Requirements and Matching Software Technologies for Sustainable and Agile Manufacturing Systems

Daniël Telgen, Leo van Moergestel, Erik Puik, and Pascal Muller
Dept. of Micro Systems Technology and Embedded Systems,
HU University of Applied Sciences Utrecht
Nijenoord 1, 3552AS Utrecht, The Netherlands
{Daniel.Telgen, Leo.vanMoergestel, Erik.Puik, Pascal.Muller}@hu.nl

John-Jules Meyer
Dept. of Information and Computing Sciences
Utrecht University
Princetonplein 5, 3584 CC Utrecht, The Netherlands
J.J.C.Meyer@uu.nl

***Abstract*—Sustainable and Agile manufacturing is expected of future generation manufacturing systems. The goal is to create scalable, reconfigurable and adaptable manufacturing systems which are able to produce a range of products without new investments into new manufacturing equipment. This requires a new approach with a combination of high performance software and intelligent systems. Other case studies have used hybrid and intelligent systems in software before. However, they were mainly used to improve the logistic processes and are not commonly used within the hardware control loop. This paper introduces a case study on flexible and hybrid software architecture, which uses prototype manufacturing machines called *equiplets*. These systems should be applicable for the industry and are able to dynamically adapt to changes in the product as well as changes in the manufacturing systems. This is done by creating self-configurable machines which use intelligent control software, based on agent technology and computer vision. The requirements and resulting technologies are discussed using simple reasoning and analysis, leading to a basic design of a software control system, which is based on a hybrid distributed control system.**

Keywords*—Sustainable Manufacturing; Agile Manufacturing; Agent Technology; Hybrid Systems; Multi Agent Systems.

I. INTRODUCTION

Sustainability and flexibility are growing more important by the day. A changing economy, technological progress and a decline of world resources fuel these needs. At the same time, consumer expectations are rising. A new phone is bought almost every year and of course preferably in the color and coating of your choice. To make this possible there is a need for more flexibility in manufacturing and a shorter time to market. The term used for this fast and flexible type of production is *Agile Manufacturing* [7]. Agile Manufacturing calls for automated flexible production which is still cost-effective. To achieve this, a change is required from original Dedicated Manufacturing Systems (DMS), i.e., classic production lines [1]. Sustainability is another important consideration in addition to agility. Not only should the products be more sustainable for the new generation of manufacturing systems, but the manufacturing systems themselves are required to be sustainable as well. This is achieved by developing machines which can build generations of different product families on small to medium scale rather than just one product. This will require a new and more flexible approach for manufacturing, hence it is necessary to create manufacturing systems that can be reconfigured for the latest demand. To be able to produce small to medium quantities and still be cost-effective, these machines should be scalable [5]. Reconfiguration should also be possible on every layer: from adding or removing entire systems for scalability towards changing a specific module or device on a single production machine to give it new capabilities. Hardware should be either changed by hand or by another machine; the software and necessary variables should automatically configure itself. This flexibility through reconfiguration is an important step towards sustainable and agile manufacturing [1]. However, the main problem is that while the flexibility of such systems increases its possibilities, it also increases the complexity of the software. For this reason there is a need to limit dependency between software systems. This requires an analysis of current platforms and architectures to move towards a new approach of the software which controls flexible manufacturing systems.

This research project focuses on a case study on flexible manufacturing systems for sustainable and agile manufacturing purposes [2]. A manufacturing machine is introduced that is not only modular from a hardware point of view, but also from a software perspective. This system should have some intelligence to be able to recognize its own environment. It needs to identify and configure the modules and capabilities automatically, or with limited support from a mechanic rather than an expert or engineer.

