Life Cycle Agent
Beyond Intelligent Manufacturing

Leo van Moergestel, Erik Puik,
Daniël Telgen, Feiko Wielsma, Geoffrey Mastenbroek,
Robbin van den Berg, Arnoud den Haring
Institute for ICT
HU Utrecht University of Applied Sciences
Utrecht, the Netherlands
Email: leo.vanmoergestel@hu.nl

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

***Abstract*—In this paper, a model is proposed where a software agent will be tied to a product during all parts of its life-cycle. This agent will enhance the possibilities of the product itself but will also play a role in collecting important data from the device that can be used by the manufacturer, as well as the end user. The software agent will be the basis for the device to participate in the Internet of Things (IoT) concept. The motivation to use agent technology, the architecture as well as the implementation in two different products are presented.**

***Keywords*–Agent technology; Internet of Things; Lifecyle agent**

I. INTRODUCTION

Today, Information technology plays a major role in manufacturing as well as in other aspects of our modern society. In manufacturing the trend is towards low-cost agile manufacturing of small batch sizes or even one product according to enduser requirements. This is also known as Industry 4.0 [1] or cyber physical systems [2]. Recycling is also an important issue that should have attention starting at the design and manufacturing phase. In our daily life, more and more devices are connected to the Internet thus creating the Internet of Things (IoT). In this paper, a concept is presented where a software entity will be responsible for a certain product in all phases of its life cycle. This software entity starts by manufacturing the product and will collect data during the manufacturing phase. Next, the software entity will be tied to or even embedded in the product, making it a part of the IoT during all phases of its life cycle. For most products, the usage phase has the longest duration. It turns out that this phase can also play a role to adapt the production. When a manufacturer has a lot of data available about the usage of a product, the manufacturing process itself can benefit. This paper will describe the concept of the software entity and the architecture used to implement the concept. The focus will be on the usage phase and two cases are elaborated where a complex device will collect usage data as well as open the possibility to be remotely monitored and controlled.

The rest of this paper is organised as follows. Section II is dedicated to the related work and definitions. In Section III, the motivation for the chosen technology and the advantages will be discussed followed by Section IV presenting two specific cases where the concept, architecture and implementation will be explained. Section V about future work and a conclusion will end the paper.

II. RELATED WORK AND DEFINITIONS

The concept of using a software entity to guide a product through its life cycle was first published by Moergestel [3]. The software entity used was a so called software agent. Nowadays, the concept of an agent is already widely known in the field of information technology. Unfortunately, there are several definitions, so we give here the definition as stated by Wooldridge [4], that will be used in this paper:

Definition: *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.*

For reasons explained in the next section, we will stick to the concept of an agent and for this paper we have a special purpose for the software agent in mind: the life cycle agent.

Definition: *a life cycle agent is a software entity that is the representative of a product in all phases of its life cycle.*

For more complex products having an environment where such an agent can run, as the actual implementation a twin agent system is proposed, where one agent lives in the product itself and another one in cyberspace. These agents will synchronise to keep the knowledge on both systems equal and up-to-date.

Every product goes through a sequence of phases as depicted in Figure 1. Starting with design, the product will be manufactured and distributed to reach the actual user. This user will use the product, perhaps hand it over to another user. During this usage phase repair and maintenance play a role. Finally the product will come to the end of its life and parts of it will be recycled.

Design
Manufacturing
Distribution
Use
Recycling

Figure 1. Different phases in the life cycle

The manufacturing starts by the agent itself. It will control the production as described in [5]. This approach is also used by Bussmann [6]. Agent-based manufacturing is also described by Paoluci and Sacile [7]. The difference with the solutions for manufacturing so far is that in our concept [5], the agent controlling the production will stay alive and embed itself in the product, becoming the life cycle agent. If embedding is not possible, the agent will live remote and keep contact with the product to send and receive information. It should be clear that a certain complexity of the product is assumed. The product should have sensors for measuring things of interest and it should be capable to communicate this to the outside world, either continuously or on demand.

