Towards an Industrial Agent Oriented Approach

João Reis [1]

[1] Faculty of Engineering of the University of Porto. Rua Dr. Roberto Frias, s/n, 4200-465, Porto, Portugal

jpcreis@fe.up.pt

Abstract. The agent paradigm is being strongly discussed for the past few years, mainly when addressed to practical and real world issues. The industry domain, and specifically the production system context, has revealed to be suitable for the use of Multi-Agent Systems, and along with it, some artificial intelligence concepts applied to process optimization [1]. The approach presented in this paper makes use of the equipment representation by means of the agent concept, and markup languages for document encoding as XML to create simulated environment for the study of shop-floor dynamics, and the reliability and effectiveness in using this kind of paradigm in production system. The present approach aims for the optimization in specific stages of the production system, like product ramp-up, scheduled maintenance and unscheduled maintenance. These different phases reveal to be very time consuming and costly, and thus, task driven communication, negotiation strategies and the agent concept can actually open a door towards a whole new holistic perspective about information system and shop-floor interactions. In this paper, a simple welding scenario is presented looking forward to foster and leverage the use of, firstly, Multi-Agent System concept within the industrial domain, and secondly, negotiation strategies to solve and handle conflict issues.

Keywords: Multi-agent System, Agent Negotiation, Agent Collaboration, Modelling, Production Systems, Manufacturing Execution System

1 Introduction

Situations like absence of equipment visibility at the shop-floor level, non-existing inter-equipment communication and lack of collaborative capabilities are just few challenges that nowadays European Industry have to deal with. Most of the times, these kind of problems lead not only to inefficiencies in the very early stages of the production system, which involves the product ramp-up – since the production is initiated, some equipment need to be calibrated until some quality parameter is not met - , but also when equipment requires maintenance. In this latter case, there are two different

Doctoral Symposium in Informatics Engineering, p. 1, 2014.
© Springer-Verlag Berlin Heidelberg 2011

situations that can be observed. On one hand, a problem in a certain equipment is identified but it can still operate at the shop-floor level, and maintenance can be schedule in further future. On the other hand, if a severe problem occur, an equipment might require an immediate maintenance forcing all the production system to stop. All these different problems represent the nowadays challenges that need to be tackled in order to minimize production costs, and maximize the product outcome, and consequently improve the competitiveness on the industry world.

The purpose of this paper is expose an agent oriented approach applied to the industry domain, in which a simple Welding scenario is explored. This agent approach aims to create a representation of each equipment on the shop-floor level using the agent paradigm, in which each agent is an extension of the communication and processing capabilities of each machine. Therefore, collaboration and negotiation strategies were explored in order to tackle shop-floor workflow optimization issues, like which equipment should be used for a specific required task and provide the best quality results.

The main goal of this approach is to explore and study the use of Intelligent Agent paradigm applied to the Industry domain. It aims to replicate the shop-floor equipment along with its dynamics and interactions and to explore the possibility of using this paradigm in a real industrial environment. To do so, the JADE platform was used to model the all the production system equipment, from the lower capacity devices like a temperature or humidity sensor, to the Manufacturing Execution System used to plan and coordinate the whole shop-floor. Taking advantage of the JADE functionalities, the Contract-Net protocol along with specific behaviors were used to model the agents' interaction and the dynamics associated with the industry environment [2].

In this paper, an agent approach is explored to deal and face some of the nowadays industrial challenges. Section 2 reviews some of the work that have been made in field, and constituted a very good basis for this approach exploitation. Section 3 presents in detail and explains the proposed agent based approach. Section 4 shows some experimental results taking into account a simple Welding scenario, in which a set of sensors and a welding machine are used. In Section 5, a discussion is raised upon previous experimental results, concluding with few remarks and future work possibilities.

2 Related Work

This this section will presented two different framework that aim to apply the agent paradigm into the industry domain, in terms of reconfiguration and agility of the manufacturing systems.