In this research project all systems and products have a virtual counterpart. These counterparts should be autonomous entities [8]. This led to a research project on an architecture that makes use of agent technology. The word agent comes from the Latin word agere, meaning: to act. Software agents are autonomous entities [3] with their own goals and the ability to communicate with other agents. Agent technology provides a way to deal with a dynamic, unpredictable environment and it enforces a system design with distributed control. There are many types of agents, for example a Belief Desire Intention (BDI) agent, see Fig. 1. A BDI agent has its background in the philosophies of Dennett [14] and Bratman [13]. The BDI agent uses its sensors to build a set of beliefs, its desires consist of goals, which lead to an intention of the agent, which characterizes the desire the agent has selected to work on. The BDI agent is also equipped with a set of plans and it will deliberately choose a plan to achieve its goals. Agents in manufacturing systems and products have been proposed before in several other research projects, like Ellis (recycle of rare earth elements) [16] and Kovacs (agent technology in car-recycling) [17]. Also Paolucci and Sacile [18] give an extensive overview of related work. Agents are used in

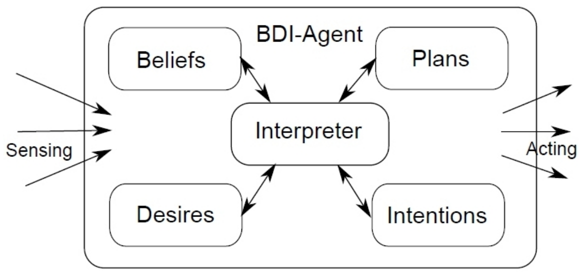

Figure 1. BDI-agent.

these systems for various reasons: scheduling, replacing human operators and control support. However, our research focuses on a production paradigm where every physical entity has a virtual agent counterpart, where all systems are distributed, and where the agents are essential for all processes: from scheduling and product logging towards direct control to the machine and module layer.

At the HU University of Applied Sciences Utrecht, the project HUniversal production was started to look at the current level of technology and use this to research and develop new modular and self-configuring machines, which are called equiplets [2]. Equiplets are meant to be low cost and have multi-purpose properties, offering generic services to any product which requires a production step that falls within its capabilities. An equiplet can be configured with a variety of modules necessary for manufacturing purposes, e.g., pick and place modules, grippers, (3D) printers, etc.

Fig. 2 shows three schematics of prototype equiplets with several modules. The particular equiplet on the left has a pick and place module (based on a delta robot), in the middle a gripper module is added to pick and place parts. On the right a camera module is added to get the location parameters of the objects it interacts with. While the hardware is changed manually or by another equiplet, software agents are used to automatically start the appropriate control software entities and set up the variables for the correct configuration of the system. This is what we call ’automatic reconfiguration’.

To deal with several equiplets, *Grid Manufacturing* [2] is introduced to show that production is not done anymore by production lines, but by flexible grids, which provide services to virtual products called *product agents* [8]. A grid is a set of equiplets. There are several agents on the grid layer. Hence the software of this manufacturing paradigm is based on a Multi Agent System (MAS). MAS consists of several interacting autonomous agents that should cooperate, coordinate and negotiate to achieve their goals. In a multi-agent system several abstract concepts are specified, for instance what the role, permission, responsibility and possible interactions are between each agent. In the concept of grid manufacturing these specifications are set for all entities. The product agent is a virtual (software) representative of the real product, which holds the requirements and knowledge that is needed to build it. There are also equiplet agents that represent the equiplets. The equiplet agents will interact with the product agents and other entities in the environment. The grid provides a flexible setup with a range of equiplets which uses modular tooling and hardware so that a large range of services can be provided

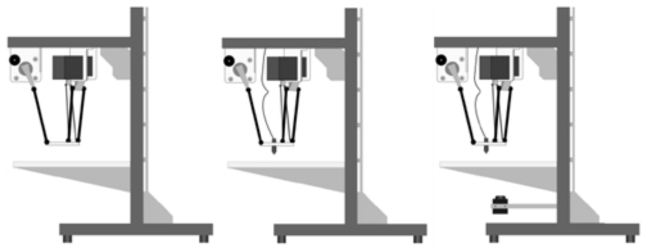

Figure 2. This particular equiplet has (from left to right) a pick and place module (based on a delta robot), a gripper module to pick and place parts and finally a camera module to get the location parameters of the objects it interacts with.

to the products. This makes it possible to create a range of different products on a single grid. Products can be produced dynamically using this concept. Intelligent agents also ease the processes on the grid layer; coordinating and configuring modules when necessary and providing the cognitive abilities that are necessary for the communication between equiplets and the products.