Monitoring systems that check and collect data during the usage phase, are widely used nowadays. An Aircraft is constantly monitored during the flight. These monitoring systems are specially designed for the aircraft involved [8]. The information collected is used for safety, preventive maintenance as well as redesign. So the purpose of monitoring is comparable to the agent-based concept presented. A term used in the transport industry is Health and Usage Monitoring System (HUMS) [9]. This system helps to ensure availability, reliability and safety of vehicles. An overview of health monitoring is given in [10]. These monitoring systems are specially designed for monitoring purposes and rarely used in other parts of the life cycle. Monitoring for medical and human health is also an interesting and important aspect. An example of an application is this area is given by Otto [11].

III. Technology used and its advantages

In this section, the choice for agent technology is motivated. Another part of this chapter presents an overview of the benefits of the model for different stakeholders.

A. Agent technology

The reason why an agent is proposed is based on some important characteristics of agents:

- autonomy; no user intervention is required. A device can operate on its own.
- communicating; devices need to communicate with the external world.
- reactive: this property will make the device work as expected.
- pro-active: this might be a property that will make the device smart.
- mobile: an agent can move from one platform to another.
- learning: this is also a property that makes a device smart
- adapting: the device can adapt itself to different situations
- reasoning: an advanced property for making the device smart
- cooperative (important in a multiagent approach).

All these properties fit well within the concept of a smart software entity to guide and represent a product and making it a part of the IoT.

When agent-based product guidance is used, two possibilities arise:

1) one single agent is developed to guide the product during its whole life cycle. This agent might have a backup or counterpart outside the device living in cyberspace.
2) a multiagent approach is used, where different agents operate at different phases but where information exchange between these agents is possible.

Without pretending to give an exhaustive overview, we will now describe some advantages of using agents in the life cycle of a product.

B. Advantages

To investigate the benefits of the approach proposed, the advantages and possibilities for different stakeholders will be presented.

1) manufacturer: Though the life cycle agent concept was introduced as a tool to implement agile manufacturing for batch size equal to one, the concept can be used for all types of products having a certain complexity and the capability to register data from sensors installed in the product itself. For the manufacturing phase, the concept is the enabler for agile production of small quantities as described in [12]. It will result in logging the production data for every single product. However, in case of a batch production the logging could be specific for the whole batch and shared by all products belonging to that production batch.

By using the feedback during usage, the manufacturing process itself and the product can be optimized, because of the availability of usage information. Over the air (OTA) updates of product software can be performed within the proposed model. The mean time between failure (MTBF) of subparts of the system will become a well-known factor because of the information collected during usage. Finally the end-user can be specifically advised by the manufacturer about a replacement of a product, knowing the type of usage by that specific end-user.

2) distributer: Logistic systems can benefit from the fact that the product involved is already ”smart” by having a software agent available. If the device is capable of using its sensors during transport, the possibility arises to check that transport has been done under acceptable conditions (temperature, shock, time involved etc.). This might be helpful because in many situations the distributor is responsible for damage during transport.

3) end-user: When a device is connected to the internet, several possibilities that can benefit the end-user arise. First, the device can be monitored and controlled by the end-user using a device like a smart phone, tablet or any other system having a web browser. By collecting the usage that is made the device can optimise itself to the usage of a specific end-user. Preventive maintenance and over the air updates can also benefit the user.

4) environment: Because the usage, MTBF and wear of several subparts is available as well as the maintenance and replacement of subsystems, during the process of recycling, subparts can be reused based on these data. This makes it possible to reuse materials as well because during the manufacturing phase, data about the materials used are collected by the life cycle agent. All these possibilities will reduce the amount of waste.

IV. Usage phase cases

Two different types of devices are shown as proof of concept. The first one is an autonomous robot vacuum cleaner. The second one is a radio device for playing internet audio streams. Though different there are also similarities. In both cases the device contains the local agent. In case of the internet radio, the hardware where the agent resides is also the hardware that controls the device itself. In the case of the vacuum cleaner, the agent has its own specific hardware that is only connected to the system to get information about the internal controls. Both cases use a Raspberry-Pi as the agent platform. The reason is the low price and extensive documentation available.

A. Generic and specific

The approach used was to build a generic system, that can be used for a wide range of devices, while extra software will be added to adapt to the specific properties of a device.