The first one is named MetaMorph II, and is an agent based architecture that aims to integrate different manufacturing activities like design phase, planning scheduling, etc, that allow the system reconfiguration. This architecture is oriented to distributed intelligent design and manufacturing that takes into account entities like suppliers, customers and partners for extended-enterprise issues [3]. However, this architecture is more oriented to the system adaptability rather than system configuration and reconfiguration [4].

The second one is called AARIA (Autonomous Agents for Rock Island Arsenal), and is an agent architecture that is mostly concerned about the systems' design. This way, it is composed by collaboration models that are requirement driven, rather than using the collaboration capabilities for designing and specifying the system [5].

3 Agent Oriented Approach

3.1 Industrial Overview

As previously said, one of the purposes of this approach is to have an agent representation for each equipment on the shop-floor. To correctly understand the meaning of each agent, we need to previously be aware, in a simplified way, of the industrial panorama on two different stages of importance: common information system oriented to the industrial domain that aims to set up the shop-floor configuration and plan the workflow according to the product specifications, called Manufacturing Execution System (MES); the different types of equipment that can compose a shop-floor production system. The first one aims to manage and control all the shop-floor equipment, like receiving the product specifications and translate them into tasks that can be delegated among all the available equipment on the shop-floor level. The second one is directly related to the machines that are displaced throughout the shop-floor. In this latter, two different types of equipment were identified: high capacity devices and lower capacity devices. The high capacity devices are equipment with high capabilities of processing and memory that can control other high capacity devices or different sensors and actuators. The lower capacity devices are just sensors and actuators that need to be controlled by another entity, due to the lack of processing and memory.

3.2 Agents' Description

For a better understanding of this approach's purpose, the concept of agent needs to be explained and clarified. The definition of agent varies in ranges of context, assuming different functions and purposes in areas like philosophy, sociology, economy, law, and others. Despite those contexts, the definition that seems to be more suitable for the industry domain, lies in the following: *"A computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objective"* [6]. An agent should have the a sense of autonomy, acting independently from human intervention, should have the ability to interact with other agents and the environment, react to the environment changes and percept from it, assume a pro-active posture in acting by its will, and should be continuous running process.

Taking into account the Industrial Overview presented in the previous subsection, the was mainly influenced and inspired by the I-RAMP3 project, a set of agents was identified to fulfil the intended representation of the industrial environment: *Device Agent*, *Sensor & Actuator Agent*, and *MES Agent*. The *Device Agent* is a direct corre-

spondence of high capacity devices and it aims to replicate the behavior of the equipment that have the high processing and memory capabilities, and is intended to control / manage other *Device Agents* or *Sensor & Actuator Agents*.

Sensor & Actuator Agent is the representation of the sensors and actuators that exist on the shop-floor. Since these kind of equipment are lower capacity devices, to operate on the production system, they need to be controlled / managed by *Device Agents* or the *MES Agent*.

Finally, the *MES Agent* is intended to partially replicate the behavior of the *Manufacturing Execution System*. This partiality is based on the equipment shop-floor planning, in which tasks are delegated to *Device* and *Sensor & Actuator Agents* to execute according to its specification.

3.3 Agents' Interaction

One of the things that is inherent to a Multi-Agent System is the interaction among all the agents. This way, a very well-structured communication needs to be specified in order to guarantee a correct interpretation of what each equipment is capable to operate on the shop-floor level, all the tasks that needs to be delegated and executed, and even additional functionalities that can be independent from the physical execution, like a service that provides a machine's information on a given period. Therefore a set of eXtensible Markup Language (XML) documents was developed to fulfil all the communication needs to promote the most reliable interaction between agents. The XML format was chosen because it is both machine-readable and human-readable, is a very well accepted standard for encoding documents [7] and there are several tools that can translate the information present on a class model into an XML-based file, and the other way around. Hence, the following set of documents were defined: *Self-Description Document* (*SDD*); *Task Description Document* (*TDD*); *Task Fulfilment Document* (*TFD*) [8].