The paper shows the problems involved in the software of agile manufacturing systems and gives an example of how such a system could function. Next it states what is required to create such a flexible architecture for reconfigurable systems. The requirements, feasible for future use in industry, are discussed and matched. Also, a software hierarchy and basic architecture is presented that can be developed using these technologies. Finally, the conclusion states the potential for the combination of these technologies and discusses future work based on the analysis in this paper.

II. Problem Description

The goal of HUniversal production is to create a grid of manufacturing machines which is fully flexible, meaning it can create any product at any scale which falls within the sum of capabilities that are determined by the parameters and configurations of all equiplets in the grid. These capabilities can change at any time, since the grid is designed as such that it can be reconfigured at runtime. Equiplets can manually be added or removed with limited interference to other systems. As a result, software systems have to act autonomously, since they are unaware of all the systems and their capabilities in the environment. Giving them all the information of every system in the grid would make the software too complex and inflexible. This implies that a distributed control system is required, which uses autonomous software entities, where every entity can have unknown abilities. Since these systems still need to interact, it is required that they communicate in a more abstract manner. This way each system can interpret the abstract communication according to its knowledge. In such a system it is not possible to program beforehand which methods to call. To achieve cooperation, there is a need for dynamic behavior.

Fig. 3 shows an example of a simplified functional design of the architecture that is currently under development [9]. The example shows one product within a grid and two modular manufacturing systems (equiplets). Since these equiplets are configured with several modules they have to acquire the right information to be able to function. In Fig. 3, six steps are