1) Functional requirements: When the needs for both manufacturer and user in the use phase are considered, the following list applies:

- the manufacturer needs data about the usage of the products
- the manufacturer needs data about component failures
- the user wants to remotely control and monitor the product.
- the user wants a generic system with similarities among different products

These needs resulted in the following functional requirements:

- interface to collect information about the usage of a product
- connection with the cloud, to prevent that data will be lost in case a product is completely destroyed.
- Easy configuration for both the user as well as the manufacturer
- Agent-based to fit in the concept of the life cycle agent. The advantages and reason to use agent technology are already explained in the previous section.

2) overview: Figure 2 shows an overview of the architecture proposed. As seen in the figure, there are actually two agents involved for every device. One residing in the device, while the other is living in the cloud. The reason for this is that in situations where the agent in the device is completely destroyed, the cloud agents still has all the information available. A collection of cloud agents store their data in a database that is accessible to the manufacturer. An API will open the possibility for the end-user to interact with the device.

3) Technical requirements: For the implementation Jade has been used. Some important properties of jade are: Jade [13] was used as a platform for the Multiagent System (MAS). The reasons for choosing Jade are:

- the system is a multiagent-based system. Jade provides most of 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;
- the life cycle agents should be capable to negotiate to reach their goals. Jade offers possibilities for agents to negotiate. If we need extra capabilities, the Jade platform can easily be upgraded to an environment that is especially designed for BDI agents like 2APL [14] or Jadex [[13]]. Both 2APL as well as Jadex are based on Jade but have a more steep learning curve for Java developers;
- agents can migrate, terminate or new agents can appear.

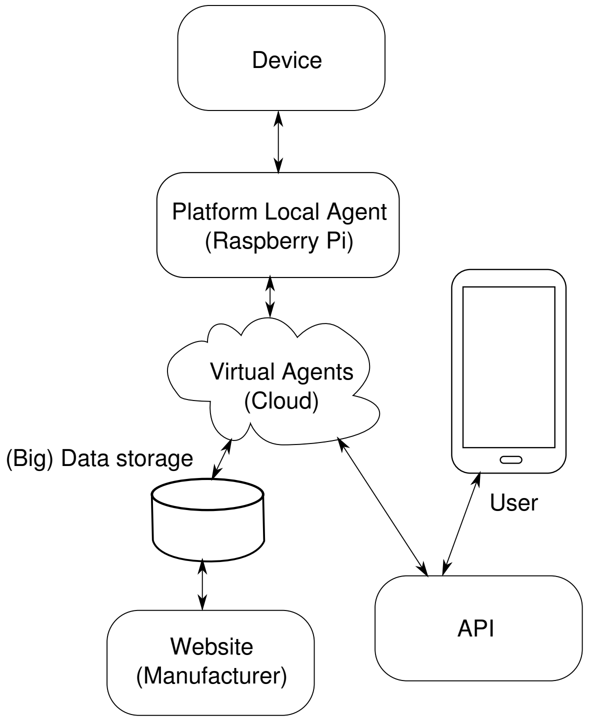


Figure 2. Architecture overview

The Jade runtime environment implements message-based communication between agents running on different platforms connected by a network. In Figure 3, the Jade platform environment is depicted.

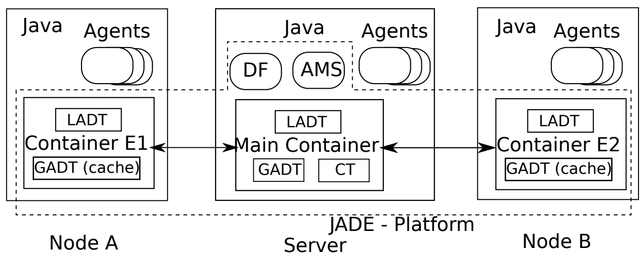


Figure 3. The Jade platform

The Jade platform itself is in this figure surrounded by a dashed line. It consists of the following components:

- A main container with connections to remote containers (in our case Node A and Node B, representing for example other computer platforms running Java);
- A container table (CT) residing in the main container, which is the registry of the object references and

transport addresses of all container nodes composing the platform;

- A global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their status and location. This table resides in the main container and there are cached entries in the other containers;
- All containers have a local agent descriptor table (LADT), describing the local agents in the container;
- The main container also hosts two special agents AMS and DF, that provide the agent management and the yellow page service (Directory Facilitator) where agents can register their services or search for available services.