Self-Description Document.
The *Self-Description Document* aims to be sent to all the system entities (*MES Agent*, *Device Agent*, *Sensor & Actuator Agent*) when an agent enters the network of devices and is the basis for all the other documents to be exchanged during the agents' interaction, since it describes the equipment basic information and all the tasks that can be executed.

Task Description Document.
The *Task Description Document* is intended to be sent from both *MES Agent* and *Device Agent* when a task needs to be delegated. On the other hand, all the agents can receive this kind of document, despite the *MES Agent* because it is the one who delegate tasks in the first place.

This document is generated by the agent that wants to delegate a task to other agent and it is a particular case of the *Self-Description Document*, since it only concerns

about the task parameterization, with the addition of when the task will start, and how long it will take.

Task Fulfilment Document.
The *Task Fulfilment Document* is the response to the *Task Description Document* and is an exact copy of it, with the updated values from the responder taking into account the worst case scenario, meaning that the update will only occur when the task execution on the shop-floor, according to the SDD, is different from what was initially required.

3.4 Agent Negotiation

When coordination and collaboration is intended to make part of the agents' dynamics in a Multi-Agent System, negotiation strategies need to be considered. The purpose of Negotiation in this specific approach is related with agent's cooperation to meet local and global goals, in which agents must act as a group and decide who executes what according to cost and utility functions.

In this case, it was used the Contract-Net Negotiation protocol to deal with resource allocation conflicts. Simply describing it, the Contract-Net protocol always need as least two different entities: the *ContractNetInitiator* and *ContractNetResponder*. The first step of the protocol is made by the *Initiator*, sending a Call for Proposal (CFP) message to the *Responder*, in order for the latter to provide a Proposal according to what was specified previously on the proposal call. Finally the *Initiator* has the main role to analyze the proposal using a *Utility Function* deciding if it should accept or refuse the proposal. The idea of this approach is to use this protocol to deal with conflicts where two or more agents want to delegate a task to the same agent, and they need to agree upon which one will make use of the task delegation.

For the evaluation of documents (both TDD and TFD) two different functions were built to quantify the cost associated to a task execution by a specific agent, and to measure the utility of a task execution that a different agent has for the agent itself. Therefore, a *Cost Function* and a *Utility Function* were defined, along with a simple *Threshold Function* that calculates how much an agent is will to negotiate a given task execution.

Cost Function.
Cost Function is intended to quantitatively evaluate a function in terms of execution impact for the production system. In equation (1) is presented the main parameters that should be taken into account to calculate a value that represents the effort applied to a task execution.

$$\frac{Sa}{Sr} + \frac{St}{T - \frac{\left((St - T) * Sr\right)}{T}} \quad (1)$$

For the task execution cost to be calculated, a set of arguments need to be specified: *Sr*: required services; *Sa*: available services; *St (serviceMinTime)*: value that describes the amount of time an equipment worth to execute a certain task on the shop-floor; *T:* task duration.

Utility Function.
Utility Function is basically a quantitative measurement of how useful a task execution can be, taking into account the differences between the requester ideal task operation and actually the possible task specification to be executed by the requested agent.

$$\frac{10}{[\sum_{i=0}^{A} dif\ i] * 0.1 + 10} \quad (2)$$

If the difference is 0, the utility will be 1, and in other cases, the utility will be any number between 0 and 1.

Threshold Function.
Equitation (3) is a simple equation which is intended to define a how much an agent should be willing to negotiate for a task execution with others. In this particular case, parameter *P* assumes the value of 10.

$$P * Utility \quad (3)$$

4 Experimental Results

The main purpose of this section is to gather all the concepts and ideas described in the previous sections, and apply them to a very simple case. It aims to provide not only a holistic overview of the approach, but also to frame all the concepts in a specific context, clarifying the main role of the proposed approach. Also this case study was inspired and based on the expertise of the all partners involved on the I-RAMP3, in which a Welding scenario takes an important role not only for project's requirements, but also as a basis for final project's demonstration purposes.