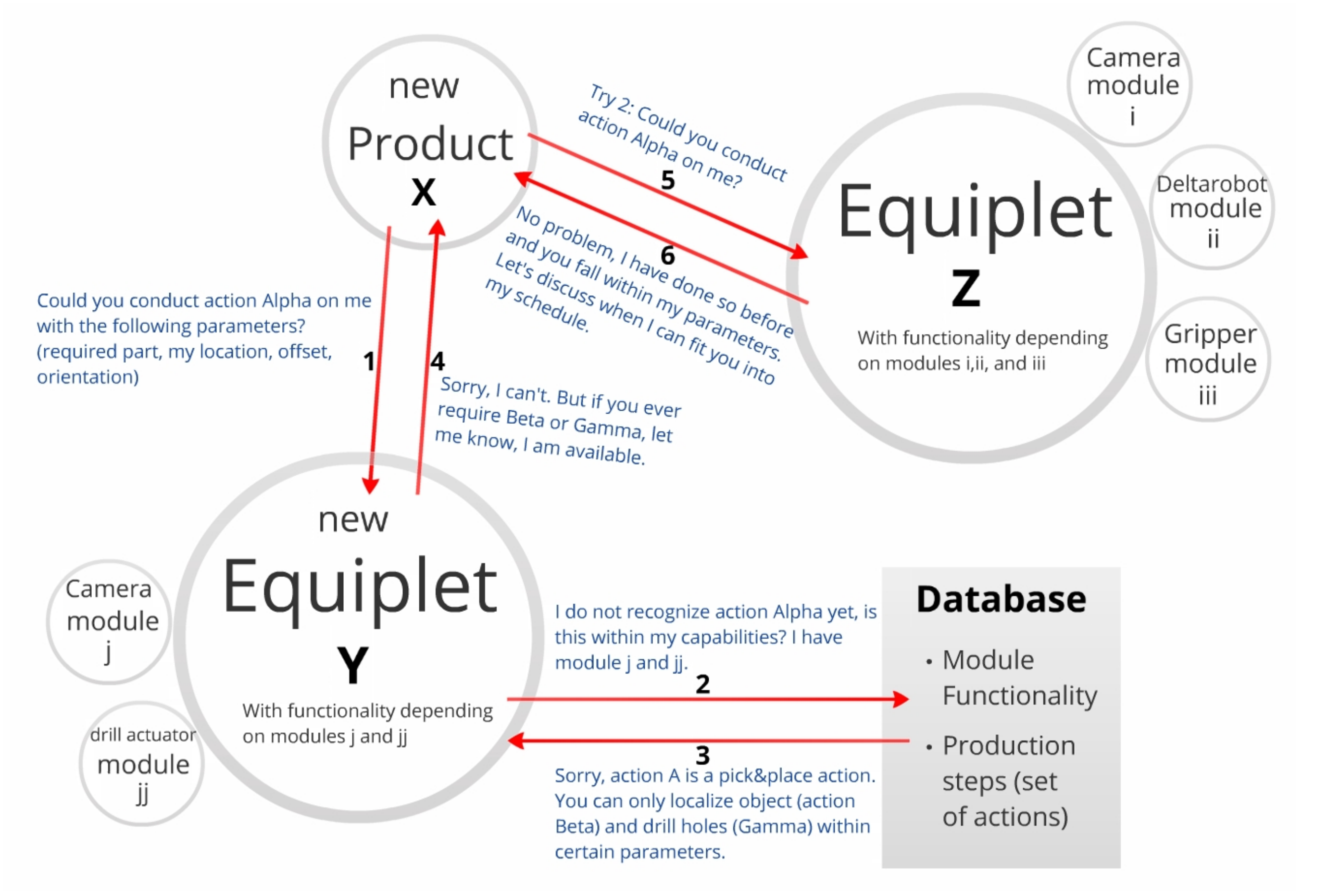
new Product X
Try 2: Could you conduct action Alpha on me?
5
6
No problem, I have done so before and you fall within my parameters. Let's discuss when I can fit you into my schedule.
Equiplet Z
With functionality depending on modules i,ii, and iii
Camera module i
Deltarobot module ii
Gripper module iii
Could you conduct action Alpha on me with the following parameters? (required part, my location, offset, orientation)
1
4
Sorry, I can't. But if you ever require Beta or Gamma, let me know, I am available.
new Equiplet Y
With functionality depending on modules j and jj
Camera module j
drill actuator module jj
I do not recognize action Alpha yet, is this within my capabilities? I have module j and jj.
2
3
Sorry, action A is a pick&place action. You can only localize object (action Beta) and drill holes (Gamma) within certain parameters.
Database
• Module Functionality
• Production steps (set of actions)

Figure 3. Simplified example of how a product communicates with equiplets in a grid where equiplets offer services within certain parameters.

presented when a product requires a pick and place action. In this example, equiplet Y is just added within the grid and has been configured with two modules. Equiplet Y has just been reconfigured with these modules and is therefore not yet aware of all its capabilities. Because the environment is completely flexible, it is possible that new equiplets, products or modules, and therefore new functionality and production steps, can be added at any time in a running system. This makes it necessary for the equiplets to inquire at the central database whether they are able to perform this specific action. To ”know” the capabilities of the equiplet we look towards the research of Järvenpää [6] who has developed a rule-based matching system for requirements and capabilities in production systems. Equiplet Z already has done several pick and place actions. Since it has not been reconfigured it can immediately answer to product X that its demands fall within the possibilities.

