



A Framework for Industry 4.0

JOSÉ BRUNO MARTINS DA SILVA

Julho de 2017

José Bruno Martins da Silva

A Framework for Industry 4.0

Dissertation in Computer Engineering,
Area of Specialization in Software Engineering

Advisor: Professor Doutor Luís Lino Ferreira

Co-advisors: Doutor Michele Albano & Cláudio Maia

Porto
June 2017

Abstract

The potential of the Industry 4.0 will allow the national industry to develop all kinds of procedures, especially in terms of competitive differentiation. The prospects and motivations behind Industry 4.0 are related to the management that is essentially geared towards industrial internet, to the integrated analysis and use of data, to the digitalization of products and services, to new disruptive business models and to the cooperation within the value chain. It is through the integration of Cyber-Physical Systems (CPS), into the maintenance process that it is possible to carry out a continuous monitoring of industrial machines, as well as to apply advanced techniques for predictive and proactive maintenance.

The present work is based on the MANTIS project, aiming to construct a specific platform for the proactive maintenance of industrial machines, targeting particularly the case of GreenBender ADIRA Steel Sheet. In other words, the aim is to reduce maintenance costs, increase the efficiency of the process and consequently the profit. Essentially, the MANTIS project is a multinational research project, where the CISTER Research Unit plays a key role, particularly in providing the communications infrastructure for one MANTIS Pilot.

The methodology is based on a follow-up study, which is jointly carried with the client, as well as within the scope of the implementation of the ADIRA Pilot. The macro phases that are followed in the present work are: 1) detailed analysis of the business needs; 2) preparation of the architecture specification; 3) implementation/development; 4) tests and validation; 5) support; 6) stabilization; 7) corrective and evolutionary maintenance; and 8) final project analysis and corrective measures to be applied in future projects.

The expected results of the development of such project are related to the integration of the industrial maintenance process, to the continuous monitoring of the machines and to the application of advanced techniques of preventive and proactive maintenance of industrial machines, particularly based on techniques and good practices of the Software Engineering area and on the integration of Cyber-Physical Systems.

Keywords: Cyber-Physical Systems; Industry 4.0; MANTIS; Solution.

Resumo

O potencial desenvolvido pela Indústria 4.0 dotará a indústria nacional de capacidades para desenvolver todo o tipo de procedimentos, especialmente a nível da diferenciação competitiva. As perspetivas e as motivações por detrás da Indústria 4.0 estão relacionadas com uma gestão essencialmente direcionada para a internet industrial, com uma análise integrada e utilização de dados, com a digitalização de produtos e de serviços, com novos modelos disruptivos de negócio e com uma cooperação horizontal no âmbito da cadeia de valor. É através da integração dos sistemas ciber-físicos no processo de manutenção que é possível proceder a um monitoramento contínuo das máquinas, tal como à aplicação de técnicas avançadas para a manutenção preditiva e pró-ativa das mesmas.

O presente trabalho é baseado no projeto MANTIS, objetivando, portanto, a construção de uma plataforma específica para a manutenção pró-ativa das máquinas industriais, neste caso em concreto das prensas, que serão as máquinas industriais analisadas ao longo do presente trabalho. Dito de um outro modo, objetiva-se, através de uma plataforma em específico, reduzir todos os custos da sua manutenção, aumentando, portanto, os lucros industriais advindos da produção. Resumidamente, o projeto MANTIS consiste num projeto de investigação multinacional, onde a Unidade de Investigação CISTER desenvolve um papel fundamental, particularmente no fornecimento da infraestrutura de comunicação no Piloto MANTIS.

A metodologia adotada é baseada num estudo de acompanhamento, realizado em conjunto com o cliente, e no âmbito da implementação do Piloto da ADIRA. As macro fases que são compreendidas por esta metodologia, e as quais serão seguidas, são: 1) análise detalhada das necessidades de negócio; 2) preparação da especificação da arquitetura; 3) implementação/desenvolvimento; 4) testes e validação; 5) suporte; 6) estabilização; 7) manutenção corretiva e evolutiva; e 8) análise final do projeto e medidas corretivas a aplicar em projetos futuros.

Os resultados esperados com o desenvolvimento do projeto estão relacionados com a integração do processo de manutenção industrial, a monitorização contínua das máquinas e a aplicação de técnicas avançadas de manutenção preventiva e pós-ativa das máquinas, especialmente com base em técnicas e boas práticas da área de Engenharia de Software.

Palavras-chave: Sistemas Ciber-físicos; Indústria 4.0; MANTIS; Solução.

Acknowledgements

The present work is the culmination of a cycle of 5 years of study, therefore consisting in the development of my academic and personal life. The people who surrounded me for the past few years, and in a daily basis, and that helped me throughout this entire cycle undoubtedly deserve my gratitude.

First of all, I would like to thank my teacher and counselor, Prof. Dr. Luís Lino Ferreira, who was extremely comprehensive and tolerant throughout the entire time when it was necessary to discuss the solution, or even when the stakeholders decided to change some of the requirements, which implied some changes in the present work, since he provided all the support and motivation I needed to fulfil this goal of mine.

There are truly no words to describe how much Prof. Dr. Luís Lino Ferreira, Prof. Dr. Michele Albano and Eng. Cláudio Maia were important during the development of this thesis. Hence, I would like to thank you all for helping me and, above all, for being true friends. I will never forget you.

To the educational institution, especially to the Director of the course, Prof. Dr. Nuno Silva, a big thank you for the opportunity and for your understanding, as well as for providing all the necessary resources to the fulfillment of this degree.

I am also grateful to my partner, Salete Coelho, and to my son, Daniel Silva, who I love the most, for always being there for me. For all your support and motivation, which I needed to achieve this important goal, thank you so much my loves.

Finally, thank you to all those people that one way or another contributed to the conclusion of this dissertation. A sincere thank you to you all.

Index

1	Introduction.....	12
1.1	Context.....	12
1.2	The Problem	13
1.3	Objectives.....	19
1.4	Solution Overview	21
1.4.1	Machine	22
1.4.2	Edge Local.....	23
1.4.3	Edge Server.....	24
1.4.4	Data Analysis	24
1.4.5	Human Machine Interface	25
1.5	Development Process and Work Methodology.....	26
1.6	Document Structure.....	28
2	Industry 4.0	30
2.1	Introduction.....	30
2.2	Background.....	32
2.3	Industrial Concerns	33
2.4	Value offer	34
2.5	Architectures	35
2.5.1	Reference Architecture Model Industry 4.0 (RAMI4.0).....	35
2.5.2	Industrial internet reference architecture (IIRA)	36
2.5.3	Service-Oriented Architecture (Arrowhead Framework)	37
2.6	Components	40
2.7	Design Principles.....	44
2.8	Development Process	49
2.9	Conclusions.....	51
3	Evaluation of Message Oriented Middleware solutions.....	52
3.1	Advanced Message Queueing Protocol (AMQP)	52
3.2	Message Queuing Telemetry Transport (MQTT)	53
3.3	Simple Text Oriented Messaging Protocol (STOMP)	53
3.4	Java Message Service (JMS).....	54
3.5	Extensible Messaging Presence Protocol (XMPP)	54
3.6	Data Distribution Service (DDS).....	55
3.7	Open Platform Communications Unified Architecture (OPC-UA) publish-subscribe	55
3.8	Conclusions.....	58
4	Value analysis	59
4.1	Stakeholders concerns	59
4.2	Business Process and Innovation.....	60
4.3	Value offer	62
4.4	Value of the Current Market	64
4.5	Business Model Canvas	65
4.6	Competition analysis.....	68
4.7	Conclusions.....	71
5	Analysis	72
5.1	Proposed architecture.....	72
	Requirements Engineering	74
5.2.....		74
5.2.1	System actors	74

5.2.2	Use Cases	75
5.3	Business Modeling	79
5.3.1	Domain Model	79
5.3.2	Event model & Semantic model	82
5.3.3	Database	84
5.4	Design and architectural patterns	86
5.5	Conclusions	87
6	Design & Implementation	88
6.1	Machine Subsystem	88
6.1.1	OPC-UA-Server Component	89
6.1.2	CNC Bender Component	93
6.1.3	Sensors Component	94
6.2	Mantis-PC Subsystem	98
6.2.1	OPC-UA Server Component	99
6.2.2	BLE Server Component	99
6.2.3	99
6.3	Edge Local Subsystem	102
6.3.1	Node-RED Component	103
6.3.2	Middleware Client Component	106
6.3.3	OPC-UA Client component	107
6.4	Data Analysis Subsystem	108
6.4.1	Middleware Client Component	108
6.5	Edge Server Subsystem	111
6.5.1	Middleware Component	112
6.5.2	Manager Component	117
6.5.3	RabbitMQ-Core Module	120
6.5.4	Middleware Client Component	121
6.5.5	History Component	121
6.5.6	Database Component	126
6.5.7	Email component	127
6.6	Human Machine Subsystem	127
6.6.1	API Component	128
6.6.2	Middleware-Web Client Component	129
6.7	Deployment	132
6.8	Conclusions	133
7	Tests	134
7.1.1	Unit	134
7.1.2	Integration	137
7.1.3	Validation	139
7.1.4	System	146
7.2	Conclusions	151
8	Conclusions	152
8.1	Accomplished objectives	152
8.2	Future work	155
9	Bibliography	157
10	Appendixes	163
10.1	Appendix-A- Acceptance Tests	
11.1	Appendix-B- Machine Component (Preliminary Solution)	
11.2	Appendix-C- UML class diagram for History Component	

- 11.3 Appendix-D- UML class diagram for RabbitMQ-Core**
- 11.4 Appendix-E- UML Class Diagram for OPC-UA Server**
- 11.5 Appendix-F- UML class diagram for full Middleware Client (Producer/Consumer)**
- 11.6 Appendix-G- UML Class Diagram for Manager Component**
- 11.7 Appendix-H- UML class diagram for API Component**
- 11.8 Appendix-I- Deployment setup and information**
- 11.9 Appendix-J- Interoperable and Interconnected CPS-populated Systems for Proactive Maintenance**
- 11.10 Appendix-L- Application of Sensors for Proactive Maintenance in the Real World**
- 11.11 Appendix-N- FlexHousing: Flexoffer concept for the energy manager**
- 11.12 Appendix-O- Maintenance Supported by Cyber-Physical Systems and Cloud Technology**

Figures Index

Figure 1 – Frontal view of the machine	15
Figure 2 – Bending process	16
Figure 3 – CNC operator interface.....	17
Figure 4 – The MANTIS projects concept	18
Figure 7 - Innovation units of Industry 4.0	30
Figure 8 - Reference Architecture Model - Industry 4.0	35
Figure 9 - An Arrowhead local cloud comprising an orchestrated service instance	39
Figure 10 - Layers of a Cyber-Physical System.	41
Figure 11 - Canvas business model elaborated for the current project.....	51
Figure 12 - Canvas business model elaborated for the current project.....	66
Figure 13 - Analytic Hierarchy Process (AHP) elaborated for the current project	68
Figure 14 - MANTIS (ADIRA Pilot): Context Diagram for the proposed solution	72
Figure 15 – Subsystem and component interaction for the proposed architecture	74
Figure 17 - Domain Model for the proposed solution	80
Figure 18 - Machine Event Model	83
Figure 20 – Entity-Relationship model of the proposed solution.....	86
Figure 23 - User Interface of <i>OPC-UA server</i>	91
Figure 25 - <i>OPC-UA server</i> Component - UA_SERVER Start	93
Figure 27 - Context Diagram for the Sensors(Arduino 101) proposed solution.....	95
Figure 29 - <i>Sensor</i> Component - Partial Node Code	97
Figure 31 - Mantis-PC Subsystem	99
Figure 34 - HMI Subsystem - Sensors Soft Real-time visualization	102
Figure 35 - Edge Local Subsystem	103
Figure 38 - Node-Red component – User Interface.....	106
Figure 39 - Data Analysis Subsystem.....	108
Figure 42 - HMI Subsystem – Data Analysis visualization.....	111
Figure 44 – Context Diagram of Middleware Component	113
Figure 45 - Middleware Component Queues Structure	114
Figure 49 - Manager Component – RabbitMQ User Interface - Based queues.....	120
Figure 50 - UML Sequence Diagram - Initialize Cloud System	123
Figure 51 - UML Sequence Diagram - Create Machine.....	124
Figure 52 - History Component – HistoryService Partial Code	125
Figure 53 - History Component – HMI history visualization.....	126
Figure 54 - Email component – User email notified example	127
Figure 57 - HMI Subsystem – Machine data Soft Real-time visualization	132
Figure 59 - Testing strategy (Source: Pressman, 2011).....	134
Figure 60 – Middleware Unit Tests Partial Code	137
Figure 61 - Middleware Integration Tests Partial Code.....	138
Figure 62 - Z-Test - Two-tail interval.....	140
Figure 63 – Histogram of Edge Local Subsystem - Data packets delay frequency.....	145
Figure 64 - W3AF OWASP TOP 10 - Audit.....	148
Figure 65 - Apache Jmeter - Graph Report 150 request per second.....	150
Figure 66 - Apache Jmeter - Tests Graph Report 300 request per second	150
Figure 67 - Graphical User Interface – Main Page.....	168
Figure 68 - UML Sequence Diagram - Machine Preliminary Solution.....	169

Tables Index

Table 1: Industry 4.0 components	40
Table 2: Design principles of each Industry 4.0	45
Table 4 - Representation of the benefits and sacrifices of the proposed value.....	63
Table 5 - AHP – Criteria Pairwise comparison	69
Table 6 - Criteria solution	69
Table 7 - Criteria with alternatives pairwise comparison	70
Table 8 - AHP -Final Solution	71
Table 9 - Types of Databases.....	85
Table 10 - REST Resources of the Manager Component.....	117
Table 11 - Z Test: Two samples for Mean.....	141
Table 12 - Descriptive analysis of the Database storage	141
Table 13 - Descriptive analysis of the Middleware delay in seconds (s).....	143
Table 14 - Z Test: One samples for Mean	144
Table 15 - Manager Component performance and stress tests	149
Table 16 – Start Machine Internal Sensors Data - Acceptance Tests	163
Table 17 – Start Machine External Sensors Data - Acceptance Tests	163
Table 18 – HMI Local - Acceptance Tests	164
Table 19 – HMI Web - Acceptance Tests.....	165
Table 20 - OPC-UA server Component Setup and Information.....	176
Table 21 - CNC Bender Component Setup and Information.....	176
Table 22 - Sensor Component Setup and Information.....	176
Table 23 - Mantis-PC Subsystem Setup and Information	177
Table 24 - BLE server Component Setup and Information	177
Table 25 - Edge Local Subsystem Setup and Information	177
Table 26 - Node-Red component Setup and Information.....	177
Table 27 - Middleware Client component Setup and Information	177
Table 28 - Edge Server Subsystem Setup and Information	178
Table 29 - Manager and History and components setup and Information.....	178
Table 30 - API Component Setup and Information	178
Table 31 - API Component Setup and Information	178

Acronyms and Symbols

AHP	<i>Analytic Hierarchy Process</i>
AMQP	<i>Advanced Message Queuing Protocol</i>
API	<i>Application Programming Interface</i>
API-Key	<i>API-Key a code passed in by computer programs calling an application programming interface (API) to identify the calling program, its developer, or its user to the Web site.</i>
Azure	<i>Microsoft Azure: Cloud Computing Platform & Services</i>
BLE	<i>Bluetooth Low Energy</i>
CAGR	<i>Compound Annual Growth Rate</i>
CI	<i>Consistency Index</i>
CNC	<i>Computer Numerical Control</i>
CPS	<i>Cyber-Physical Systems</i>
CR	<i>CR Consistency Ratio</i>
DDS	<i>Data Distribution Service</i>
DTO	<i>Data Transfer Object</i>
ESB	<i>Enterprise Service Bus</i>
FTP	<i>File Transfer Protocol</i>
GATT	<i>General Attribute Profile</i>
HighCharts	<i>Interactive JavaScript charts for your webpage</i>
HMI	<i>Human Machine Interface</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
IBM	<i>International Business Machines</i>
ICT	<i>Information and communication technology</i>
IIoT	<i>Industrial Internet of Things</i>
IIRA	<i>Industrial internet reference architecture</i>
IIS	<i>Internet Information Services</i>
IMDSS	<i>Incident Management Decision Support System</i>
IMU	<i>Inertial measurement unit</i>
IoS	<i>Internet of Services</i>
IoT	<i>Internet of Things</i>
IoT-A	<i>Architectural Reference Model</i>
JMS	<i>Java Message Service</i>
JSON	<i>JavaScript Object Notation</i>
M2M	<i>Machine-to-Machine</i>
Mimosa	<i>An Operations and Maintenance Information Open System Alliance</i>
MOM	<i>Message Oriented Middleware</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
MVC	<i>Model–view–controller</i>
NCD	<i>New Concept Development</i>
Npm	<i>NodeJS package manager</i>
OPC-UA	<i>Open Platform Communications Unified Architecture is a machine to machine communication protocol</i>

PLC	<i>Programmable logic controller</i>
PMM	<i>Proactive Monitoring and Maintenance</i>
QoS	<i>Quality of Service</i>
RabbitMQ	<i>Rabbit Message Queueing</i>
RAMI	<i>Reference Architecture Model Industry 4.0</i>
REST	<i>Representational State Transfer</i>
RFID	<i>Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves.</i>
RI	<i>Random Consistency Index</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
SOA	<i>Service-Oriented Architecture</i>
SQL	<i>SQL is a standard language for storing, manipulating and retrieving data in databases.</i>
SSL	<i>Secure Sockets Layer</i>
STOMP	<i>Streaming Text Oriented Messaging Protocol</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol.</i>
TDD	<i>Test-driven development</i>
TTL	<i>Time to live or hop limit is a mechanism that limits the lifespan or lifetime of data in a computer or network.</i>
UML	<i>Unified Modelling Language</i>
UUID	<i>Universally unique identifiers</i>
VPN	<i>Virtual private network</i>
XMPP	<i>Extensible Message and Presence Protocol</i>
β	<i>Bandwidth</i>

1 Introduction

This chapter introduces the project context, the proposed objectives, the solution overview within the project, the concern of the stakeholders, the expected results of the development process and the methodology of work as well as the general structure of this document.

1.1 Context

Industry 4.0 is one of the fastest growing research topics for both academics and practitioners (Hermann, Pentek & Otto, 2015). It was first mentioned in 2011, consisting in a strategic plan developed by the German federal government to transform the manufacturing industry (Koch *et al.*, cit. in Åkeson, 2016; Nathan, 2015). Nonetheless, there is not a consensual definition of Industry 4.0. According to (Hermann *et al.*, 2015), this is a major problem, since the subject has received more and more deviant definitions each day. Nonetheless, some organizations achieved the role of key promoters, thus strengthening the correctness of their definition. “Plattform Industrie 4.0” (2014, cit. in Åkeson, 2016, p. 1) defines Industry 4.0 with the following words: “Industry 4.0 is best understood as a new level of organizational control over the entire value chain of the life cycle of products, it is geared towards increasingly individualized customer requirements. The basis of the fourth industrial revolution is the availability of all relevant information in real-time by connecting all instances involved in the value chain.”

Given the fact that the definition is quite blurred, (Hermann *et al.*, 2015) established the key aspects and principles for the Industry 4.0 concept. Regarding the key aspects, the authors found that there are four aspects to consider: Cyber-Physical Systems (CPS), Internet of Things (IoT), Smart factories and Internet of Services (IoS). On the other hand, and according to (Koch *et al.*, 2015), the key perspectives and motivations for Industry 4.0 are:

- 1) The industrial internet will change the entire business and it is very important to have dedicated management;
- 2) Integrated analysis and use of data are the main capabilities of Industry 4.0;
- 3) Digitalization of products and services (Smart products);
- 4) New disruptive business models;

5) Horizontal cooperation throughout the entire value chain.

All key aspects of Industry 4.0 have been strengthened by Mittermair (2015) and by Nathan (2015), since both authors showed that its foundation is based on data and it is made possible through CPS. Therefore, it is important to establish the scope of such term, which will be now addressed.

The evolution of Cyber-physical Systems has modified several areas, from deeply embedded systems to smart cities. Industrial systems can also benefit from the advances that have been developed in the field of CPS in multiple areas. One area where these kind of systems are important is on the maintenance of industrial machines. The access to large volumes of data can be instrumental in order to forecast machines' malfunctions, to schedule maintenance operations (Mobley, 2002; Faisal & Mahmoud, 2003) and to determine the root cause of failures. Nonetheless, the collection, transport and management of those large volumes of data poses several challenges, which span from technical to organizational, the latter being frequently related with the integration of existing industrial processes. This thesis deals with the problem of providing an integrated solution, following the Industry 4.0 guidelines, to tackle the problem of maintenance of industrial machines.

1.2 The Problem

The potential developed by Industry 4.0 will allow the national industry to be able to develop new procedures, specifically at the level of the competitive differentiation, by introducing the ability to satisfy specific requirements of customers in terms of design, configuration, planning, manufacture and maintenance. In turn, this flexibility allows a dynamic configuration of different aspects of the value chain, combining, simultaneously, an optimization of the decision model to cope with the real needs of the market (Correia, 2014).

Both the reliability and the safety of industrial machines depend on their timely maintenance. This thesis focus on providing existing industrial machines with capabilities for extended maintenance, by acquiring detailed information about the machine operation and storing that information locally or on the cloud. The collected information can be used to detect and predict machine failures. Based on this knowledge it will be possible to correctly schedule maintenance operations, detect failures when they occur and predict its occurrence supported by detecting deviations on the machine behavior.

The integration of Cyber-physical Systems within the maintenance process, actually enables the continuous monitoring of machines, as well as the application of advanced

techniques for the predictive¹ and proactive² maintenance of machines. All building blocks for such revolution (e.g.: embedded sensors, efficient preprocessing capabilities, ubiquitous connection to the internet, cloud-based analysis of the data using prediction and proactive algorithms) are in place, despite having to cope with the hurdle of their application in real scenarios, which require the integration with new and existing machines and with the existing maintenance process. It is very important to note that industrial machines usually have a lifecycle of more than 20 years, consequently it is of paramount importance for such a system to be sufficiently flexible to be able to be used in machines with very advanced controllers and old machine with minimal capabilities.

This thesis is also structured upon the Cyber-Physical System based Proactive Collaborative Maintenance (MANTIS) European project which consists in an international initiative that aims to build a platform for proactive maintenance of industrial machines (Jantunen *et al.*, 2016). In this European project, where the CISTER Research Unit (the student's hosting institution), plays a key role on the provision of the communication infrastructure and data storage for the MANTIS project. The main goal of the MANTIS project is the reduction of the maintenance costs by adopting novel monitoring techniques.

However, the maintenance requires the detection of some faults, and as early as possible, in order to avoid failures, to predict failures, to facilitate the scheduling of the parts' replacement and to provide tools that actually ease the diagnosis of such problems. This particular approach reduces machine down-time, eliminates excess spare-parts' stocks, improves the product's quality, increases operator safety and lowers the overall cost of maintenance, as discussed by some authors, e.g. by (Holmberg *et al.*, 2010). In MANTIS, several machine learning algorithms are used, aiming to generate the prediction and detection models. Such algorithms are parametrized according to the data that is obtained from the sensors, which may be: i) sensors that are already in use for the control of a manufacturing process; ii) sensors that are specifically designed and deployed to maintenance purposes, or even iii) sensors that are temporarily installed on the machine.

¹ Predictive maintenance techniques are designed to help determine the condition of in-service equipment in order to predict when maintenance should be performed. This approach promises cost savings over routine or time-based preventive maintenance, because tasks are performed only when warranted.

² Proactive maintenance is the maintenance philosophy that supplants "failure reactive" with "failure proactive" by activities that avoid the underlying conditions that lead to machine faults and degradation.

This thesis consists of two main results:

- 1) A generic architecture for proactive maintenance;
- 2) An industrial pilot, which validates and demonstrates the usefulness of the architecture.

The implementation will be based on Press Braking machine produced by ADIRA (Figure 1). Press Braking is the process of deforming a metal sheet (workpiece) along a given axis by pressing it between clamps (tools) (Figure 2). In order to have a finished part, a metal sheet will be consecutively bent at several places – e.g. to make a computer box.

Brake forming can bend sheet-metal from 0.6 to 50 mm thick and lengths from 150 mm to 8 m long. The angle and type of the bend are determined by the shape of the punch and die and the depth with which a punch penetrates a die. The dies may have "U", "V" or channel shapes.

A movable ram is attached to the beam and is covered by a shroud. The punch is attached to the bottom of the ram and the die to the top of bed (covered by the lower shroud). When the ram descends on the table, a bending force is exerted on the sheet-metal between the punch and the die. The bending force and bending speed have to be carefully controlled in order for the material being used to maintain their physical characteristics and insure the required bending precision. Additionally, the machine structure deforms due to the forces involved and those deformations have to be compensated in order to guarantee the machine precision. For more details on the machine operation see (Ferreira *et al.*, 2016).



Figure 1 - Frontal view of the machine

The machine maintenance procedures require detecting faults as early as possible to avoid catastrophic failure, predict failures in order to facilitate the scheduling of the parts replacement and providing tools that ease the diagnosis of the problems. This approach reduces machine down-time, eliminates excess spare-parts stock, improves product quality, increases operator safety and lowers the overall cost of maintenance.

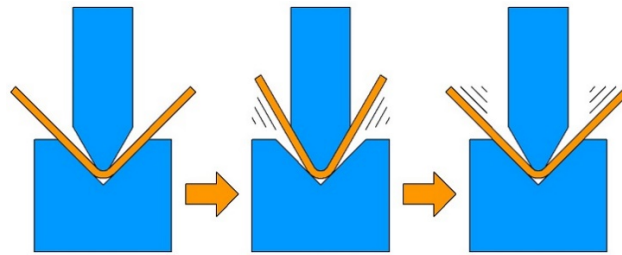


Figure 2 – Bending process (source: <http://sheetmetal.me/tooling-terminology/bottom-bending/>)

The machine (Figure 1), contains several different sensors and actuators. As an example of sensors we can highlight the ram vertical positions sensors (Y1 and Y2) which allow to determine position of the ram on both extremes of the machine. Machine failure occurs when the difference between Y1 and Y2 is too high. More than 50 other devices are available on the machine.

The machine is controlled by a PLC which is responsible for its execution of machine cycles and a Numerical Controller, which is responsible for the interface between its operator and the PLC. This interface is supported by a Windows XP based machine, which can be connected to a network.

Data from the internal machine sensors and actuators is available on the CNC, through RS485 connection, with the PLC making these values available to other application through shared memory.

The machine operator interfaces with the machine using the *CNCBender* application, from which we show a screenshot in Figure 3, which allows to configure the machine according to the kind of material to be bend and the tools being used.



Figure 3 – CNC operator interface

For maintenance purposes, other sensors can also be added to the machine, which are not usually available on these machines, due to their low cost/benefit ratio. An oil sensor is one of those cases (costs between 300€ to 3000€) that can be an interesting addition to this machine, enabling to determine the quality of the oil (e.g. by detecting water and particles). Accelerometers, can also be used to monitor the operation of the ram and other mobile components of the machine, detecting vibrations not compliant with the normal machine operation.

The solution to the maintenance problem will be based on the MANTIS project approach.

The overall concept of MANTIS is to provide a proactive maintenance service platform architecture based on Cyber Physical Systems and on Industry 4.0 building blocks that allows to estimate future performance, to predict and prevent imminent failures and to schedule proactive maintenance in different types of equipment. In MANTIS Physical

systems (e.g. industrial machines, industrial processes, vehicles, renewable energy assets) and the environment they operate in, are monitored continuously by a broad and diverse range of sensors, resulting in massive amounts of data that characterize the usage history,

operational condition, location, movement and other physical properties of those systems. These systems form part of a larger network of heterogeneous and collaborative systems connected via robust communication mechanisms – this part will most of the focus of this thesis.

A partial objective of the MANTIS project is to support distributed processing chains that efficiently transform raw data into knowledge while minimizing the need for communication bandwidth. Eventually some of these sophisticated functions are performed at different levels in a collaborative way, ranging from local nodes to locally optimize performance, bandwidth and maintenance; to cloud-based platforms that integrate information from diverse systems and execute distributed processing and analytics algorithms for global decision making. Finally, users of these systems will interact with it through advanced **HMI** systems. Figure 4, shows the different components.

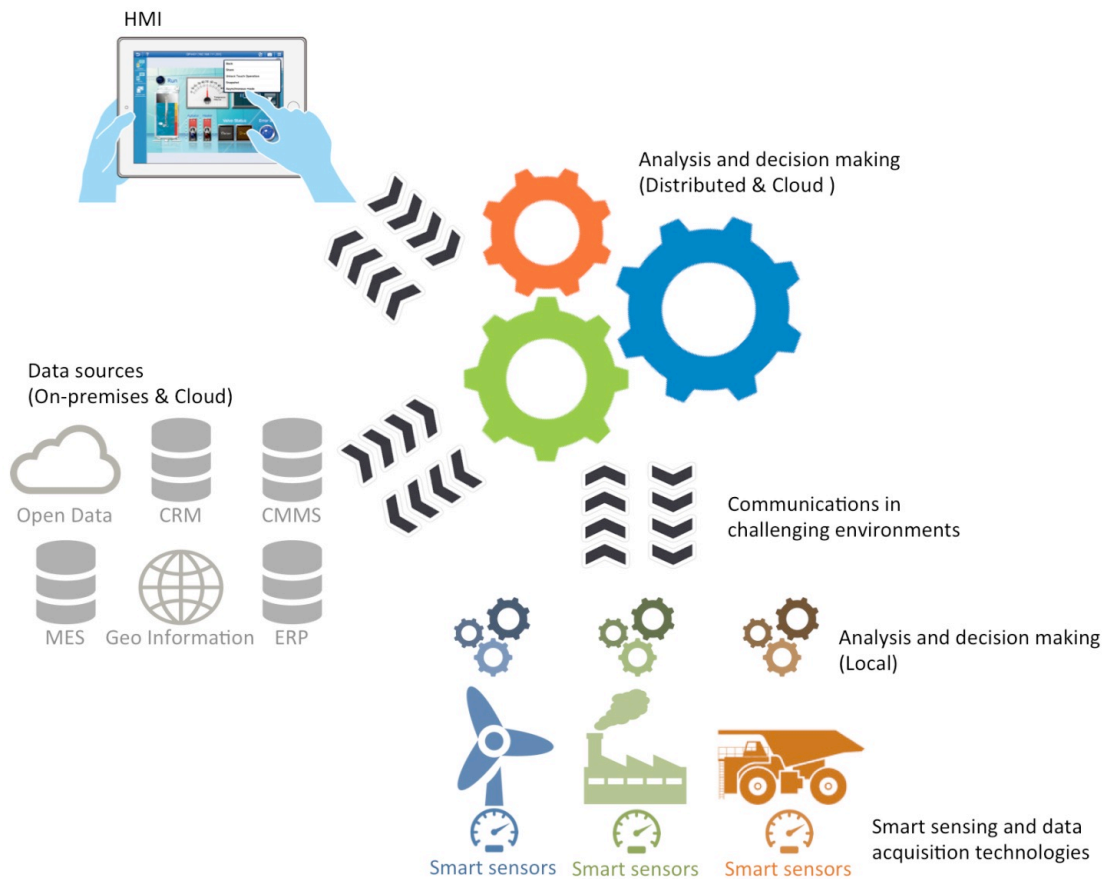


Figure 4 – The MANTIS projects concept (Source: <http://www.mantis-project.eu/>)

The MANTIS project is multimillion European project with 42 partners, an overall budget of 34 Million Euros and a total of 3900 Persons Month. The main contributions of this thesis to the project are not only limited to the design, implementation of the communication infrastructure of the Portuguese pilot and to the integration of new sensors on the machine, but

were also a decisive input to the overall MANTIS architecture, defined in deliverable D3.6 (Ferreira *et al.*, 2017), which integrated most of features used in this pilot, since it was the most advanced solution when the deliverable was written.

1.3 Objectives

The grand vision of the Industrial IoT (IIoT) is nowadays attainable, due to the advances of the last decade in the areas of the Internet of Things, Big Data, embedded systems and communications protocols. IIoT systems are actually composed by sensors and actuators that collect data and that act upon Cyber-physical System. One of the biggest challenges of dealing with this complex structure of connected components, which can actually be seen as a large distributed system, is related to the way that all these components must be interconnected concerning the exchange of information.

A Middleware that acts as an abstraction layer for a transparent data management can facilitate such integration, more precisely by making data available to advanced processing algorithms, by coping with the technical constraint that are related to data management in existing industrial processes, and finally by liberating the programmer from lower-level details.

The resulting framework for a proactive maintenance of industrial machinery can be used in several industrial monitoring contexts, some of which are analyzed in the present work.

Many of these Industry 4.0 implementations rely on industrial protocols, which are very reliable protocols, but lack in terms of flexibility and interoperability. IoT applications are also mostly supported by service oriented protocols. In this thesis, we want to explore solutions built over protocols that are standard message-based protocols. Thus, we define objective 1 as:

- **O1:** Study and understand how existing standard message-based protocols for middleware can cope with the IIoT requirements, with a special emphasis on industry 4.0 design principles (Interoperability, Virtualization, Decentralization, Real-Time Capability, Service Orientation and Modularity);

One of the main initial problems was to choose an adequate message oriented middleware that supports the communication infrastructure in an efficient manner. The market already offers many commercial and open source solution, which have to be compared in multiple dimensions and among which a choice has to be made. To support such decision, we can formulate the second objective of this thesis as:

O2: Provide an analysis, supported on the Analytic Hierarchy Process of existing middlewares;

This framework will only be successful in production if supported by an adequate business model, which points out how the solution being proposed can reach the market, that constitute the third objective of this thesis

- **O3:** Analyze the Business Modelling and propose a solution;

Obviously, the analysis, design and implementation of the overall framework are the main objective of this thesis and of the work performed on the MANTIS project. This objective can be further divided into several sub-objectives. We can then formulate the following objective and its sub-objectives as follows:

- **O4:** Analyze, design and implement all component and *APIs* for the framework subsystems;
 - **O4.1:** collecting machine data from internal sensors and actuators and handle interconnection;
 - **O4.2:** external maintenance-specific sensors on the machine and its interconnection;
 - **O4.3:** a component that has the responsibility of collecting data from machines inside the factory, pre-processing it and sending to the cloud through a middleware;
 - **O4.4:** a communication middleware, on the cloud, which has the responsibility of enabling communication and management of data between distributed components, more specifically the **Data Analysis** and **HMI** components;
 - **O4.5:** an interface to connect the **HMI** with the middleware and other components;
 - **O4.6:** a component to store and retrieve historical data related with the sensors and actuators;
 - **O4.7:** interfaces to be used by other MANTIS partners to access the middleware.

To enable the integration of all modules and components needed for the overall MANTIS project there is also the need to implement some functionalities which allow the **HMI** to integrate with the middleware, to maintain and administer the system and to support the display of graphical data related with the accelerometers (which required a high-performance **HMI**).

- **O5:** Support and implement some of the framework's external modules:
 - **O5.1:** set up real time graphical libraries for **HMI**;
 - **O5.2:** create library that provides the **HMI** with direct access to the middleware;

- **O5.3:** configure the main server with FTP, IIS, middleware and services security (SSL/HTTPS), etc.

