adapt all control systems to the changes.

To enable the automatic translation system, and simplify reconfiguration, several problems are identified that have to be solved:

1. Identifying the reconfigured hardware
2. Adding the specific parameters of the new reconfigured system
3. Identifying the new capabilities of the reconfigured system

To discuss the required actions to solve these problems, the systems architecture will be discussed first.

4. Software Architecture

The software architecture is based on a hybrid system using Multi Agent System (MAS) for the deliberative aspects and Robot Operating System (ROS) for the reactive aspects [7]. ROS is a software framework that provides middleware system and libraries for hardware abstraction. Basically, autonomous entities in the MAS have cognitive abilities to take the necessary decisions using entities in ROS for performing direct hardware interfacing.

a. Capabilities

Each equiplet in the production environment has certain properties and behaviors which can be classified as functional capabilities [8]. In a grid each equiplet has certain capabilities which is provided as a services to the grid. All the services in a grid can be used by the product agent to manufacture a product. A capability is defined as a service description combined with the limitations of the service. Examples of capabilities are pick and place, or draw Line. Limitations of these capabilities include the boundaries of the workspace. Basically these are defined as such:

Service	=	*Abstraction of the provided service*
Limitation	=	*Min. Weight, Max Weight, Dimensions Limits, etc*
Capability	=	*[Service, Limitation]*
ProductStep	=	*<Service, Criteria>*
Criteria	=	*Weight, Dimensions, etc*

A product agent divides the manufacturing of his product into product steps. For each product steps the agent uses a service of an equiplet. The product agent defines a product steps by the required service and criteria's of the service. Criteria's of a product step include the dimensions and weigth of the part on which the service needs to be executed, e.g., one of the criteria would be the dimensions of an object that needs to be pick and placed, so an equiplet agent knows if he is able to perform the service.

b. Directory Facilitator

A product agent can find the services in a grid through a Directory Facilitator (DF). The DF that provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals

c. Hardware Controller

As was shown in Figure 1, the original design was placed in three levels. Product steps, service steps and equiplet steps. Product steps are an abstract description of the step that has been performed, without any knowledge of possible manufacturing means. Service steps were defined as a translation from the product steps to available standardized capabilities of the grid. A equiplet step was a standardized capability translated to specific hardware that will perform the actual manufacturing.

As seen in Figure 3 this design has been implemented with some slight changes. The first translation from Product to service step has been taken out, considering that this was not a direct responsibility of the grid, i.e., in this example a product agent is created explicitly for this grid. As such the manufacturing steps are already designed or translated using the capabilities of this grid, eliminating the first required translation from abstract product steps to service steps. Also the hardware agent has been changed in two parts, the translation part and intelligent part. The hardware controller does only provide translations using the specific configuration parameters and is used by the intelligent equiplet agent. With this implementation the hardware controller does not require to be autonomous and it has no interactions anymore with more than one agent, as such it has become an integral part of the equiplet. Hence, it has been implemented as an object on the Hardware Abstraction Layer (HAL).

d. Overview

Figure 3 shows an overview of the current software architecture. As shown the ROS and MAS platforms use a blackboard to uncouple communication and therefore isolate performance issues to the individual platforms. ROS represents its processes as nodes, where each node commonly controls an object with its own process. Nodes use streaming topics to communicate. In the shown architecture each reconfigurable module, commonly a sensor or actuator, has a node. Besides the modules the equiplet also have an equiplet node that sends instructions to the modules on an industrial bus, a lookup handler node to find objects in its physical surroundings, and an environment cache node that holds several parameters and data received by the lookup handler.