Several requirements justified this choice.

- Open source, so the further development and integration with other sofwtare developments would be possible, without dependence on developers of closed software.
- Based on a standard widely accepted programming language, making the adoption by third parties easy.
- Jade has been designed as a platform for a distributed multiagent system. This is exactly the type of multiagent system that is needed in our case.

4) Software architecture: The generic system has a modular setup. The main modules of the system are depicted in Figure 4 (LCA stands for Life Cycle Agent). LCADevice is the module where the life cycle agent residing in the device is created, while LCACloud is the module for the agent living in cyberspace. LCAWebAPI contains a REST API (REpresentational State Transfer Application Programming Interface) [15] that enables message transfer (by HTTP requests) to the JADE platform that has been used for the implementation (see the subsection Technical Requirements). All messages used by the system use the same concept and are available to all modules by LCAMessage.

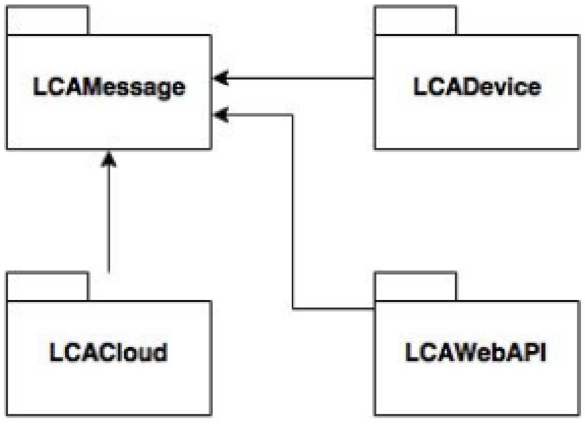


Figure 4. Modules used

B. case vacuum cleaner

The model used in this experiment was a Roomba vacuum cleaner. This brand had been chosen because of the availability of an Open Interface (OI). A document is available where this OI is described [16]. By using this interface most of the sensors and actuators can be used. A simple serial interface makes the connection to the device. A Raspberry-Pi was added to the Roomba to enable a JADE runtime environment for the agent and for establishing a WiFi connection.

1) sensors and actuators: The sensors can give information if there are walls or holes in the floor near the device. The amount of current used by the motor is available. A raise can signal a wearing-out of the brushes. The buttons on the device can be read. The distance travelled in millimetres since the previous request can be read. Voltage and current from the battery as well as the capacity and maximum capacity are available. Bumper sensors are also available. Apart from the sensors there are actuators.

- The motors driving the wheels. Speed and turning can be controlled.
- Motors moving the brushes. The speed can be controlled by Pulse Width Modulation (PWM).
- Several LEDs are available on the device. The intensity is also controllable.
- The device contains a speaker for making sounds. Sounds can be loaded and played.

2) Roomba application: As a proof of concept, an application has been developed to monitor and control the vacuum cleaner by the end-user. The vacuum cleaner should be connected to the LifeCycleAgent platform as explained earlier in this paper. The following list of functionalities is available (see the start menu in Figure 5):

- user login;
- battery status (see figure 6);
- playing music;
- start cleaning;
- return to dock;
- remote control for driving the cleaner around.

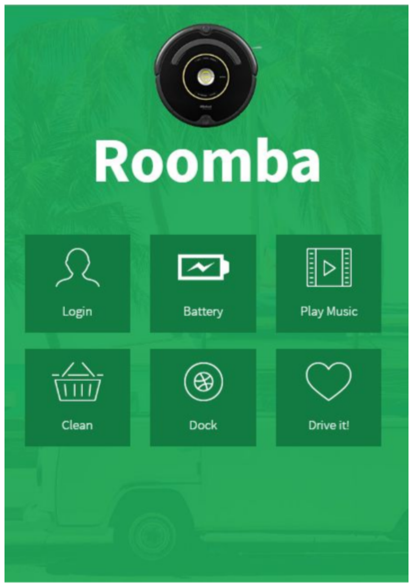
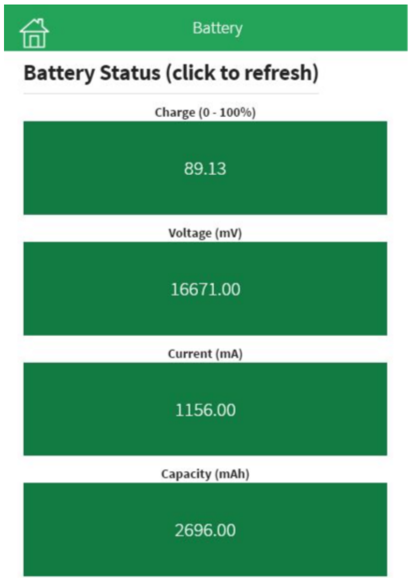