From the objectives stated above it is also clear that such a complex distributed system demands thoroughly tests to all of its features. It is also required to be aware of the performance of the system under different several loads.

- **O6:** Provide verification and validation tests of the framework;

The Framework must also be validated in a real scenario, showing its applicability to different scenarios. In this case it will be applied to the machine displayed in Figure 1, the GreenBender ADIRA Steel Sheet bending machine.

- **O7:** Deploy the framework in a real-world scenario at ADIRA company.

For better understanding the flow of this thesis the following section gives a high-level overview of the solution, which is later described in detail on the following chapters.

1.4 Solution Overview

Based on the objectives defined before, this section provides the reader with an overview of the adopted solution. This solution is then detailed on later chapters.

Figure 5 depicts a scenario of the ADIRA Pilot. This scenario is composed of a number of components that can be grouped into 6 subsystems: **Machine**, **MANTI-PC**, **Edge Local**, **Edge Server**, **Data Analysis** and Human Machine Interface (**HMI**) connected together by a Communication Middleware, which can be split into a Local part, which usually belongs to the internal network of a factory and Cloud parts (**Edge Server** and Data Analysis), which collects analysis information from multiples machines, located on multiples factories, from multiple clients.

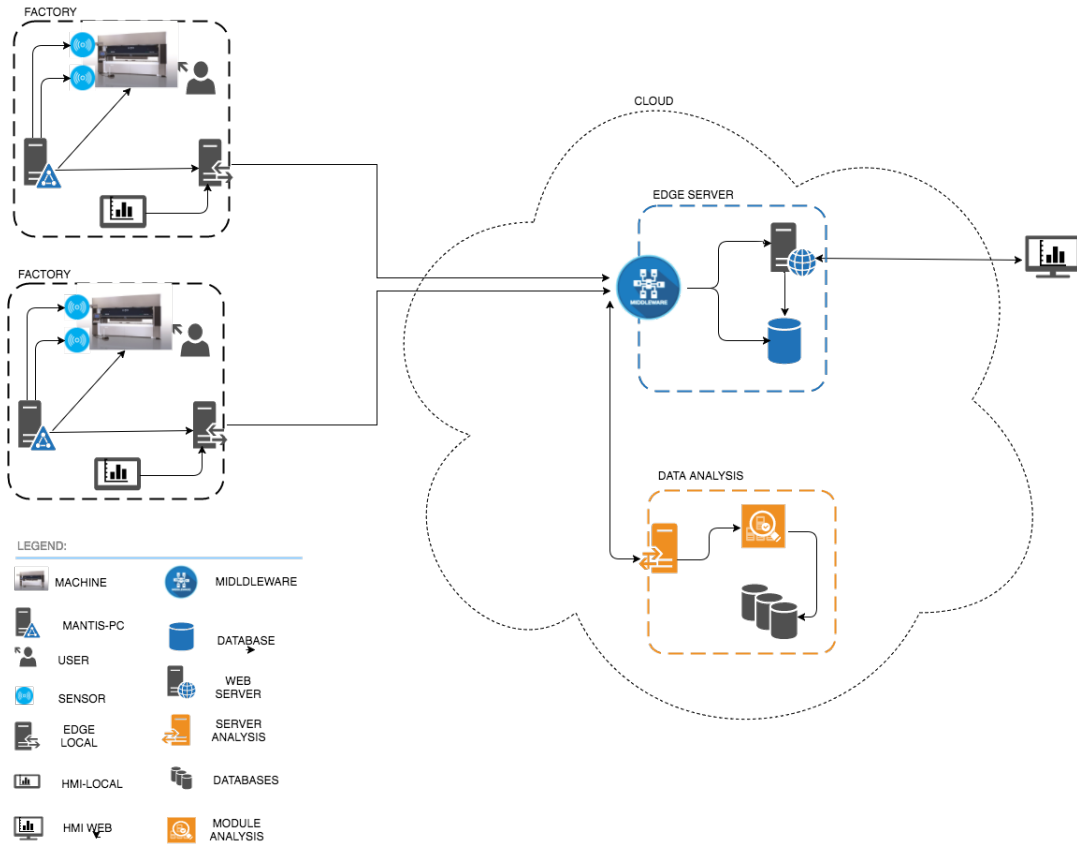


Figure 5 – MANTIS (ADIRA Pilot): Context Diagram for the proposed solution

1.4.1 Machine

Data on the machine are collected by means of sensors that are part of the machine's control systems, or from sensors that were added specifically for maintenance purposes. This subsystem can consist of several modules, each part of a machine's subsystem. The number and kind of modules depends on the machine being monitored. Usually, these modules provide access to four different data sources: the machine CNC, which controls the machine; the PLC-connected sensors and actuators; the Safety PLC; Maintenance Sensors.

The Machine subsystem is comprised of both pre-existing modules, and ones added specifically for the maintenance platform itself. The PLC sensors are already part of the machine, and are used internally to control its operation. These range from buttons and pedals to advanced electric motor drives, with positioning information. Although used primarily for control functions, these sensors can also be used to determine anomalous events or states, to diagnose problems and even to infer the root cause of problems. The CNC, normally, is also able to perform some diagnosis functions that allow the identification of some failures and the generation of warnings on the condition of the machine.

The PLC works in close cooperation with the CNC, by controlling all automation functionalities and, at the same time, it is able to send information from its sensors to the CNC.

The Safety PLC handles only safety-related functions for the machine, such as preventing humans from being too close while the machine is working, detecting critical conditions, etc. Data from sensors controlled by the Safety PLC are mainly used to distinguish between component failures and safety-related events.

Finally, the Maintenance Sensors are sensors placed on the machine, usually communicating over an independent channel (e.g. a wireless network) that only acquire specific maintenance-related information, such as oil quality and data on the machine's moving parts. These data are then aggregated by the MANTIS-PC, which establishes the interface between protocols (e.g. BLE, OPC-UA, MQTT, etc.), provide data conversion standardization, and make the data available regarding a component to be collected by **Edge Local**.

1.4.2 Edge Local

The main objective of this subsystem is to isolate the factory from the outside world, at the same time providing some functionalities at local level. From the security point of view, the **Edge Local** can be seen as creating a DeMilitarized Zone (DMZ) in the sense that it is the only module in the factory premises that has network access, and thus concentrates all the security requirements on itself.

Data collected from multiple Machines, usually inside a factory, has to be made available to the other subsystems of the platform. The **Edge Local** subsystem provides mechanisms to support communication and management of the data acquired across multiple heterogeneous and distributed data sources. This is accomplished by providing an abstraction layer that detaches the application development from the intricacies of the lower level details. It acts as a virtualization platform and as data broker that connects the Machine subsystem to the Cloud Middleware, capable of extracting, collecting, distributing/sharing, pre-processing, compressing, and semantically enhancing the data produced in an efficient manner. Therefore, the one of the fundamental goals of the **Edge Local** subsystem is – from one side is to support the data integration of multiple data sources and – from the other side is the provisioning of data to the cloud where more complex and resource consuming data processing takes place.

The **Edge Local** subsystem is composed by several components - the Virtual Device, the local **HMI** service, the Data Broker, etc. The Virtual Device is responsible for virtualizing physical entities (machines and industrial assets) available in the shop floor. These machines

and assets are virtualized in terms of their capabilities to facilitate and enhance the process of exchanging data (machine data readouts). The local **HMI** service is responsible for visualizing all the necessary information generated within the **Edge Local** subsystem, i.e. Virtual Devices available, machine data readouts. Finally, the Data Broker operates as a gateway allowing the indirect connection between the machines in a factory and the Cloud Middleware, which performs **Data Analysis** and supports advanced **HMI** features.

1.4.3 Edge Server

This subsystem manages the data, by storing and transporting them between the **Edge Local** (eventually from multiple factories), and both the **Data Analysis** and **HMI** components. The **Edge Server** operates through four components, the middleware, the history, the manager and the database.

The **Edge Server** manages a communication middleware, which receives data from several **Edge Local** devices through publish/subscribe interaction, and saves the data to a *Database Component*, which is structured according to the MIMOSA/IoT-A standards. Additionally, the publish/subscribe protocol supports the communication with the **Data Analysis** and **HMI** subsystems, and between those two subsystems.

The **Edge Server** also makes available a set of services which are used by the **HMI** to configure the systems. This configuration information is then permanently stored on the database and used to support system startup and resume system operation in case of a crash. As explained in more detail later in Analysis chapter. This solution provides an adequate level of security and enables the isolation of data between all factories, at the same time providing a highly scalable solution.

1.4.4 Data Analysis

The **Data Analysis** subsystem includes three modules. The first is a set of Prediction Models module, used for the detection, prognosis and diagnosis of machine failures. The models can be built for one machine family, or can be generic and adapted to different machine families. The second module is a Prediction Application Programming Interface (API) that outputs predictions from the models, and provides data to feed and train the models. The third module is an Intelligent Maintenance Decision Support System (IMDSS), which is used to manage the models (model generation, selection, training and testing), for example on reception of training data, or when the API is contacted. The IMDSS is composed of a Knowledge Base

that uses diagnosis and prediction models and the data sent by sensors. On top of this Knowledge Base there will be a Rule based Reasoning Engine which includes all the rules that are necessary to deduce new knowledge that helps the maintenance crew to diagnose failures.

In addition to the data and algorithms, expert knowledge has been encoded as a set of rules that are used to detect and flag possible failures. Each rule indicates what sensor and CNC signals need to be acquired, how they are segmented, the type of analysis to be executed and what failure is associated with these signals.

As an example, let us consider when the brake press is working in automatic mode, terminates its bend cycle and has parked the ram on the top position waiting for the next task. If no failure exists then the ram must remain still in the same position where it stopped. Because the hydraulic system is constantly losing pressure, the CNC compensates for any deviation. Normally, such deviations are minor (imperceptible to the naked eye) and occur at very low rates. However, if a hydraulic pump fails or a hydraulics tube ruptures, leaks will cause large deviations as the CNC compensates for the condition.

In order to detect such problem, the positions of the pistons are recorded when the control signal indicates that the ram is at top dead center (segmentation). Statistical tests are used to check that the deviation is within a specific tolerance threshold. This threshold is determined via the machine learning algorithm (stream based) and is tweaked in order to reduce the false positive and false negative rates.

1.4.5 Human Machine Interface

This subsystem provides a Human interface for the proactive maintenance system. It has two main components, one for data visualization and another for data management.

In the visualization component it is possible to view historical and live data, which are collected from specific machine sensors (e.g. machine status, speed, positioning and pedals state). It is also possible to show the results generated by the **Data Analysis** module, more specifically the alarms for unusual sensor data and the warnings regarding impending failures. It is possible to match the warnings from the **Data Analysis** subsystem with historical data collected from the sensors.

The *Manager API* Component includes all the administrative operations, like users and roles management, as well as factories and machines setup. Role management is a very important process that allows one to dynamically assign specific permissions to each type of

user (e.g. a user with the “operator” role can view historical and live data only). Factories and machines management, allow an authorized user to setup a new factory and its machines.

The **HMI** follows a web-oriented design and therefore can be accessed from anywhere, at any time and through all sort of electronic devices with the only requirement being the use of the Internet to do so. This allows both remote (administrative) and on-site operations such as analyzing the machine’s state or view its past performance.

1.5 Development Process and Work Methodology

The adopted methodology in the project’s management goes through a follow-up with the customer where the ADIRA Pilot is being implemented. The objective is to adjust the reference architectures and the business models, as well as the requirements defined in the MANTIS European project, to the needs of the stakeholder in a real context of its own applicability. Hence, the development process evolves through the following macro phases:

- 1) Detailed analysis of business needs;
- 2) Preparation of architecture specification;
- 3) Implementation/development;
- 4) Tests and validation;
- 5) Support;
- 6) Stabilization;
- 7) Corrective and evolutionary maintenance (European’ project time);
- 8) Final analysis of the project and corrective measures to apply in future projects.

All phases are accompanied by specific documentation, which helps to formalize the scope of the project and to validate and measure the success of the project. The used methodology by our team (Figure 6) is very similar to the Rational Unified Process (RUP) (Traa, n.d.). At the moment, it was only possible to execute until macro phase 4.

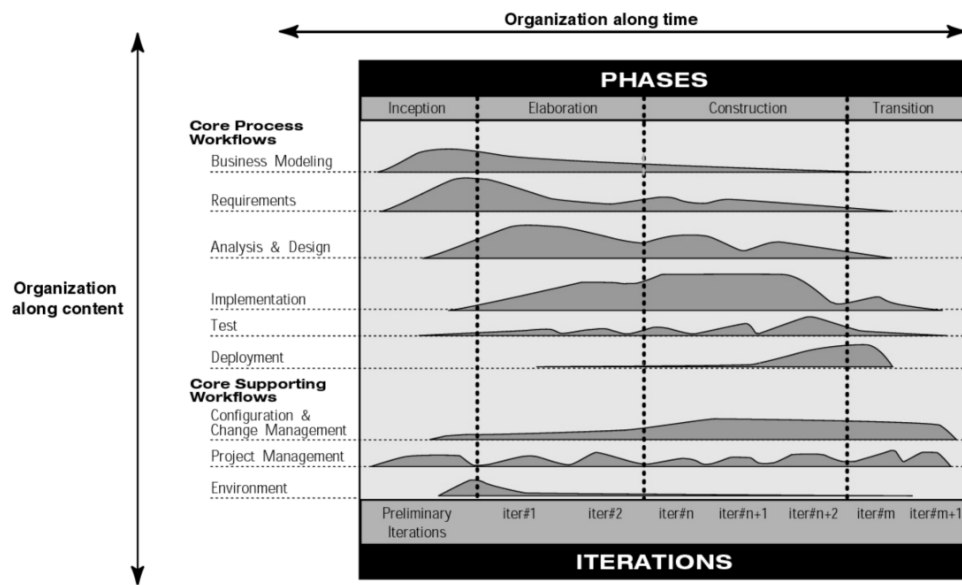


Figure 6: The Iterative Model graph shows how the process is structured along two dimensions (Source: Xiong, 2008).

This project was supported by a team consisting of a Project Manager (person that is responsible for the project's management), a Project Owner (an organization responsible for the delivery and fulfillment of the project according to the plan and architecture), a Quality Control (a person who assures the quality of the project ensuring the correct delivery of all functionalities) and by the Developers (people responsible for the implementation and executing the Test Plan of the project) (Xiong, 2008).

In this particular project, the Project Manager role was allocated to Luis Lino Ferreira; the Project Owner was ADIRA company; the Quality Control was divided between Luis Lino Ferreira and the author of this document; the other functions (Developer and Tests) were allocated on the author of this document.

The project development process was performed iteratively, with periodic deliveries of prototypes of the solution to the Project Manager, and also the Project Owner, ADIRA. The main purpose is to provide a real context of tests, aiming to control and monitor the state of the project and the existence of possible changes. This control will be essential to a correct development of the solution.

1.6 Document Structure

The present work has a very specific structure, being divided into several chapters.

- 1) **Introduction:** current chapter, which aims to give the reader the basic information needed, in order to facilitate the framing of the topic of the thesis. Therefore, this chapter begins with a brief explanation of the objectives that supported the choice of the present theme. In addition, this chapter refers to the adopted methodological and technological approach, presents the expected/achieved results, describes the development process, the work methodology and also the structure of the project.
- 2) **Industry 4.0:** this chapter frames Industry 4.0 in articulation with the problem's context, referring to its relevance and value at the industry level; introduces the most relevant definitions as well as existing SOA-based solutions in which the author of this document have been involved.
- 3) **Value analysis:** this chapter aims to prove the importance and necessity of a value proposition for solving the problem at hand. A general analysis of the potential and impact of the project is carried out, followed by the developed component, focusing on its subsequent commercialization. Hence, the analyze the value proposition and the value in the current market is provided, as well as defined the Canvas business model for it will be made.
- 4) **Evaluation of Message Oriented Middleware solutions:** this chapter presents the solutions based on Message Oriented Middleware (MOM) evaluation, since the intended solution that respond to industry 4.0 design principles, with special focus on decentralization, modularization, real-time capability and standardization.
Analysis: this chapter presents, in a detailed manner, all phases carried out during the process of developing a software solution, starting with the description of the requirements that the system must fulfil, the report of the concepts of the problem domain, and ending with its analysis and design.
- 5) **Tests:** this chapter presents the tests performed and evaluation of the solution developed as well as the metrics and methodologies that will be involved.
- 6) **Design & Implementation:** this chapter presents the design of the solution for the problem previously identified, as well as different alternatives as well as different alternatives, that will be compared. Furthermore, the implementation of the use cases that were previously identified in the Chapter 5.

- 7) **Bibliography:** this chapter presents all the references that supported the present work, which were necessary to fully understand the problem in hand.
- 8) **Appendixes:** this complementary chapter includes several documents that are considered to be essential to the present work.

2 Industry 4.0

This chapter presents Industry 4.0 in articulation with the problem's context, referring to its relevance and value at the industry level. Furthermore, introduces the most relevant definitions as well as existing SOA-based solutions in which the author of this document have been involved.

2.1 Introduction

Förderschwerpunkte Industry 4.0 (2017) argues that the German Federal Government follows a particular initiative, the called “High-Tech Strategy 2020 for Germany”, which actually aligns with the global challenges in five areas: climate/energy, health/nutrition, communication, mobility and security. Hence, this initiative promotes key technologies, more precisely by formulating ten future projects in order to achieve their goals and to fulfill their visions. As a matter of fact, one of the government's high-tech-strategy projects is the so called “Industry 4.0”, which justifies the existence of several publications, conferences, and practical articles that primarily focus on that project (Drath & Horch, 2014).

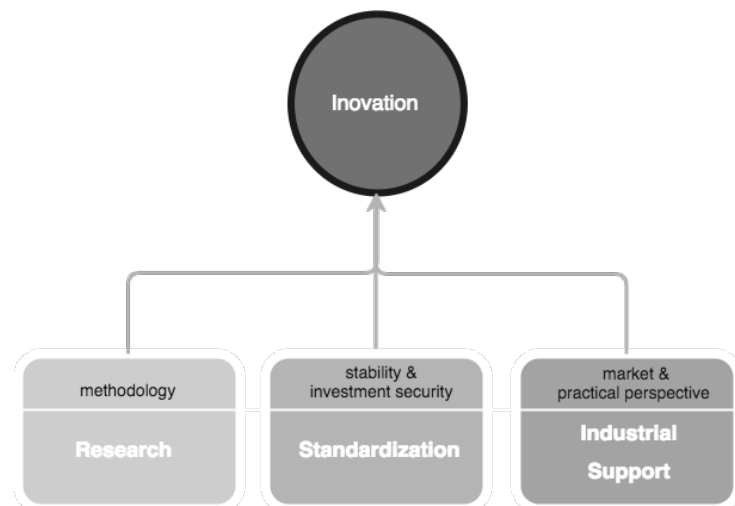


Figure 7: Innovation units of Industry 4.0

Furthermore, the “German Standardization Roadmap” (DKE, 2014) argues that an innovation like Industry 4.0 requires a quite close cooperation between several elements, namely research, industry and the standardization. As it is shown in Figure 7, research institutions are the ones that bring the methodological foundations for a new innovation, while

standardization ensures stability and investment security and the Industry tests the new concepts in a and regarding their market relevance.

Regarding the interest in Industry 4.0, there are essentially two main reasons that explain it. The first one is related to the fact that it never before was an industrial revolution announced a-priori and not observed ex-post. Hence, several research centers and companies had the chance to actively form such revolution. The second reason for the interest in Industry 4.0 is the forecast of a vast economic impact. In fact, this project has potential to develop new business models, services and products, as well as considerably increased operational efficiency (Kagermann, Lukas & Wahlster, 2013; Kagermann, 2014). According to Bauer, Schlund, Marrenbach and Ganschar (2014), developments on such project will contribute as much as 78 billion euros to the German GDP by the year of 2025.

As it is explained by the Bundesministerium für Bildung und Forschung (2017), the project Industry 4.0 essentially promotes a computerization in the field of production technology. In addition, this project also facilitates the vision of a Smart Factory, which is mainly characterized by mutability, resource efficiency and ergonomic design, as well as the integration of stakeholders and business partners in value-added processes of the factory. Therefore, the goal is that the factory extends with ideas from information technology, which in turn will enable intelligent behavior. However, the questions regarding how factories should actually be actually built, organized and structured in the future, and how existing factories can adapt, are still to be answered.

Nowadays, and despite presenting a high importance for several research institutions, companies and universities in the German-speaking area, Industry 4.0 does not have a commonly established definition. Hence, it is quite difficult to debate such topic on an academic level, as well as implement Industry 4.0 scenarios. As a matter of fact, all contributions of several participants during the last past three years only made the term more unclear, instead of clarifying it (Bauernhansl, ten Hompel & Vogel-Heuser, 2014). Organizations such as “Platform Industry 4.0” and “Industry 4.0 Working Group” also do not provide a clear definition of the concept, since they merely describe basic technologies, the selected scenarios and the vision that Industry 4.0 aims at. As it is clearly stated by Hermann, Pentek and Boris (2015), practitioners and companies need a certain systematization of knowledge in order to identify and to implement Industry 4.0 scenarios. These same authors, in their paper, formulate six design principles of the Industry 4.0 project, more precisely: interoperability, virtualization, decentralization, real-time capability, service orientation and modularity, which will be later discussed. Likewise, Hermann, Pentek and Boris (2015) also present four trends of Industry 4.0,

which will be later discussed as well: “Cyber-Physical Systems” (CPS), “Internet of Things” (IoT), “Internet of Services” (IoS), and “Smart Factory”.

Kagermann, Lukas and Wahlster (2013) describe their own vision of the project in the final report of the “Industry 4.0 Working Group”. They argue that companies of the future will found global networks and will then be capable of incorporating their warehouse systems, machinery and production facilities. Regarding the manufacturing environment, the so-called Cyber-physical Systems consist of smart storage systems, machineries, and production facilities, which autonomously exchange information, prompt actions and control each other independently. Such automation actually implies essential advances for the industrial processes within manufacturing, material usage, engineering, life-cycle and supply chain management. On the other hand, products of the future will be distinctively identifiable, able to be located at all times, and know their own history of events, current status and alternative courses to reach their target. The networks within the embedded manufacturing factory are vertically connected to all business processes, as well as horizontally networked to detached value networks while being able to communicate and make decisions in real time. Therefore, and to obtain all of this, end-to-end engineering from the moment an order is placed up to the outbound logistics is required, more precisely throughout the entire value-chain.

2.2 Background

The term “Industry 4.0” was firstly presented to the public in 2011, more precisely at the Hannover Fair, by an association of representatives from politics, academia, and business as an approach to strength the competitiveness of the German manufacturing industry (Kagermann, Lukas & Wahlster, 2011). In fact, the term “Industry 4.0” refers to the fourth industrial revolution, thus being preceded by three other industrial revolutions in the history of mankind (Posada *et al.*, 2015). The first industrial revolution happened in the second half of the 18th century, namely with the introduction of mechanical production systems that used water and steam power. The second one introduced, on the other hand, mass production in the 1870s, by the division of labor (Taylorism), and the use of electric power. Finally, the third industrial revolution introduced the used of advanced electronics and information technology, which were used to develop to further automate production processes. Such revolution was also called the “Digital Revolution” and happened in the 1970s (Lasi *et al.*, 2014).

It is relevant to mention that the German federal government supported the idea of Industry 4.0, which justifies its integration into their high-tech strategy program. Later on, the

government formed the “Industry 4.0 Working Group”, which published a set of implementation recommendations in 2012. The final report of this working group was presented in April of the following year (Kagermann, Lukas & Wahlster, 2013).

Simultaneously to the final report of the Industry 4.0 Working Group, the “Platform Industry 4.0” actually formed out of the industry associations Bitkom, VDMA, and ZVEI. Their main goal was to coordinate future activities of Industry 4.0. As a matter of fact, they are currently working on a reference model to structure basic ideas of this project (Platform Industry 4.0, 2017). Coordination and funding activities in Germany were created by the Economic Affairs and Energy (BMWi) and the Federal Ministries of Education and Research (BMBF).

Even though the term “Industry 4.0” is not so familiar outside the German-speaking area, we can find similar activities throughout the entire world. As a matter of fact, in the United States an initiative known as the Smart Manufacturing Leadership Coalition (SMLC) is also working on the manufacturing’s future. Essentially, SMLC is a non-profit organization of suppliers, manufacturing and technology companies, government agencies, universities and laboratories (Shin, Woo & Rachuri, 2014). In addition, the US government also supports research and development activities for the use of the Industrial Internet with a two billion dollar fund and General Electric, which are precisely the names of their initiatives of the “Industrial Internet”. According to Evans and Annunziata (2012), complex machinery with sensors and software connected to the network are defined in order to better predict, plan and control business and societal outcomes. Furthermore, additional similar ideas to Industry 4.0 can be found under the terms of “Integrated industry” or “Smart Manufacturing” (Bürger & Tragl, 2014; Dais, 2014).

2.3 Industrial Concerns

Firstly, it is essential to mention that the introduction of Internet technologies in the industry causes some quite understandable concerns for several plant operators. As a matter of fact, their facilities combine investments, know-how, production and profitability. Therefore, many visions of Industry 4.0 seem hardly intangible for today’s production systems, which justifies the fact that the communication of production-relevant devices, with a called “cloud”, is frequently perceived as a potential hazard. Hence, the industry actually sets some demands on Industry 4.0, which establishes the following requirements for an industrial acceptance:

- 1) Security of investment: Industry 4.0 must be gradually introduced into existing production facilities and equipment;
 - 2) Stability: Services that are available through Industry 4.0 (IoS) must never jeopardize the production, neither by failure or malfunction nor by uncoordinated intervention. Actually, production systems place increased demands on characteristics such as availability, real-time, reliability, durability, robustness, productivity, costs, security and other so-called nonfunctional properties. That is, the demands on these properties must remain unaffected by Industry 4.0;
 - 3) Controllability: Access to plant-related data and services is a prerequisite for value creation in Industry 4.0, but it has to be controllable. Write-access to production-related equipment, machines or systems actually require a special review body, which ensures the validity of the procedure in the context of the total production;
- Security: Prevention of unauthorized access to data or services must exist (DKE, 2014).

2.4 Value offer

In the production industry, a value-chain represents all steps within the production of a product order, more precisely as a structured string of activities (Kolberg & Zühlke, 2015). These same activities create values, consume resources and are also connected to each other through several processes. This concept was first published in 1985, more precisely by Michael E. Porter, in his book *Competitive Advantage*. According to DKE (2014), CPS will contribute to the autonomy of sub-processes within different value-chains in a production facility, which will support both short-term flexibility as well as the medium-term variability in response to the increasingly shorter and weightier external influences and, therefore, improve the resilience of production. Figure 8 represents the four dimensions of the value-adding processes, namely in the industrial production, which are essentially the following: business process and the product, factory and technology life cycles (Brettel *et al.*, 2014).

It is crucial to refer that the foundation of Industry 4.0 is the availability of all the relevant information in real time, namely through the integration of all entities that are involved in the value-added processes, as well as the ability to derive the optimum value flow from the available data at all times. In fact, dynamic, real-time optimized, self-organizing enterprise encompassing value networks are created by connecting objects, humans and systems. Furthermore, they can be later optimized according to several criteria such as costs, availability and consumption of resources (Lee, Bagheri & Kao, 2015).

2.5 Architectures

In the particular context of Industry 4.0, some architectures were identified in this research work. The evaluation of architectures concentrates on fulfilling the industry requirements in order to expand the modelling and standardization of the architectures for different purposes. Furthermore, a service-oriented architecture (SoA) related to Arrowhead Framework is presented.

2.5.1 Reference Architecture Model Industry 4.0 (RAMI4.0)

The authors of the RAMI 4.0 model developed a 3D model in order to represent all the different types of characteristics of the technical-economic properties (Figure 8). The RAMI 4.0 model is a small modification of the SGAM (Small Grid Architecture Model) and allows the appearance of different aspects. The small layers on the vertical axis represent different aspects, such as information, communication and functions from an integration capacity (Manzei, Schleupner & Heinze, 2016).

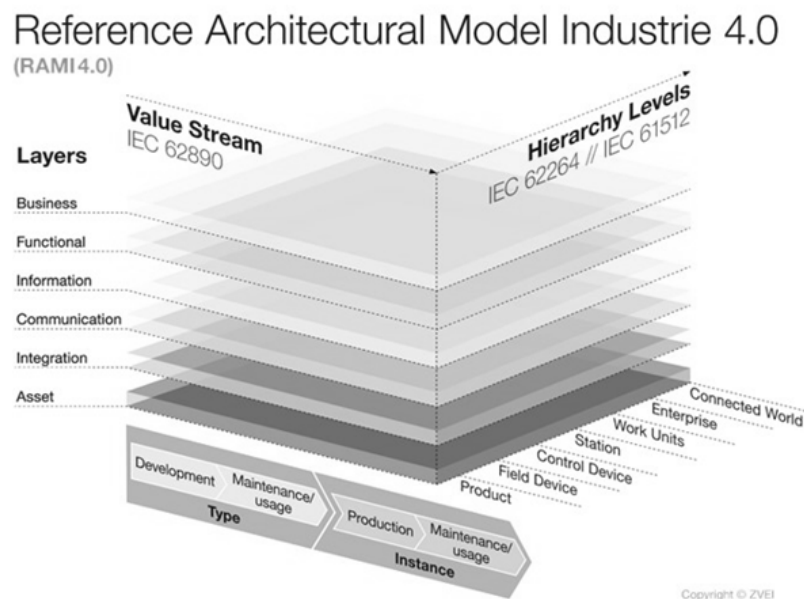


Figure 8: Reference Architecture Model - Industry 4.0 (Source: Zezulka, Marcon, Vesely, & Sajdl, 2016).

One of the most important criteria in modern engineering is the product's life cycle with value flow. The left horizontal axis demonstrates this same feature, and expresses the constant

acquisition of data throughout life. At the horizontal and right level is the function of the components in Industry 4.0.

The asset layer represents reality and physical components such as ideas, files, documents, linear axes, metal parts, and diagrams. The human part is connected to the world of virtual reality by the integration layer, the passive connection of the assets to the higher integration layer effected through QR codes. This layer provides information about the assets (HW/SW, components) in a form available for computer processing (Zezulka, Marcon, Vesely, & Sajdl, 2016).

There is also computer control of the asset event generation process and contains elements that are connected to IT. Integration of people is part of the integration layer via **HMI**.

The communication layer provides the standardization of communication through a uniform data format in the direction of the information layer. It also provides integration layer control services.

The information layer provides the runtime for event preprocessing, execution related to events, and allows a formal description of event preprocessing rules. The following functions ensure data integrity, consistent integration, new and higher quality data delivery and structured data service delivery. It also receives the events and data that are available for the next layer.

The functional layer allows the formal description of functions and creates a platform for the horizontal integration of specific functions. Contains runtime and modeling environment for business process support services and an environment time for applications and technical features. Rules and decision logic are generated in the functional layer. Some use cases can be run on other layers. Although remote access and horizontal integration may occur within the functional layer due to data integrity.

The business layer is the layer that ensures the integrity of functions in the value stream, and allows planning the business models and the results of the overall process. The legal and regulatory content allows the modeling of the rules that the next system creates and, at the same time, a link between different business processes.

2.5.2 Industrial internet reference architecture (IIRA)

Industrial Internet is an internet of things, machines, computers and people, intelligent industrial operations that use advanced **Data Analysis** for transformational business results. It incorporates the convergence of the global industrial ecosystem, advanced computing and manufacturing, penetrating detection and ubiquitous network connectivity. There are many

interconnected systems deployed today that combine hardware, software, and networking capabilities to feel and control the physical world. These industrial control systems have built-in sensors, processors and actuators that provide the ability to service commercials. These systems have not been connected to larger systems or with the people who operate with them.

The concept of industrial internet is a concept that has evolved over the past decade to be globally interconnected by trillions of ubiquitous devices representing the physical world. The Internet Industrial effort will bring industrial control systems online to form large end-to-end systems, connecting them with people and integrating them fully with business systems, business processes and analytical solutions. These end-to-end systems are referred to as Industrial Internet Systems (IIS). Within these IIS, operational sensor data and the interactions of people with systems can be combined with advanced organizational or public information and other means of data processing (for example, policies based on rules and systems). The result of this analysis and processing will allow, in turn, advances in the optimization of decision-making, operation and collaboration among a large number of autonomous control systems (Robert, Bradford, 2015).

Industrial Internet Reference Architecture (IIRA) is an open architecture, to maximize its value, it has a broad industrial applicability to drive and plan applicable technologies, and guide technology and standards development. The description and representation of the architecture are generic at a high level of abstraction to support the broad applicability of the industry.

The IIRA design aims to transcend today's available technologies and, by doing so, is able to identify technological ones based on architectural requirements. This, in turn, will boost the development of the new technologies of the Industrial Internet community.

2.5.3 Service-Oriented Architecture (Arrowhead Framework)

Briefly, a service-oriented architecture (SoA) is an architectural pattern of information technology in the field of distributed systems that structures and uses the services of IT systems. Then, a special focus is towards the orientation of the business processes whose abstraction levels are the basis for concrete service implementations. For example, a loan granted by a bank is a service abstraction at a higher level of business processes. Behind this particular service there is a number of people and IT-systems, such as “opening a business relationship”, “opening of one or more bank accounts”, “credit agreement” and several others. In fact, and by an effective orchestration of lower level services, some services of a higher abstraction level can

be created in a quite flexible manner, thus allowing for a maximum of reusability (Chung & Chao, 2007).

Regarding the SoA's goals, they are essentially the long-term reduction of costs in software development and a higher flexibility of business processes, namely by reusing the existing services. Also, the programming costs of the n^{th} with the SoA realized application should be considerably reduced, since all the necessary services are already available and only need to be effectively coordinated. Therefore, the remaining costs are related to the cost of business analysis and of software configuration (Douglas, 2015).

The Arrowhead project consists in a large European effort, which aimed the normalization through SOA design and the interaction between IoT applications. Such effort actually targeted many application domains, more precisely those that comprise industrial production, smart buildings, electromobility, and energy production. Regarding services, they are exposed and consumed by (software) systems, which are executed on devices that consist in physical or virtual platforms that provide computational resources.

More specifically, these devices are grouped into local automation clouds, which are self-contained, geographically co-located, independent from one another, and mostly protected from external access through several security measures. Arrowhead services are frequently considered either application services (when implementing a use case), or core services (that provide support actions such as service discovery, security, service orchestration, and protocol translation). Therefore, and aiming to ease the development of new application, the core services are actually included into the common Arrowhead Framework (Varga *et al.*, 2016).

As a matter of fact, the Arrowhead Framework is intended to be either deployed at the industrial site, or accessed securely, for example through a VPN. Regarding distributed IoT automation requirements, these include latency, security and packet delivery. Such application of IoT-based automation systems (Figure 9) would greatly benefit from QoS capabilities, including service-oriented management and monitoring of different QoS characteristics. It is important to add that industrial applications depend on the quality of the information communication, since they drive actions on industrial processes, which in different contexts are inherently time dependent, require communication robustness, sufficient bandwidth, or other stringent QoS requirements (Varga *et al.*, 2016).