4.1 Case Study: Sensor Negotiation

Scenario Dynamics.
The first step to be taken is to define the agents that will make part of scenario. As previously explained, there are two different entities that aim to operate on the shop-floor level - *Resistance Spot Welding Machine* and *Metrology Station* - that implies the use of two instances of *Device Agents*, and one Mote that required a representation of one *Sensor & Actuator Agent*, with the capability of providing three different types of measurement at the same time – Temperature, Humidity and Camera.

For a simple and clear explanation of the scenario, **Fig. 1** will be used as a reference. It has the representation of two *Device Agents* (*RSW Machine* and *Metrology*

Station) and one *Sensor & Actuator Agent* (Mote with three different sensors integrated). Before starting to make use of **Fig. 1**, an initial interaction between agents need to be explained. It was previously described that when an agent enters the network (environment where all the devices can virtually see each other), they needs to broadcast throughout all the devices its internal information and all the tasks that can be executed. Therefore, taking into account the entities of the present scenario, when the two *Device* and *Sensor & Actuator Agents* step into the network, they need to generate a document that describe themselves, and send them to all the devices that make part of the network. The purpose of sending the SDD to all the network components, is to make them aware of what the other ones can do, and therefore, giving the possibility to locally make decisions about which agents are suitable to collaborate with.

After this registration process on the device network, everything is aligned to start the task delegation process, where agents create documents (TDDs) where they can specify on which conditions a task should be executed. Thus, it can be seen from **Fig. 1** that the initial message from the *Device Agents* is a *Task Description Document*s requiring the execution of a specific task from the *Sensor and Actuator Agent*. This step corresponds to the Call for Proposal message on the Contract-Net Protocol, where agents send the ideal required task execution. Consequently and regarding this scenario, the Mote should be able analyze the two received TDDs, and do two different things in each case: Update TDD into the TFD; Calculate the cost of the updated task execution.

The first thing is basically to compare and update what was defined in the TDD with what the Mote can actually execute on the shop-floor (SDD specification). The second thing is it to make use of the defined *Cost Function* to estimate the price that a *Device Agent* needs to pay for the task execution. This value is a measure used to compare the effort that needs to be applied for different tasks by different devices, and to see if it worth the use of that equipment or it makes sense to reject that specific task execution. After those two steps are completed, the Mote device needs to send the TFD response to the *RSW Machine* and *Metrology Station*, along with the cost of task execution.

For the *Device Agents* to measure how suitable a task response (TFD) is comparing with the task request (TDD), it need to make use of the *Utility Function*. This function, as previously explained, measures the distance between the ideal task execution and what can really be executed on the shop-floor level, and it aims to quantify if the proposal sent from the *Sensor & Actuator Agents* is useful or not for the task execution to be used.

In the case of both *Device Agents* accept the proposal, the *Sensor & Actuator Agent* is responsible to increase the costs of the task execution to see how's willing to pay the higher value. This negotiation strategy aims to take the maximum advantage of the proposal requesters, since in cases of where the demand is higher than supply, the product prices will increase and maximize its profits. In this case, since we are talking about collaboration, these prices are not actually to be paid, but just a measure to determine whose component will make use of the task execution.