Figure 5. Main menu for the user



Battery Status (click to refresh)

Charge (0 - 100%)	89.13
Voltage (mV)	16671.00
Current (mA)	1156.00
Capacity (mAh)	2696.00

Figure 6. Battery status page

To make this implementation work, two device-specific software modules were developed. This is depicted in Figure 7.

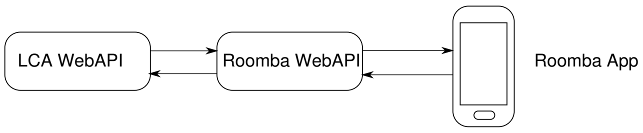


Figure 7. Architecture

The Roomba App will be installed on the smartphone or tablet and is the actual user interface as presented in Figures 5 and 6. Because this is a device-specific application, a second specific application has been added to control the communication between the Roomba App and de LCA WebAPI. Normally, these two parts should be supplied for every specific device.

C. case internet radio

In Figure 8 a blockscheme of the internet radio is shown. The core of the system is a Raspberry-Pi enhanced with a touchscreen and a powerful soundsystem that drives the speakers.

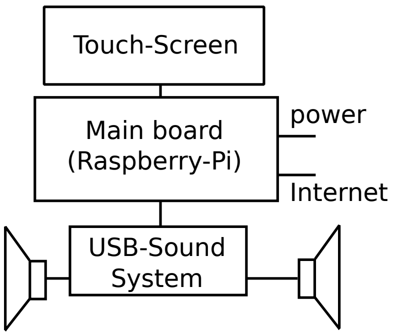


Figure 8. Block schematic

1) sensors: This is the list of input data available: The volume setting, currently chosen audio stream, play (yes/no), type of music chosen. Strictly speaking this information is not directly originating from a sensor, but it is data provided by the software entity that is the internet radio itself.

2) actuators: The actions that can be performed are:

- change the selected audio stream;
- change volume
- start playing
- stop playing
- pause playing

3) realisation: An application for remote control has not been completed yet, but the radio is integrated as a system in the life cycle agent platform, having its own virtual agent to synchronise with. The usage of the radio is monitored by the agent that lives inside the radio and the information is also available for the agent in the cloud.

The display is shown in Figure 9. The radio itself is shown in Figure 10.

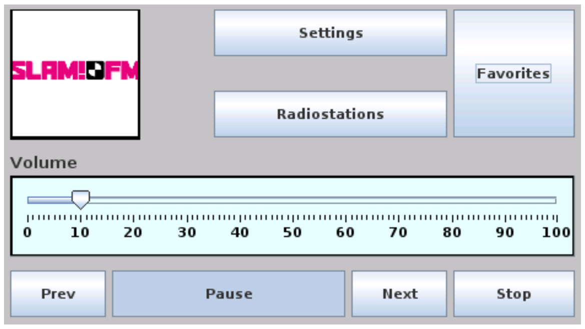


Figure 9. Display

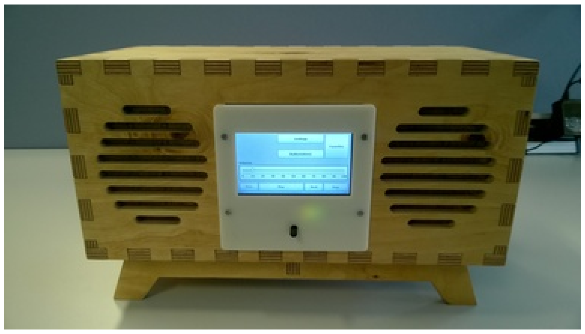


Figure 10. Radio

In Figure 10, a design is shown that was made by using a water jet cutter to make the different panels. To these panels several components are attached. The radio consists of six panels (front, back, top, bottom, left and right) with zero or more attachments. The end-user can select a shape, resulting in these six panels. This results in a toplevel XML-file where the radio is defined to consist of a these six panels in combination with actions describing how to assemble the radio using these panels. A simplified example of this XML-file that describes the components of the radio as well as the actions to be taken to construct it, looks like:

```
<Radio>
  <Source>
    six panels
  </Source>
  <Actions>
    assembly instructions
  </Actions>
</Radio>
```

Other XML-based information is added to refine the description, that is used to actually construct the radio.