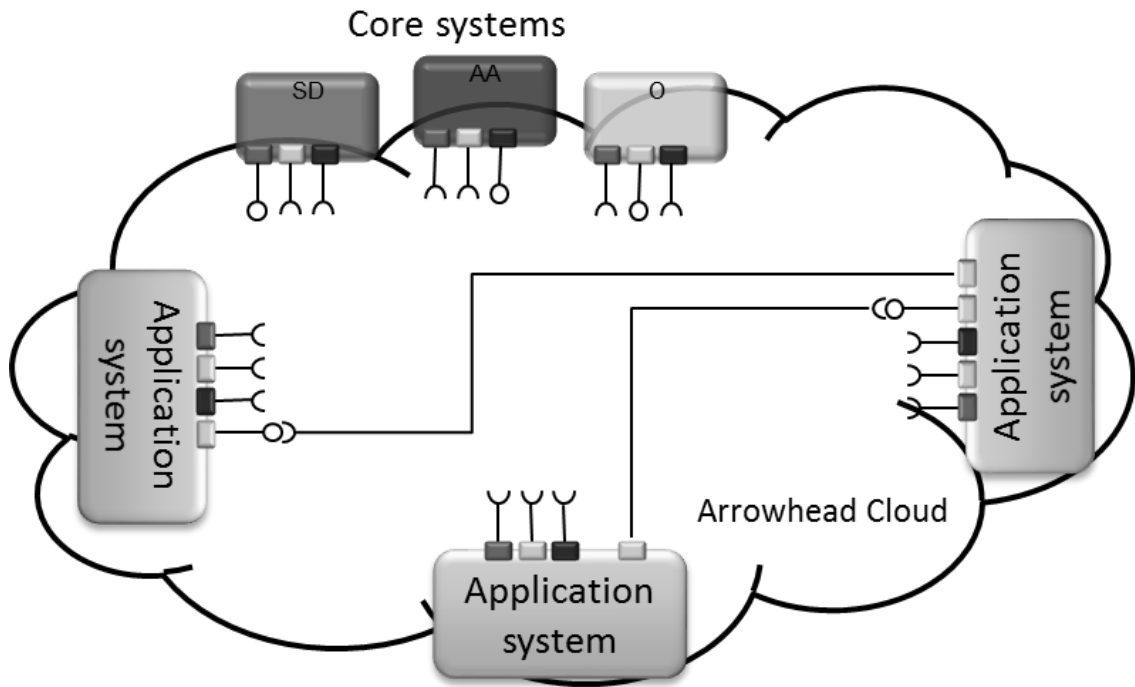


Figure 9: An Arrowhead local cloud comprising an orchestrated service instance (Source: Albano et al., 2017).

The QoS Setup and the Monitor are two core services that are devoted to supporting QoS in Arrowhead local clouds. The first one is provided by the QoS Manager system and it is consumed by systems in order to verify that QoS requirements are feasible in a local cloud, and to request the configuration of network activities and devices in order to grant the given QoS, this latter including performing reservation on resources such as network bandwidth and device processing time. On the other hand, the Monitor service, which is produced by the QoS Monitor system, is used in order to instruct the system to collect data from network actives and devices, regarding the performance of a specific service, and to compare it with the required QoS (Ta, 2006).

Given the current context, the person responsible for the elaboration of the present document have been simultaneously working for the Arrowhead project, more precisely assisting in the development of the services for the Arrowhead Framework: services of QoS, of QoS Manager, of QoS Monitor and of Event Handler. Therefore, a demonstrator was created in order to test the QoS on the Arrowhead Framework, using QoS Manager and QoS Monitor capabilities, which are used on top of the Flexible Time Triggered-Switched Ethernet (FTT-SE) technology.

Finally, the author of the present report made contributions to the paper "Quality of Service on the Arrowhead Framework" (Albano et al., 2017).

2.6 Components

Aiming to structure and specify the idea of Industry 4.0, Hermann, Pentek and Boris (2015) did an intensive literature review about the project. In fact, the authors reviewed 200 publications that could be found by applying the search term “Industry 4.0”, subsequently identifying 15 keywords that frequently occurred in the context of Industry 4.0. Then, they grouped the keywords into eight representative keyword groups, which are presented in Table 1. Out of the eight groups, Hermann, Pentek and Boris (2015) identified four key components of Industry 4.0: Cyber-Physical Systems, Internet of Things, Internet of Services, and Smart Factory. The latter four groups in Table 1 are not found to be independent key components. As a matter of fact, Smart Products can be considered as a subcomponent of Cyber-Physical Systems, Machine-to-Machine communication is considered to be an enabler for Internet of Things, and Big Data and Cloud Computing are data services that use the information inside Industry 4.0 applications, but that do not represent independent Industry 4.0 components.

Table 1: Industry 4.0 components (as identified by Hermann, Pentek and Boris, 2015).

Keyword (group)	Number of publications in which keyword (group) occurred
Cyber-Physical Systems (CPS)	46
Internet of Things (IoT)	36
Smart Factory	24
Internet of Services (IoS)	19
Smart Product	10
Machine-to-Machine (M2M)	8
Big Data	7
Cloud	5

In the next sections, the explanation of the identified Industry 4.0 components is proceed. Furthermore, the link to Industry 4.0 is elucidated and some application examples are provided.

- **Cyber-Physical Systems (CPS):** Kagermann (2014) argues that a substantial element of Industry 4.0 is the union of the physical and virtual world. Such fusion is actually achieved through Cyber-Physical Systems (CPS). The technical committee 7.20 “Cyber-Physical Systems” and 7.21 “Industry 4.0” of the VDI-GMA(2015, cit. in Allenhof, p. 11) describe a CPS as follows: “A CPS is a system that connects real

physical objects and processes with virtual objects and processes through an information network, which is open, partly global, and continuously connected. Optionally, a CPS uses local or external services, uses human-machine interfaces, and offers the possibility to dynamically adapt the system at runtime”.

Furthermore, the CPS research agenda explains that CPS are comprised of embedded systems, production, logistic, engineering, coordination and management processes and Internet services (Mikusz, 2014). Therefore, they immediately capture physical data through sensors and generate physical actions through actuators. Overall, the systems are connected over digital networks, using worldwide available data and services.

Drath (2014, cited in Allenhof, 2015) presents an example of a Cyber-Physical System, more precisely regarding future possibilities for traffic light systems. In this particular example, the real physical traffic lights log into a centralized registration system, instantiate a virtual copy of their identity and publish their planned schedule. Furthermore, vehicles that participate in road traffic are also able to log into the registration system and to download the schedules of all nearby traffic lights. They are able to adjust their route and velocity to optimize their journey by basing on the vehicle’s location, velocity, weather data and other traffic-sensitive information. On the other hand, and regarding emergency vehicles, such as ambulances or police cars, they are also able to initiate a change in the scheduled behavior of the traffic lights, thus ensuring that those traffic lights are green as they approach them.

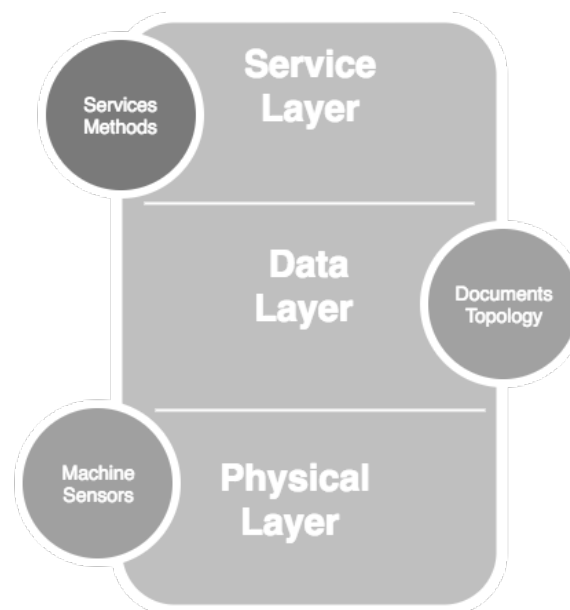


Figure 10: Layers of a Cyber-Physical System.

Essentially, Figure 10 summarizes the concept of CPS according to Drath's (2014, cited in Allenhof, 2015) arguments, which is subdivided in three layers. In the bottom layer are the physical objects, which are the intelligent, self-exploratory, and self-diagnostic assets in systems (compare to the real vehicles and traffic lights presented in the previous example). In the middle layer are the data and models of the physical objects (virtual instances of traffic lights). Finally, in the upper service layer are new products and services, which will be further developed (for instance, and based on the presented example, a service that evaluates the traffic light schedules and subsequently changes vehicles' velocity in order to improve the fuel consumption) (Wang *et al.*, 2016).

- **Internet of Things (IoT):** Regarding the concept of Internet of Things (IoT), it derived from the idea that the computer will progressively disappear as a device in the future, being replaced by “intelligent objects and things” (Sadeghi, Wachsmann & Waidner, 2015). It is crucial to add that, instead of being a subject of human attention itself, such intelligent, ever-smaller embedded objects aim to support human activities quietly in the background, without attracting or demanding any attention. Weiser (1991) actually explained, and for the first time, this particular vision.

In particular, the IoT designates the existing link of clearly identifiable physical objects (or things) with a virtual representation in an Internet-like structure. In this perspective of “Future Internet”, humans and representations of the “things” are participant (Saint-Exupery, 2009). As a matter of fact, the automatic identification using RFID is quite frequently regarded as the foundation of the IoT. Nonetheless, a unique identification of objects can also be obtained through a barcode or even a 2D code. Several devices, such as sensors or actuators, actually extend the functionality to the collection of states and to the execution of actions. In summary, Giusto *et al.* (2010) argue that IoT allow for “things and objects” to interact with each other and with their own neighboring intelligent objects through unique addressing schemas. However, the intelligent things and objects can be understood as CPS. Hence, IoT is essentially a network in which CPSs communicate through quite unique addressing schemas (Anderl, 2014).

A particular example of IoT is related to the tracking of an ordered product over the internet, since delivery services now offer the possibility to follow the ordered product through their several delivery stages. Such process is done by a unique identification using barcodes or 2D codes (Weyer *et al.*, 2015).

- **Internet of Services (IoS):** Buxmann *et al.* (2009) argue that Internet of Services (IoS) allows service suppliers to provide their services over the Internet. As a matter of fact, the IoS is quite complex, since it comprises an infrastructure for services, certain business models, the services themselves, and participants requesting the services. These services are, on the other hand, combined into value-added services and offered by various vendors and communicated to several users and stakeholders through numerous channels. Such development actually enables new possibilities, more precisely in terms of dynamic variation in the distribution of specific value-chain activities. Furthermore, it is quite imaginable that the idea of IoS transfers from single factories to entire value-added networks. Despite offering some production types of their products, factories of the future could also offer some special production technologies, which would be offered through the IoS network, thus enabling a customer of the services to simply manufacture a specific part, or compensate for limited production capacities (Scheer, 2013).

This concept of IoS has actually been applied in a project that was initiated by the Ministry for Economic Affairs and Energy, within the program “Autonomics for Industry 4.0” (Allenhof, 2015). In that same project it is presented a new distributed production control system, which is developed for the automotive industry, comprehending assembly stations that are of modular nature. Such concept actually allows the flexible modification and expansion of some stations. Regarding the transportation between the modules, it is realized through automated guided vehicles (AGV). Considering that the project is based on a SoA, both the AGV and the assembly stations publish their services to the IoS, which allows the products, in this particular case the vehicles, to choose their course autonomously through the production process, having been pre-programmed with customer specific configurations (Shafiq *et al.*, 2015).

- **Smart Factory:** According to Kagermann, Lukas and Wahlster (2013), the so-called “Smart Factories” are a key feature of Industry 4.0. As a matter of fact, such term constitutes the vision of a production environment in which production facilities, as well as logic systems, largely self-organize without any type of human intervention. Actually, the “context-aware” Smart Factory considers both the position and the status of a product within the entire process, also assisting machines and people in the execution of their own tasks. Briefly, it is important to mention that the technical

background of Smart Factories is built on CPSs, in which the systems communicate and cooperate with each other and humans with the assistance of IoT. More specifically, such vision addresses the communication between the product or work piece and the manufacturing facility, where the product carries its own production information in machine-readable form, which allows machines to retrieve their workload from the product itself. Furthermore, these systems gather information both from the real physical and the simulated virtual facilities in order to determine their next production steps. Hence, physical information frequently means the position of the product or tool of a machine, whereas information from the virtual model is the optimal tool choice for a specific task, or the determination of optimal production schedules (Schlechtendahl *et al.*, 2015).

Schlick, Stephan, Loskyll and Lappe (2014) present an example of Smart Factory, namely the “Future Urban Production” facility in Fellbach, Germany. Among all the manufactured products in such facility are gear wheels. In the past years, the physical transport of the goods between several delivery and pick-up spaces was done by an electric truck that drove around the factory every hour. Such inflexible procedure has been superseded by material supplies on demand in the framework of an Industry 4.0 pilot project. Therefore, and in order to implement a demand-driven supply, nowadays intelligent work piece carriers are used, which basically consists of a report by the carrier, and when the piece is actually ready to be picked-up, who assesses the status to the transportation control unit. Such procedure is quite helpful, since it decreases the number of transportation runs and saves personnel superfluous work.

2.7 Design Principles

Hermann, Pentek and Boris (2015), to assist organizations in the process of identification and implementation of Industry 4.0 pilot projects, argue that specific design principles are required in order to be defined. Their work actually derives six design principles, which are summarized in Table 2. Furthermore, it is crucial to establish that such principles are obtained by an evaluation of the literature review regarding the Industry 4.0 components. Hence, Table 2 demonstrates which Industry 4.0 components lead and are linked to which design principle.

Table 2: Design principles of each Industry 4.0 component, according to Hermann, Pentek and Boris (2015).

	Cyber-Physical Systems	Internet of Things	Internet of Services	Smart Factory
Interoperability	X	X	X	X
Virtualization	X			X
Decentralization	X			X
Real-Time Capability				X
Service Orientation			X	
Modularity			X	
Standardization	X		X	X

These design principles for Industry 4.0 are illustrated by an example of the key finder plant of SmartFactory^{KL} in Germany (SmartFactory, 2015, cited in Allenhof, 2015). This project is actually an initiative of the German Research Center for Artificial Intelligence, more precisely to the development of a vendor independent technology that operates in a production facility. Such demonstration plant assembles and processes parts for key finders. Hence, all relevant data for the production process is saved on a RFID tag, which is directly attached to the work piece itself. Regarding the data, it is written on the transponder in the commissioning station of the production line. Afterwards, the production line basically consists in a milling station, in an automated assembly and in a manual workstation. At the manual workstation two tasks can be developed: the complete manual assembly of the product or the final manual assembly after the automated assembly station is performed. In addition, this manual station represents a context-sensitive work environment, which guides people with the assistance of Augmented Reality, more precisely through the complex manufacturing processes.

- **Interoperability:** Since CPSs are connected with each other and humans over IoT and IoS, interoperability is quite an important enabler for Industry 4.0. Therefore, a key success factor in order to enable a rule-based and situation-controlled communication between CPSs of several manufacturers will be the creation, establishment and integration of standards. Actually, the German Commission for Electrical, Electronic &

Information Technologies of DIN and VDE addressed such requirement in their publication (DKE, 2014). More specifically, its publication claims that the existing system landscape of both technologies and structures is still not coherent and entirely defined in a global manner. Thus, it is not sufficient to only define the emergent behavior and additional level of integration of Industry 4.0, but also important models of the classical architecture, which clearly needs to be overworked and integrated in addition to Industry 4.0 (Monostori, 2014). According to the same resource (DKE, 2014), when the interoperability is address in the context of the demonstration plant at SmartFactory^{KL}, all CPSs of such plant (namely assembly stations, work piece carriers and products) communicate with each other through open nets and semantic descriptions. As an example, the order information can be transferred from the application system to the control of the picking station and further stored in the digital product memory by using a recently established communication standard OPC-UA.

- **Virtualization:** Regarding the principle or virtualization, it basically represents the ability of CPS to monitor physical processes. Afterwards, the monitored data is linked to simulation plant models, thus producing a virtual copy of the physical world. In addition, the key finder demonstration plant also contains a virtual representation of itself, which includes the monitorization of the condition of all CPSs in the plant (SmartFactory, 2015, cited in Allenhof, 2015). In case of occurring any errors in the facility, a human can be notified and assisted during the resolution of such failure. Furthermore, humans can also be supported by the system in handling the rising technical complexity of the next working steps, maintenance, or safety arrangements (Fast-Berglund *et al.*, 2014).
- **Decentralization:** Both the progressive reduction of mass production and the increasing demand for individual products gradually impede the central control of a production system. Therefore, and through developments in the embedded systems, CPSs can actually become more intelligent, being able to make decisions on their own. It is only in the case of maintenance or failure that de CPSs need to communicate to a centralized, higher level system. Thus, the entire production facility moves towards becoming more decentralized. Considering the given example, the decentralization in the SmartFactory^{KL} plant is realized through the RFID tag in the product, since all the information about the individual production stages can be directly retrieved from the product itself (DKE, 2014).

- **Real-Time Capability:** Allenhof (2015) claims that, in order to organize a factory and to react to time critical events quite immediately, it is necessary that the Industry 4.0 collects and analyzes data in real time, considering that the increasing amount of information that circulates in a factory sets some sophisticated demands on the processing units. Therefore, it is also crucial to determine judicious selections and cycle times for the data that will be collected for a further processing. Also, in the key finder production the status of the plant is continuously tracked and analyzed. If an unpredicted event occurs, the organization/facility must be able to react immediately and reroute products that can get stuck in the production line of another machine.
- **Service Orientation:** Essentially, service orientation is a design paradigm that has several principles that stress the separation of concerns in the software, leading to components of software that are divided according to operational capabilities, each one designed to solve an individual and particular problem. Also, these components can be qualified as services. It is crucial to refer that the services of companies, humans and CPSs are accessible over the IoS, thus being able to be used by other participants. Nonetheless, all services can be offered both internally and globally across the organization's borders (Allenhof, 2015).

Regarding the example of the SmartFactory (2015, cited in Allenhof, 2015), its plant is also designed according to a service-oriented architecture. As a matter of fact, all CPSs in the plant post their functionalities as a web service within the factory. Therefore, the customer order details that can be retrieved from the RFID chip on the products and a suitable production process can actually be composed based on the available services.

- **Modularity:** By designing a factory in concordance to the paradigms of modularity, companies provide the advantage to flexibly adapt to requirements, more precisely by replacing or by expanding certain modules or CPSs in the factory. These same flexible modules are frequently called as “plug and produce”-modules. Basically, they describe the ability of certain machines and tools within the facility to communicate its services, as well as location, in the factory. Furthermore, the “plug and produce”-ability, that should also be vendor independent, sets some demands for necessary standardized interfaces (Shafiq *et al.*, 2015).

In the key finder facility, new modules can be added or removed, more specifically by using the “plug and produce”-principle. Also, new CPSs can be automatically recognized, when based on standardized hardware and software interfaces, as well as used instantaneously by using IoS. Hence, modular systems can be easily adjusted in

case a product changes its characteristics or even if there are some seasonal instabilities (Allenhof, 2015).

- **Standardization:** As it is demonstrated by Figure 7, standardization is considered as a key component for new innovations, more precisely to ensure stability, security for investments and confidence among every users and manufacturers. Furthermore, standardization is also understood as being the entirely consensual establishment of a recognized organization of strategies, guidelines and regulations for recurrent or general activities. The previously established standards of standardization bodies, such as IEC and ISO, are actually accompanied by a set of certain specifications in several forms. As an example, these specifications can be the DIN Specifications (DIN SPEC), the VDE Codes of practice, the Publically Available Specifications (PAS), the Technical Specifications (TS), Industry Technical Agreements (ITA) or even Technical Reports (TR) (DKE, 2014).

Regarding the Industry 4.0, a particular difficulty arises for both the standardization and the terminology. In fact, and previously to defining the behavior and additional levels of integration of this project, the existing landscape of structures and technologies needs to be coherent, as well as entirely defined in an internationally standardized manner (Drath & Horch, 2014).

Even though digital simulation can enable an increased variety of possibilities for tests of several scenarios, there are still few available standards among the diversity of IT solutions. Some of these solutions include several data models and interface protocols, which subsequently requires an enhanced maintenance, changes and new implementations. Also, there is a certain lack of transparency in the way that these same changes are implemented. The most classical way to program has to be replaced by a new system of rules, more precisely by one that sets some specifications, a coherent terminology and libraries. For instance, new Industry 4.0 terminology could be added to the existing specifications, such as it occurs in the specification IEC 51131-3, on terminology for industrial automation and information that orient instrumentation and control technology (Weyer *et al.*, 2015).

The ever-rising information technological expenditure behind facilities in the automation environment can limit the programmability, more precisely due to a steady increase in its complexity and comprehensibility. Thus, it actually can lead to an abandonment of absolute control, which requires a particular solution to ensure room for some developments, while minimizing the described problems. Most solutions that

are proposed by Industry 4.0 suggest that an architecture of decentralization through a service-oriented architecture (SoA) must be implemented (Posada *et al.*, 2015).

2.8 Development Process

The previous explanations given on the topic regarding Industry 4.0 largely describe the convergence of technologies that were formerly applied separately. Such combination is, and for example, the fusion of mechanical engineering and electrical engineering into mechatronic engineering. In the field of the production industry, a possible example is, for instance, the development of smartphones, where the technical areas of physics, electrical engineering and IT actually converge. This modifies competence requirements for engineers, pushing them to become more and more interdisciplinary. However, such convergences of technologies need to be taught to students and engineers. Zoitl (2015, cited in Allenhof, 2015) argues that the coming industry requires engineers that are specialists in terms of having the overview over an entire system. Furthermore, he also claims that the degree program “System Engineer” is, for instance, one trendsetting course.

On the other hand, Geisberger and Broy (2015, cited in Allenhof, 2015) suggest that the biggest engineering challenge is to satisfy two of the main features of CPS. In fact, engineers must develop secure, safe and dependable systems that can also add value in a sustainable manner. However, those systems must be open to innovation networks that facilitate new applications, as well as networks. Furthermore, the authors argue that these procedures need an iterative approach of exploratory work, from multi-vendor industries and research associations. In addition, the interconnectedness and merging of all technologies and applications needs an integrated approach, as well as the development of a new, or at least extended, engineering concepts. Hence, a quite substantial amount of effort is required, more precisely to be devoted to the fields regarding the integration, modelling, interdisciplinary requirement engineering and software and systems engineering Hermann, Pentek & Boris, 2015).

Methods for Requirement Analysis, Design and Evaluation

Firstly, and for the requirement analysis, iterative design and evaluation of architectures and solution designs, there are two methods that can currently be used. One of them is related to the use of interactive prototypes and realistic virtual simulations, while the other is related to the use of mathematical modelling, where a simulation of a system is applied. In the particular context of Industry 4.0, some new challenges related to the development of open systems have

been arising. According to the users' needs, the systems are quite capable of interacting in a highly dynamic context with a global network. However, fulfilling all these requirements involves the expansion of the research beyond the horizon and the development of new analysis methods. Nonetheless, these new methods must be aligned to the previous separate designed models that ensure an end-to-end verification and validation (Allenhof, 2015).

Requirements Engineering

Allenhof (2015) states that requirements engineering is the key to the conception, design, validation and verification of CPSs. In fact, the main tasks of requirements engineering include determining user, business, process requirements and customer, identifying potential problems, setting targets and priorities, resolving inconsistencies and possible conflicts, as well as establishing requirements for the architecture, components and communication methods of a particular system. Regarding the wide range of possible Industry 4.0 scenarios, the major requirements engineering's concerns are essentially the identification, analysis and specification of the needed requirements (Drath & Horch, 2014).

Human, System and Architecture Models

At last, and in order to ensure that integrated and comprehensive models of systems, humans and architectures are adequately designed, configured and managed, several technologies and knowledge of fields are required, namely:

- 1) Engineering and science, specifically in electrical and mechanical engineering, chemistry, physics and biology;
- 2) Information and communication technology (ICT), sensor technology, embedded systems, system and networks management and the Internet;
- 3) Neuroscience, cognitive psychology and brain research on human behavior, thought and complex problem solving;
- 4) Social science and social networks (Lee, Kao & Yang, 2014).

Essentially in the field of engineering, research focuses on methods to reduce complexity and, therefore, simplifies the design and management of CPS. The specific areas of research are the following:

- 1) Human-machine systems and human factors;
- 2) Model-based systems engineering;
- 3) Virtual engineering;

- 4) Evolutionary software development;
- 5) Generative programming and synthesis;
- 6) Reuse of software product lines;
- 7) Formal verification, validation and testing (Monostori, 2014).

2.9 Conclusions

All the explanations given in this topic, regarding Industry 4.0, clearly describe technologies that already exist. Therefore, the challenge lies in the structured connection of these same technologies, in the standardization of some accomplished principles and in the reference models. As a matter of fact, system architectures, as well as business models, need to be created and sorted into application dependent groups. All the presented design principles support both the scientific and practitioners' communities, thus helping in the implementation of scenarios and in the selection of potential use cases for future investigations.

According to Drath (2014, cited in Allenhof, 2015), Industry 4.0 is an inevitable phenomenon. It is quite analogous to the introduction of the Internet itself, in the beginning of 1990, which subsequently brought out an unimaginable world of online stores, online brokerage, auctions, E-mail, video streaming, Facebook and app stores. Nonetheless, the scope of Industry 4.0 is still not graspable in its entirety.

In addition, and in order to summarize it all up, the research on the topic Industry 4.0 was essentially applied in the languages English and German. Hence, a certain limitation of the results actually persists. Therefore, it is quite possible that other results, from journals in different languages, may have been left unobserved in the present work. Finally, an evaluation of middleware solutions follow the industry 4.0 design principles is given in the Chapter 3.

3 Evaluation of Message Oriented Middleware solutions

In this chapter, the solutions based on Message Oriented Middleware (MOM) will be evaluated, since the intended solution focuses on a reliable exchange of messages around a particular network and on the use of queues as a reliable load balancer and of topics, in order to implement the options of publishing and subscribing (Table 3).

Hence, solutions that are based on Enterprise Service Bus (ESB), which provides Enterprise Integration Patterns for smart routing, transformation, orchestration and working with other technologies, are not taken into consideration in the present work because of the low performance. In this chapter an analysis of the most relevant MOMs is performed.

3.1 Advanced Message Queueing Protocol (AMQP)

The Advanced Message Queueing Protocol (AMQP, 2017) is an open standard application layer that allows message exchange between applications. Hence, this means that AMQP is a protocol for message-oriented middleware (MOM). The most important components of this protocol are the message brokers, with they act as a bridge between two applications, receiving messages from producers and routing those messages to the correct consumers. Essentially, the AMQP protocol is constituted by several entities, namely producers, consumers, exchanges, bindings and queues. Each entity plays a distinct role enabling the communication and cooperation between two or more applications. As a matter of fact, producers are applications that are quite compliant with the protocol and publish messages into the broker. Then, messages are published to an exchange in the broker (Teixeira, 2015).

The AMQP protocol presents the following characteristic:

- 1) **Interoperable:** all AMQP clients interoperate with all AMQP servers;
- 2) **Reliable:** AMQP is capable of removing communication gaps and slowdowns between different platforms, critical systems and applications' components, both within a company with external systems;
- 3) **Unified:** it provides a core set of messaging patterns via a single manageable protocol;
- 4) **Complete:** AMQP provides a wire level transport for applications;
- 5) **Open:** vendor and platform agnostic;

- 6) **Safe:** offers some of the most reliable security and authentication mechanism to do its entities (Fernandes, 2011).

3.2 Message Queuing Telemetry Transport (MQTT)

The Message Queuing Telemetry Transport (MQTT, 2017) is a lightweight network protocol that is used to publish and subscribe messages that are sent between devices. Basically, the MQTT works on top of the TCP/IP protocol, being ideal for use in constrained environments or low-bandwidth networks with limited processing capabilities, small memory capacities and high latency. Its design actually minimizes network bandwidth requirements, while attempting to ensure reliability on delivery (Chen, 2017).

Furthermore, the MQTT protocol provides publish-and-subscribe paradigm between clients and a brokers. In this context, a client is any device, from a micro controller up to a server, that has a MQTT library running and connecting to a MQTT broker. Hence, clients can subscribe and publish on topics, the latter being the routing information for the broker. On the other hand, a client app is the responsible for collecting information from the telemetry device, thus connecting and publishing the information to the server. Finally, the broker is essentially responsible for receiving all the messages, filtering all of them and routing. Afterwards, it sends the message to all subscribed clients. In other words, brokers receive messages from publisher clients on a certain topic, which they forward to the interested subscribers (Seebacher, 2013).

3.3 Simple Text Oriented Messaging Protocol (STOMP)

The Simple Text Oriented Messaging Protocol (STOMP, 2017) is a lightweight and quite simple human readable text messaging protocol. STOMP provides an interoperable wire format, in order to allow clients (publishers/consumers) to communicate to any message broker that actually supports the protocol, which is based on the HTTP protocol. Essentially, the messages consist in a frame header with certain properties and a frame body (Alves, 2014).

It is important to establish that this protocol does not deal with queues and topics, since both the semantics and the detailed syntax of the destination tag are not defined in the official specification. Therefore, different brokers can actually interpret the destination in different manners, which compromises the protocols interoperability. At last, this protocol has several open source implementations available for clients and brokers, which provide libraries in different programming languages (Alves, 2014).

3.4 Java Message Service (JMS)

The Java Message Service (JMS, 2017) provides a standard java API to create, send, receive and read messages. Furthermore, JMS provides:

- 1) Two different kinds of communication models: PTP (Point to Point) and publish/subscribe;
- 2) Reliable message transport;
- 3) Transaction;
- 4) Message filtering mechanism.

Nonetheless, there are several terms used in the context of JMS, which need to be clarified. In fact, there is the JMS provider, which basically consists in the message broker that actually implements JMS. On the other hand, PTP is the term used to define Point to Point messaging model, which provides durable buffering of the message in queue. The queue refers to a message domain that contains the message, which can be consumed by only one consumer each time. The topic, which is also related to the message domain, contains the actual message that can be consumed by several active subscribers at the same time. The connection factory is basically a factory object that is used to establish the connection, while the latter is established between the client and the message broker. The term destination is related to the message domain that is managed by the JMS provider and it stores the message that is produced by all clients. The session is basically a thread that receives and sends messages. Finally, the term message producer refers to the object that is created by one session to send the message, while the message consumer is the object created by that same session in order to receive the sent message (Yunpeng, 2010).

3.5 Extensible Messaging Presence Protocol (XMPP)

The Extensible Messaging Presence Protocol (XMPP, 2017) is a protocol for near-real-time messaging, presence, and request-response services. This protocol uses the Extensible Markup Language (XML) as the base format for message exchange. Basically, XMPP is a protocol that provides an infrastructure to allow the exchange of small pieces of XML among entities and in close real-time (Saint-Andre, Smith & Tronçon, 2009).

Because of the features that are provided by the XMPP, this protocol has been used in order to build large-scale distributed systems, Internet gaming platforms, search engines and video or audio conferences. It is precisely the substantial usage of XMPP in several distinct

applications that demonstrates how versatile, flexible and powerful this protocol can truly be (Moffitt, 2010).

Finally, the XMPP protocol is for message-oriented middleware systems and is categorized as a message passing paradigm, since the identity of clients are known by other clients. As a matter of fact, and even if XMPP is used for client-server architectures or for peer-to-peer, it can actually be extended with some new features, namely through the definition of XMPP Extensions (XEPs), that are essential to enable XMPP to be used in different contexts (Moffitt, 2010).

3.6 Data Distribution Service (DDS)

The Data Distribution System (DDS, 2017) for Real-Time is a standard that is mainly used to implement the publish-subscribe communications for real-time and embedded systems, also providing a set of QoS policies. Furthermore, it is a data-oriented middleware that is based on the Data Centric Publish-Subscribe (DCPS) model, where it actually implements a distributed peer-to-peer architecture that provides a reliable and efficient communication among applications (Albano *et al.*, 2015).

DDS provides high interoperability among heterogenous systems, more precisely through a Global Data Space (GDS), where several applications publish messages into the GDS (publishers) and other applications are able to access the same GDS and subscribe to the information of interest (subscribers). However, and every time a publisher changes data in the DDS, the latter is also responsible for propagating that changed data to all subscribers.

Finally, in DDS, the data that is available in the GDS actually follows a data model that is based on specific structures. Each one of them is identified by a topic that uniquely identifies the structure and a type that provides structural information that is used by the middleware in order to perform actions on the existing data. A set of QoS policies is also provided by DDS, aiming to guarantee the delivery of all data, the real-time systems performance, the bandwidth reservation, redundancy and data persistence (Albano *et al.*, 2015).

3.7 Open Platform Communications Unified Architecture (OPC-UA) publish-subscribe

Even though OPC-Classic was quite popular, the security, the platform's dependency and some scalability issues led to the development of alternative standards or to a change in the

OPC paradigm. Therefore, the open platform communications unified architecture (OPC-UA, 2017) is essentially the collective effort of collaboration between manufacturers and OPC Foundation. It is important to establish that OPC Classic was a merely client-server based connectivity solution, while the OPC-UA takes the SOA (Service Oriented Approach). Therefore, the latter introduces security, reliability, scalability and also eliminated the platform dependency. The OPC-UA is open, thus allowing small embedded systems to be connected to the internet in a secure and uniform manner. Some of its most important features are:

- 1) **Platform independent:** OPC-UA has a quite low sized protocol stack that is written in ANSI C, being able to be ported on to small embedded systems. This freedom, which is related to the platform, made OPC-UA to be considered for IoT;
- 2) **Services:** it provides a suite of standardized services for data access, events, alarms and historicizing, among others;
- 3) **Address space flexibility:** since it is object oriented. It addresses space support methods that can be executed from remote clients;
- 4) **Common protocol suite:** it uses common protocol suites and encodings, making it possible to be used with the internet;
- 5) **Information model:** it defines the means to exchange useful information through its adaptable information model, which can also be integrated with industrial data models;
- 6) **Security:** it defines a robust security model that provides application to application security. Such security model aims to provide user authentication, access rights, secure end to end communications, namely through encryption;
- 7) **Process transparency:** it helps representing the underlying process data transparently to the client-side users. Therefore, users do not need to understand the underlying technology or the data representation (Pujari, 2016).

Table 3 - Message Oriented Middleware (MOM) protocol

MIDDLEWARE PROTOCOL	IMPLEMENTATION PROTOCOL	DISCOVERY	TRANSPORT	QUALITY OF SERVICE	ENCODING	OPEN SOURCE
AMQP	Broker	No	TCP/IP	Up to 3 Parameters	SASL and/or TLS	ActiveMQ ¹⁹ , RabbitMQ ²⁰ , OpenAMQ ²¹ , Apache Qpid ²²
MQTT	Broker	No	TCP/IP	Up to 3 Parameters	TLS	ActiveMQ, RabbitMQ, Mosquitto ²³
STOMP	Broker (Server)	No	TCP/IP	Application Dependent	Text-based or Binary	ActiveMQ, RabbitMQ, HornetQ ²⁴
JMS	Broker (Server)	No	TCP/IP	Up to 3 Parameters	Binary	ActiveMQ, HornetQ, Open MQ, Apache Qpid
XMPP	Broker (Server)	Yes	TCP/IP	None	Plain Text	ActiveMQ
DDS	Global Data Space	Yes	UDP/IP or TCP/IP	Up to 22 Parameters	Binary	OpenDDS ²⁵ , Vortex OpenSplice ²⁶
OPC-UA publish-subscribe	Broker (Server)	Yes	UDP/IP or TCP/IP	Not defined	Binary	None

¹⁹ <http://activemq.apache.org/>

²⁰ <https://www.rabbitmq.com/>

²¹ <http://www.openamq.org/>

²² <https://qpid.apache.org/>

²³ <https://mosquitto.org/>

²⁴ <http://hornetq.jboss.org/>

²⁵ <http://opendds.org/>

²⁶ <http://www.prismtech.com/vortex/vortex-opensplice>

3.8 Conclusions

The proposed analysis for the Message Oriented Middleware solutions under study, was presented in this chapter by analysing the message protocols and its properties for each solution. Furthermore, in the subsection competition analysis of the next chapter, is compare the message protocols of each solution, for the defined alternatives.