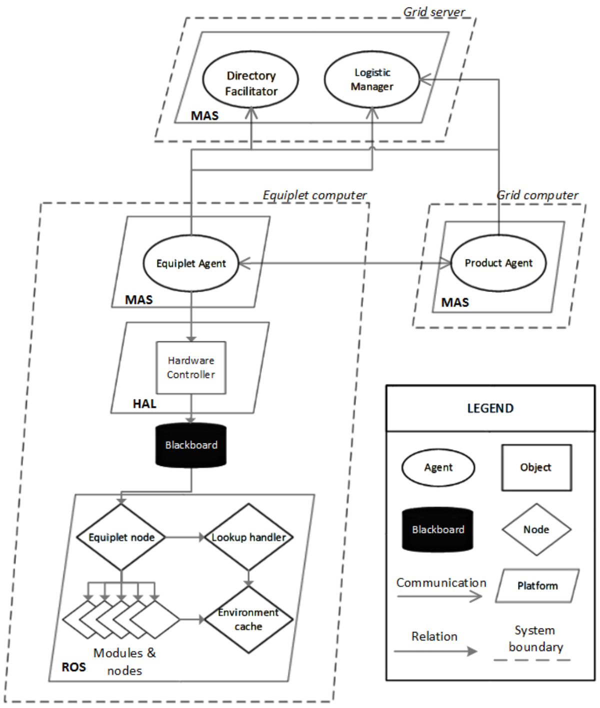


Figure 3: Software Architecture

On the grid level the logistic manager agent takes care of logistic processes that happen on a grid level, like transportation for the products between equiplets and the flow of resources. While each grid only has one grid server, directory facilitator and Logistic Manager, there can be many Product agents and equiplets. Since each equiplet is autonomous each has its own computer with all required software to control the entire equiplet.

5. Reconfiguration

This chapter describes the three steps that are required for a reconfiguration action. First the reconfiguration itself has to be detected by a trigger, which detects that a module has been changed. The module is then identified after which the new required software can be loaded. After the identification, specific parameters that are required for correct use of the new modules should be added to the equiplet. Finally the equiplet should recognize its new capabilities and offer its new services to the grid. These steps will be discussed in more detail:

a. Reconfiguration trigger and identification

As described in the problem description it is required to identify reconfiguration actions. Depending on the changes the software and capabilities of the equiplet should be updated at the grid level. Since hardware needs a trigger to start up a system was devised that simplifies this operation. An equiplet can be set in a ‘safe’ reconfiguration mode where it uses a camera system to detect 2D barcodes that identifies a module, see Figure 4 for an example. This way the according nodes that belong to the modules can be removed or added from the equiplet only by scanning the correct barcodes. Combined with a tool described in the next subsection parameters can be added without the need for a specialist or engineer to completely reconfigure the equiplet. Basically the barcodes provide an abstraction to the corresponding hardware.

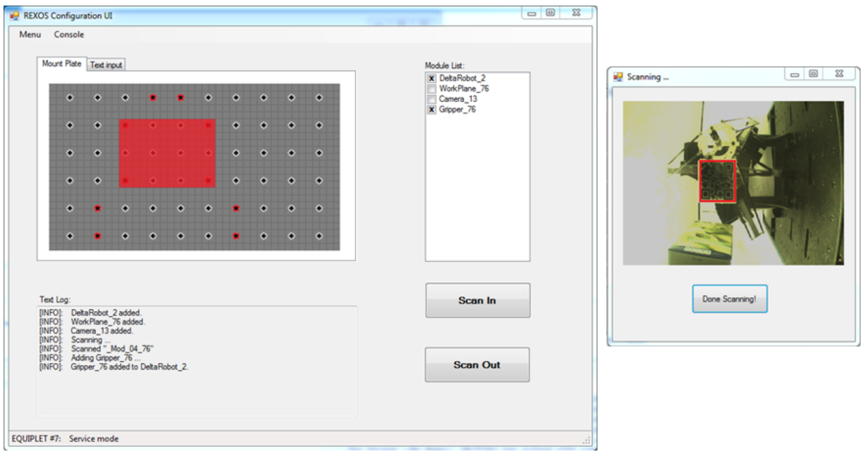
Figure 4: 2D barcode

b. Parameter input

The barcode is used to identify the type and unique ID of the reconfigured module. This can be used to load the required nodes to be able to control it. However, for control specific parameters are required. Current parameters that are defined for modules are: *moduleType, moduleID, mountSocket, tilt, extraLimitations, parentModule*

moduleType	*= Containing its type based on its capabilities*
moduleID	*= Contains a unique ID based on model and serial number*
mountSocket	*= x and y location of attachment points*
tilt	*= tilt of the object (e.g., to detect an inverted camera)*
extraLimitations	*= unique limitations of the module*
parentModule	*= if a modules is connected to another (e.g. a gripper)*

Most of these parameters are known through the unique ID and moduleType. However, the position of the module attached to the equiplet depends on the unique configuration of the equiplet. For this purpose a specific tool has been added to simplify this process. Figure 5 shows a screenshot of this tool. It can be used to scan the barcodes, but more specifically it also provides a model of the equiplet. An equiplet uses specific mounts which are known to the equiplet. As such parameters of the location of the module can easily be extracted by indicating to which mounts the modules are connected. With the use of the tool a module add or remove command is triggered, followed with the parameters put in the tool. These parameters are filled in at the hardware abstraction layer and are used by the hardware controller that will send the command to add the required nodes so that they can receive instructions.