The largest problems that need to be overcome are the high software complexity which is a result of the flexible automation and the different requirements of the software on different levels within the software architecture. While the direct control of the hardware requires high performance, stability and real-time capabilities, the flexible agents require cognitive abilities, which are inherently slow and not real-time. These cognitive abilities are necessary for automatically reconfiguring the systems. The agents have to load the necessary software and setup the correct variables that it needs to function. Hence the modules and machines have to be able to recognize their capabilities, necessary parameters for their configuration, and dependencies. This requires an entirely new hybrid architecture, whereof the requirements and the hierarchy design are presented in this paper. The scope will be limited to the decisions on the requirements and necessary specifications of the technologies that are required to develop a proof of concept system using the current equiplet prototypes. The problem description brings us to the main research questions which are important at this phase:

- What are the requirements for the software architecture of an equiplet, using agents and MAS?
- Which technologies are available at this time that could be used in a proof of concept system and fit the requirements?
- What architecture can be used for industrial applications which works within a dynamic environment?

III. Requirements and Matching Technologies

The system is meant to become applicable in industry. The system is also reconfigurable and should work in a dynamic, non-deterministic environment. Hence, the logistics, i.e., where are the product parts that are necessary and the parts it interacts with, should be dynamic as well. The equiplet has to be able to find the component and important objects within the environment to interact with on a visual level. Since systems, objects and therefore functionality can be added at any moment in runtime, the system needs to have distributed control, where dependencies between systems have to be kept to a minimum [10]. It is unlikely that the dependencies between different systems and modules can be nil. Take for example a system that transports the product between equiplets. If such a system is in an error state, this will directly influence the equiplets it is connected to. It is important to decouple the depending systems as much as possible. This is done by using a data driven system. The different systems, e.g., 'higher' logistic systems and 'lower' control systems, are decoupled by data that can be influenced from both sides. This way an error in one autonomous system will not block another, also the performance of several systems will only be influenced by interdependent systems. Finally it is important that the abstraction of the system is as high as possible, for the hardware, as well as the software, interfaces and communication between systems.

Summarizing the requirements:

- Since the configuration and the logistics are flexible, every equiplet system should be visually aware of objects it needs to interact with within its working environment.
- There should be a minimum amount of dependencies between systems, since not all systems are aware of each other's existence, hence distributed control is necessary.
- To further decouple systems and be able to change and update information, the system should have some properties of a data driven software design. This way, it is easier to add new functionality for new products without directly interfering with the running systems. Required data for new actions could be requested from the database when necessary.
- There should be a high level of abstraction for the communication between the software entities themselves and the hardware. By using a higher abstraction level, communication can be made less specific which makes the system more flexible. Systems can use their own beliefs to interpret the messages they receive and act according to their own design goals and capabilities.
- Manufacturing machines interact within the environment. While most objects (products and modules) in the environment have an agent as virtual representative, there are also (very) simple hardware devices and software objects that only have to be used and as such are not depicted as a (software) entity. These objects in the environment should also be as abstract as possible such that all entities can interact with them when necessary.

This brings us to some specific requirements that are set to be the basis of an architecture that will be researched and developed in the next phases:

- Hardware systems should be able to communicate as abstract as possible with the higher order intelligent agents in the software. Therefore there is the need for a hardware abstraction layer where modules can be added in runtime.
- An adaptive system is required based on computer vision that can interpret the environment in which the manufacturing machine operates to distinguish production parts and their position.
- A higher level abstract architecture is necessary that can communicate with other unknown entities in the system and can deal with a dynamic and changing environment.

To apply all these requirements and to limit the scope of the research, we have decided to use the following technologies:

1) ROS - Robot Operating System is a software framework and provide tools and libraries. ROS can be used as an abstraction layer for the hardware in modular machines and robots. ROS utilizes the C++ program language and can be used directly to control hardware systems in real-time.
2) MAS - Multi Agent Systems [4] are the most likely option for the higher layer systems. Since a dynamic environment is used with many reconfigurable modules, it is important to decouple dependencies and lower the overall complexity by defining specific entities that each have their own responsibilities. Agents can act independently and have flexible behavior. Especially the Java Agent DEvelopment framework (JADE) seems of interest because of its low learning curve (based on the Java programming language), negotiation abilities, and its ability to migrate, terminate and add agents in runtime [15], the Jade platform has been in development since 2001 and has matured enough to be thrustworthy for industrial application.
3) OpenCV - Open Computer Vision is included with ROS, as such it is logical to choose for the OpenCV library that is integrated in this system. The computer vision is used to identify and localize parts within the working space of the equiplet and is used for other logistic processes necessary for configuration and calibration of the systems, e.g., identification of a new gripper.
4) Environment Programming - As mentioned in the requirements, the MAS system needs to interact with parts of the environment that might not be directly part of the MAS. Ricci introduced environment programming that considers the environment as an explicit part of the MAS and clearly distinguishes responsibilities between the agents and the environments. [11]. In Environment Programming there are agents and artifacts. Artifacts are non-autonomous, function-oriented entities that operate as a first level abstraction of usable objects in the environment.

IV. Discussion

The requirements and matching technologies provide the possibility to start the design of an architecture for control

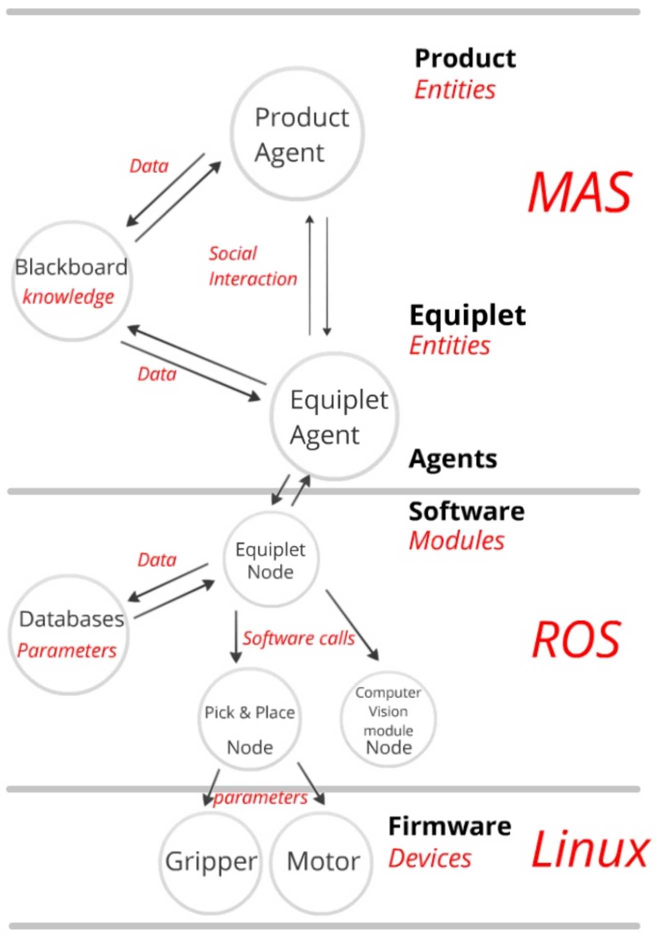

Figure 4. An example of the software hierarchy showing a single product and equiplet with several modules and the ROS, MAS and Linux layers.

of all systems within the equiplet and grid layer. To be able to create a system based on industrial specifications, it is necessary to combine these systems, hence to create a hybrid system [12]. The hybrid system is based on an architecture where the real-time capabilities with the physical abilities and cognitive abilities are split in two. In practice this means that real time capabilities like the direct control of sensors and actuators are managed by ROS, which runs on the Linux operating system and is especially tailored to interface with modular hardware. Cognitive abilities, like reasoning, are to determine if an equiplet has the capability to perform a new action on request by a product. This is handled by the MAS platform based on the JADE platform. The equiplets and grids have a modular design and are configurable in real-time, hence the system should be able to change the software when necessary. Both MAS and ROS platforms have these capabilities. Several MAS platforms are able to start new agent entities at runtime. ROS also works with entities that can interface with each other and it calls these software entities nodes. Nodes can be launched and stopped in runtime, but do not have the autonomous and cognitive capabilities like an agent. In other words, they are less intelligent.