4 Value analysis

This chapter presents the identified stakeholders concerns then contextualize the 5 key elements of Peter Koen's model ("The New Concept Development Model" – NCD) in order to define what is a value analysis, its importance and usefulness. This solution was developed for the MANTIS European project but in future the commercial purposes is intended. Thus, we proceeded to an analysis of the offer and the current market value of the project solution followed by the canvas model to describe the proposed business ideas. Finally, the competition analysis defines the communications layer related to the solution.

4.1 Stakeholders concerns

The identified stakeholders are the industry companies and one of the partners and stakeholder to highlight is ADIRA. This company, formed in the year 1956 that export to more than 40 countries worldwide, aims to predictive and proactive maintenance of their machines. Since the new industry business models became quite popular due to the fact that it enables some flexibility on IIoT. However, the lack of efficient solutions had a negative impact on the exploitation of new industry business opportunities.

Since an efficient solution is intended, that reach all stakeholders, the present work respects some critical and less technical aspects:

- 1) Proprietary development: The development will be done by the same company. The technology and architecture will be documented in order to be able to be picked up by another company and/or developers;
- 2) Consistency between services: The architecture must be adapted to its needs, that is, to the corresponding service, but must also respect the main architecture design. In sum, it must maintain consistency between all the services that use middleware;
- 3) Costs of development: Initial development will have a higher cost, which is necessary to start up the project into its initial production phase. However, the present work must have in mind the future work, in the sense of adding new services;
- 4) Confidentiality: Some practices must be put in place, more precisely best practices according to security guidance. The aim is to give a sense of security to who is using the service.

4.2 Business Process and Innovation

Nowadays, and considering the globalization's complexity, the vast majority of top managers of organizations have to try to adopt some innovation practices, specifically to gain a competitive differentiation within their specific market, and among their competitors. According to Armbruster *et al.* (2008), the organizations that present a specific focus on innovation can actually increase their market share and profits, which emphasize the important role that is developed by innovation within a particular organization.

Therefore, we can conclude that, to succeed in their innovation goals, top managers must focus on innovative products, since they deliver some value. Nevertheless, it is important to add that both the business and the innovation process must be divided into three distinct parts, more precisely: 1) the Fuzzy Front End (FFE), which is essentially a theoretical proposal where the pre-development activities are concentrated; 2) the development of a new product, which consists in a transformation of the theoretical proposal into a new product; and 3) the commercialization of the new product (Koen *et al.*, 2001).

It should be noted that the initial phase of the product's creation is quite determinant to the generation and selection of ideas, considering that the dimension of resources, time, costs, deadlines and the process quality is generated by the FFE. Furthermore, and to define the main components of this FFE, as well as to create a common language, it is essential to use the model that was proposed by Peter Koen, namely "The New Concept Development" (NCD) model, which basically follows five key elements, particularly: 1) Identification of opportunities; 2) Opportunity analysis; 3) Generation and enrichment of ideas; 4) Selection of ideas; and 5) Definition of the concept.

Regarding the first key element, it is essentially through the identification of opportunities that the organization, either by request or by research, identifies the main goals to be achieved. By request, some industrial organizations know that reliability and safety of industrial machines depend on their timely maintenance (Ardichvili, Cardozo & Ray, 2003). Hence, the ADIRA company, where the Pilot will be tested, identified the need to satisfy its customers, as well as other potential ones, more precisely through the creation of a Pilot.

Such need arises from the integration of the maintenance process, which allows a continuous monitoring of the machine and the application of advanced technologies for the preventive and proactive maintenance of the machine, specifically to create conditions for results improvement in several organizations, which makes them more competitive.

At the moment, the number of companies that have solutions to solve this particular problem is quite small, considering that they do not present several of the functionalities that are predicted within the project. However, and since there are still some companies that do not use effective mechanisms in order to solve this problem, they will become potential stakeholders (Ardichvili, Cardozo & Ray, 2003).

Hence, the technique that is used to analyze this particular key element is based on the analysis of the customers' trends, allowing to conclude that some companies are truly interested in an industrial maintenance solution, since it provides a continuous monitoring that ultimately improves their own results. Nonetheless, the remaining problem is that those companies do not know how to effectively perform it or do not have the financial resources to implement partial solutions that may solve such problem (Koen *et al.*, 2001).

In terms of the second key element, it is important to firstly mention that additional information is needed in order to translate the identification of business opportunities. In this particular context, a solution was found, considering that it actually met the needs of companies in general. The method that was used was actually based on the evaluation of stakeholders by the organization, ultimately selecting potential stakeholders and determining their main needs that still had not been reached by the existing products. The competition was also analyzed, thus allowing the identification of the main competitors that act within the area of industrial maintenance, as well as the determination of the product that is necessary to provide a competitive advantage (Koen *et al.*, 2001).

The third key element, related to the generation and to the enrichment of ideas, is associated with the evolution of a particular idea, namely from its birth until its own realization. Hence, it is essentially an evolutionary process, due to the fact that an idea is built through several combinations, readjustments and remodeling. In turn, to enrich the idea the company must follow an iterative process, discussing it with both the stakeholders and the future users (Bhuiyan, 2011).

When the functional prototype to be developed is proposed by the stakeholders, the idea to develop a certain performance evaluation tool, specifically by objectives, is already defined and ready to be realized. Hence, all the prototype requirements were developed through a study of several industrial maintenance methods, as well as by the application of distinct and advanced techniques for the preventive and proactive maintenance of the machines. Finally, it should be noted that the proposed solution is strictly subjected to this particular design, since the main goal is to improve and experiment the solution before proceeding to its implementation.

After generating and enriching the ideas, it is necessary to proceed to their selection, which is the fourth key element. One of the main difficulties of this particular process is related to the fact that the ideas to be selected need to possess a business value in order to be viable (Hoyer *et al.*, 2010). As it was previously mentioned, the functional prototype was requested by the stakeholder, which validates the selected idea in terms of viability and business value. At last, and regarding the requirements of the prototype, we selected a particular set that includes the most relevant ones to achieve the proposed objectives.

Finally, the fifth and final key element of the model is the definition of the concept, which involves the development of a business case. Essentially, this business case is based on several factors, such as the investment requirements, the estimates of market research, the customer needs and the competitive assessment. Therefore, its level of formality is related to the own pertinence of the opportunity (new technology and market existence). To analyze this particular key element, it is important to adopt a technique that allows an early participation of the customer(s) in the testing of the product, since it allows to evaluate if the developed solution actually satisfies the stakeholders needs (Edvardsson & Olsson, 1996).

4.3 Value offer

The value proposition is characterized by a particular set of products (tangible) or services (intangible), which create value to a specific segment of stakeholders. Nonetheless, it should be noted that the value of a product/service is only achieved when it actually satisfies the stakeholders needs (Clarke III, 2001).

According to the stakeholders perspective, the value is defined as the difference between the costs and the benefits of a product/service. Thus, the value is positive when the performance (benefits) is higher than the total cost (financial costs) (Woodruff, 1997). Nevertheless, the value is distinct for different stakeholders, namely due to the fact that they all present different perspectives, which corroborates the discrepancy in the value of the same products/services. As a matter of fact, the perceived value actually presents two distinct aspects, since it has a meaning for the producer and a complete different one for the customer: the first considers the value in terms of organizational costs and of business possibilities, while the latter regards aspects such as costs and sacrifices, being this last one translated into benefits when the product/service meets his needs (Sweeney & Soutar, 2001).

Overall, the value analysis aims to evaluate how it is possible to increase the value of a product/service without sacrificing quality and at the lowest cost. Zeithaml (1988) argues that

the products' development should be a continuous and interactive learning process, focusing on customer value, considering that stakeholders are the most valuable asset, creating business value for every organization.

In this particular project, we develop a solution that integrates the process of industrial performance, namely to perform a continuous monitoring of machines. Therefore, this system will be characterized, among others, by the application of advanced techniques. Despite being linked to the industry, where it will actually be disclosed, the solution will be easily adopted by other sectors inside organizations, more precisely by the construction, commerce, tourism and transport. The adoption of this tool will be of particular interest to organizations, since it will significantly contribute to the increase of their competitiveness and to the solutions that are phased in IoT.

Furthermore, it is crucial to mention that there are some types of value that are truly related to the implementation of the proposed solution, more specifically those that are related to are innovation, modularity, decentralization, Interoperability and standardization, performance (Real-Time), safety, integrity, confidence and price (Smith & Colgate, 2007).

Table 4 - Representation of the benefits and sacrifices of the proposed value

Domain/ Scope	Product/ Services	Relationship
Benefit	Innovation Performance (Real-Time) Modularity Virtualization Interoperability & Standardization Decentralization Service Orientation	Integrity Safety (Security & Privacy) Confidence
Sacrifice	Quality/ Price	Effort/ Time Conflict

Through Table 4, the time of effort associated with the creation and use of the product and the research actually stand out as sacrifices. Conflict has also been identified as a sacrifice, given the fact that, despite the possibility of coexistence with existing platforms in the market, there may be some friction to change by users who already resorted to other solutions.

Nonetheless, it should not be forgotten that quality and price have been identified as sacrifices, since the solution to the problem in question truly aims to be an industrial environment tool.

Increasingly, the industry presents the need for IoT, given the fact that companies need to address in order to expand the adoption in the following fields:

- 1) Recreate templates. Companies will have to redesign their organizations, partnerships and operations. For example, agrochemical companies will have to collaborate with software vendors, climate data providers and satellite operators in order to improve crop yields in specific locations and conditions. Furthermore, manufacturers can also decentralize operations, as technologies such as 3D printing enable products to be produced closer to stakeholders;
- 2) Capitalize on the value of the data. This includes establishing interoperability and security standards to ensure that data is shared with confidence among enterprises. New financial models will also be needed to support pay-per-use and other service-based offerings;
- 3) Prepare for the future work. With more access to data, decentralized work environments will be essential to support front-line employee decision making. New organizational structures will also be needed, more precisely to enable workers to collaborate in a more creative manner with their own peers in partner companies (Baldassarre *et al.*, 2010);

Given the fact that the solution consists of a system of information in the cloud, it is crucial to guarantee its security and integrity, due to the fact that it contains confidential and critical data (of employees of an organization – data integrity).

4.4 Value of the Current Market

Nowadays, companies need to acquire certain technology capabilities to be able to offer Proactive Monitoring and Maintenance (PMM) products and/or services, which means that they need to invest in technical, human and financial resources. Hence, and to achieve such purpose, they can follow or implement different strategies, namely: in-house development, diverse collaboration and partnership efforts and technology vendor acquisitions. As a matter of fact, PMM implies the addition of intelligence and connectivity to the end product, requiring the promotion of service based business models (Lu, Morris & Frechette, 2016).

According to some previsions, the global operational predictive maintenance market will actually grow at a CAGR (Compound Annual Growth Rate) of 26.6% between 2016-2022, foreseeing a total market value of EUR2.900 million by the end of that period. Nonetheless, such development will be boosted by the IIoT market rise, which is growing at a CAGR of

42%, acting as an enabler for its rapid industrial penetration. One key sector where predictive maintenance will have a huge impact is manufacturing. As a matter of fact, the European manufacturing sector accounts for 2 million companies and about 33 million jobs, thus representing 15% of the total EU GDP. Thus, and to increase this particular contribution to 20% by 2020, European manufacturing industry faces a significant challenge (Brisk Insights, 2016).

It is important to emphasize that competitiveness goals will be achieved through a full digitization of the European industrial ecosystems. Within such framework, the predictive maintenance will account for a huge improvement potential to all actors, more precisely: relevant productivity increase (users and asset), new revenue streams with higher profit margins (asset manufacturers) and also new business opportunities that are based on analytics (asset service providers). Finally, it is argued that predictive maintenance in factories within the industrial sector could reduce maintenance costs by 10 to 40 percent, which leads the manufacturer's savings of 215 to 580 billion euros in 2025, resulting from reduced downtimes and minimized manufacturing defects among others (Lu, Morris & Frechette, 2016).

4.5 Business Model Canvas

Some questions need to be answered in order to realize any business idea, such as:

- Who will be our clients (stakeholders)?
- What is valued by them?
- How do we get to them?
- What skills are required?
- What kind of partners should we have?

To answer these questions, and in a structured manner, the recurrence to the Canvas business model (Figure 12) is quite useful, since it is an important tool to define the Business Model of any single project. This particular model is divided into nine different blocks, in an integrated and visual way, focusing on several aspects, namely key partners, key activities, value proposition, customer relationships, customer segments, key resources, channels, cost structure and revenue streams (Barquet *et al.*, 2013).

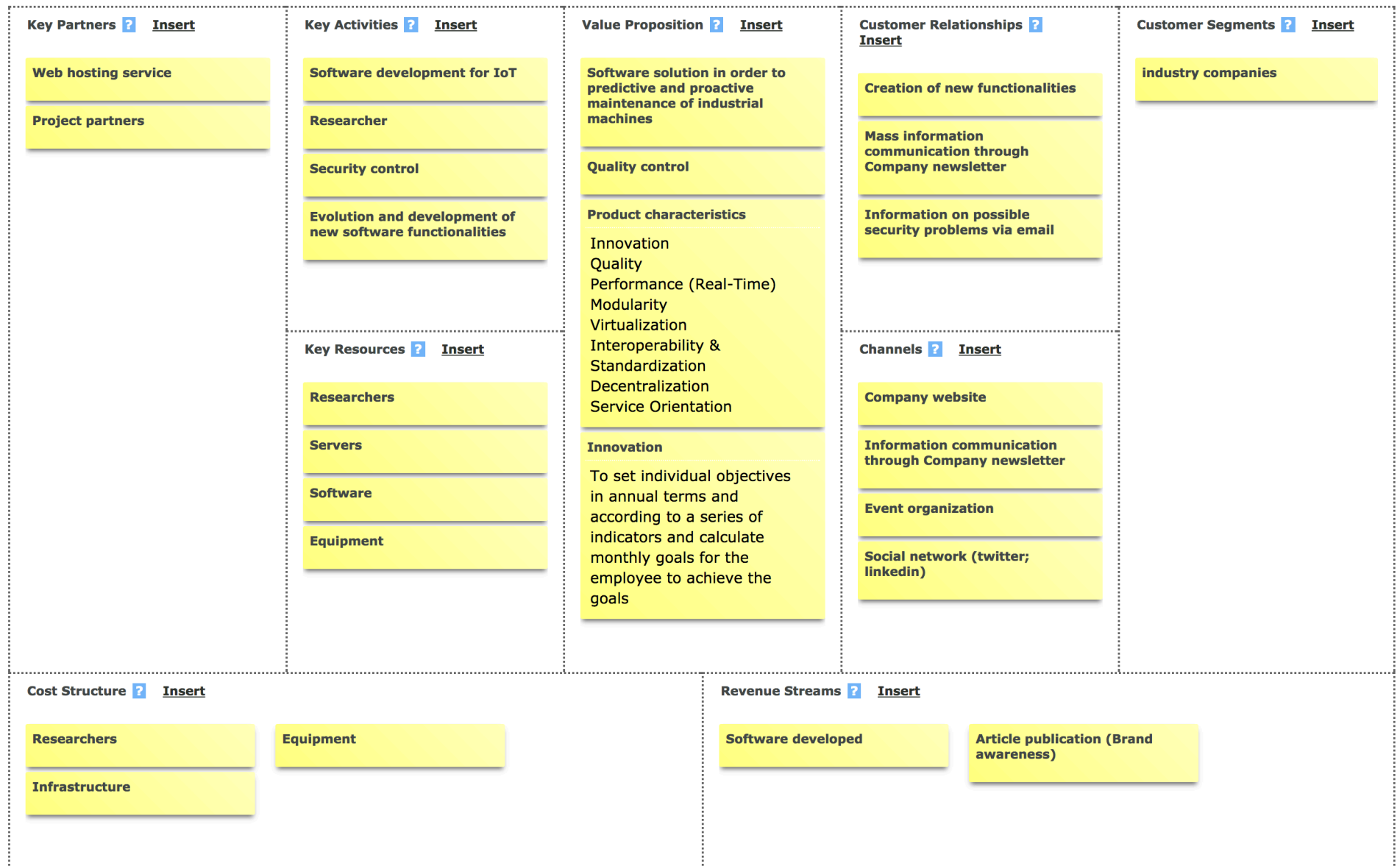


Figure 12 - Canvas business model elaborated for the current project

Key Partners, who provide services, identify the Web hosting services that, in turn, provide services in the cloud, which are necessary to fulfill the project. It is also important to refer that the Project Partners are essential to the solution, more precisely the **HMI** component and Data Analysis (Trimi & Berbegal-Mirabent, 2012).

Regarding the Key Activities, and of this particular model, they are associated with the area of Software development for IoT and to the evolution and development of new software functionalities. The Researcher is fundamental to the creation of innovative solutions and to the validation of the existing ones, as well as to the security control, aiming to enhance the final quality of the product and responding, at the same time, to the project's needs (Osterwalder *et al.*, 2014).

The Key Resources to this model's implementation are the servers, the use of softwares, specifically of software development tools, the developed software itself and all the equipment that is provided to this department of the company (Barquet *et al.*, 2013). The distribution channels for the product dissemination are the company's website and newsletters, event organization and social networks.

The customer relationships proceed to its establishment by the creation of new features or improvements, more specifically through suggestions on that basis, as well as through mass communication with stakeholders and email about possible security problems. The Customer Segments that represent the proposed potential solution are mainly related to industrial companies that actually want an IoT software solution (Meertens *et al.*, 2012).

At last, Value Proposition defines Software solution in order to obtain a predictive and proactive maintenance of industrial machines, thus proposing an innovative solution with quality control, which is inherent in the software development process. Product characteristics such as quality, performance, modularity, standardization, virtualization, interoperability, decentralization, service orientation and security are focused on creating a solution that actually follows Industry's 4.0 Design Principles (Osterwalder *et al.*, 2014).

Therefore, the implementation of this model with respect to the software development process implies the quantification of the value creation through the result in applications that offer greater security, which also provides quality control, management and time benefits by all those that are involved. It is possible to prove what was previously argued, considering that this process is subjected to an evaluation.

4.6 Competition analysis

The Analytic Hierarchy Process (AHP) was applied in order to compare different solutions. This model was created by Thomas L. Saaty, in 1970, to validate both quantitative and qualitative characteristics. The main focus is to prove that the best communications layer related to the solution to create a specific design project (Saaty, 2008). The Figure 13 depicts a list of the important criteria that this system should have, as well as a list of alternatives. It is important to add that all criteria are evenly essential, what makes them quite easy to compare. Possibly, some criteria were not included, since our purpose is to thoroughly analyze all solutions, which include performance, stability, tools and technology aspects.

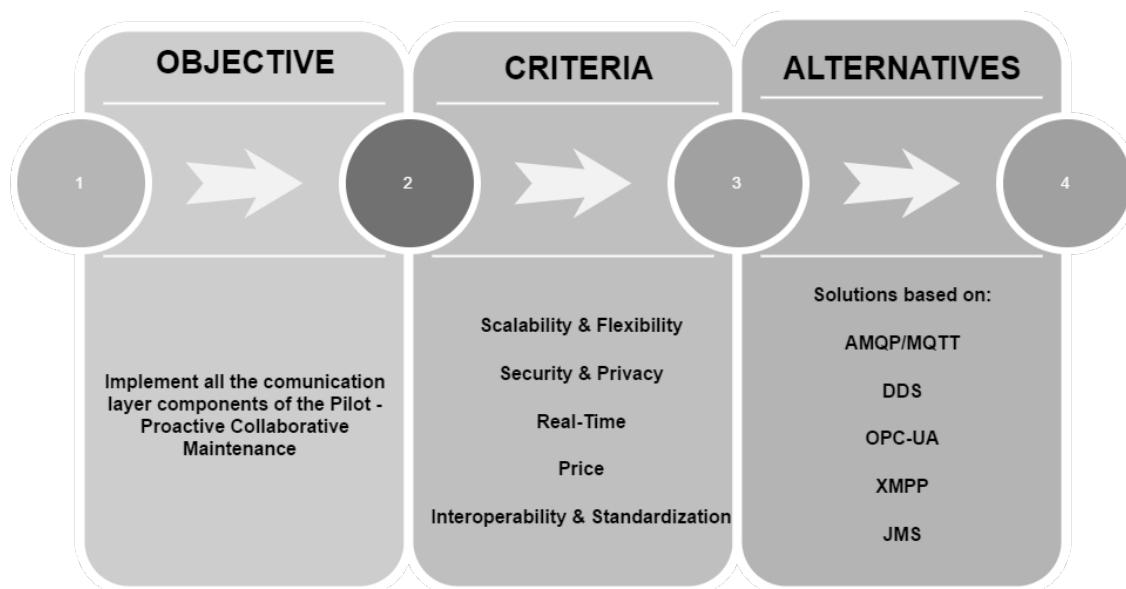


Figure 13 - Analytic Hierarchy Process (AHP) elaborated for the current project

Then, and using pairwise comparisons, the weight of each criteria is matched in order to demonstrate those that are more important in this context. In this case, the used scale goes from 1 to 9, 1 meaning equal, 3 moderate, 5 strong, 7 very strong and 9 extreme (Table 5).

Table 5 - AHP – Criteria Pairwise comparison

	Scalability & Flexibility	Security & Privacy	Real-Time	Price	Interoperability & Standardization
Scalability & Flexibility	1	2	1	2	1
Security & Privacy	1/2	1	1	1	1/2
Real-Time	1	2	1	2	1/2
Price	1/2	1	1/2	1	1/3
Interoperability & Standardization	2	2	2	3	1

To get a ranking of priorities from the pairwise comparison, it is necessary to normalize the matrix, squaring the same and then calculating the eigenvector (to 4 decimal places), summing up all rows and normalizing the rows by dividing the row sum by the row totals.

Such process is repeated until the eigenvector is the same as the previous iteration. The results are the following:

Table 6 - Criteria solution

Criteria	Score	Preference	Consistency (product of Score vs Criteria)	Consistency divided by Preference
Price	0,223	2	1,241	5,558
Compl. & Light	0,135	3	0,722	5,341
Flexibility	0,202	3	1,074	5,319
Security & Privacy	0,106	4	0,565	5,325
Scalability	0,334	1	1,773	5,315
λ_{max} (average)				5,372

The previous eigenvector gives the score for each criteria (Table 6), which allows us to conclude that the bigger is the score is the better is the result. In this case, Scalability is the most important criteria.

In order to prove that the values are consistent, the Consistency Index (CI) must be calculated:

$$CI = (\lambda_{max} - n) / (n - 1) \quad (1)$$

$$CI = (5,3718 - 5) / (5 - 1) = 0,093$$

According to Thomas Saaty's (2008) table, the Random Consistency Index is:

$$RI(5) = 1.12$$

Finally, the Consistency Ratio (CR) can be calculated:

$$CR = CI / RI \quad (2)$$

$$CR = 0,093 / 1.12 = 0.083$$

Since the CR, 0.083, is lower than 0.1, it can be assumed that the values of the weight of each criteria and the eigenvector are both consistent.

The previous process is repeated for all alternatives under each criteria, which results in an eigenvector for each alternative vs criteria.

Final result (AHP)

For the final result, a matrix with the eigenvectors from the previous step is squared with the eigenvector from the criteria (Table 7). Thus, is possible to conclude that the biggest value refers to the best choice.

Table 7 - Criteria with alternatives pairwise comparison

	Scalability & Flexibility	Security & Privacy	Real-Time	Price	Interoperability & Standardization
JMS	0,152	0,140	0,159	0,185	0,135
XMPP	0,224	0,218	0,203	0,192	0,168
OPC-UA	0,134	0,251	0,158	0,184	0,320
DDS	0,135	0,169	0,182	0,186	0,176
AMQP/MQTT	0,355	0,221	0,297	0,253	0,200
Criteria Eigenvector	0,223	0,135	0,202	0,106	0,333

Table 8 - AHP -Final Solution

Alternatives	Product with Criteria eigenvector	Preference
JMS	0,150	5
XMPP	0,197	3
OPC-UA	0,222	2
DDS	0,168	4
AMQP/MQTT	0,263	1

Therefore, the final result (Table 8) proves that the solutions that are based on “AMQP/MQTT” are the preferred ones under such criteria and alternatives.

4.7 Conclusions

All the stakeholder's concerns were identified regarding the business process and innovation following the Peter Koen's model. Therefore, the analysis of the offer defined the benefits and sacrifices of the proposed value for a specific segment of stakeholders. Besides, was defined the current market value proactive monitoring and maintenance (PMM) where the of the project solution is part of were provide. The business model was defined through a canvas business allowing to define the business idea. Finally, the competition analysis defined the communications layer related to the solution and prove through an analytic hierarchy process (AHP) and will be studied in more detail in previous chapter.

5 Analysis

This chapter presents the proposed architecture of this solution and describes the steps that were taken throughout the development process of the software solution that supports this thesis. It follows the Rational Unified Process (RUP), namely, identification of the system requirements, business modelling, and architectural patterns.

5.1 Proposed architecture

Software architecture development is an activity that refers to the organization of the software components of a particular system and how they communicate with each other. Such activity is essential, since any subsequent changes to the architecture later in the software development process may be difficult to realize, implying high costs either in time and resources (Tang *et al.*, 2010).

In this section, we establish a connection between the overview of the proposed architecture given in Chapter 1 and its representation in a more formal and detailed way (see Figure 14), specifically as a UML component diagram (see Figure 15)

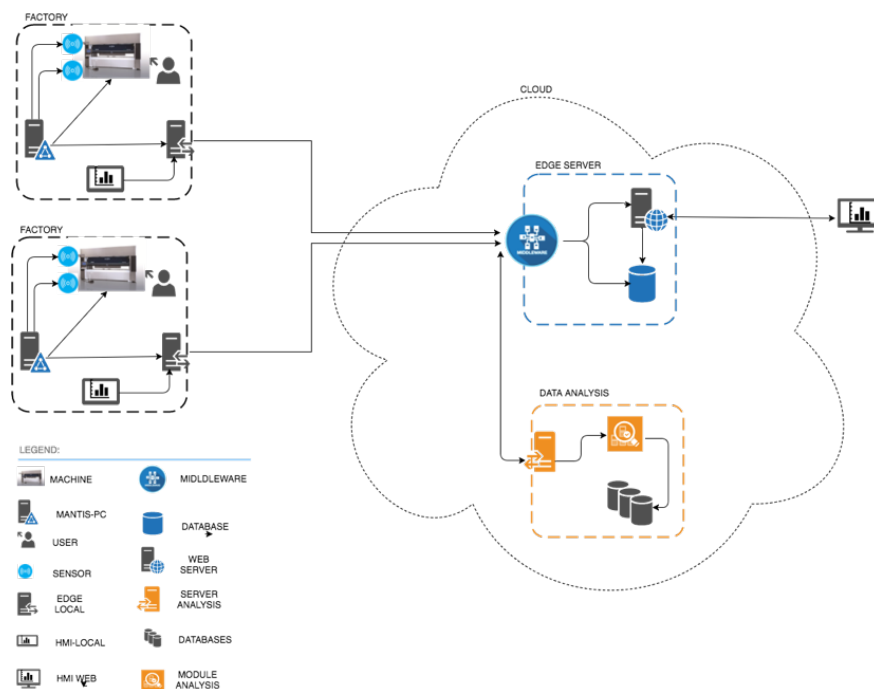


Figure 14 - MANTIS (ADIRA Pilot): Context Diagram for the proposed solution (equal to Figure 5 present in the Chapter 1)

The proposed architecture relies on seven different subsystems with a complex communication layer involving 7 different communication protocols, as described next:

- **Mantis-PC:** contains a *BLE server component* that communicates with the external sensors of the controlled **Machine** via BLE protocol and an *OPC-UA server component* that communicates with an *OPC-UA client* in the **Edge Local** subsystem via OPC-UA protocol.
- **Machine:** contains an *OPC-UA server* that communicates with an *OPC-UA client* in an **Edge Local** subsystem via OPC-UA protocol. Moreover, within the **Machine** subsystem there is a component named *CNCBender* (detailed in the Chapter 6) whose internal state is monitored by *OPC-UA server*.
- **Edge Local:** contains two *Node-Red* client components: (1) one that communicates with the *Middleware component* via AMQP, the *Middleware Client*; (2) one that may communicate with several **Machines** and several **Mantis-PC** components via *OPC-UA client*.
- **HMI:** contains a *Middleware-Web-Client component* that communicates with the *Middleware component* via STOMP (Websockets²⁷); a *Manager API component* (for simplicity in the figure and further in this report we use the term *API*) that communicates with the *Manager Component* existing in the **Edge Server** subsystem via HTTPS and the **Database** via TPC/IP protocol.
- **Data Analysis:** contains a *Middleware-Client component* that communicates with the *Middleware* in the **Edge Server** via AMQP protocol.
- **Edge Server:** contains a *Manager Component* that communicates with the *API component* of the **HMI** subsystem and *Middleware* via HTTPS protocol; a *History component* that communicates with a *Database* via TPC/IP protocol; a *Middleware component* that communicates with the **Data Analysis** and **Edge Local** subsystems via AMQP protocol; and finally, the *Middleware component* also communicates with the **HMI** subsystem via STOMP (Websockets).
- **Email:** this subsystem communicates with the *Manager Component* in the **Edge Server** subsystem via SMTP protocol.

²⁷ WebSockets is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request.

Figure 15 depicts all the subsystem and component interactions as described above.

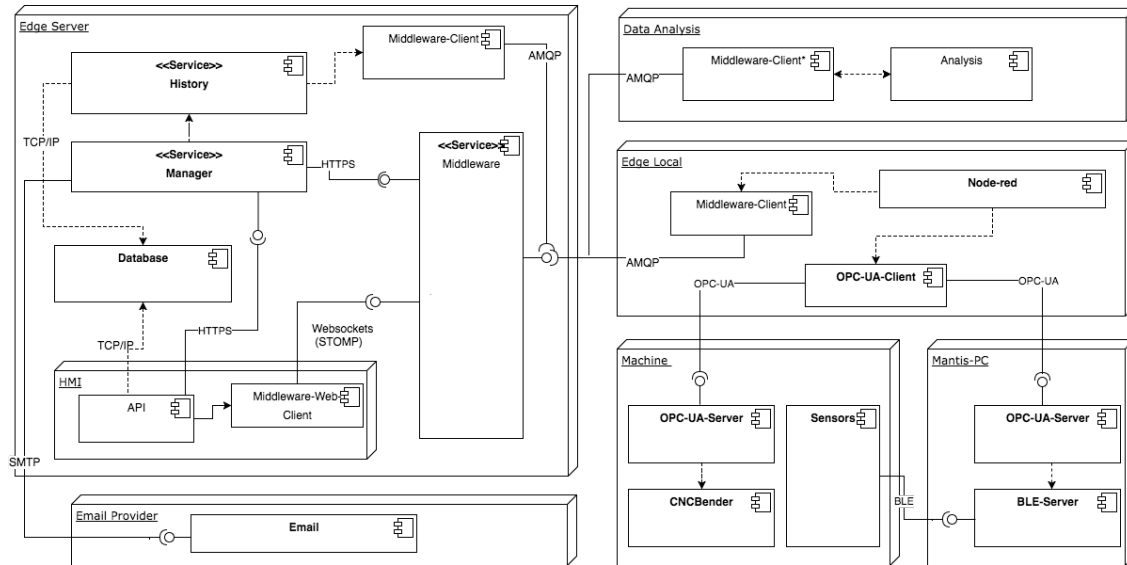


Figure 15 – Subsystem and component interaction for the proposed architecture

5.2 Requirements Engineering

Requirements engineering is a software engineering process that creates, analyzes, develops and maintains both functional and non-functional requirements that must be satisfied by a system in order to solve a given problem. Specifically, it aims at understanding and documenting the needs of the stakeholders in order to minimize the risk in the development of a certain software product. Thus, the requirements process started by extracting the requirements in the work package 1 (deliverable of the MANTIS platform requirements) of the MANTIS project (Jantunen et al., 2016).

In this subsection, the systems actors²⁸ are identified together with the mapping of functional requirements into use cases.

5.2.1 System actors

Each system actor has its own access profile which determines what features it is able to use. Moreover, each system actor can access two distinct levels, namely, the Local level (in Figure 16 the local level encompasses the two represented Factories) managed by the *OPC-*

²⁸ An actor of a system represents any entity (e.g., person, machine, external system, etc.) that interacts with the system under analysis (Sommerville, 2011).

UA servers at the **Machine** subsystem (within a factory there may be several *OPC-UA servers* with centralized permissions, where each server has the same configuration), and the Cloud level which is managed by the *Middleware* component. Each level has its own set of actors as described next.

The **Cloud actors** that were identified in this research work are the following:

- **Edge Server:** it is the *Middleware* component administrator being responsible for all the *Middleware* component parameterization (e.g., configuration, users, access permissions, etc.).
- **Data Analysis:** acts as a consumer for the *Middleware* component, being responsible for the consumption of the information of the queues. In fact, this actor is both a producer and a consumer of data.

Data Analysis and Edge Local: act as producers for the *Middleware* component through the usage of queues where the produced data is inserted. The **Local actors** that were identified are the following:

- **Edge-Local:** it is the entity responsible for the execution of the available methods and for receiving the information from one or more *OPC-UA servers* within the **Machine** and Mantis-PC subsystems.
- **Human:** it is the **Machine's** subsystem user and is responsible for controlling the **Machine** according to the information received from one or more *OPC-UA servers*.

5.2.2 Use Cases

Functional requirements correspond to the functionalities of a system. That is, they describe how the system should react (i.e., generate outputs) according to the given inputs. Moreover, functional requirements can be mapped into UML use cases in order to not only show system functionality but also to define system boundaries. Therefore, in order to present the functional requirements of the system, a use case diagram was elaborated, as depicted in Figure 16.