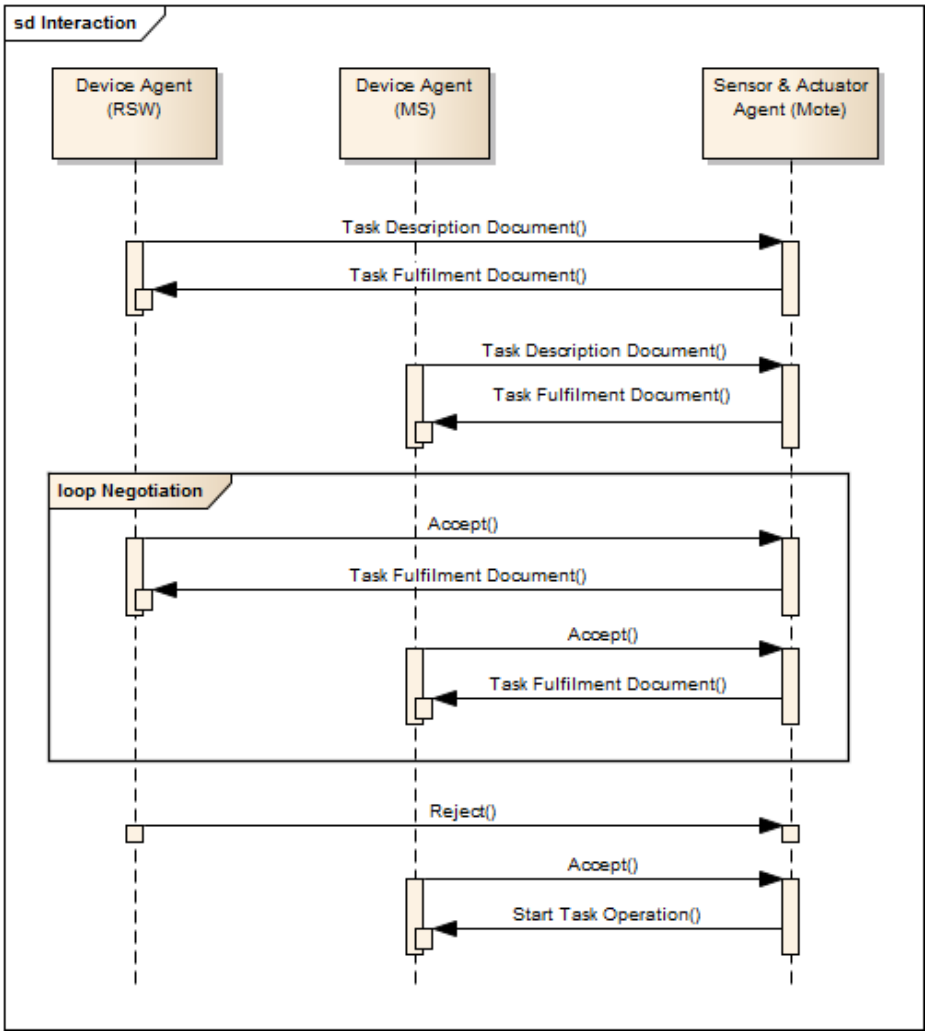

Fig. 1. Simple Negotiation Sequence Diagram

Finally, when the task execution cost is higher enough for only one *Device Agent* is willing to pay for the execution, a message will be sent informing about the success of task execution allocation, and it will start to operate in the specified time, during the specified task duration.

Simulated Scenario.

For full and complete understanding of information exchanged regarding the negotiation process, a simple example will be presented using the previous explained equipment, protocols and information documents. As a basis for this exposure, **Table 1**

shows the information that was specified regarding the task execution. It can be seen that task type is *Measurement*, and it can use three different types of sensors at the same time: Temperature, Humidity and Camera. From the table, we can see information on the task level, like *ServiceMinTime* and *NumberServices*, and on the sensors level we can see a set arguments that assume the operation conditions for the Mote device.

In the initial negotiation phase, *RSW Machine* and *Metrology Station* should create a TDD and send it to the Mote device. In this paper, we will not focus on TDD and TFD definition, but on the process of reaching an agreement in whom will make use of the task execution. Therefore, only the differences between TDD and TFD will be considered, rather than the whole task description. The first equipment wants to make use of all the three different sensors, and the second one wants to make use of only one sensor: Camera. From this point, we can assume that *RSW Machine* TDD should have the information of the three sensor parameterization, and the *Metrology Station* should have only one sensor parameterization.