In Fig. 4 the hierarchy is shown. On the MAS layer there is social interaction using the abstract communication on the equiplet and product layers between several entities. Note that there can be any number of equiplet and product agents and every entity represents a physical counterpart. The knowledge that is required by the data driven environment is stored on Blackboards, which can also be used to decouple the data from the agents that needs to be shared amongst the agents. On a lower layer ROS also uses (autonomous) entities. However, these are called nodes. Nodes are usually programmed on the ROS platform using the program language C++. While they are autonomous entities, they do not offer the abstract level of communication that is common among agents. However, ROS is designed to be used for Robot or Industrial Machine applications and does provide better performance and a more rigid and stable platform that is required for industrial applications. Also ROS provides a large amount of libraries, making it easier to develop new functionality. Therefore ROS provides a more robust method of the essential control of hardware systems. Finally at the lowest layer of the hierarchy there is the firmware layer, where devices are shown that are connected to the control system using an Industrial bus.

Figure 5. A QR code is used to identify items and modules in the environment, this particular QR code leads to a demonstration of an older prototype showing its flexible adaptation to the environment.

A. Reconfiguration

As determined, MAS and ROS can be configured with new software in runtime. However, how does any system know that it has been reconfigured? In other words, what is the trigger to start or update a node with the necessary software to use it? For this a system is designed that uses QR codes to ”scan” items. Every module or device has its own QR code, see Fig. 5, which uniquely identifies this module. This way the machine can be configured just by scanning a configuration QR code and a specific QR code that belongs to the new module that is being added or removed. When the Computer Vision system detects a reconfiguration code it will inform the ROS control system, which will in turn message the MAS system to find the appropriate software, either on the cognitive (MAS) or physical (ROS) layer. While Computer Vision is used to trigger changes in the hardware it is also used to provide other parameters. The environment is monitored by the system, providing information of the working field and objects within sight. By communicating with product agents and other equiplets also objects and systems are identified. This way the physical configuration of systems can also be determined automatically. If necessary, QR codes are also used to help with the identification. The QR codes can be identified accurately without the usual margin of error that is common in computer vision applications.

V. Conclusion and Future Work

The analysis shows that a proof of concept of a combination of ROS (using C++), MAS (using Java) and Computer Vision has potential for an industrial environment. Problems like stability and performance of the agent platforms are minimized by using the hybrid system with a combination of JADE and ROS. The more stable and performant ROS is used for critical systems that should be running at all times and MAS provides the cognitive abilities that are necessary in a dynamic distributed system. The hierarchy of these systems also lowers the overall complexity, because of the decoupling of systems and the distributed (autonomous) architecture. The distributed nature of the systems lowers the complexity of the software and makes it possible to address problems with a black box approach, focusing on the interfacing behaviour between the entities. The software auto configuration is made possible by using the MAS and ROS architecture, that are both based on autonomous entities, which can be added and configured in runtime. In combination with the computer vision systems based on OpenCV the parameters of the environment are used to quickly determine the state of the environment, identifying objects of interest and making it easy to configure new hardware systems when necessary.

Currently, due to the analysis in this paper, the research group has decided to start the development of the software architecture as presented. The functional design of the software has been completed and the hardware of several modules is in its beta phase. Progress has also been made on the hardware, in Fig. 6 the newest (4th generation) equiplet prototype is shown. This prototype uses the distributed system as proposed, including the suggested hybrid architecture using agents and nodes coupled by blackboards. The programming of various subprojects are in progress, like research on the lifecycle of a product [8], which is used to acquire information on the product for (sustainable) goals like recycling and repairs. Several other projects are planned for the future, for example how grid agents can be used for optimalisation, intelligent behavior within an equiplet and the development of more hardware and software modules.