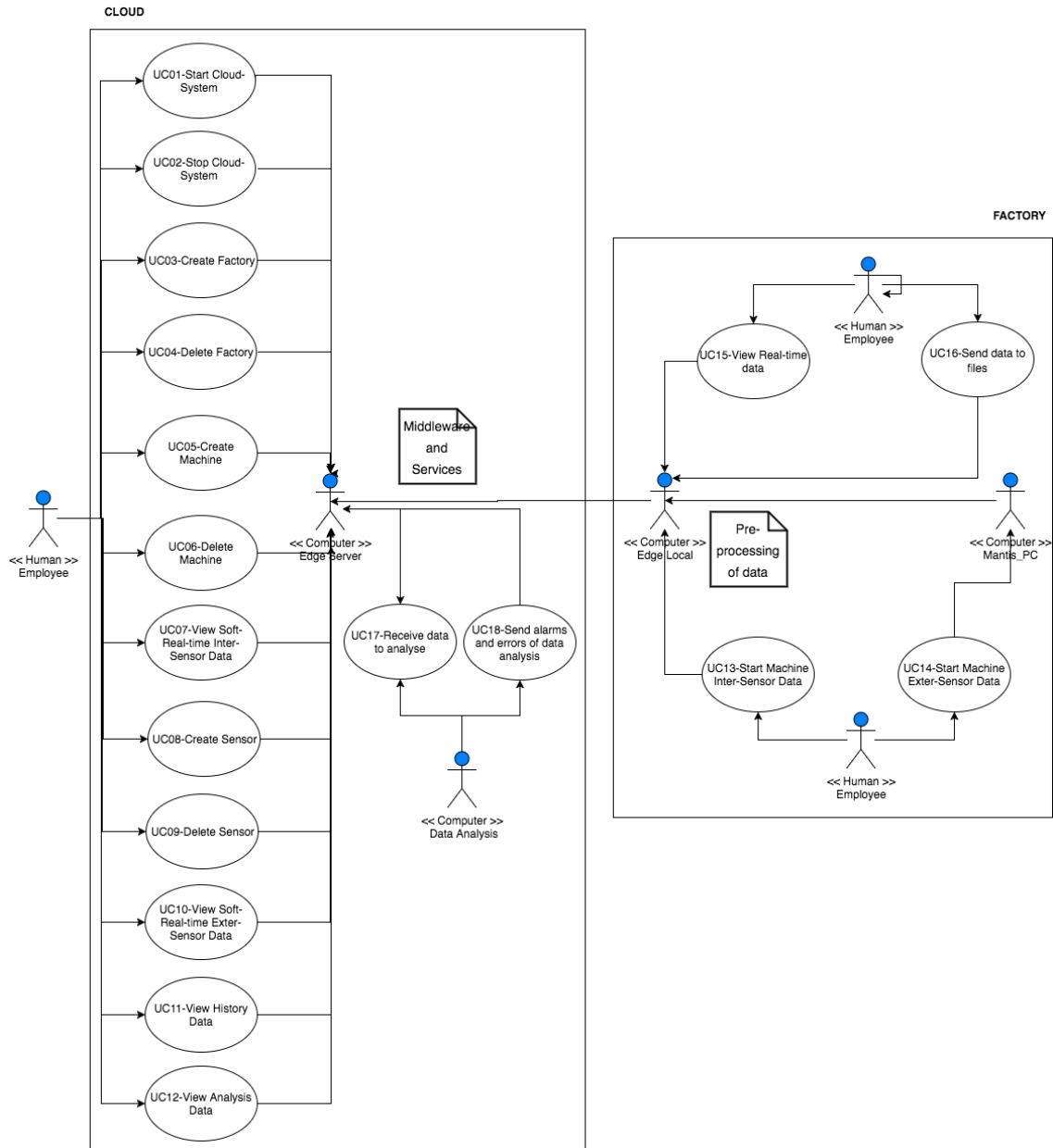


Figure 16 - Use Case Diagram for the proposed solution

According to Figure 16, there are 18 use cases divided in two distinct environments (as identified in the previous section): the factory environment including the Local actors at the Local level and the cloud environment including the Cloud actors at the Cloud level. Below, each of the use cases identified in Figure 16 is described in detail.

- UC01: Start Cloud-System:** A user accesses the system and initiates the Cloud-System using the HMI. This action configures the entire system (e.g., user, permissions, queues and history). If the system has already been started and there is configuration data in the database, this will trigger the actions of UC3, UC5 and UC8. The History system starts and configures the middleware queues according to the Data Analysis system.

- **UC02: Stop Cloud-System:** A user accesses the system and terminates the Cloud-System using the **HMI**. This action eliminates all basic system configurations (e.g., user, permissions, queues and history). In particular, this action does not trigger UC4, UC6 and UC9 actions, due to safety rules. However, the entire history system is deleted.
- **UC03: Create Factory:** A user accesses the system and creates a factory using the **HMI**. This action configures the system for that factory (e.g., user, permissions, queue and history). The system, when creating the user access permissions, sends an email with the user credentials.
- **UC04: Delete Factory:** A user accesses the system and deletes a factory using the **HMI**. This action erases the system settings for that factory (e.g. user, permissions, queue and history). It should be noted that all database content is preserved.
- **UC05: Create Machine:** A user accesses the system and adds a machine, using the **HMI**. This action configures the system for that machine (e.g. queues and history). Each machine can have up to two queues, one for internal sensors and another for the internal database.
- **UC06: Delete Machine:** A user accesses the system and deletes a machine using the **HMI**. This action erases the system settings for that machine in the middleware and the history service stops. The database data is maintained.
- **UC07: View Soft Real-time Machine Internal Sensor Data:** A user accesses the system and visualizes data from the machine's internal sensors using the **HMI**. This action generates a graph in real time using the selected sensors. A direct link is created between the **HMI** and the middleware. This use case depends on UC13, UC14, UC01, UC03 and UC05.
- **UC08: Create Sensor:** A user accesses the system and adds an external sensor using the **HMI**. This action configures the history system for this sensor.
- **UC09: Delete Sensor:** A user accesses the system and deletes an external sensor using the **HMI**. This action erases the history system for that sensor. The database data is maintained.
- **UC10: View Soft Real-time Machine External Sensor Data:** A user accesses the system and displays data from the machine's internal sensors using the **HMI**. This action provides a graph in real time using the selected sensors. A direct link is created between the **HMI** and the middleware. This use case depends on UC13, UC14, UC01, UC03 and UC05.

- **UC11: View History Data:** A user accesses the system and displays a machine's internal or external sensors' history data, using the **HMI**. This action provides a graph with data, within a defined time interval. The history service is responsible for inserting data into the database, data which is then depicted by the **HMI**. This use case depends on UC13, UC14, UC01, UC03 and UC05.
- **UC12: View Analysis Data:** A user accesses the system and views **Analysis Data** from a machine's internal or external sensor, using the **HMI**. This action provides a graph of the error or alarm detected by the **Data Analysis** subsystem. The history service is responsible for inserting data into the database, data which is then depicted by the **HMI**. This use case depends on UC13, UC14, UC01, UC03 and UC05.
- **UC13: Start Machine Internal Sensor Data:** A user accesses the machine and starts the software using the CNC. This action starts the machine software, which has the responsibility of reading the internal sensor values of the machine and sending them to the **Local Edge**.
- **UC14: Start Machine External Sensor Data:** A user accesses the **Mantis-PC** computer and initiates the external sensor software. This action initiates external sensor software, which is responsible for reading the values from the machine's external sensors and sends them to the Local Edge. All external sensors use wireless technology.
- **UC15: View Real-time data:** A user accesses the **Edge Local** computer. This action starts the software, which has the responsibility of connecting with the internal and external sensors of the machines and collecting data values, making them available in real time. This use case depends on UC13 and UC14.
- **UC16: Send data to files:** A user accesses the **Edge Local** computer and configures the software (delay, join, sensor, file type, etc.). This action starts the software, which has the responsibility of connecting with the internal and external sensors of the machines and collecting the values and adding the information to the files. This use case depends on UC13 and UC14.
- **UC17: Receive data to analyze:** A **Data Analysis** computer connects to the Middleware to receive data from internal and external sensors of the machines. This action starts the software that has the responsibility of analyzing all data and starting UC18, if necessary. This use case depends on UC13, UC14, UC01, UC03, UC05 and UC08.
- **UC18: Send alarms and errors of data analysis:** A Data Analysis computer connects to the Middleware to send data. This action initiates the software, which is responsible for

the errors or alarms related to the analyzed data of the internal and external sensors of the machines. This use case depends on UC13, UC14, UC01, UC03, UC05, UC08 and UC17.

As a side note, in this work we have also identified non-functional requirements as well. However, we opted to not cover them all in this work but to only focus on the most relevant categories into which they fit. Therefore, the most important non-functional requirements fit into the following categories:

- **Reliability:** refers to the system's integrity and conformity . The possibility of forecasting with accuracy the average time of failures, as well as their frequency and severity;
- **Performance:** evaluates the system's development requirements. Several aspects can be used as measurements, such as: response time, memory consumption and CPU utilization.

5.3 Business Modeling

Given the complexity of the problem under study, it is important to understand the problem domain and the main business entities that are part of it. Therefore, in a joint effort with the project partner UNINOVA (Instituto de Desenvolvimento de Novas Tecnologias), the Domain Model, Event, Semantic and Data models were created.

5.3.1 Domain Model

The domain model, also designated by a conceptual class model, is an artifact that is used in the business modeling discipline of RUP in order to decompose the problem domain into different concepts. Therefore, it represents the conceptual entities of the domain and their relations, thus allowing the definition of the domain's vocabulary (Fettke & Loos, 2007).

The domain model diagram of the proposed solution is presented in Figure 17. Below the entities of the domain model and their relations are described following a bottom-up approach. This model is based on IoT-A model²⁹ and has as its ambition the use of static

²⁹ The IoT-A model provides a reference architecture for the Internet of Things with applicability in industrial maintenance processes.

information through the transformation of the model Information that is represented by the semantic model of the Chapter 6.

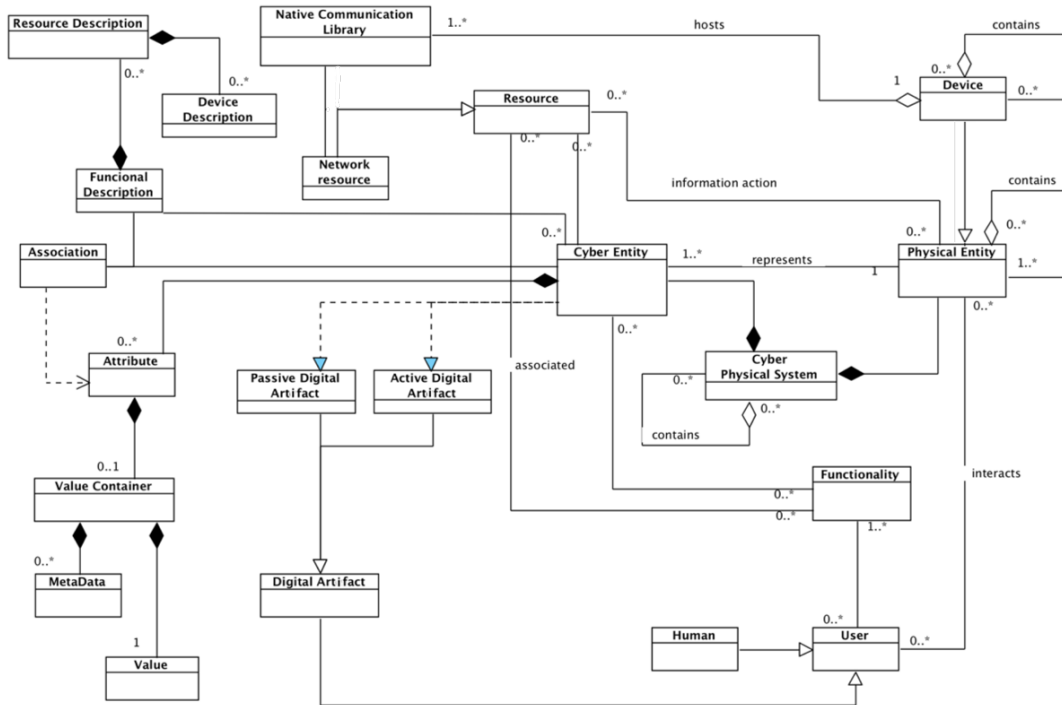


Figure 17 - Domain Model for the proposed solution (based on IoT-A model)

Let us start by defining the **Attribute** entity. Each Attribute has a specific name (attributeName), a semantic Digital type (attributeType) and one or more values (Value Container). The **Association** entity keeps the information about the relation that is established between **Cyber Entity** and the **Functional Description** entity, specifically through a Functional Description – Cyber Entity association.

Each **Value Container** entity groups a single **Value** and zero or more **Metadata** information units that belong to a given **Value**. Moreover, a **Value** is considered as a measurement unit of a particular datatype, for example, a certain value can represent the oil temperature of a give machine. The **Metadata** entity can be used in order to save the timestamp of the **Value**, as well as other quality parameters, such as accuracy or the measurement unit.

The **Functional Description** entity stores information about the functionality of the **Cyber Entity**. The **Cyber Entity** refers to an identifiable part of the physical environment that potentially could be used by users to accomplish their goals. The connection between an **Attribute** of a **Cyber Entity** and the **Functional Description** is only modeled through a specific **Association** (e.g., the Functionality could act as a “get” function for an **Attribute** value of the **Cyber Entity**).

The **Device Description** describes the **Native Communication Library**, which is exposed by the **Functionality**. Nonetheless, the **Resource Description** might contain some information regarding the **Physical Entity**, more precisely on which the **Resource** is hosted (e.g.: the **Physical Entity Description**).

The **User** entity is a quite transparent one, considering that it refers to a human person or to some kind of a **Digital Artifact** (such as a service, an application or a software agent) that interacts with a certain **Physical Entity**. The **Digital Artifact** entity can be active or passive. It is active when the artifact actively runs software applications, agents or services that may access some other services or resources, and it is passive when the **Digital Artifact** is essentially a passive software element, such as database or other digital representations of the **Cyber Entity**.

A **Cyber Entity** is the representation of **Physical Entities** existing in the digital world and/or in the cyber-space. These **Cyber Entities** present two fundamental properties, namely:

- 1) They may use the interfaces exposed by the **Digital Artifacts**, and through the **User** entity they may interact with one or more **Physical Entities**. On the other hand, **Physical Entities** are associated to different **Cyber Entities**. It is essential that each **Cyber Entity** has one and only one ID, in order to univocally identify each **Cyber Entity**;
- 2) **Cyber Entities** are synchronized representations of a particular set of properties of the **Physical Entity**. This means that all the relevant digital parameters that represent a **Physical Entity** are included in the **Cyber Entity**.

Therefore, the **Cyber-Physical System** entity represents the composition of **Cyber Entities** and **Physical Entities**. In other words, the **Cyber-Physical System** entity enables everyday objects and/or devices that interact with the real world to be included in the system under analysis.

The concept of **Resource** includes all entities that actuate over the **Physical Entities**. Given the fact that it is the **Functionality** entity that makes a **Resource** accessible, the relations established between **Resources** and **Cyber Entities** are modeled as associations between **Cyber Entities** and **Functionality**. Resources can be classified into:

- 1) **Network resource**: includes the resources that are available through a network, as for instance a database in a cloud system;
- 2) **Native Communication Library**: includes the resources that accesses sensor information.

The **Device** extends the **Physical Entity**, thus allowing the latter to be a part of the digital world. It provides the technological interface (Native Communication Library) to

interact with the **Physical Entity**. It is important to emphasize that **Devices** can be aggregations of several devices of different types.

Finally, and complementing what has been previously presented, the **Functionality** allows the user to indirectly interact with the **Physical Entity**. Therefore, it allows one to act on **Physical Entities** and/or to provide certain information about them. However, the user needs a client in order to access a particular functionality, such as a software with an accessible user interface.

5.3.2 Event model & Semantic model

This subsection focuses on the event and semantic models for one the **Machine** subsystem. Thus, the design of a generic event system which allows the registration of several event handlers, describes the event flow and provides basic contextual information for each event.

Thus, two standardized models were proposed in the project, namely the IoT-A model and the Mimosa³⁰. Mimosa model is oriented towards operations and maintenance in manufacturing. However, the Mimosa Model has not been designed for Internet of Things reason why it was not used. Although the adopted model is the IoT-A model the presented event models and semantic model that are part of this section provide a partial mapping with the Mimosa model. The diagram in Figure 18 specifies the proposed event model for the Cyber Entity identified in the previous section. In this instantiation of the model, the Cyber Entity represents the **Machine** subsystem identified in Subsection 1.1. In a machine's context, the machine itself generates events which are consumed by other entities in the system.

The relevant information related to each event is described below and depicted in the Figure 18.

³⁰ <http://www.mimosa.org/>

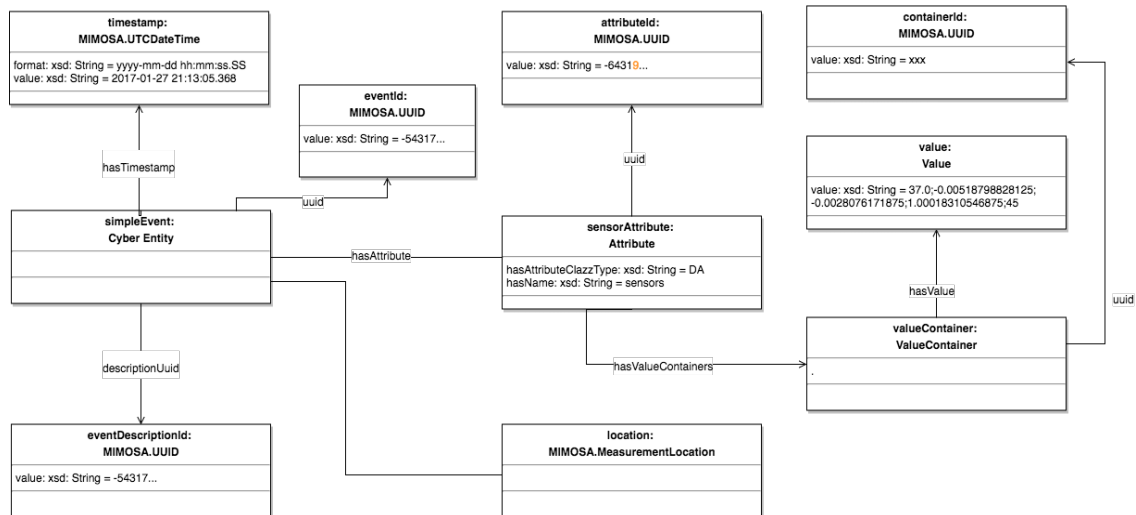


Figure 18 - Machine Event Model (based on Mimosa/IoT-A models)

Each event produced by a Cyber Entity has a timestamp, a unique identifier named eventId, a sensor attribute (e.g., name, type) that has a unique identifier, and a container that has a unique container identifier (containerId) and stores a data value (e.g., raw data from a sensor). There is also location (i.e., physical space where the Cyber Entity is located) and a unique identifier named eventDescriptionID that matches the static information with the event itself. Through the usage of static information the amount of data transferred through the cloud may be decreased.

Furthermore, the use of static information (Figure 19) allows the amount of data to be reduced. This technique because we have a large amounts of data with very short acquisitions.

Figure 19 depicts the metadata (static information) modelling relative to the machine event model Figure 18. In this way, the model refers to a sensor attribute that is part of a cyber entity (**Machine**) and has as metadata the name, type, etc. The container also has associated metadata and this metadata may have another associated metadata.

Finally, there should be metadata related with functional and resource descriptions which would be added in the configuration of this component by the **HMI**.

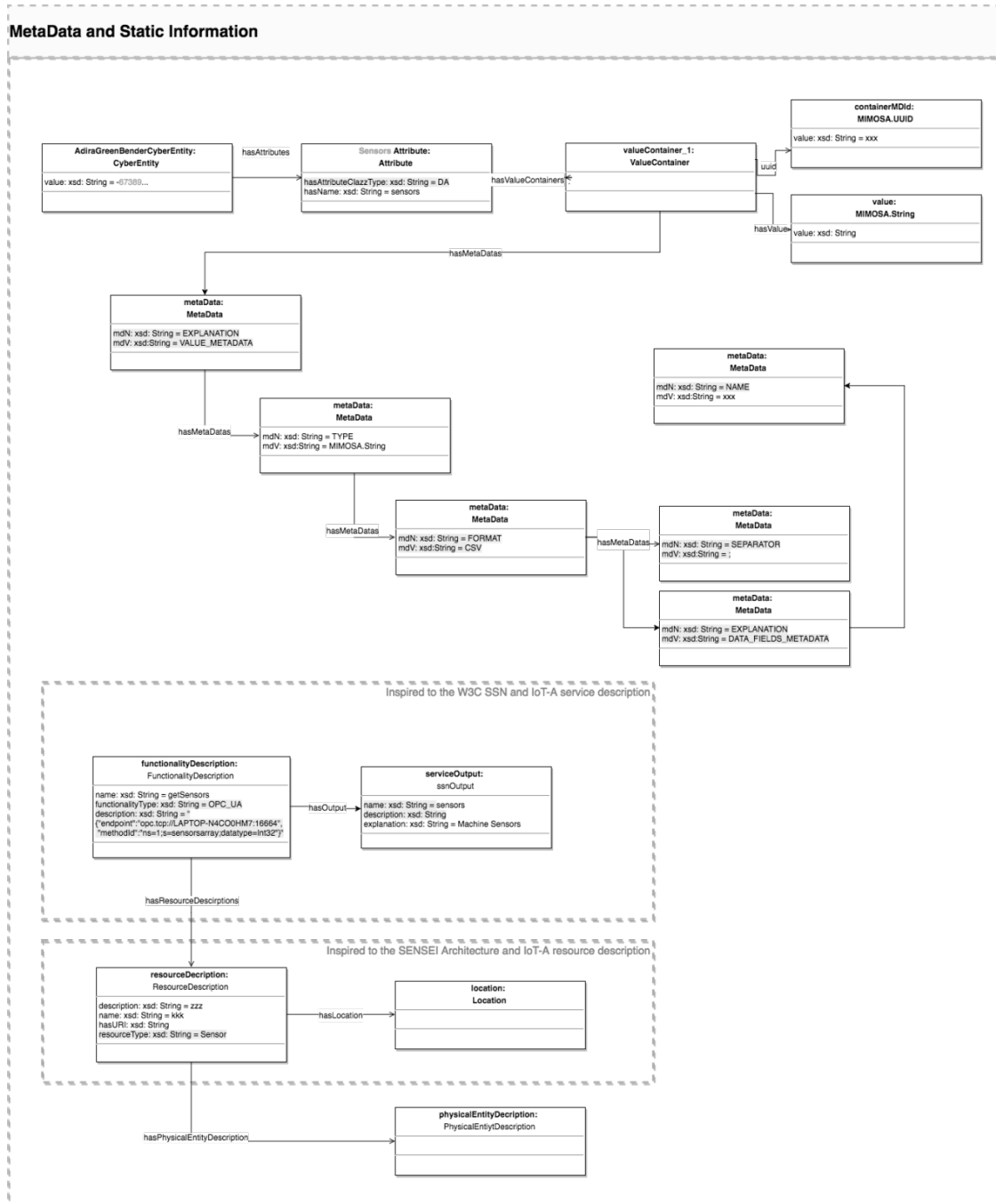


Figure 19 - Machine Sensors Semantic Model (based on Mimosa/IoT-A models)

5.3.3 Database

The proposed solution uses a database to store the system information. This way and following the SOA strategy, it is possible to choose the type of database according to the business needs. As for the different types of databases, we highlight the relational databases (the ones that use SQL as its main query language), the document databases (the ones that use

noSQL as its query language) and finally, the graph databases (there is no standard query language yet adopted by this type of databases).

In the scope of this thesis there was the need of selecting a database engine to be used by the several subsystems identified above. The process of selection that was used is based on the strengths of each database type. Table 9 presents proposed types of databases that could be adopted for the proposed solution.

Table 9 - Types of Databases

Subsystem – Information to store	Type of database
HMI - Authentication Database	Relational
Sensors (internal and external) - Metadata, Other relevant information	Documental
Data Analysis - Data Analysis Information	Relational or Graph

According to the context, there are more suitable databases than the relational-type. For instance, document-type databases are more suitable for content catalogues, since they enable faster readings. As for the graph-type ones, these are ideal to represent both the data and their connections. Hence, these are quite useful when constructing the mechanism that allows the access to detection, prediction and diagnosis models (Fong, Wong & Cheng, 2003).

Despite the proposed databases, all the partners involved in this project decided to use a single type of database in order to improve the development speed. This reason justifies the choice of a relational-type database, along with its Entity-Relationship model³¹.

Figure 20 presents the Entity-Relationship model that represents the Cyber-Physical System universe depicted in the domain model of subsection 5.3.1. As a side note, the structure of the **HMI** and **Data Analysis** subsystems is not contemplated in the proposed model below, as they are developed by other partners.

³¹ The Entity/Relationship model is constituted by several entities and by the relationships that are established between them.

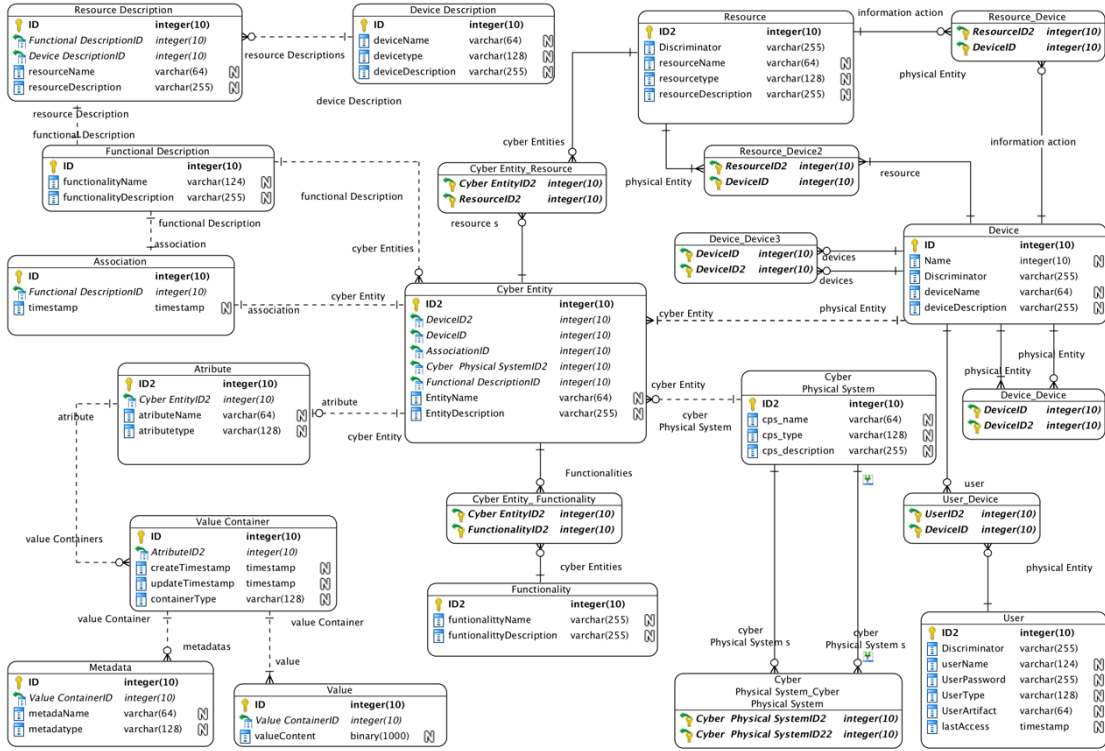


Figure 20 – Entity-Relationship model of the proposed solution

5.4 Design and architectural patterns

Throughout this thesis several architectural patterns were used in different parts of the system. The relevant patterns are identified next:

- **Single Responsibility Principle:** This pattern determines that each module or class must bear responsibility for a well-defined part of the features provided by the application. This pattern is applied in the development of all the system components.
- **N-Layer:** This pattern separates the responsibilities in different layers. This pattern is applied in the development of the History and *Manager Components*.
- **Inversion Control (or Dependency Injection):** This pattern allows the creation of a modular application, which facilitates testing. This pattern is applied in the development of the *Manager Component*.
- **Broker pattern:** This pattern provides access to all the distributed services, through the high-level message exchange between software objects. This pattern is applied in the *Middleware component*.

5.5 Conclusions

The proposed analysis for the problem under study was presented in this chapter by establishing a connection between the proposed architecture, system actors and system components. The subsystems, components and communication protocols were identified and described. Moreover, the business model for the problem under study was presented, including the main business entities.

Next chapter details the design and implementation of the entities identified in this chapter.

6 Design & Implementation

In RUP, after the Requirements Engineering phase there is the discipline of Analysis and Design. In the previous chapter the Analysis was covered. In this chapter, we present the adopted design decisions, the used patterns and rules, followed by a discussion of the design alternatives. Furthermore, in RUP, the implementation phase of the life cycle of a software product follows the Analysis phase. Hence, in this chapter we also cover the implementation details, including the hardware/software requirements of the installation, examples of the component's instantiation and their functionalities.

All of its components followed the principle of single responsibility, more precisely to avoid the attribution of several responsibilities to a single class and the unwanted coupling and resistance to change when introducing those new attributes.

In the following subsections, each subsystem and its components are described in detail (please refer to Figures 14 and 15 for a visual guide on the components diagram) and ends with the deployment view of the solution. Since implementation and design cover many components of each subsystem that have been integrated into work packages deliverables of the MANTIS project, only the most relevant details are presented.

6.1 Machine Subsystem

This subsystem depicted in Figure 21 is composed by several components, such as the OPC-UA-Server that communicates with the OPC-UA-Client in an Edge Local subsystem via OPC-UA protocol. Moreover, within the Machine subsystem there is a component named CNCBender which is monitored by OPC-UA-Server. Furthermore, ADIRA request a preliminary solution for the Machine subsystem and a description of the implementation is available in Appendix B. The preliminary solution provides limited functionalities that are covered by the solution of this thesis.

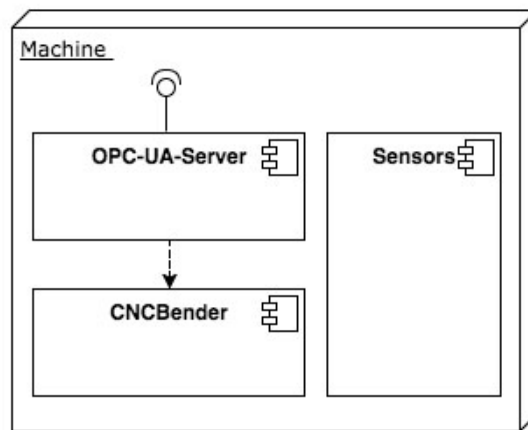


Figure 21 - Machine Subsystem

6.1.1 OPC-UA-Server Component

The class diagram of the OPC-UA-Server component is presented in Appendix-E. The base class is the Server class which contains information about credentials and network parameters. This base class incorporates elements of the File and Menu classes. The ShareMemory class is not represented in the class diagram since it was developed in partnership with an employee of the ADIRA company.

The use of a server by a user triggers a complex sequence of operations, more precisely, the ones related to the reading of several parameters that are written within the files' properties and to the storing of those same parameters in the memory (Figure 22).

Furthermore, this functionality is responsible for the integration of the Information Model within the Server Address Space, as well as for the static writing of this same model, more precisely by reading the shared memory of the log files.

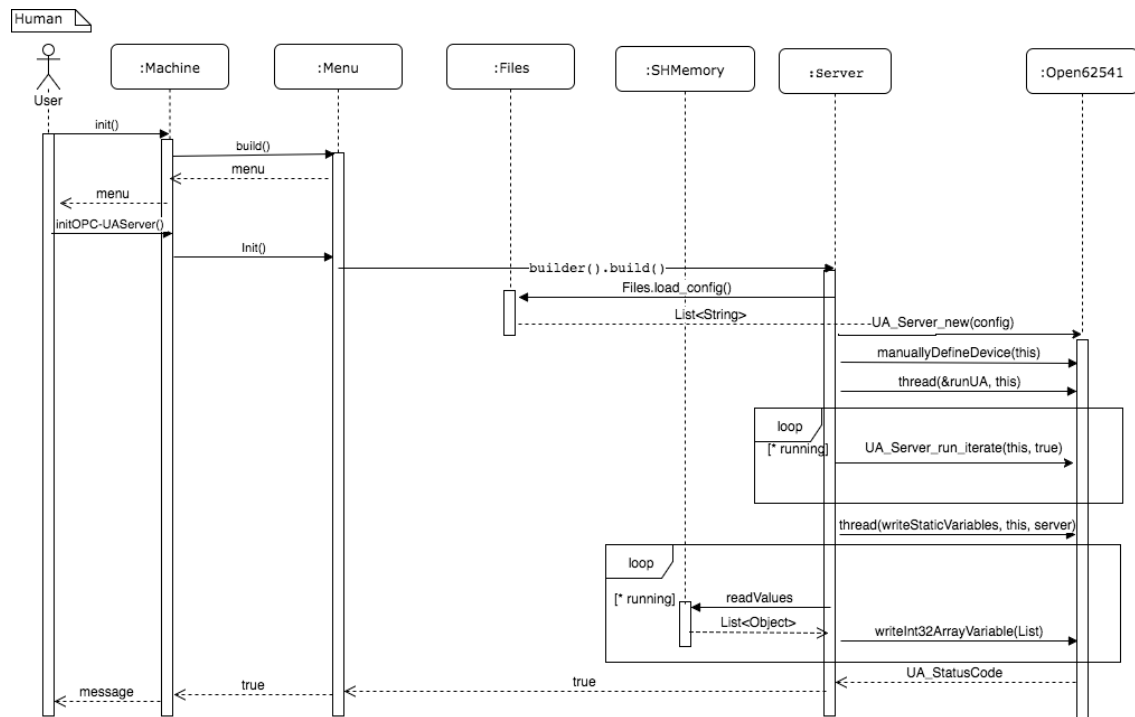


Figure 22 - UML sequence diagram that presents the initialization of the OPC-UA server

Furthermore, this specific component uses the primitive Data Types of the OPC-UA protocol, namely the UA_TYPES_STRING, the UA_TYPES_INT32, the UA_FALSE or the UA_TRUE and the UA-DATETIME, thus not being necessary to create specific object types. Regarding its connection, it is established by TCP/IP, by using a Secure Channel and a Session with the same structure of the OPC-UA binary messages.