[✓ DeltaRobot_2]
WorkPlane_76
Camera_13
[✓ Gripper_76]

Text Log:

```
[INFO]:   DeltaRobot_2 added.
[INFO]:   WorkPlane_76 added.
[INFO]:   Camera_13 added.
[INFO]:   Scanning ...
[INFO]:   Scanned "_Mod_04_76"
[INFO]:   Adding Gripper_76 ...
[INFO]:   Gripper_76 added to DeltaRobot_2.
```

EQUIPLET #7:   Service mode

Figure 5: Shows the prototype input tool to choose which sockets connect the module and an overview of the modules and the scanning tool.

Figure 6 gives an overview of all systems with their dependencies, the user input is relevant as it gives the reconfiguration parameters that are necessary for the automatic product step translation system.

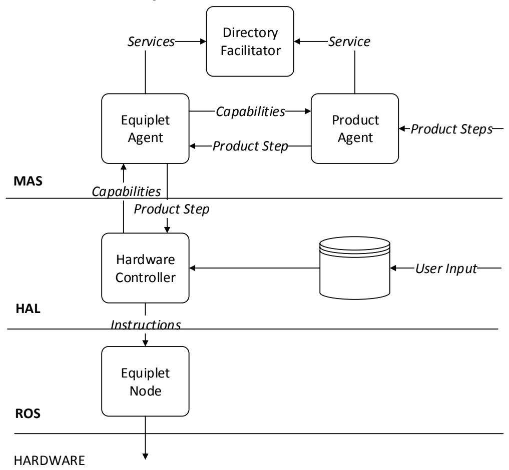


Figure 6: System dependencies

The directory facilitator matches services from the equiplet agent and the product. When a number of possible equiplets are acquired the product negotiates with several equiplets to schedule a product step. Scheduling algorithms that can be used are Earliest Deadline First or Least Slack First, which are discussed specifically for this case in related work by van Moergestel [5]. User input that is required by the hardware controller is given by the reconfiguration tool, which can contact a database with the required module information. The equiplet Agent also uses the hardware controller to acquire the new capabilities.

c. Capability change

When the reconfiguration is complete and the software and parameters of the new modules are loaded the capability has still to be adapted. Figure 7 shows an example of an equiplet who changes its capabilities and how a product performs a step at this equiplet.

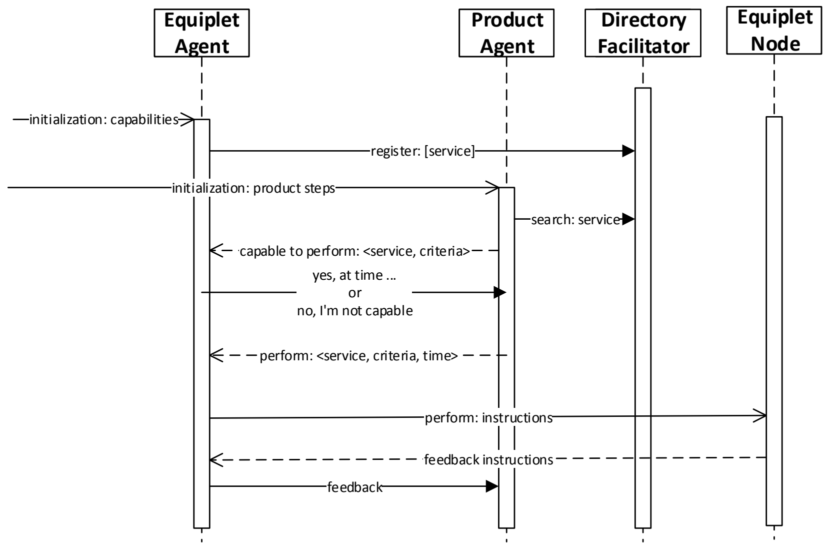


Figure 7: Initialization of Capabilities

The first trigger is given by the tool that triggers the configuration action. When the reconfiguration is complete the tool has a list of all configured modules. The combined list of modules with its explicit parameters will determine which capabilities the equiplet has and send these to the equiplet agent. The agent offers its capabilities to the grid by registering its capabilities as services by the Directory Facilitator. This will make the equiplet available for product agents that require these services. When a product is scheduled at the equiplet it will check if the detailed capability with the equiplet, including specific criteria. If accepted the product will be scheduled at the equiplet. When the scheduled time arrives the equiplet agent will use the translated steps to send the appropriate instructions to the equiplet nodes through the hardware controller as described in section 2.b.