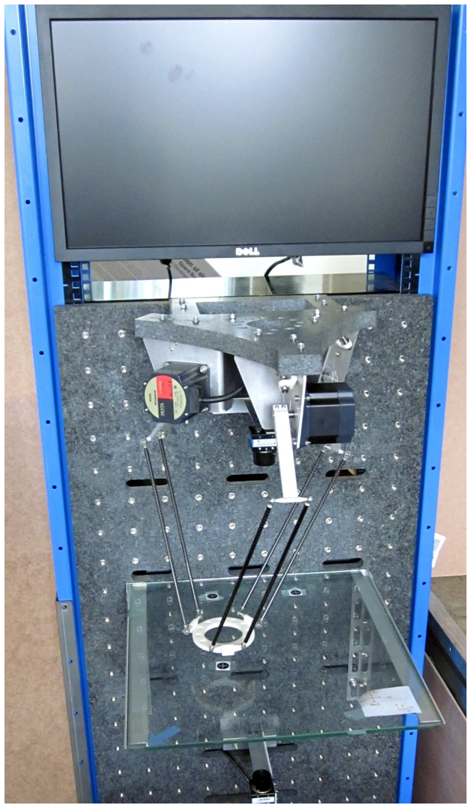
Figure 6. One of the new prototype equiplets currently under assembly.

References

[1] Y. Koren et al., Reconfigurable Manufacturing Systems, CIRP Annals - Manufacturing Technology, vol. 48, no. 2, 1999, pp. 527-540.

[2] E. Puik and L.J.M. van Moergestel, Agile multi-parallel micro manufacturing using a grid of equiplets, IPAS 2010 proceedings, 2010, pp. 271-282.

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

[4] M. Wooldridge, ”An Introduction to Multi Agent Systems”, Second Edition, Wiley, 2009.

[5] E. Puik, L. van Moergestel, and D.Telgen, Cost Modeling for Micro Manufacturing Logistics when using a Grid of equiplets, ISAM2011, IEEE, Finland, 2011.

[6] 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, Helsinki, 2012, pp. 449-456.

[7] A.Gunasekaran, Agile manufacturing: the 21st century competitive strategy, 2001.

[8] L. van Moergestel, E. Puik, D. Telgen, H. Folmer, M. Grnbauer, R. Proost, H. Veringa and J.-J. Meyer, ”Monitoring Agents in Complex Products - Enhancing a Discovery Robot with an Agent for Monitoring, Maintenance and Disaster Prevention”, ICAART2013 proceedings, volume 2 , 2013, pp. 5-13.

[9] D. Telgen, L. van Moergestel, E. Puik, and J-J. Meyer, ”Agent Manufacturing Possibilities with Agent Technology”, FAIM2012 proceedings, Helsinki, Finland, 2012, pp. 341-346.

[10] L. van Moergestel, E. Puik, D. Telgen, and J-J. Meyer, Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing, ISADS 2011 Japan, 2011, pp. 281-288.

[11] A. Ricci, M Piunti, and M. Viroli, ”Environment Programming in Multi-Agent Systems An Artifact-Based Perspective” Autonomous Agents and Multi-Agent Systems journal 23(2), September 2011, pp. 158-192.

[12] R.C. Arkin, ”Behaviour-Based Robotics”, MIT press, Hybrid Systems, 1998, pp. 205-234.

[13] M.E. Bratman, ”Intention, Plans, and Practical Reason”, Harvard University Press, Cambridge, Mass, 1987.

[14] D.C. Dennett, ”The Intentional Stance”, MIT Press, Cambridge, Mass, 1987.

[15] N.R. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, ”Multi-Agent Programming”, Springer, 2005.

[16] T.W. Ellis, F.A. Smith, and L.L. Jones. ”Methods and opportunities in the recycling of rare earth based materials”, The Metallurgical Society (TMS) conference on high performance composites, (IS-M796), 1994.

[17] G. Kovacs and G. Heidegger, ”Car-recycling sme network with agent-based solutions”, European Research Consortium for Informatics and Mathematics, (73), 2008, pp. 53-54.

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