The graphic interface (Figure 23) of the *OPC-UA servers* provides the following functionalities:

- 1) Start Server
- 2) Stop Server
- 3) Connected Clients
- 4) Access Credentials
- 5) User Manual
- 6) Development Guide
- 7) Copyright

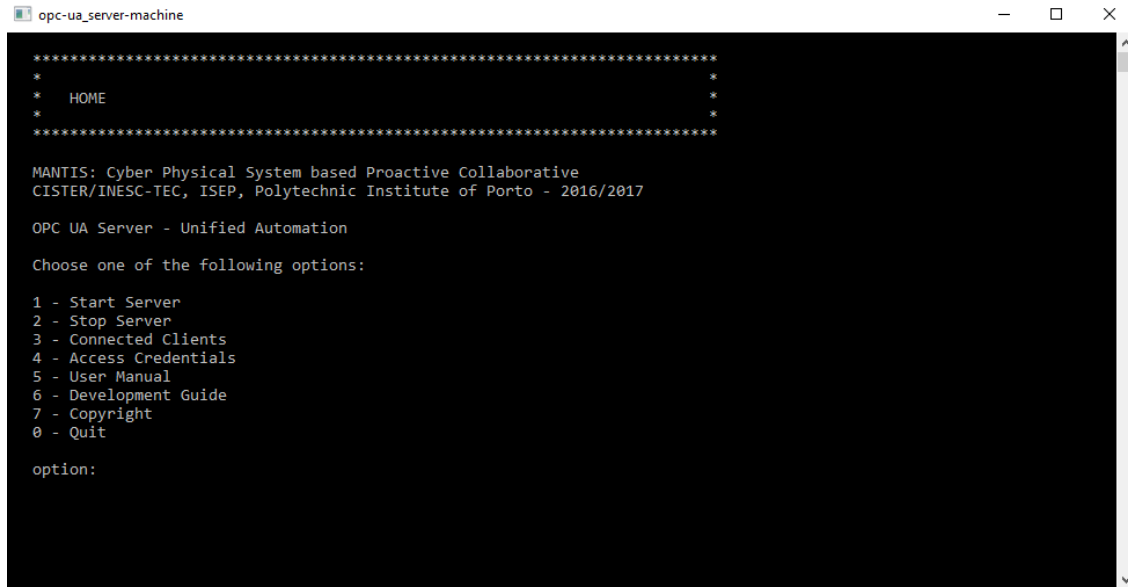
A screenshot of a terminal window titled 'opc-ua_server-machine'. The terminal displays a menu for the OPC UA Server - Unified Automation. At the top, it says '*****' followed by '* HOME *' and another '*****'. Below this, it reads 'MANTIS: Cyber Physical System based Proactive Collaborative CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto - 2016/2017'. Then, 'OPC UA Server - Unified Automation' is displayed. A prompt 'Choose one of the following options:' is followed by a numbered list: '1 - Start Server', '2 - Stop Server', '3 - Connected Clients', '4 - Access Credentials', '5 - User Manual', '6 - Development Guide', '7 - Copyright', and '0 - Quit'. At the bottom, there is a prompt 'option:'.

Figure 23 - User Interface of OPC-UA server

The implementation used the C++ Programming Language, which created an application project that used, in turn, the NetBeans IDE. It is essential to add that such application project was developed in the Windows 10 operating system, more precisely by using the Cygwin10 compiler, which targets the Windows XP operating system to be complaint with CNC Bender Component.

Furthermore, this project also used some external libraries, specifically Boost and Cppunit, which were installed via Cygwin (therefore, they must be compiled in the target Operating System). Even though the *OPC-UA* implementation (open62541³²) was developed in the C language, there were no compatibility issues in the integration of the project.

It is quite important to add that this component uses a configuration file (config.ini) which defines several parameters of the software, specifically files, connection, settings and variables (Figure 24).

³² <https://open62541.org/>


```

[files]
dir=files
tmpdir=files\tmp
currentFile=12052017.log
currentLine=26265
[connection]
port=16664
username=adira
password=mantis
[settings]
machineId=M0000017
dataAccess=files
period=1000
[vars]
var1type=UA_DateTime
var1isArray=false
var1Name=timestamp
var2type= UA_ DATETIME
var2isArray=false
var2Name=machineId
var3type=UA _INT32
var3isArray=true
var3Name=sensorsarrav

```

Figure 24 - OPC-UA server Component - Config File

All the data were modelled according to the OPC-UA standard, thus resulting in the following information model³³, which was, in turn, modelled in the Address Space Server³⁴:

1) Device root

- a. machineId - *UA _STRING*
- b. timestamp - *UA _DATETIME*
- c. sensorsarray - *UA _INT32 (array)*

As soon as it starts, this software defines the structure of the information model and loads the values for that same structure, providing default values for the timestamp and sensorsarray of the Share Memory and of the Log files. Furthermore, this software also uses a thread, to execute and load all the data in a loop (which is defined in the config.ini file), as it is shown in Figure 25.

³³ https://open62541.org/doc/0.2/information_modelling.html

³⁴ <https://open62541.org/doc/current/types.html>

```

/**
 * Starts the UA_Server with the appropriate config,
 * executes the manually Define Device information model
 * and a thread that writes Static Variables
 * @return UA_StatusCode
 */
UA_StatusCode Server::start_ua_server() {
    UA_StatusCode retval;
    this->running = true;
    try {
        signal(SIGINT, stopHandler); /* catches ctrl-c */
        this->n1 = UA_ServerNetworkLayerTCP(UA_ConnectionConfig_standard, this->port);
        config = UA_ServerConfig_standard;
        config.networkLayers = &this->n1; // NetworkLayer TCP
        config.networkLayersSize = 1;
        // defines the user authentication
        config.enableUsernamePasswordLogin = UA_TRUE;
        config.usernamePasswordLogins->username = UA_STRING(this->username);
        config.usernamePasswordLogins->password = UA_STRING(this->password);
        config.enableAnonymousLogin = UA_FALSE;

        this->_server = UA_Server_new(config);
        this->manuallyDefineDevice(this->_server);
        // thread that handle the static variables of the infoemation model
        this->thread_server_writeSV = new boost::thread(&Server::writeStaticVariables,
        this, this->_server);
        this->thread_server = new boost::thread(&Server::runUA, this);

    } catch (const std::exception& e) {
        cout << "\n\n  Error Server::start  UA server " << e.what() << endl;
        this->running = false;
        return retval;
    }
    return retval;
}

```

Figure 25 - OPC-UA server Component - UA_SERVER Start

6.1.2 CNC Bender Component

Regarding this component's responsibility, it has been previously described, in the **Machine** Subsystem's section 6.1. Nonetheless, it is important to specify that it uses a Windows XP operating system, which no longer is supported. Security is a very relevant factor and the *OPC-UA server* Component uses an authentication method to restrict access on the exposed data.

This component uses the AdControl50 version. Afterwards, which were collected about this particular software, considering that they conditioned the choice of the used technology in the *OPC-UA server* Component. We can easily verify that the Operating System is discontinued, bringing security problems, and that the hardware (processor, memory and hard disk) actually influences the performance.

6.1.3 Sensors Component

The **Sensor** Component is constituted by two accelerometers, which are essentially off-the-shelf sensors, monitoring the blade that actually performs the bending of the metal sheet. The **Machine** Subsystem is where the two sensors are highlighted, considering that the sensors collect the data from the own movement of the blade in the press, especially from the vibration patterns that are caused by the hydraulics. In fact, and given the fact that the vibratory pattern can be associated to the condition of the machine's bending motors, the collected data can be used in order to perform PM of the machine.

For instance, if the hydraulic pistons start malfunctioning, possibly due to the existence of some particles in the oil, then a different vibration pattern can be detected. Nonetheless, these same sensors can be placed on a different machine location, for example for malfunction diagnosis.

One of the project's requirements was related to the use of specific sensors with Bluetooth 4.0 in order to measure both the noise and the acceleration. Therefore, after analysing and testing the available hardware, the Micaz and TelosB sensors using the Tinyos operating system, we concluded that they did not meet the mentioned requirements due to battery consumption problems.

Hence, we selected the new Arduino 101 based on its specifications, such as its low energy consumption, its reasonable CPU and its memory capabilities.

The specifications of Bluetooth 4.0 introduce the Bluetooth Low Energy (BLE). BLE is optimized for low power use at low data rates, and was designed to operate from simple lithium coin cell batteries.

The Bluetooth LE protocol operates on multiple layers. General Attribute Profile (GATT) is the layer that defines services and characteristics and enables read/write/notify/indicate operations on them. The Figure 26 depicts the interaction between peripheral and the central, where the peripheral is the GATT server (since it provides the services and characteristics), while the central is the GATT client.

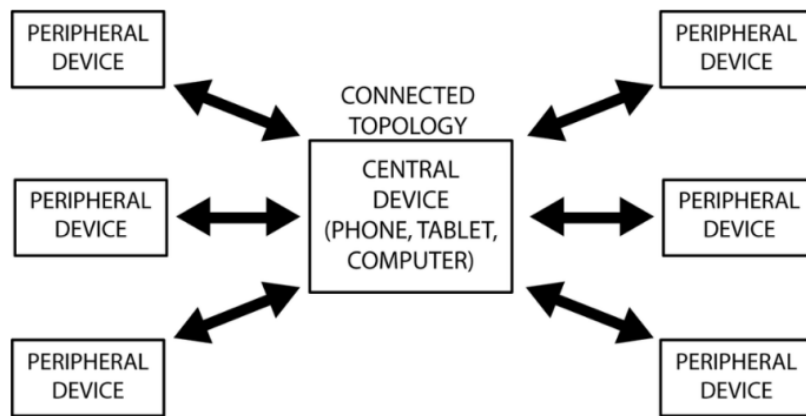


Figure 26 - BLE - General connected topology

Despite our architecture considers that the BLE devices connect to a single Mantis-PC (Central Device), the GATT protocol would allow the Peripheral Devices to connect to multiple Mantis-PCs when and if necessary.

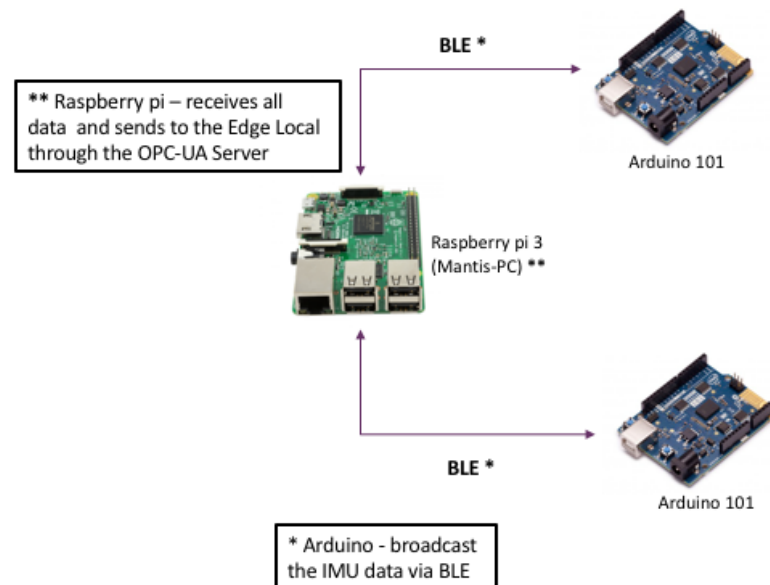


Figure 27 - Context Diagram for the Sensors (Arduino 101) proposed solution

Figure 27 depicts accelerometers (2 x Arduino 101) broadcasts the Inertial Measurement Unit (IMU) of the data collected by the accelerometers, using the GATT protocol via BLE for Mantis-PC (Raspberry PI 3 Model B).

The sensors are prepared to only initiate the sending data once they are paired with the Central Device (*Mantis-PC*). They must use Arduino Low Power strategy, in order to make better use of the battery. For the present project, two sensors are used to detect unevenness

between axis of the machine as well unexpected vibration. They are able to perform self-calibration, synchronization and security by validating the UUID of the exposed services.

The sensors are based on the Arduino 101 platform, which provide a 3-axis accelerometer with a maximum amplitude range of 8g. They are powered by two 9V batteries (Figure 28), to ease the components' installation and testing. In this pilot case, the sensors were configured for a lower measurement range (between 0 and 2g), aiming to attain a better accuracy. The CurieBLE library³⁵ is used to support communication between the sensors and the **Mantis-PC** by using of the Generic Attribute Profile (GATT). According to some preliminary experiments, the maximum distance for this technology is 30 meters, which is compatible with what is stated in the BLE specifications.

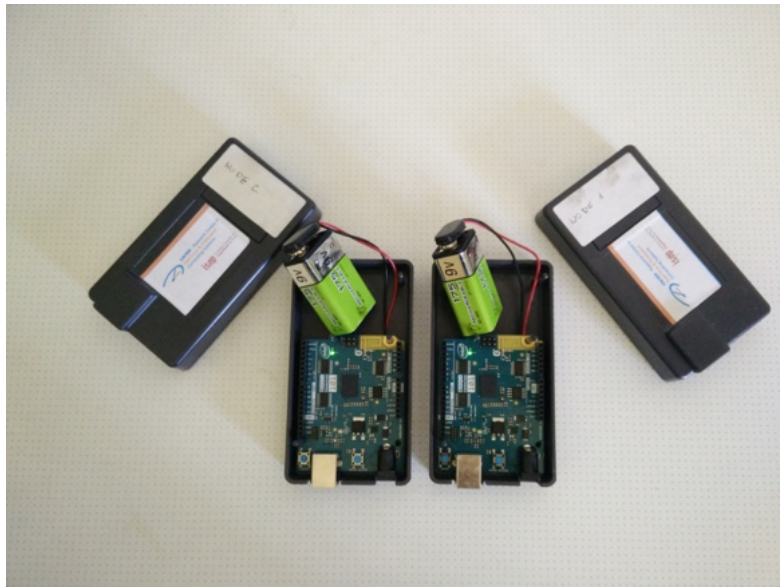


Figure 28 - Sensor Component Hardware

Regarding the data conversion, raw values are mapped to the 2g range (-2g maps to a raw value of 32768 and +2g maps to a raw value of 32767), as it is depicted in Figure 29. This module sends information to the *BLE server* Component of the Mantis-PC Subsystem, which forwards that same information to the **Edge Local** Subsystem, witch is the exit point to the Cloud. On the other hand, the **HMI** Subsystem makes available an interface that is based on the Highcharts³⁶ library, enjoying the “full-responsiveness” capability and providing data.

³⁵ <https://www.arduino.cc/en/Reference/CurieBLE>

³⁶ <https://www.highcharts.com/>

```

#define IMU_NODE1_SERVICE_UUID "2947ac9efc3811e586aa5e5517507c66"
#define ACC_NODE1_CHAR_UUID "2947af14fc3811e586aa5e5517507c66"
// compare and correct values
unsigned long microsPerReading, microsPrevious; float a[3];

// Arduino 101 acts as a BLE peripheral
BLEPeripheral blePeripheral;
// IMU data is registered as a BLE service
BLEService imuService(IMU_NODE1_SERVICE_UUID);
// Each IMU data point is its own characteristic
BLECharacteristic accChar(ACC_NODE1_CHAR_UUID, BLERead | BLENotify, 12);
// Assign pin to indicate BLE connection
const int INDICATOR_PIN = 13;
int aix = 0; int aiy = 0; int aiz = 0; long previousMillis = 0;
int gix, giy, giz; // error in readMotionSensor

void setup() {
  Serial.begin(9600); // console write
  // Initialize IMU
  CurielMU.begin();
  CurielMU.autoCalibrateAccelerometerOffset(X_AXIS, 0);
  CurielMU.autoCalibrateAccelerometerOffset(Y_AXIS, 0);
  CurielMU.autoCalibrateAccelerometerOffset(Z_AXIS, 1);

  // Set the accelerometer range to 2G
  CurielMU.setAccelerometerRange(4);
  // initialize variables to pace updates to correct rate
  microsPerReading = 1000000 / 25;
  microsPrevious = micros();

  // Initialize BLE peripheral
  blePeripheral.setLocalName("IMU");
  blePeripheral.setAdvertisedServiceUuid(imuService.uuid());
  blePeripheral.addAttribute(imuService);
  blePeripheral.addAttribute(accChar);
  const unsigned char initializerAcc[12] = { 0 };
  accChar.setValue(initializerAcc, 12);
  ...
  // Now, activate the BLE peripheral
  blePeripheral.begin();
}

void loop() {
  // Check if the connection to the central is active or not
  BLECentral central = blePeripheral.central();

  if (central) {
    unsigned long microsNow

    microsNow = micros(); // check if it's time to read data and update the filter

    digitalWrite(INDICATOR_PIN, HIGH);

    while (central.connected()) {
      if (microsNow - microsPrevious >= microsPerReading) {
        updateImuData();
        ...
        // increment previous time, so we keep proper pace
        microsPrevious = microsPrevious + microsPerReading;
        ...
      }
      ...
      digitalWrite(INDICATOR_PIN, LOW);
    }
    ...
  }
}

```

Figure 29 - Sensor Component - Partial Node Code

At an early stage, a simulator was created in order to facilitate the testing of the two *Sensor* Components. Furthermore, the tests also assessed real-world scenarios, specifically at ADIRA (Figure 30). The obtained data, which comprehended different periods (20/30 and 100ms) were truly important to assess the sensors' precision, as well as the intensity of the noise that was detected.

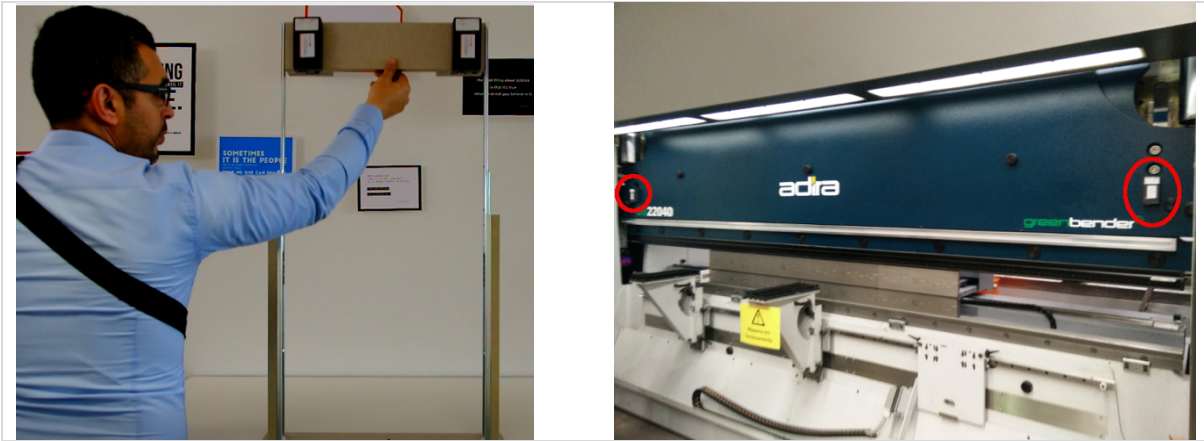


Figure 30 - Sensor Component - Simulator and Tests in real-world scenarios (ADIRA company)

6.2 Mantis-PC Subsystem

This subsystem is composed by two distinct components (Figure 31), namely the *OPC-UA server* Component and the *BLE server* Component, which run in different instances and exchange information through sockets.

In terms of its responsibilities, the *BLE server* Component receives data from the Sensors Component, sending it to the *OPC-UA server* Component, which in turn places the data in the Address Space Server. Regarding the hardware of this subsystem, it is important to emphasize that an economic version was used, namely to test the component (the same that occurs in the *Sensor* Component). If it is necessary to pre-process the data or to perform a local history, then the hardware must be improved.

These components act as an intermediate point collecting the data obtained by the machine's external sensors. Afterwards, the converted data is sent to the local Edge, with a minimum period of 30 milliseconds, using the OPC-UA protocol and the OPC-UA data structure. In the present project, we use the Raspberry PI 3 Model B hardware (Figure 28) with the Raspbian Operating System, which is based on Debian distro.

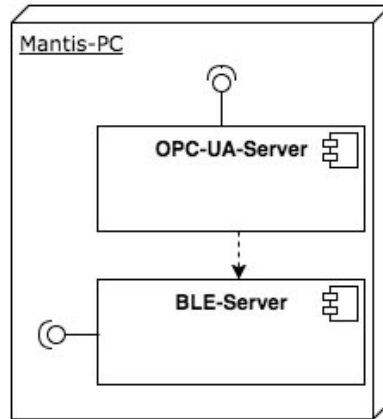


Figure 31 - Mantis-PC Subsystem

6.2.1 OPC-UA Server Component

This particular component presents the same structure of the OPC-UA Server Component, which was described in the **Machine** Subsystem section 6.1. Nonetheless, all data that is loaded into the Server Address Space is realized through the *BLE server* component. In turn, the communication between components is fulfilled through socket.

6.2.2 BLE Server Component

This component is responsible for receiving data from the sensors, as well as for the conversion and sending of that data to the *OPC-UA server* Component of this Subsystem.

Considering that we use a new technology, there is only one library in Node.JS³⁷ that is capable of acting as a *BLE server* (Central Device). Therefore, such component uses a server side JavaScript program, which is built over Node.JS, and the library collects values from both sensors. However, this particular functionality is also responsible for the connection with the Socket server that is provided by the *OPC-UA server* Component, which is represented in Figure 32, thus sending all the messages that it receives from the **Machine** Subsystem Sensors Components.

³⁷ <https://nodejs.org/en/>

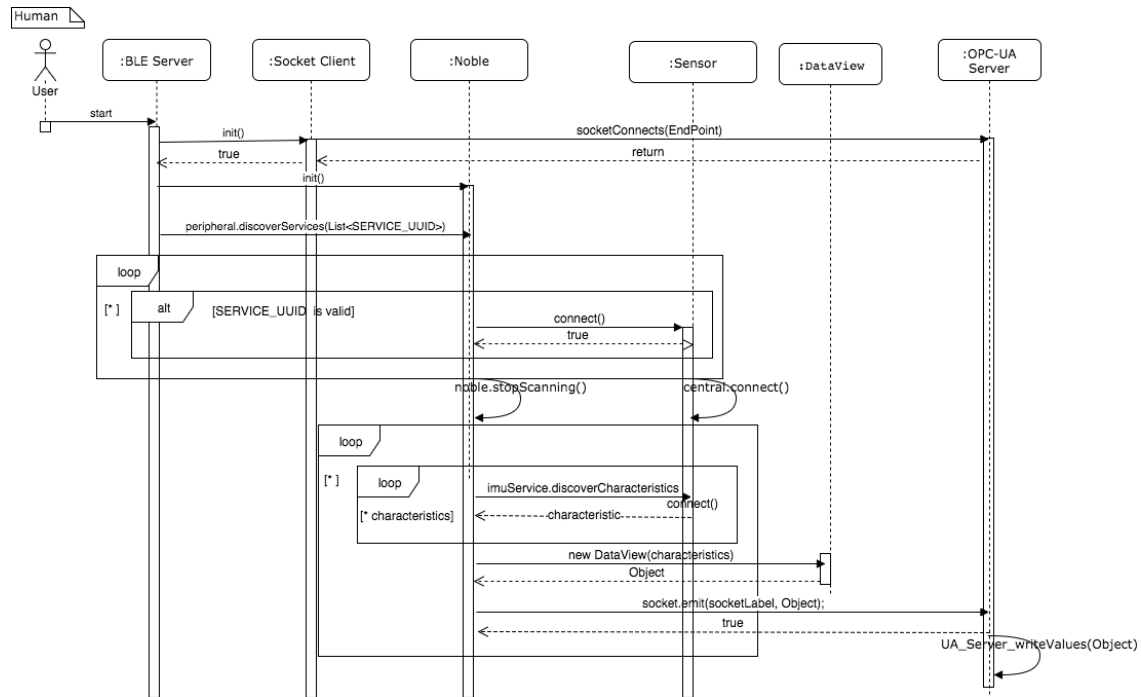


Figure 32 - UML Sequence Diagram for BLE server

Furthermore, the noble library was used to instantiate the *BLE server* (Figure 33) that connects it to the peripherals that have the service adequately registered (SERVICE_UUID). This is one way to ensure that no malicious software connects to the central.

Another responsibility of this component is to convert the received raw values (-2g maps to a raw value of 32768 and +2g maps to a raw value of 32767) and to send them to the *OPC-UA server* Component by the net library.

```

/*
|-----|
| noble |
|-----|

| A Node.js BLE (Bluetooth Low Energy) central module.
*/
// #peripheral's service and characteristic UUIDs - node 1#
var IMU_NODE1_SERVICE_UUID = '2947ac9efc3811e586aa5e5517507c66';
var ACC_NODE1_CHAR_UUID = '2947af14fc3811e586aa5e5517507c66';

// #peripheral's service and characteristic UUIDs - node 2#
var IMU_NODE2_SERVICE_UUID = '11a6779eb70b11e680f576304dec7eb7';
var ACC_NODE2_CHAR_UUID = '11a67a64b70b11e680f576304dec7eb7';

noble.on('stateChange', function (state) {
  if (state === 'poweredOn') {
    // BLE scan..
    ...
    noble.startScanning([IMU_NODE1_SERVICE_UUID, IMU_NODE2_SERVICE_UUID], false);
    ...
  } else {
    // Cannot scan... state is not poweredOn
    noble.stopScanning();
  }
});

// #Discover the peripheral's IMU service and corresponding characteristics#
noble.on('discover', function (peripheral) {
  noble.stopScanning(); // avoid block
  ...
  peripheral.on('disconnect', function () {
    peripheral.disconnect();
  });
  peripheral.connect(function (error) {
    if (error) {
      peripheral.disconnect();
      console.log(error);
    }
    // Connected to peripheral
    peripheral.discoverServices([IMU_NODE1_SERVICE_UUID, IMU_NODE2_SERVICE_UUID],
function (error, services) {
  var imuService = services[0];
  // Discovered IMU service
  imuService.discoverCharacteristics([], function (error, characteristics) {
    ...
    characteristics.forEach(function (characteristic) {
      // Emitting Sensor Data
      emitSensorData(characteristic, peripheral.uuid);
    });
  });
});
...

```

Figure 33 - BLE server Component - Partial noble Code

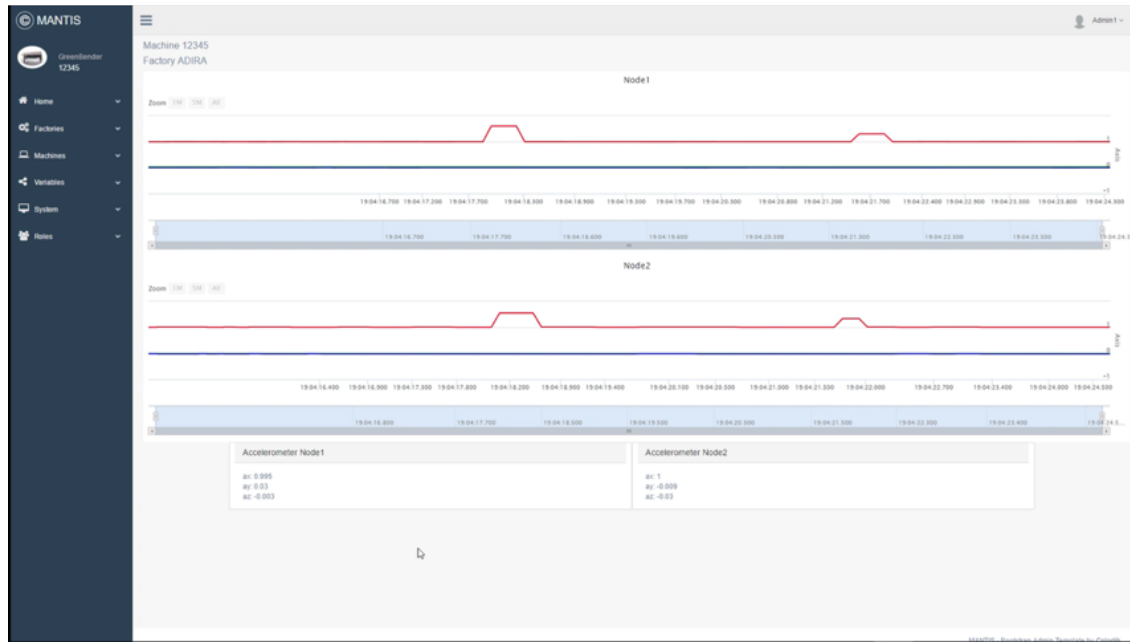


Figure 34 - HMI Subsystem - Sensors Soft Real-time visualization

In Figure 34 we can verify the visualization of the sensors' data in soft real-time. These data were sent by two *Sensors* Components and transmits with a period of ~30ms and sent to the **Edge Local** Subsystem (*OPC-UA Client* Component) through the *OPC-UA server* Component. Afterwards, the Edge Subsystem sends them to the *Middleware* Component, which allows the **HMI** to access the *Middleware* Component and to receive those data (some details regarding this implementation/integration are presented in the subsection on the Human Machine Subsystem implementation).

6.3 Edge Local Subsystem

This subsystem is composed by three components (Figure 35): the *Node-Red*, the *OPC-UA Client* and the *Middleware Client*. The *Node-Red* is responsible for instantiating and connecting all the entities (both machines and industrial assets) that are available in the shop floor. Such machines and assets are virtualized in terms of their capabilities in order to facilitate and enhance the process of exchanging data (machine data readouts). On the other hand, the **HMI** service module is responsible for provide a web application with all necessary information that is generated within the **Edge Local** Subsystem (Devices available and machine data readouts). At last, the *Middleware Client* operates as a gateway, allowing the indirect connection between the machines in a factory flow and the *Middleware*, which performs **Data Analysis** and supports advanced **HMI** features.

The **Edge Local** Subsystem is composed by three different components, namely the *Node-Red* Component, the *OPC-UA Client* Component and the *Middleware Client* Component, being these last two added by the first one.

Considering their responsibilities, we can conclude that the *Node-Red* Component and the *OPC-UA Client* Component receive data from the Address Space Server, from one or more *OPC-UA server* Components, while the *Middleware Client* Component sends data to the Cloud (*Middleware Component*).

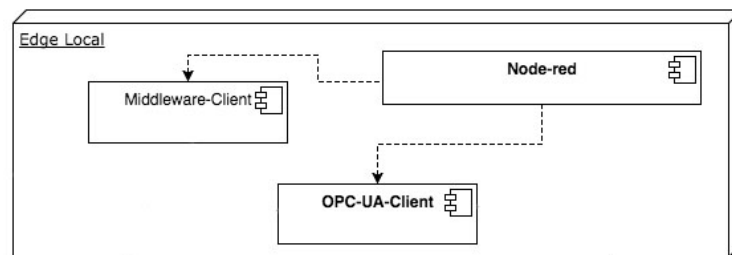


Figure 35 - Edge Local Subsystem

6.3.1 Node-RED Component

This component uses a software tool that was developed by IBM, specifically to wire hardware devices, *APIs* and online services as a part of IoT environment (the *Node-Red* provides a user interface that is quite revolutionary in the field, matching with the Local **HMI** (Figure 36).

The *Node-Red* creates an application by using *Nodes*, which present special capabilities individually. As an example, there are several *Nodes* that are capable of reading and writing files, http pages, emails, twitter messages and connects to *Middleware Clients*. At the present, the *Node-Red* has about 40 out-of-the-box *Nodes*. With a simple drag-and-drop operates a workflow it is possible to create a *Node* within the application, as well as to configure it and connect it with the remaining *Nodes*, thus forming a network. Figure 36 presents an illustration of the workflow of the message states among the several *Nodes*.

uses the previous one to point to the Address Space Server variables of its *OPC-UA server* Component;

- Clean-Payload is of type Function *Node*, being responsible for the removal of characters (e.g.: “;”), which are incompatible with the topic conversion with the JSON format;
- Join is a Join *Node* type, being responsible for the aggregation of *Nodes* messages. In this particular case, it would be the payload of the Inject *Nodes* per key/value;
- PayloadY1, PayloadX1, and PayloadZ1 are of type Function *Nodes*, and in this particular flow they serve as an example of filtering one or more variables of a join;
- Delay is a type of Delay *Node*, being responsible for the creation of the desired delay in the graphics;
- All Value* and Chart* are of the Gauge *Node* and Chart *Node* respectively, being responsible for making the data of different types available on the dashboard (Figure 37);
- Merge-Payload and Erase-duplicate-Payload are of the Function *Node* type and are responsible for pre-processing the data (e.g.: clearing repeated and invalid data) and for transforming the OPC-UA model into an Interoperability model (MIMOSA/IoT-A);
- Logs and csv (“json to csv converter” are not used for incompatibility issues of the different topics) are of File *Node* type and have the responsibility of sending the payload message to files of different formats;
- Msg. payload is of type Debug *Node*, being responsible for debugging one or more *Nodes*;
- RabbitMQ is the AMQP Out *Node* type and is responsible for sending messages to the Cloud (*Middleware* Component).

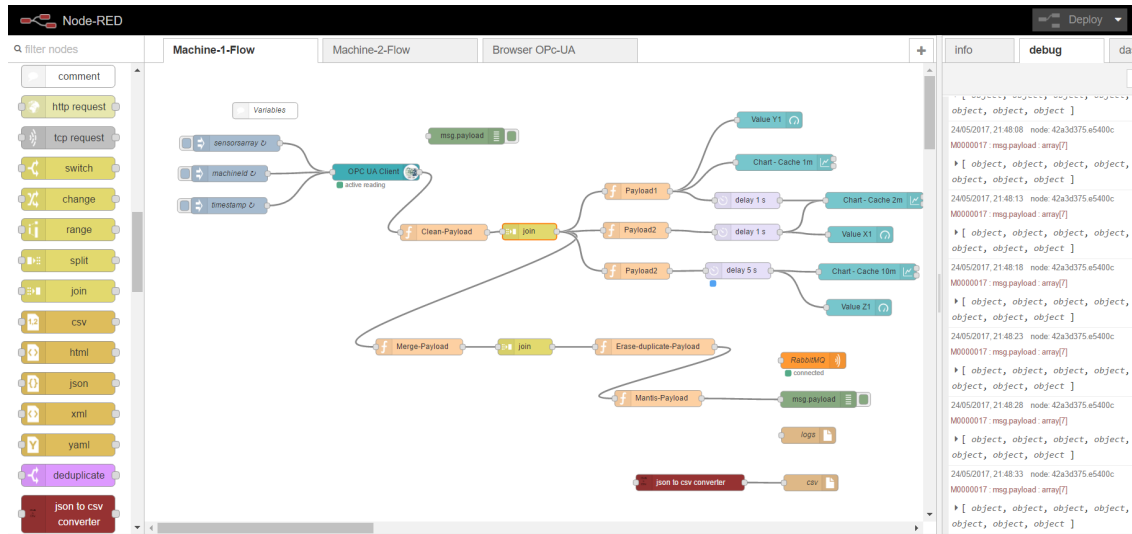


Figure 37 - Node-Red component - Flow

In order to visualize the graphs (Gauge and Chart), a dashboard plugin was used to create a web page, which is presented in Figure 38. The figure features examples of those same graphs, even though they are differently configured. The graph can show the delay (refresh rate), the number of variables (second column has two variables, Y1 and X1) and cache (data history time).

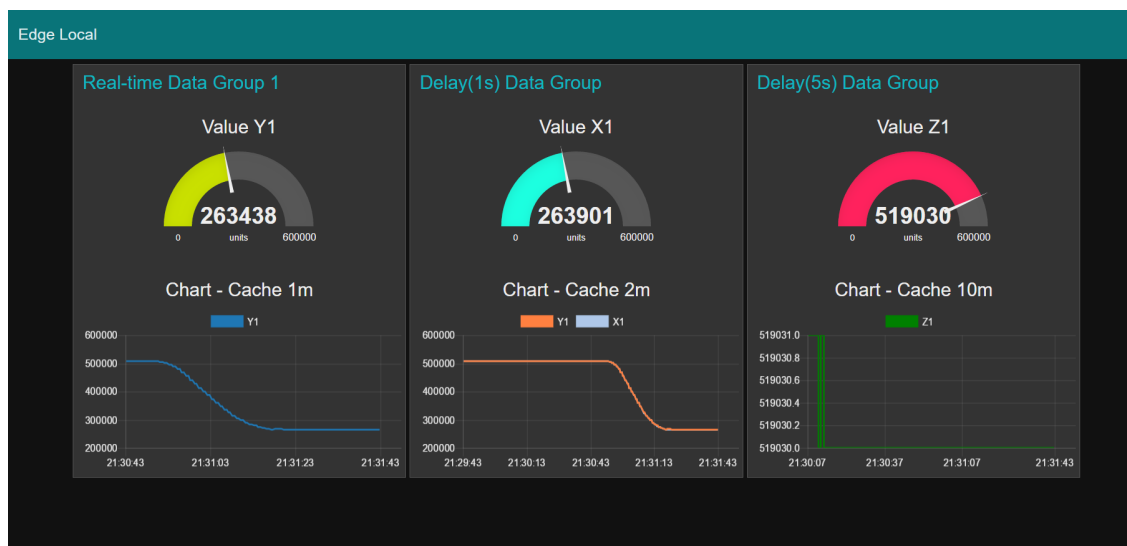


Figure 38 - Node-Red component – User Interface

6.3.2 Middleware Client Component

This component acts as a gateway that connects to the Cloud, to the *Middleware* Component, which is a part of the **Edge Server** Subsystem with the *OPC-UA Client* component.