6. Discussion

Distributed control systems have been discussed in several papers [9]. In many cases there has been some discussion on the term *distributed* itself. In the context described in this paper an equiplet is completely autonomous. From a grid perspective this makes all equiplets have distributed control, without overall (grid level) hierarchy. This is in contrast with some that mention sensors and actuators that are distributed throughout a system with multiple manufacturing machines, which are hierarchically controlled from one centralized control system. This is relevant to this paper, since the only adaptation that is required after a reconfiguration action is to update the capabilities at the Directory Facilitator. This way the impact on other systems of a reconfiguration action is minimal to none.

To use intelligent agents has also been attempted before. Leitão discusses several research and industry projects using agents for manufacturing systems [10]. Results of these projects are mostly positive from a technological perspective. However, while they show potential they are barely adopted by industry. This becomes even more clear in the work of Bussmann [11], who demonstrated a successful flexible production line for DaimlerChrysler that was controlled by agents. However, after several years the production line was not renewed, basically because of that the economically measurable advantage was hard to measure. Flexibility makes in principle a difficult business case. Because of this we focus on not only the flexibility of a manufacturing line, but also the flexibility of the manufacturing equipment. The expectation is that combining the use of new technologies like additive printing and simplified reconfiguration of manufacturing systems opens up new business cases that might be economically viable.

7. Future Work

Current focus is on the reconfiguration aspects and the automatic adaptive control of the equiplets. Current status is that the translation system and prototypes of the reconfiguration tool has been developed, all critical systems have been tried in proof of concepts. However, Some aspects of the architecture as shown in Figure 3 have not been fully developed. In the near future the full system should be demonstrated with a larger set of possible modules and configurations. Aspects that will be looked into are the automatic calibration of actuators and camera computer vision systems to make the reconfigurable even more plug & playable. This must eventually lead to a system that is completely reconfigurable with the abilities of standard mechanics, i.e., without interference to other systems or input from engineer level personnel.

8. Conclusion

This paper considers an approach to automatically adapt the software when a Manufacturing Systems is reconfigured. It discusses the problems that occur and gives solutions to how parameters and changes in capabilities can be mitigated using standard solutions from agent technology. The reconfiguration action automatically updates its services at a central system. As such the grid automatically adapts to reconfiguration system without overall system impact. Products can immediately start using the reconfigured system after this process. The impact of this work is a greatly reduced time to market, since the manufacturing systems can be easily reconfigured without impact on other systems in the grid.

References

[1] H.A. ElMaraghy, "Flexible and reconfigurable manufacturing system paradigms", International journal Flexible Manufacturing Systems, 17:261-276, Springer, 2006

[2] D. Telgen, L. van Moergestel, E. Puik, A. van Zanten, A. Abdulamir, J.-J. Meyer, "Agile product manufacturing by dynamically generating control instructions," Assembly and Manufacturing (ISAM), 2013 IEEE International Symposium on , vol., no., pp.282,284, July 30 2013-Aug. 2 2013

[3] E. Puik, and L. van Moergestel, (2010). Agile multi-parallel micro manufacturing using a grid of equiplets. In Ratchev, S., editor, Precision Assembly Technologies and Systems, volume 315 of IFIP Advances in Information and Communication Technology, pages 271– 282. Springer Berlin Heidelberg.

[4] M. Wooldridge and N. Jennings, “Intelligent Agents: Theory and practice”, The knowledge Engineering Review, page: 115-152, 1995.

[5] L. van Moergestel, E. Puik, D. Telgen, and J-J. Meyer, ”Production scheduling in an agile agent-based production grid”, IAT2012 proceedings, 2012.

[6] Ronald C. Arkin, ”Behaviour-Based Robotics”, MIT press, page 205-234, Hybrid Systems, 1998.

[7] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Y. Ng, ”ROS: an open-source Robot Operating System”, Open-source software workshop of the International Conference on Robotics and Automation (ICRA), 2009.

[8] E. Järvenpää, P. Luostarin, M. Lanz, and R. Tuokko,”Development of a Rule-base for Matching Product Requirements against Resource Capabilities in an Adaptive Production System”, FAIM2012 proceedings, page. 449-456, 2012.

[9] D. Trentesaux, Distributed control of production systems, Engineering Applications of Artificial Intelligence, Volume 22, Issue 7, Pages 971-978, 2009.

[10]P. Leitão, ”Agent-based distributed manufacturing control: A state-of-the-art survey”, Journal Engineering Applications, Volume 22, Issue 7, Pages 979–991, 2009.

[11]S. Bussmann, K. Schild: An Agent-Based Approach to the Control of Flexible Production Systems. In Proc. of the 8th IEEE Int. Conf. on Emergent Technologies and Factory Automation (ETFA 2001). Antibes Juan-les-pins, France, Pages 481-488 (Vol. 2), 2001.