V. FUTURE WORK

The manufacturing phase is still under development. Several production machines have been constructed. The flexible transport system is still under development. For the internet radio the user specified production is also under development. That means that a webinterface will be created where a user can specify his or her specific implementation of an internet radio. A list of so-called production steps (explaining what to do) and materials (what to use) should be the result of this design phase. This would be the input of an agent that will guide the manufacturing and will become the life cycle agent as presented in this paper. The first step in that direction will be to replace the production machine agents by real humans. These human agents will be instructed by the life cycle agent during the manufacturing phase to actually make the product. Meanwhile the recycling phase is studied by implementing a marketplace for devices to buy and sell parts for maintenance and repair. In all these situations we try to stick to the same multiagent-based architecture, so migration from one phase to another should be easy.

A special focus should be on the ethical aspects of the approach proposed in this paper. When information is collected during the usage phase, all kind of questions arise, like who owns this information. Who should have access to this information and so on. An end-user owning a device might have concerns about his or her privacy and should be capable to decide who should have access about the usage data of the device.

VI. CONCLUSION

In this paper, the concept of a life cycle agent has been introduced. The motivation for using agent technology was that this fits all the requirements that the system proposed should have. The multiagent architecture of the distributed system has been presented. To test this architecture in the use phase the system has been implemented. By using two different cases the generic and specific parts of the architecture became clear. The system worked as expected and can be further developed.

REFERENCES

[1] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, “How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective,” International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, vol. 8, no. 1, 2014, pp. 37–44.

[2] R. Rajkumar, I. Lee, Insup, S. L., and J. Stankovic, “Cyber-physical systems: The next computing revolution,” Proceedings of the 47th Design Automation Conference (DAC), Anaheim, California, 2010, pp. 731–736.

[3] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, “The role of agents in the lifecycle of a product,” CMD 2010 proceedings, 2010, pp. 28–32.

[4] M. Wooldridge, An Introduction to MultiAgent Systems, Second Edition. Sussex, UK: Wiley, 2009.

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

[6] S. Bussmann, N. Jennings, and M. Wooldridge, Multiagent Systems for Manufacturing Control. Berlin Heidelberg: Springer-Verlag, 2004.

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

[8] URL, HindSight-Eurocontrol Publications. at http://www.skybrary.aero/index.php/HindSight_-_EUROCONTROL, june, 2017.

[9] D. He, S. Wu, and E. Bechhoefer, Use of physics-based approach to enhance HUMS prognostic capability, 2007, vol. 1, pp. 354–361.

[10] H. Sohn, C. R. Farrar, F. M. Hemez, D. D. Shunk, D. W. Stinemates, B. R. Nadler, and J. J. Czarnecki, “A review of structural health monitoring literature: 1996–2001,” Los Alamos National Laboratory, 2003, pp. 1–7.

[11] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, “System architecture of a wireless body area sensor network for ubiquitous health monitoring,” Journal of mobile multimedia, vol. 1, no. 4, 2006, pp. 307–326.

[12] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, “Implementation of manufacturing as a service: A pull-driven agent-based manufacturing grid,” Proceedings of the 11th International Conference on ICT in Education, Research and Industrial Applications (ICTERI 2015), Lviv, Ukraine, 2015, pp. 172–187.

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

[14] M. Dastani, “2apl: a practical agent programming language,” Autonomous Agents and Multi-Agent Systems, vol. 16, no. 3, 2008, pp. 214–248.

[15] R. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[16] URL, pdf file name iRobot_ Roomba_ 500_ Open_ Interface_spec.pdf. at http://www.irobot.lv/uploaded_files/File, june, 2017.