After conducting a specific analysis, the “*Node-Red AMQP*³⁸” was chosen, due to the fact that it only works with the Producer. Essentially, the *Node-Red AMQP* is a *Node-Red* package that directly connects to an AMQP server (e.g.: RabbitMQ), containing an input, an output and a configuration node in order to connect to the AMQP exchanges or queues to the *Node-Red*.

Delivers incoming the message payload to the specified exchange or queue. It expects an object called `msg` containing the following fields:

- **msg.payload**: string or an object containing the content of the AMQP message to be sent;
- **msg.topic**: string containing the routing-key of the AMQP message to be sent;
- **msg.options**: object containing specific AMQP properties for the message to be sent, see the `amqplib` publish documentation for more information.

If a topic is defined in the output node definition, that it is sent as a routing-key instead of the `msg.topic`. If the `msg.payload` field does not exist, the whole `msg` object is sent. In the settings, it is only possible to define the exchange type or queue, as well as its name. Hence, if it is necessary to use an exchange or a queue with some specific settings, it is possible to define the exchange or queue in the topology tab of the AMQP server configuration node. Thus, the output node uses the exchange or queue that was previously defined in the topology.

6.3.3 OPC-UA Client component

This component is responsible for the connection to the *OPC-UA servers* through TCP/IP with the OPC-UA protocol, namely by using the OPC-TCP binary as a transport. After some analysis, it was concluded that there was only one implementation of a “*node-red-contrib-opcua*” client for the *Node-Red*, which is, in turn, based on *node-opcua*.

The *Node-opcua* is a *Node-Red* package that takes advantage of the asynchronous nature of the `node.js`, creating highly responsive applications. Furthermore, it has been developed using Test-driven development (TDD) and other benefits from more than 1200 unit tests and 96% code coverage. It is also relevant to add that the *NodeOPCUA*³⁹ uses Travis as a continuous integration service.

Let’s highlight the following functionalities:

³⁸ <https://www.npmjs.com/package/node-red-contrib-amqp>

³⁹ <http://node-opcua.github.io/>

- Discovery Service
- Secure Channel Service
- View Service
- MonitoredItems Service
- Subscription Service

6.4 Data Analysis Subsystem

This subsystem is constituted by the Analysis component developed by another project partner and the *Middleware Client* Component developed by the author of this document (Figure 39).

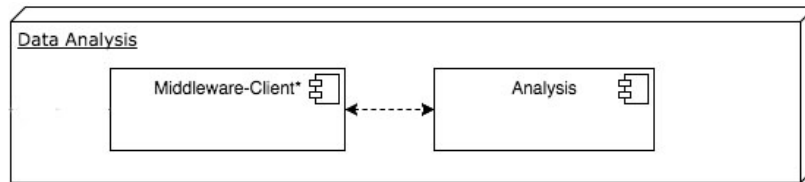


Figure 39 - Data Analysis Subsystem

6.4.1 Middleware Client Component

The *Middleware Client* Component acts as producer and consumer of the *Middleware* Component. The consumer receives data from all sensors in the several plants through the *Middleware* Component, which is later on processed by the Analysis Component.

Finally, the producer sends the detection of the prognosis and of the diagnosis of all the identified machine failures to the *Middleware* Component.

In the Middleware Client class diagram of Appendix-F presents the structure that allows an ease integration with different types of messages, as well as the simple integration of this same component into an existing project, which also allows its own configuration through a configuration file.

On the other hand, Figure 40 illustrates the Consumer of the Middleware Client Component, more precisely in the context of the system of messages exchange between several objects.

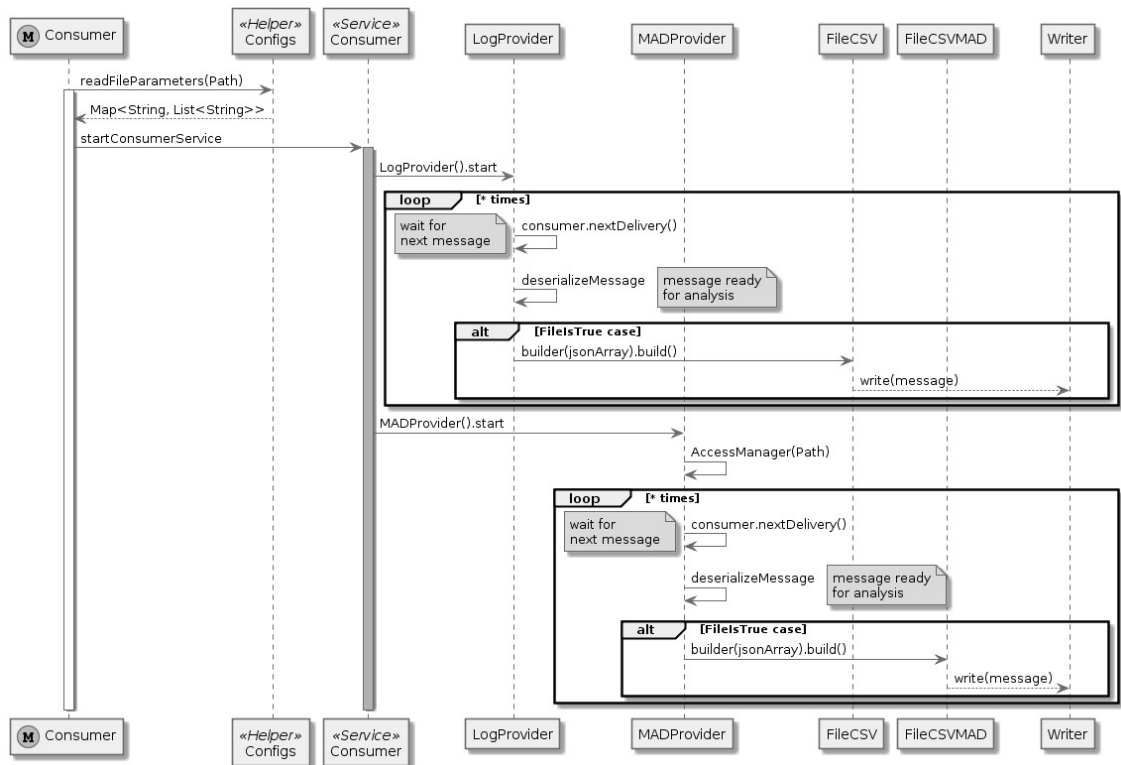


Figure 40 - UML Sequence Diagram - Middleware Client Component (Consumer)

This component was developed in the Java language using Netbeans IDE, and uses the AMQP protocol, more precisely a client of the RabbitMQ implementation. Regarding the message topology, it is based on AMQP topics for sake of flexibility, as defined by the *Middleware* Component.

The main capabilities and responsibilities of these components are:

- Configuring properties in file, namely, Server IP address, files export paths, queues, exchanges, routing keys;
- Connecting to queues using AMPQ Server and collecting or sending data;
- Making a request using a data interval to collect data from the RPC Server using RPC messages pattern;
- Converting the messages to interpret data;
- Collecting data into CSV files (optional).

According to all the specifications presented so far, this implementation followed good practices of programming and design patterns. In Figure 41 we can see the Endpoint class, which is responsible for creating the connection to the *Middleware* Component, specifically through a *ConnectionFactory*. Therefore, this class can be used by both clients (producer/consumer) of this component.

```

...
public abstract class EndPoint {
    // The config file parameters
    protected Map<String, List<String>> valueMap;

    protected Channel channel;
    protected Connection connection;
    protected String ROUTING_KEY;
    /**
     * Instantiates a new end point.
     *
     * @throws Exception the exception
     */
    public EndPoint(String type) throws Exception {
        this.valueMap = Configs.getParametersMap();
        ConnectionFactory factory = new ConnectionFactory();
        // hostname of your rabbitmq server
        factory.setUri(valueMap.get(type).get(0));
        factory.setRequestedHeartbeat(60);
        factory.setConnectionTimeout(
            Integer.parseInt(valueMap.get("CONNECTION_TIMEOUT").get(0)));

        // attempt recovery every 10 seconds
        factory.setNetworkRecoveryInterval(10000);
        factory.setAutomaticRecoveryEnabled(true);

        // getting a connection
        connection = factory.newConnection();
        // creating a channel
        channel = connection.createChannel();
        //declare exchange, queue, binding
        channel.exchangeDeclare(valueMap.get("EXCHANGE_NAME").get(0), "topic");

    }

    /**
     * Close.
     *
     * @throws IOException Signals that an I/O exception has occurred.
     */
    public void close() throws IOException {
        this.channel.close();
        this.connection.close();
    }
}
...

```

Figure 41 - Middleware Client Component - Partial Client Connection Code

Finally, it is mentioned that this component actually allows the integration of the **Data Analysis** Subsystem with the **HMI** Subsystem, which makes it possible to visualize the errors/alarms in the analyzed data, specifically through a graph (Figure 42).

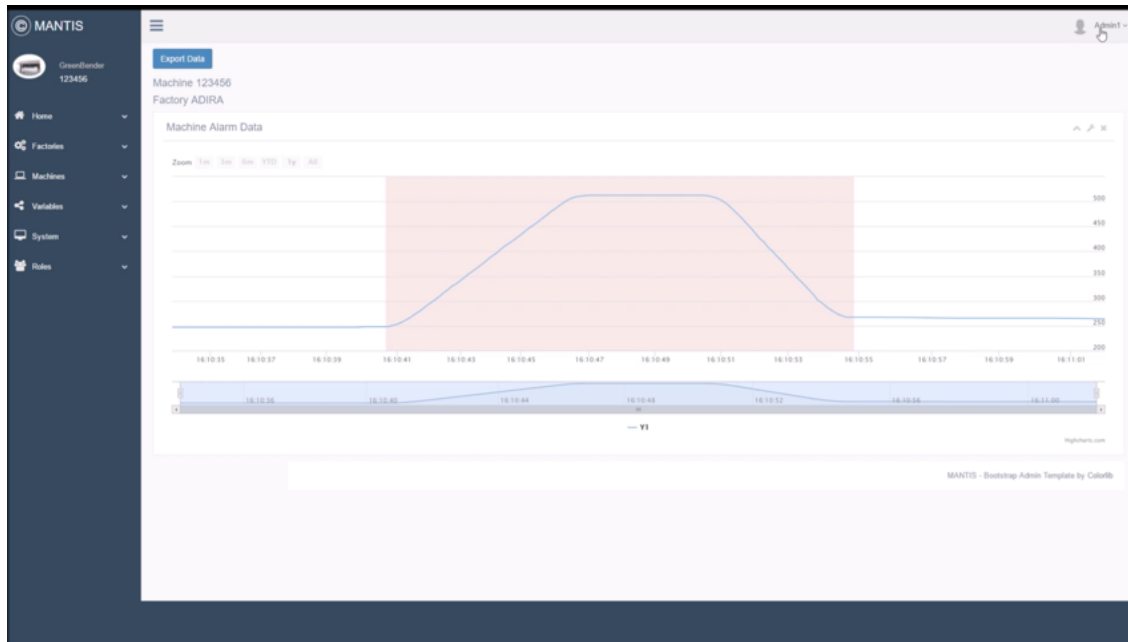


Figure 42 - HMI Subsystem – Data Analysis visualization

6.5 Edge Server Subsystem

This subsystem is composed by several components, such as History, Manager, Database and Middleware, as well as the **HMI** Subsystem with the respective components (Figure 43).

In order to be able to develop and implement the proposed subsystem, the first step is related to the configuration of the Windows server 2012, both for the project and its services:

- **IIS:** web server (Microsoft), where the **HMI** subsystem is deployed;
- **FTP:** to transfer files, thus facilitating the deployment/configuration of all components;
- **RabbitMQ (Middleware Component):** it is a message broker that works as the primary node. The remaining cluster *nodes* are located on other servers;
- **SQL (Database Component):** it is a database relational that is used by the History Component and by the **HMI** Subsystem.

It should be noted that, even though Appendix-G suggests the hardware for this component, this server should be easily scalable, which justifies the suggestion of using, for example, a server that is hosted in the Azure10.

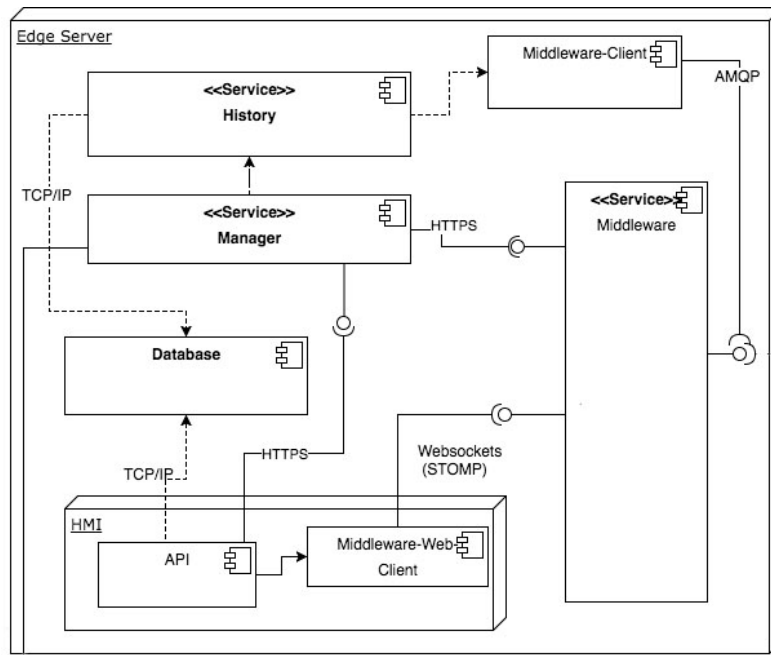


Figure 43 - Edge Server Subsystem

6.5.1 Middleware Component

This component manages the data by storing and transporting them between the **Edge Local**, the **Data Analysis** and **HMI** modules. The communication used in this component obeys the message-oriented paradigm, and messages between components are transported over an Advanced Message Queuing Protocol (AMQP) messaging bus (defined on chapter: “Evaluation of solutions and Existing Approaches”).

Of the four most popular AMQP implementations, namely ActiveMQ, RabbitMQ, OpenAMQ, Apache’s Qpid the choice fell on the RabbitMQ platform. RabbitMQ is an open source platform that implements all the features of AMQP protocol, provides user-friendly configuration tools and useful extensions to the protocol. Furthermore, it enables secure communication between applications and includes interfaces for several major programming languages, enabling them to communicate through the same message broker.

Initially manufactured by Rabbit Technologies Ltd., RabbitMQ increased its market penetration after being released to the open source community. Since then it has offered scalable, robust messaging with flexible routing and high availability. This platform fits the requirements of the project primarily for its routing capabilities using the queue structure, and for being equipped to handle a high volume of concurrent operations.

The basic elements used to build message distribution systems are the exchange and the queue. The exchange is the recipient of a message from a message producer, and its duty is to

deliver the message to one or more queues, the latter being buffers from which the message consumers must pull the messages. An exchange can be connected to multiple queues, and the exchange can be configured to treat messages in different ways, such as relaying the messages to the queues in a round-robin fashion or broadcasting the messages to all the queues. Finally, the decision on which queue(s) receives each message from the exchange, is done by means of a routing key, which is a meta-datum assigned to each message.

The RabbitMQ platform is fault tolerant since, if a message delivery fails, the queue buffers the messages and retransmits them when the message consumer is back online. Moreover, if the broker malfunctions, messages in the queues are not lost since they can be saved in the persistent memory of the broker.

In order to provide data acquisition from machines in different locations, and as suggested in RabbitMQ documentation, the Producer/Consumer design pattern should be used, together with a clear definition of the message format.

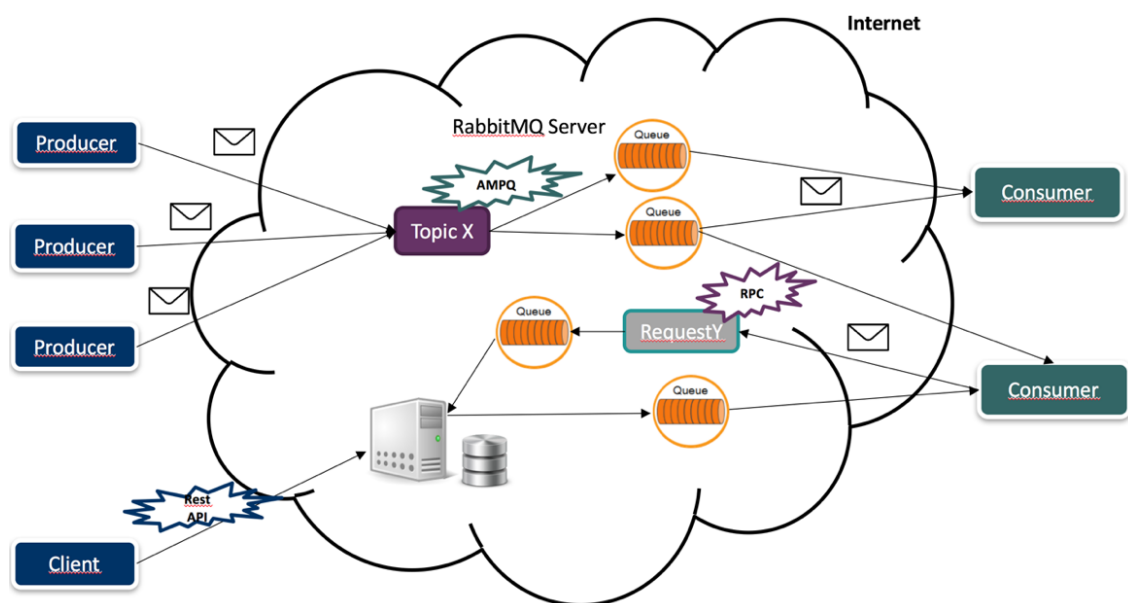


Figure 44 – Context Diagram of Middleware Component

In the implemented communication subsystem, RabbitMQ is the AMQP Broker the Figure 44 exemplifies multiple Producers publishing messages using different routing keys to the topic and the topic, these messages are redirected to each queue and finally are delivered to the respective consumers. Remote Procedure Call (RPC) mechanism is also utilized, it allow running program functions on remote computers.. Finally, the management rest API is available and provide the configuration on the *Middleware Component* via HTTP/HTTPS.

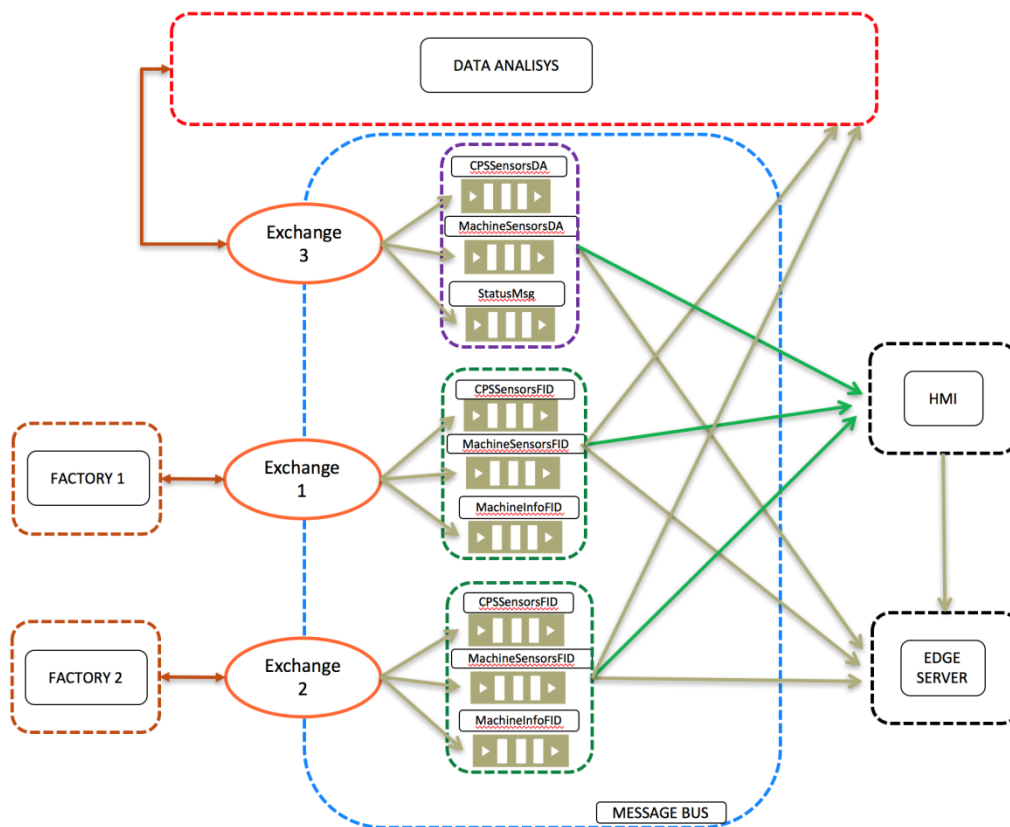


Figure 45 - Middleware Component Queues Structure

The middleware was configured using a topic exchange topology since it supports various publish/subscribe patterns that facilitate message routing. More specifically, this topology can route the same message to multiple queues by matching the message routing key and a queue pattern. Consequently, this allows sending the same message to all queues whose pattern matches the routing key. This is required because several modules must consume the same data. Figure 45 depicts the queues, the modules publishing to those queues and the module consuming from those queues.

Each factory (identified by its FID tag) monitored by this system has its own set of independent queues to where its data is published with different encryptions keys by each factory. The queuing system, for each factory, is composed by three queues (CPSSensorsFID, MachineSensorsFID, MachineInfoFID) and a unique topic exchange. This configuration allows the separation of data from different factories, which may belong to different owners. Thus, coping with privacy concerns of the factory owners which do not allow sharing production-related information with other concurrent in the same markets. Additionally, it provides

increased flexibility if in the future there is the need to support the middleware over a computing cluster – i.e. this solution supports the scalability of the system.

The MachineSensorsFID queue handles data from the internal sensors of all CNC machines in a factory. The MachineInfoFID handles data from the CNC machine databases. This database contains information about the operations performed by the machines and any event detected by its control program. The CPSSensorsFID queue handles data from sources external from the Machine, whose data is collected by the **Mantis-PC** Subsystem. By the way, sensors publishing information to this queue are wireless devices that collect information related to the moving parts of the machine, and are not integrated into the machine control system.

The communication infrastructure also comprises queues to feed data into the **Data Analysis** Subsystem and from this module to the **HMI** Subsystem. The **Data Analysis** Subsystem receives a copy of the data published to the CPSSensorsFID and MachineSensorsMID queues from each factory, which is used to run a series of different analysis on the data and publishes results concerning alarms and errors to the CPSSensorsDA and MachineSensorsDA queues. The StatusMsg queue handles data sent from the **Data Analysis** Subsystem to the **HMI** Subsystem, conveying information regarding the data that has been processed by the **Data Analysis** Subsystem. Note that this processing can be time consuming.

All data are stored on a database, structured upon the MIMOSA /IoT-A open standard for operations and maintenance in manufacturing. Due to its volume, historical data is only stored for ~ 5 months before being deleted or backed up. The calculations leading to this design decision are detailed in the section.

The Factory ID (FID) parameter referred in each queue, identifies the factory from where data comes from. It is unique and allows, together with other credentials (user name, password, etc.) for the **Edge Local** in a factory to connect to the middleware. Similarly, data consumers must also know the FID and connect with their own credentials.

Furthermore, the **HMI** is able to configure all these queues for each factory through a Representational State Transfer (REST) API, being able, for example to create a factory queue structure, as described above.

The queues are configured to temporarily retain messages for a given time to live (TTL) period of 3 hours, thus avoiding an excess of in-memory data if no subscriber reads the messages.

Also note in Figure 46 that the connections in green represent the usage of the STOMP over WebSockets protocol in order for the Web-based **HMI** to be able to connect with Middleware. All other connections in grey colour are supported over AMQP protocol.

As a matter of fact, it has been configured to use SSL and its supporting plugins. It is important to add that the Management Plugin provides a User interface, as well as the REST API that is consumed by the RabbitMQ-Core via HTTPS. Several tests were performed on different Operating Systems (Windows/Linux), on which the Middleware always responded without failing, considering that its installation is an easy process. However, and for security reasons, the Guest (default) was removed and the default port was changed, through the use of the rabbitmq.config File.

In order to provide a test for scalability, we created a cluster (Figure 46), where a Disk and a RAM Node were created. Afterwards, we disconnected the Disk node without affecting the system's functioning. The results show that even if a node fails the system continuous to work properly.



Figure 46 - Middleware Component - Middleware Cluster example

6.5.2 Manager Component

This Component functions as a gateway between the **HMI** Component, with the *Middleware* Component, and the *History Component* through a REST architecture. Therefore, it is possible to disaggregate such Components without impairing any system functionalities or even without adding more responsibilities to it.

In Table 10 the routes that are defined for each service and the type of request are presented. However, it is important to refer that this particular component works with the HTTPS protocol (SSL) and with an API-Key, more precisely to guarantee a better security and privacy.

Table 10 - REST Resources of the Manager Component

Resource	URI	Request
User	/api/createUser/{apikey}	POST
	/api/deleteUser/{apikey}	POST
	/api/createUserPermission/{apikey}	POST
Factory	/api/createFactory/{apikey}	POST
	/api/deleteFactory/{apikey}	POST
	/api/reloadFactory/{apikey}	POST
	/api/getFactorySatus/{apikey}	GET
Machine	/api/createMachine/{apikey}	POST
	/api/deleteMachine/{apikey}	POST
Sensor	/api/createSensor/{apikey}	POST
	/api/deleteSensor/{apikey}	POST
System	/api/initCloudSystem/{apikey}	POST
	/api/closeCloudSystem/{apikey}	POST
	/api/getSystemStatus/{apikey}	GET

In the Manager Component class diagram Appendix-G, is depicted the structure that allows to access History through HistoryProvider, provide RestService which provides all the

resources described in the Table 10 and a ManagerService that is responsible for manipulating all the functionalities RabbitMQ-Core.

There is also a package domain that represents all models and the package helper that contains all static functions that support. Java 8 is used, it is possible to have global variables through an interface (GlobalConstants interface).

When creating a factory, there are several operations that are involved, such as the validation of the requests, the setting up of the *Middleware* Component, specifically by using the Rabbit-MQ-Core Module, the generation of credentials, the sending of an email to the user and the initiation of the service's history through the *History Component*. Hence, the particular workflow of these same operations is presented Figure 47.

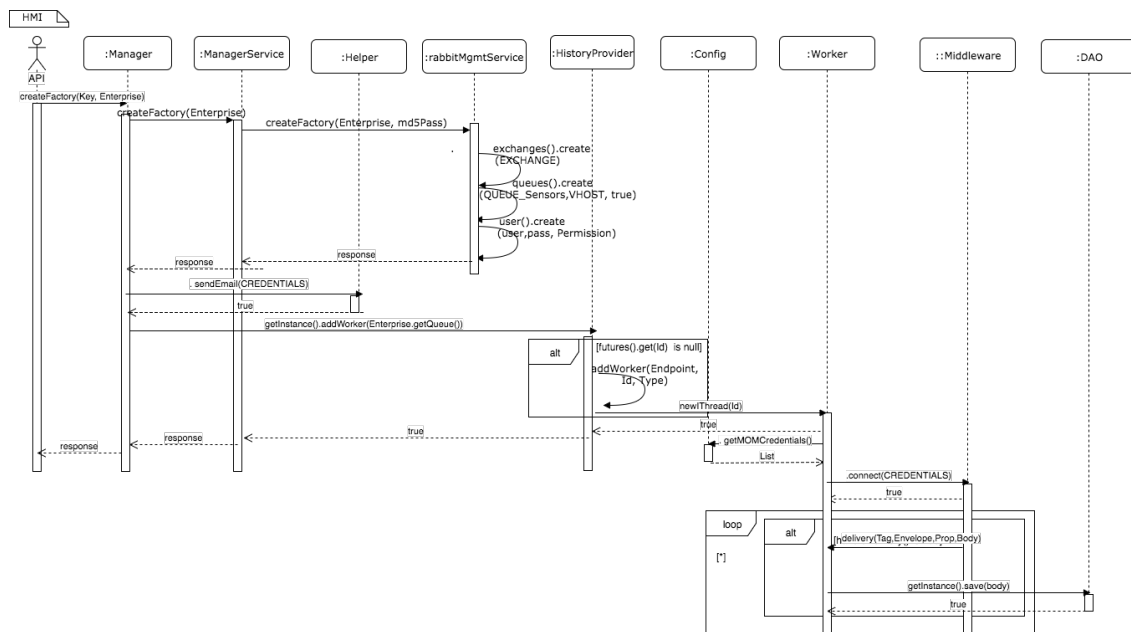


Figure 47 - UML Sequence Diagram - Create Factory

This component uses a REST architecture, which is mainly used by the **HMI** Subsystem, and delegates the work to the History and *Middleware* Components. Hence, a Maven MATIS-IIOT project was created, where the **Edge Server** Module, with the respective Java Application Projects (History, Manager and RabbitMQ-Core), was added, specifically to unselect the projects for a parallel development and without any dependencies or reuses in future projects.

Furthermore, this component also uses the Grizzly library to create the HTTPS server and the Jersey library to make the REST Service available, all with the appropriate resources, as it was previously argued in the design section.

In this project, it was necessary to inject the History Component and the RabbitMQ-Core Module as dependencies, considering that they are actually used.

As it can be seen in Figure 48, Java 8 functionalities (Lambda Expressions, Streams, Optional, etc.) were used, especially whenever it was possible.

```
// get properties (user/Permission)
String values = ManagerService.getRabbitMgmtService().getValueMap().get("USERS");
List<List<String>> usersAndPermissions;
try{
    usersAndPermissions = Configs.getLists(values);
    // create Middleware user and Permission
    usersAndPermissions.stream().forEach((userP) -> {
        ManagerService.getRabbitMgmtService().createUser(userP.get(0), userP.get(1));
        ManagerService.getRabbitMgmtService().setPermission(userP.get(0));
    });

} catch (IOException ex) {
    Logger.getLogger(ManagerService.class.getName()).log(Level.SEVERE, null, ex);
}

// get all Factories from the Database
List<String> factories = DAO.db.queryFactories();

factories.stream().forEach((factory) -> {

    // get all Machines in a Factory from the Database
    Map<String, Boolean> machines = (Map<String, Boolean>) DAO.db.query(factory);

    // create Exchange and queues for each Factory
    ManagerService.getRabbitMgmtService().exchanges().create(new Exchange(factory));
    ManagerService.getRabbitMgmtService().queues().create(new Queue(factory, SENSOR_QUEUE,
VHOST, true, false));
    ManagerService.getRabbitMgmtService().bindings().create(new Binding(factory,
SENSOR_QUEUE));

    // create queues for each machine
    machines.entrySet().stream().forEach((entry) -> {
        if (entry.getKey() != null) {
            ManagerService.getRabbitMgmtService().createMachine(factory, entry.getKey(),
entry.getKey(), entry.getValue());
        }
    });
...
}
```

Figure 48 - Manager Component - Partial initCloudSystem Code

As it was previously mentioned in the design section, this component provides a REST interface that is consumed by the **HMI** Subsystem *API Component*. In Figure 49 we can see that, as an example, a call from the *API Component* to the initCloudSystem use case creates the Queue Basis in the Middleware. Hence, and since the Middleware (RabbitMQ) has been configured with the Management Plugin, we can observe all the settings that are established by the *Manager Component*.

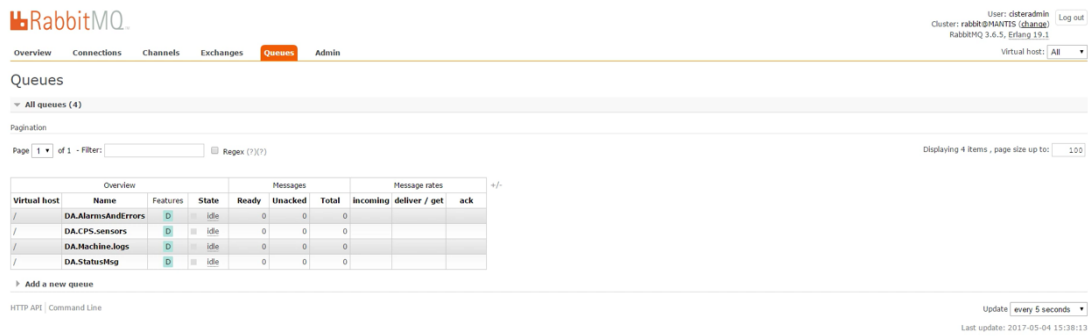


Figure 49 - Manager Component – RabbitMQ User Interface - Based queues

6.5.3 RabbitMQ-Core Module

This module allows the connection and use of the *Middleware* Component (REST) API, with provide the next functionalities:

- List virtual hosts to which they can log in via AMQP;
- View all queues, exchanges and bindings in "their" virtual hosts;
- View and close their own channels and connections;
- List all virtual hosts, including ones they could not log in to via AMQP;
- View other users connections and channels;
- Create and delete virtual hosts;
- View, create and delete users;
- View, create and delete permissions;
- Close other users' connections.

All the mentioned features can be modelled as objects, as it is shown in the package models represented in the Appendix-D. Particularly, there is a package helper that actually contains all the static functions that support and configure the files. Additionally, the builder pattern is also presented, since it is responsible for the instantiation of the objects GsonMessageHandler (DTO), BasicAuthClientProvider and RabbitProvider, which handles all the logic and client requests that use HttpContext.

Furthermore, the class Polymorphism was used, namely on Basefluent class, which can derive in type: ConnectionOperations, ExchangeOperations, NodeOperations, QueueOperations, ParametersOperations, PermissionOperations, VirtualHostOperations, BindingOperations or UserOperations. Nonetheless, it is important to mention that this particular module is prepared to easily add some new functionalities, which are provided by the

REST API, more precisely those that are related to policies and global statistics for all the virtual hosts.

This module is integrated in the Maven project, thus consisting in an **Edge Server** module. Essentially, it is a Java Application Project, making part of the *Manager Component* and working as a client of the REST Middleware service. Thus, it makes available all the functionalities that were defined in the design section of this same Module.

6.5.4 Middleware Client Component

This *Middleware Client* Component is used by the *History Component*, acting solely as a consumer of the *Middleware* Component. The consumer receives data from all the sensors through the *Middleware* Component. This component also uses the same class structure as the *Middleware Client* Component that was presented in the Analysis Subsystem, thus allowing the configuration through a specific configuration file.