With the present information, we can start specifying some values for the *Cost Function* parameters, to be used by the *Sensor & Actuator Agent* for each TDD received. Considering the *RSW Machine*, and taking into account equation (1), we can define $Sa = 3$, $Sr = 3$, $St = 100$ and $T = 500$. Considering the *Metrology Station*, we can define $Sa = 3$, $Sr = 1$, $St = 100$ and $T = 20$. As can be seen, there are two differences on the functions arguments. One of them is the number of services required - *Metrology Station* only requires one sensor - and task duration - *RSW Machine* task is required to be longer than the *Metrology Station* one. Hence, the cost associated to the required task from *RSW Machine* is 1.199 and from *Metrology Station* is 9.25. The big difference between costs is due to two major aspects: Number of sensors consumed; Task duration.

As explained in the previous sections, considering a Mote, it is more costly to use only one sensor, avoiding others to use the full capabilities of the device, and a minimum time that makes sense to use the device, for energy consumption reasons. Regarding the two TFDs that should be defined by the *Sensor & Actuator Agents*, we will assume there are two differences comparing with the *RSW Machine* TDD (two different task arguments cannot be fulfilled by the mote specifications), and no difference comparing with the *Metrology Station* (the task can be executed without restrictions).

Therefore, *Sensor & Actuator Agent* should send the TFDs to the corresponding entities, and wait for either an acceptance or rejection response. On the *Device Agent*'s side, a utility for each TFD should be calculated. To do so, we only need to know the number of sensor arguments that can be parameterized, since the *Utility Function* makes use the differences between what is ideal to be executed, and what can actually be operated on the shop-floor. Thereby, TFD utility for *RSW Machine* is 0.909, and for the *Metrology Station* is 1. The calculated utility for the *RSW Machine* is only 0.909 because, as previously depicted, there was not a full matching between the presented TDD and the proposed TFD.

Table 1. Sensor & Actuator Agent - Measurement Task Description

Type	**Service Min Time**	**Number Services**			
Measurement	100	3			
Parameters	**ID**	**Description**	**Arguments**		**Unit**
	Temperature	Temperature Sensor	Min	-55	ºC
			Max	200	ºC
			MinError	10	%
			MinResponseTime	55	Ms
	ID	**Description**	**Arguments**		**Unit**
	Humidity	Humidity Sensor	Min	0	%
			Max	100	%
			MinError	20	%
			MinResponseTime	55	Ms
	ID	**Description**	**Arguments**		**Unit**
	Camera	Camera Sensor	maxFrameRate	24	Img/sec
			minLatency	100	Ms
			MinError	20	%
			MinResponseTime	55	Ms

The next phase of the negotiation protocol is the first step of the negotiation loop, in which *Sensor & Actuator Agent* iteratively increases the cost of the task execution to determine which *Device Agent* is willing to pay the higher price for the task execution. As explained before, the cost value is just representative and with no influence. For instance, if a *Device Agent* accept a task execution with a cost value of 8.5, anything will change or influence in further operation, since we are dealing with a collaborative environment, and not a competitive one. Therefore, the *Threshold* value that *RSW Machine* and *Metrology Station* are available to pay for a task execution agreement is 9.09 and 10, correspondingly. Whenever the *Sensor & Actuator Agent* receives the acceptance messages from the two *Device Agents*, no agreement is reached, and it will increase the cost of each function in 1 unit. Hence, in the next iteration, task execution costs will assume the values of 2.119 and 10.25, and taking into account the previous thresholds, the *Device Agent* representing the *Metrology Station* is not willing to pay for the task execution, since the new cost exceeds it, and will send a *reject* message to the *Sensor & Actuator Agent* saying that it is no longer interested in the task execution. On the other hand, since the new *RSW Machine* task execution cost does not exceeds the calculated threshold, it will send an acceptance message to the Mote.

Finally, and after *Sensor & Actuator Agent* checks that only one *Device Agent* is willing to pay the cost associated with the task execution, it will send a *Start Task Operation* message informing about the success of the negotiation.

5 Discussion

Throughout the whole paper, new concepts and ideas were explored with the intention of improving some of the nowadays difficulties in the industry domain. Regardless of the simple example presented in the section 4, the strategies applied to this context along with the agent paradigm and well structure communication processes, proved to be a good and reliable approach. Nevertheless, more industrial scenarios and validation processes are required and need to be exploited, until this approach reach the necessary consistency to gain credit among the industrial reality.

One of the most important advantages of the presented concepts, is undoubtedly the decentralized approach, which seems to be verified the fault tolerant property. This means that in case of sudden equipment fail, the network of equipment will maintain its communication and collaboration activities, avoiding stopping the production system due to component dependency issues.

Another concept introduced in this paper, is the task driven communication, in which not only equipment execution on shop-floor level, but also internal programmed services, are specified in XML-based format, and used to delegate responsibilities of providing certain results and operation according to precise specifications. This concept allow the automatic reconfiguration of equipment for shop-floor operation, in which, in some of industrial contexts, is made manually and reveal to be very costly and ineffective.

Obviously that when we talk about a complex and ambitious approach like this, the other side of the coin needs also to be revealed, and therefore discussed and explored. When we talk about industrial systems, latency is always an issue that needs to be considered when exploring and developing new concepts to be applied to it. In this approach, to deal with resource conflict, a negotiation protocol was used. In an environment where there are dozens of devices communicating with each other, the Contract-Net protocol with the negotiation loop can represent an undesirable network overhead.

The last issue that will be dissected in this section, is the additional computational resources that this approach involves. As said in previous sections, there are components like sensors and actuators that don't have a high processing and memory capability, and therefore, cannot be added more complex functionalities and communication means. Hence, specific set of sensor and actuator information must flow to a central point, e.g. gateway, and be intelligently encapsulated by more capable devices.

The next steps towards this specific industrial agent oriented improvement are related with the use of learning techniques for negotiation improvement and development of predictive maintenance models. The first one lies in the threshold learning, in which the amount effort a device is willing to apply in negotiation can be adapted to certain devices with very specific shop-floor needs. There's a possibility of whenever

a conflict occurs, in which a specific device is involved, and due to specific needs, it would never get a task executed by other device. The second one is a more complex challenge, due to the fact that data must be correctly analyzed and predictive models would need to be created for further adaptation as runtime information is becoming available. This would support the maintenance personal to, firstly, know which part of a component needs to be maintain, and secondly, to when a sudden fail is probable to occur. For this latter, Bayesian Networks will be explored since it reveals to be suitable for this kind of problems, like finding the root cause of a certain problem, and what was the most successful prescriptive measure for a specific malfunction.

6 Acknowledgements

This work was carried out with the support from the European Commission under the Seventh Framework Programmer, in the Integrated Project I-RAMP3 (FoF.NMP.2012-3).

7 References

1. Shen, W., Norrie, D. H., and Barthès, J.-P. A.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. Taylor & Francies, London and New (2001)
2. Bellifemine, F., Poggi, A., Rimassa, G.: Developing Multi-agent Systems with JADE. In : Intelligent Agents VII. Springer-Verlag Berlin Heidelberg (2001)
3. Shen, W., Maturana, F., Norrie, D. H.: MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing. In : Journal of Intelligent Manufacturing. (2000)
4. Ferreira, P.: An agent-based self-Configuration Methodology for Modular Assembly Systems. PhD Thesis, University of Nottingham, Nottingham (2001)
5. Parunak, H. V. D., Baker, A. D., Clark, S. J.: The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design Integrated. In : Computer-Aided Engineering. (2001)
6. Wooldridge, M.: An introduction to multiagent systems. 2nd edn. John Wiley & Sons, Chichester, England (2002)
7. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F.: XML 1.0 Specification., W3C (2008)
8. Peschl, M., Link, N., Hoffmeister, M., Gonçalves, G., Almeida, F. L. F.: Designing and implementation of an intelligent manufacturing system. In : Journal of Indutrial Engineering and Management. (2011)