6.5.5 History Component

This particular component is responsible for creating a permanent record of all the data that is sent by the Mantis-PC Subsystem, namely through the *Middleware* Component. This entire configuration is done by the **HMI** Subsystem, which uses an API created for this purpose.

In order to respond to all the requests that are received in the *Manager Component*, this component creates several Workers, being responsible for their connection with the *Middleware* Component. Afterwards, it uses the Strategy pattern to respond to different requests and to save them into a database, thus recurring to the *Database Component*. It should be noted that this module performs many tasks simultaneously to the concurrent access to the database, which, in a certain way, makes scalability quite limited.

This component provides for the following functionalities:

- Handle Delivery for Machine sensors Alarms/Errors send by **Data Analysis** Subsystem;
- Handle Delivery for external Sensors Alarms/Errors send by **Data Analysis** Subsystem;
- Handle Delivery for Status Messages Alarms/Errors send by **Data Analysis** Subsystem;
- Handle Delivery for Machine sensors send by **Edge Local** Subsystem;
- Handle Delivery for Machine internal database send by **Edge Local** Subsystem;
- Handle Delivery for external Sensors send by **Edge Local** Subsystem.

It should be noted that this component provides a Consumer Contract that defines the implementation of new Handle Delivery functionalities.

Several patterns are used in the class diagram of Appendix-C namely strategy patterns (ConsumerFactory class that uses ConsumerContract interface, This interface can be implemented by type classes ConsumerMessage, ConsumerSensor, ConsumerMAD, ConsumerLogs and ConsumerAlarms), the builder pattern that is responsible for instantiating the objects

HistoryService, ConsumerService e ProducerService and the repository pattern which is represented in the package DAL. There is also a package model that represents all models and the package helper that contains all static functions that support. Java 8 is used, it is possible to have global variables through an interface (GlobalConstants interface).

In order to illustrate the beginning of the Cloud System and the creation of a Machine as regards exchange of messages between several objects, in an orderly way and at the very moments in which the messages between the objects are sent, two sequence diagrams are shown in Figure 50 and 51.

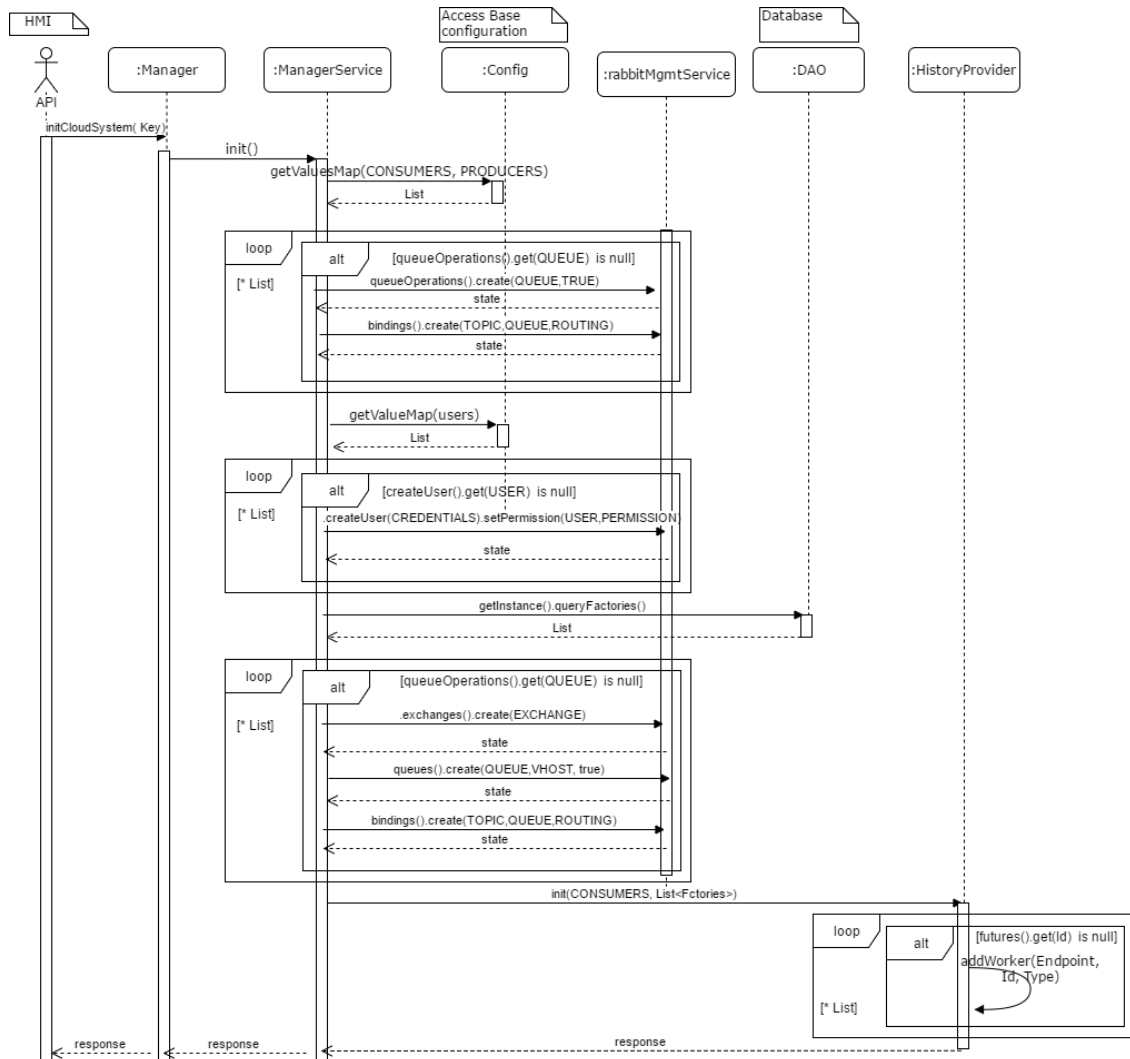


Figure 50 - UML Sequence Diagram - Initialize Cloud System

The diagram of Figure 50, relates to the *History Component* feature that has the responsibility to create the history service, after the Component Manager has configured *Middleware Component*. This configuration creates the basic users (**HMI** and **Data Analysis Users**) and base queues (CPSSensorsDA, MachineSensorsDA and StatusMsg) in *Middleware Component*.

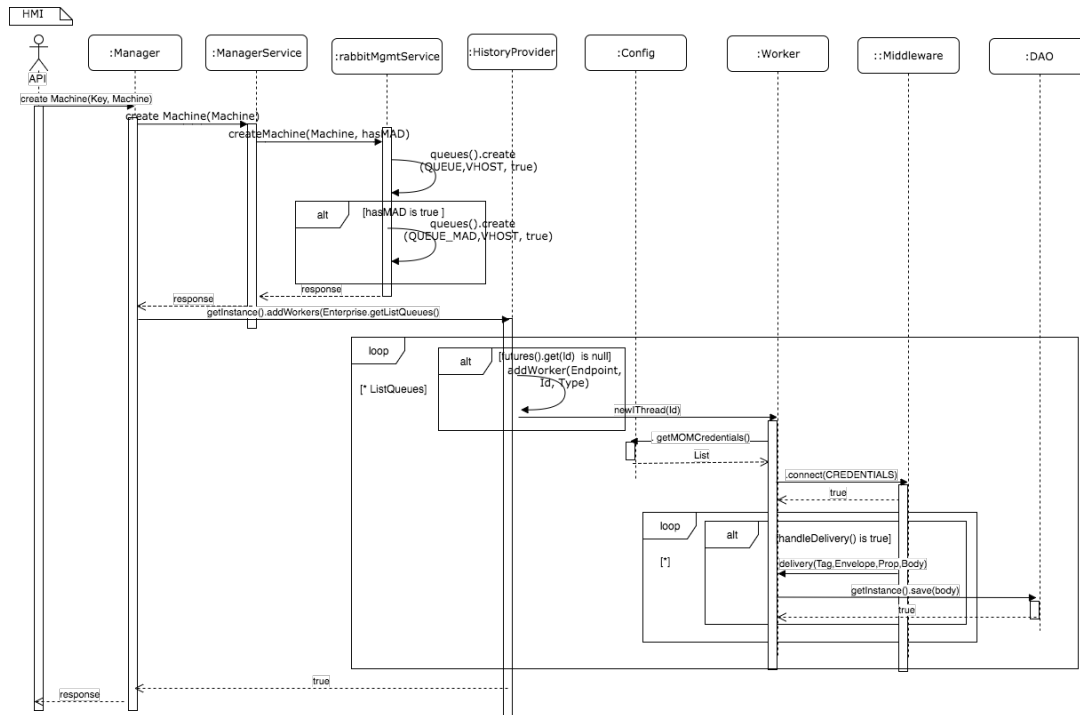


Figure 51 - UML Sequence Diagram - Create Machine

The Create Machine feature, as discussed in the Sequence diagram of Figure 51, has the responsibility of creating queues for the machine, namely the queue for internal machine sensors and the queue for the internal machine database, through *Manager Component* which, in its turn, configures the *Middleware Component*.

History Component then creates the history service for both queues. This component is integrated into the Maven project, consisting in an **Edge Server** module. Overall, it consists in a Java Application Project, thus being a part of the *Manager Component* that works as a service. Therefore, it provides all the functionalities that were defined in the design section, involving problems that are related to parallelism and competition.

For example, whenever the *Manager Component* delegates to the *History Component* the task of creating a Machine with MAD (internal database), the *History Component* actually has to create two Threads (Callable<Worker>) that run in parallel and that connect to the *Middleware Component*, more precisely to receive data (consumer) and to access the *Database Component*, aiming to correctly write them.

As a matter of fact, if we create 50 machines in the system, the *History Component* may have to run to 100 Threads. Hence, the way to scale up to this node would be justified by the existence of several *nodes* within this component and by the distribution that is performed through a load balancing, which truly improves the distribution of workloads.

This implementation used the Executor Service, namely by adding a *List<Future<?>>* in order to control the *Callable<Worker>* (Figure 52). On the other hand, the Worker is dependent on the RabbitMQ Client and has the responsibility of connecting to the Middleware and receiving data.

```

/** Class HistoryService
 * @package pt.cister.services */
public class HistoryService implements GlobalConstants{

    // init the static builder
    public static Builder builder() {...}
    // provides methods to manage termination and methods that can produce a Future for
    //tracking progress of one or more asynchronous tasks.
    private final ExecutorService threadExecutor;
    private List<Future<?>> listProducers; // Object type Producer
    private List<Future<?>> listWorkers; // Object type Worker
    // config properties
    private Map<String, List<String>> valueMap;
    // constructor with Builder
    public HistoryService(Builder builder) throws Exception {
        super();
        ...}
    // init the cloud system
    public boolean init() {
        // get all Machines from the Database
        Map<String, Boolean> lists = DAO.db.queryMachines();
        lists.entrySet().stream().forEach((entry) -> {
            try {
                String queue = entry.getKey();
                // create a Worker with an ID and add it to the respective List<Future<?>>
                this.addWorker(new Worker(queue+ GlobalConstants.MACHINE_QUEUE.logs,
queue));
            }
            ...}

        ...
        // add a Producer or Worker to the respective List<Future<?>>
        public void addProducer(Producer producer) {...}
        public void addWorker(Worker worker) {...}

        //removes a element of List<Future<?>>
        public boolean removeWorker(String id) {... }

        //before removes a Worker from List<Future<?>>it is necessary to stop
        // the middleware consumer Service
        private boolean closeConsumerService(String id, List<Future<?>> listRemoveFutures,
boolean valid) {
            for (Future<?> index : listWorkers) {
                try {
                    Worker worker = (Worker) index.get();// get a Worker
                    if (worker.getId().equals(id)){
                        worker.consumerService.close();();// stop consumer Service
                        listRemoveFutures.add(index); // remove Worker from List<Future<?>>
                        valid = true;
                    }
                } catch (...){...}
            }
            return valid; }

        // get the Worker List size, is essential to control the List<Future<?>> of this object
        public int getWorkerListSize() {...}
        // stop the cloud system
        public void close() {}
        // specifies an abstract interface for creating parts of a HistoryService object.
        public static class Builder {...}

```

Figure 52 - History Component – HistoryService Partial Code

This feature allows the **HMI** to receive data from the *Database Component*, making them available on a graph (Figure 53) or even in a table.

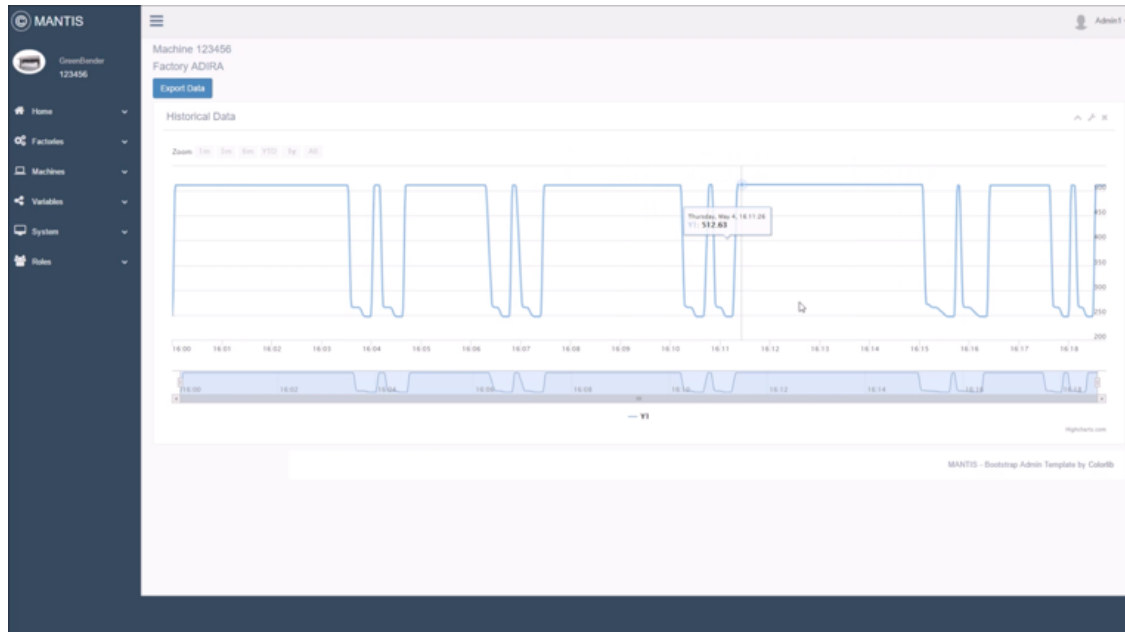


Figure 53 - History Component – HMI history visualization

6.5.6 Database Component

This database is used by several components, namely the **HMI** Subsystem and the *History Component*, which is structured according to the MIMOSA/ IoT-A standard. Due to the simplification of the creation of the base structure of such standard, a relational SQL database was used, which also creates performance constraints when compared to a non-SQL database such as MongoDB.

The *History Component* creates an abstraction layer in the data access layer through Repository Interfaces so that the database management system can be easily changed if it is changed.

This component uses a relational database (Microsoft SQL Server 2012), being used, in turn, by the *History Component* and by the **HMI** Subsystem. Considering that it is a centralized database, such as several read/write accesses in very short periods of time, this feature questions the performance and the scalability of the entire system. Therefore, we propose an exchange to a documental and decentralized database. Even though this technique is used as an index definition or even as cache, the performance is still quite low. Additionally, it should be added that this database still does not follow the proposed data model (IoT-A), since **HMI** Subsystem has not yet been updated for this new structure.

6.5.7 Email component

This component is responsible for using the SMTP email protocol to send emails (Figure 54). In this way, it is possible, for example, when creating a factory, to send the user credentials to the respective recipient through this Component.

In order to respond to a use case UC03 functionality (create factory) for the notification of the user's credentials, the Gmail provider was used, namely to provide the SMTP protocol. Furthermore, and to use this same component in the *Manager Component*, by recurrence to the Java Programming Language, the native “javax.mail” was used.

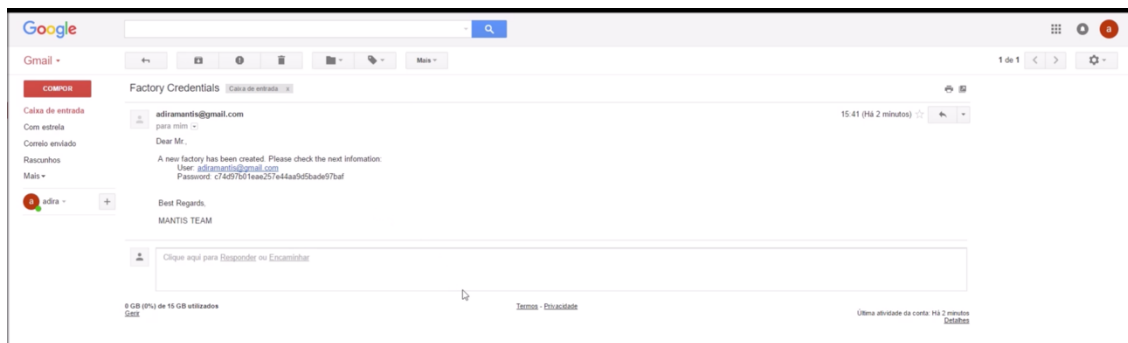


Figure 54 - Email component – User email notified example

6.6 Human Machine Subsystem

This subsystem is constituted by a Middleware-Web-Client component that communicates with the *Middleware Component* via STOMP (Websockets) and a *Manager API Component* (for simplicity in the figure and further in this report we use the term API) that communicates with the *Manager Component* existing in the **Edge Server** subsystem via HTTPS.

The author of this document developed the *API Component*, the *Middleware-Web Client Component* and provide the API module for the real-time graph visualization for this Subsystem. The **HMI** Subsystem web interface and other functionality were developed by a project partner. Due to the fact that only the *API Component* and the *Middleware-Web Client Component* have been developed by the author of this project, the remaining components of this subsystem, such as the **HMI** interface and other services, are not described from an implementation perspective. Similarly, it should be emphasized that the Graph Module was also developed to support the **HMI** interface (Graphs of Soft Real-time visualization).

6.6.1 API Component

This component is related to the REST architecture and to the separation of concerns, which is essentially the principle behind the client-server constraints.

Therefore, it provides a Service Contract that corresponds to the Endpoints of the *Manager Component*. Such implementation also facilitates the addition of new features, despite allowing the use of an API-key and the serialization of the objects that must be integrated in the Data Transfer Object.

As it is shown in Appendix-G, all of the above features are modelled into 6 different class/interface objects:

- **Service Contract:** interface that defines all the functionalities of the contract between the *Manager Component* and **HMI**;
- **Management Service:** responsible for manipulating all the functionalities related to factories, machines, sensors, system, etc.;
- **Security:** hold the API-key essential to ensure more security;
- **Helper:** contains all static functions that support;
- **Helper EndPoint:** contains all static Endpoint of the *Manager Component* (of the **HMI** .Net MVC project);
- **BootStrap:** contains all static information for testing purpose.

The API Component was developed in Visual Studio, in order to make the integration with the project compatible, which was, in turn, integrated within the main project (HMI interface). Hence, the component was made available with an example, thus following the same MVC structure of the main project.

Considering that we are integrating a REST architecture via HTTPS, a contract was created to ensure an assertion in the code. This implementation was integrated to the services layer of the HMI interface project, despite responding to all the functionalities that defined in Table 10 of the Manager Component (Figure 55).

```

namespace GECAD_Web.Services
{
    interface ServiceContract
    {
        /// <summary>
        /// Creates a factory topic in the MOM,
        /// the based queue/s, and the User
        /// The defined email will receive the password
        /// for the MOM
        /// </summary>
        /// <returns/> ResponseContent with message
        ResponseContent CreateFactory(string factoryID, string email, string Permissionype,
string regex);

        /// <summary>
        /// Deletes a factory topic in the MOM,
        /// the based queue/s and the User
        /// </summary>
        /// <returns/> ResponseContent with message
        ResponseContent DeleteFactory(string factoryID, string email);

        /// <summary>
        /// Create a Machine queue/s in the MOM
        /// and start the History for the define queue/s
        /// </summary>
        /// <returns/> ResponseContent with message
        ResponseContent CreateMachine(string factoryID, string machineID, bool hasAlarms);

        /// <summary>
        /// Init the Cloud System with the default queues
        /// and Users in the MOM. Also creates the history for
        /// the default queues
        /// </summary>
        /// <returns/> ResponseContent with message, result(bool) and resultType
        ResponseContent InitCloudSystem();

        ...
        // ----- SuperAdmin access

        /// <summary>
        /// Create a User in the MOM
        /// </summary>
        /// <returns/> ResponseContent with message
        ResponseContent CreateUser(string userName, string password);

        ...
        /// <summary>
        /// Create Create User Permission in the MOM,
        /// define the type (write/read/all) and
        /// regular expressions
        /// </summary>
        /// <returns/> ResponseContent with message
        ResponseContent CreateUserPermission(string userName, string type, string regex);
        ...
    }
}

```

Figure 55 - API Component – Partial Contract Code

6.6.2 Middleware-Web Client Component

This component uses a different protocol when compared to the *Middleware Client* Component, which is designated by the STOMP protocol. However, and considering that it is a web application, a WebSocket API needs to be used to directly connect the **HMI** Component to the *Middleware Component*, thus being able to acquire data in soft real-time. After conducting a thorough analysis, it was possible to conclude that the STOMP protocol library

presents a security problem, considering that it exposes the client-side access credentials. Hence, it is highly recommended to change the protocol, for example to the MQTT, aiming to solve such problem.

This component was developed in IntelliJ, by using the *npm* as a dependency manager. As it was previously mentioned in the design section of this component, the choice of this protocol (STOMP over Websocket) presents some security problems (Figure 56). Nonetheless, the partners have decided on the protocol, thus creating the component and integrating it in the main project (**HMI** interface).

Since we are integrating an interface with a JavaScript component, we can conclude that the integration was quite trivial. On the other hand, and considering that we are providing a significant volume of data in a single graph, it was necessary to collect and redraw some techniques, especially by using a set of data rather than a single point (`setData`: Apply a new set of data to the series). Hence, we managed to have Soft Real-time data with periods shorter than 200ms, avoiding the browser's overload.

```

// create Stomp socket client
var ws = new WebSocket('ws://193.136.60.47:15674/ws');
var client = Stomp.over(ws);

// RabbitMQ SockJS does not support heartbeats so disable them
client.heartbeat.outgoing = 0; client.heartbeat.incoming = 0;

/*
|-----
| Graph highcharts
|-----
*/
var pointsNodeX = []; var pointsNodeY = []; var pointsNodeZ = []; var pointsNodeT = [];
...
/**
 * Add points to series chart node 1 and use setTimeout function
 * @method requestDataNode1
 */
var requestDataNode1 = function () {
    var chart = $('#containernode1').highcharts();

    var series = chart.series;
    series[0].setData(pointsNodeX); series[1].setData(pointsNodeY);
    series[2].setData(pointsNodeZ); series[3].setData(pointsNodeT);
    setTimeout(requestDataNode1, 200); // 200ms
    chart.redraw();
};
...
$(function () {
    // Create the chart for a node
    $('#containernode1').highcharts('StockChart', {
        ...
    });
});
// connect to the server
var on_connect = function (x) {

    id = client.subscribe("/queue/" + @ViewBag.sensorId, function (m) {
        // here is the loop with the data
        var obj = JSON.parse(m.body);
        /*
        |-----
        | Parse DTO
        |-----
        */
        var date = new Date().getTime();
        if (obj.type == @ViewBag.UUID) {
            pointsNodeX.push([date, obj.values.ax]);
            ...
            document.getElementById('az-node1-value').innerText = obj.values.az;
            ...
        });
    });
};
// debug function
client.debug = function (e) {};
// this function is passed on client.connect and catch all errors
var on_error = function (error) {};

// this function connect to client and exposes the user and pass - this user have limit
rights (only read from queues)
// in future we will use the MQTT protocol the grant more security
client.connect('gecad', '7ecec341e14481debea8917cee063580', on_connect, on_error, '/');

```

Figure 56 - Middleware-Web Client Component with Graph Module - Partial Code

At last, it is important to add that this component actually allows the integration of the **HMI** interface with the *Middleware Component*, being possible, for example, to visualize all the data that are related to the sensors/machines in Soft Real-time through a graph (Figure 57).



Figure 57 - HMI Subsystem – Machine data Soft Real-time visualization

6.7 Deployment

The deployment diagram of the architecture presented in Figure 15 of subsection 5.1 is shown in the Figure 58. In particular, Figure 58 depicts the setup and information of subsystems and components of this solution.

Considering that the software to be developed has a local part, which is located within the companies (the **Machine**, **Mantis-PC** and **Edge Local** subsystems), and another part in the Cloud (the **Edge Server**, **HMI** and Email subsystems), which in turn is located in ISEP, by using a Windows Server 2012 and the IIS service it is possible to support the **HMI** that was developed in .NET. Furthermore, in Appendix-I the deployment setup and information is presented for all component of each subsystem.

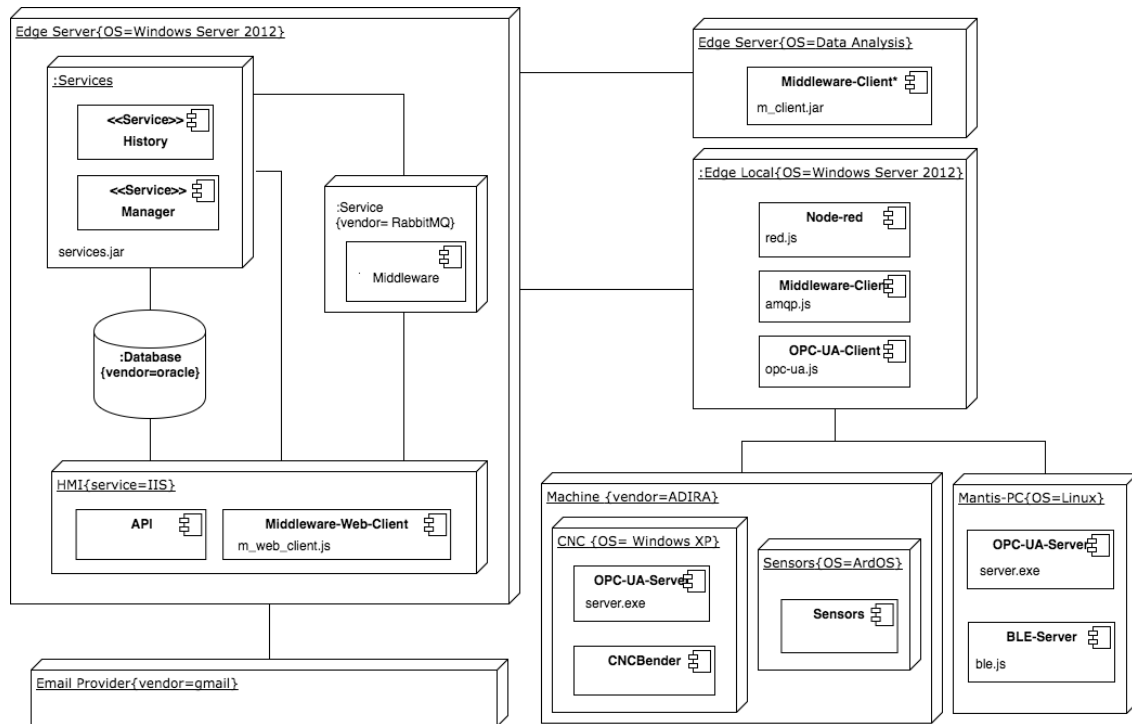


Figure 58 - UML Deployment View diagram for the proposed solution

6.8 Conclusions

All the components of each subsystem were designed and implemented regarding the requirements. Consequently, the analysis of the previous chapter that improves the understanding of the previously established requirements. The justify the adopted design decisions using patterns and rules followed by a discussion of the design alternatives were provide whenever the component requirements defined it. Furthermore, technical details and the hardware/software requirements of the installation were provided following the deployment of the solution.

7 Tests

In this chapter, the software testing is one element of a broader spectrum, which is often referred to as verification and validation (Figure 59). The check refers to the set of activities that ensure that the software correctly implements a specific function. The validation refers to a different set of activities, which ensures that the built software meets the Stakeholders requirements.

Verification - Are We Building The Product Properly?

Validation - Are we building the right product?

The tests effectively offer the ultimate stronghold in which quality can be assessed and, more pragmatically, errors can be discovered. Quality is incorporated into the software during the software engineering process. Proper application of methods and tools, effective formal technical revisions, and solid management and measurement lead to quality standards that are confirmed during testing (Pressman, 2011).

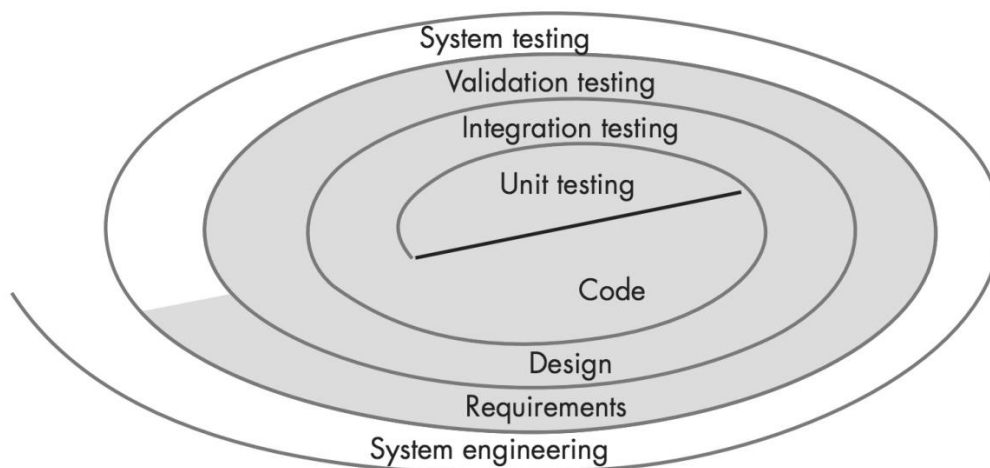


Figure 59 - Testing strategy (Source: Pressman, 2011).

7.1.1 Unit

Unit testing focuses on the smallest software design unit: the software component or module. Using the component-level design description as a guide, important control paths are

tested to discover errors within the module boundaries. The relative complexity of the tests and errors discovered is limited by the restricted scope established for the unit test (Pressman, 2011).

These tests focus on an internal processing logic and on data structures, within the limits of a component, and were conducted in parallel for various components. Furthermore, all components have been tested and performed before coding, following the adopted agile methodology (RUP). Figure 60 depicts the partial code of the *Middleware Component* unit tests focusing on important control paths.

```
public class MiddlewareAssert {
    ...

    /** Assert that Middleware has the specified node.
     * @param nodeName Name of the node to ensure exists.
     * @param matchers Modifies the assertion by adding criteria to match.
     * @return this. */
    @Test
    public MiddlewareAssert hasNode(String nodeName, NodeMatcher... matchers){
        Optional<Node> node = mgmt.nodes().get(nodeName);
        assertTrue(String.format("Node '%s' does not exist.", nodeName), node.isPresent())

        if (matchers != null && matchers.length > 0) {
            MatchResult result = isMatch(node.get(), matchers);
            assertTrue(result.getReason(), result.isMatch());
        }
        return this;
    }

    /** Assert that Middleware does not have the specified node.
     * @param nodeName Name of the node that should not exist.
     * @param matchers Modifies the assertion by adding criteria to match.
     * @return this. */
    @Test
    public MiddlewareAssert doesNotHaveNode(String nodeName, NodeMatcher... matchers){

        Optional<Node> node = mgmt.nodes().get(nodeName);
        if (node.isPresent()){
            if (matchers != null && matchers.length > 0) {
                MatchResult result = isMatch(node.get(), matchers);
                assertFalse(result.getReason(), result.isMatch());
            }
            else {
                fail(String.format("Node '%s' should not exist but does.", nodeName));
            }
        }
        return this;
    }

    /** Assert that Middleware has the specified Virtual Host.
     * @param vhostName VHost that should exist.
     * @return this. */
    @Test
    public MiddlewareAssert hasVHost(String vhostName){

        Optional<VirtualHost> vhost = mgmt.vhosts().get(vhostName);
        assertTrue(String.format("VHost '%s' shou exist but does not.", vhostName), vhost.isPresent());
        return this;
    }

    /** Assert that the specified VHost does not exist in the Middleware cluster.
     * @param vhostName Name of the vhost that should not exist.
     * @return this. */
    @Test
```

```

public MiddlewareAssert doesNotHaveVHost(String vhostName){

    Optional<VirtualHost> vhost = mgmt.vhosts().get(vhostName);
    assertFalse(String.format("VHost '%s' should not exist but does.", vhostName), vhost.isPresent());
    return this;
}

/** Assert that Middleware has the specified User.
 * @param username Name of the user that should exist.
 * @return this. */
@Test
public MiddlewareAssert hasUser(String username){

    Optional<User> user = mgmt.users().get(username);
    assertTrue(String.format("User '%s' should exist but does not.", username), user.isPresent());
    return this;
}
...
/** Assert that the current authenticated user
 * (interacting with the Management Console) is the one specified.
 * @param username Name of the user that should be interacting with the console.
 * @return this. */
@Test
public MiddlewareAssert iAm(String username){
    User clientUser = mgmt.users().whoAmI();
    assertEquals(String.format("Current user should be '%s', but is actually '%s'.",
        username, clientUser.getName()), username, clientUser.getName());
    return this;
}
/** Assert that the User has the specified configure permission.
 * @param vhost Virtual Host with the permission.
 * @param user Name of the user.
 * @param permissionExpression Value of the permission.
 * @return this. */
@Test
private MiddlewareAssert userHasPermission(String vhost, String user, String permissionExpression, int
permissionType){

    Optional<Permission> permission = mgmt.permissions().get(vhost, user);
    assertTrue(String.format("User '%s' does not have permission '%s' on vhost '%s'",
        user, permissionExpression, vhost),
        permission.isPresent());
    String actualPermission;
    String permissionTypeDescription;
    switch (permissionType){
        case 1:
            actualPermission = permission.get().getRead();
            permissionTypeDescription = "read";
            break;
        case 2:
            actualPermission = permission.get().getWrite();
            permissionTypeDescription = "write";
            break;
        default:
            actualPermission = permission.get().getConfig();
            permissionTypeDescription = "configure";
            break;
    }
    assertEquals(String.format("User '%s' permission '%s' should be '%s' but is '%s' on vhost '%s'.",
        user, permissionTypeDescription, permissionExpression, actualPermission, vhost),
        permissionExpression, actualPermission);
    return this;
}

/** Assert that Middleware has the specified queue.
 * @param vhost Name of the vhost with the queue.
 * @param queueName Name of the Queue.

```

```

* @param matchers Modifies the assertion by adding criteria to match.
* @return this. */
@Test
public MiddlewareAssert hasQueue(String vhost, String queueName, QueueMatcher... matchers) {

    Optional<Queue> queue = mgmt.queues().get(vhost, queueName);
    assertTrue(String.format("Queue '%s' does not exist and should on vhost '%s'.", queueName, vhost),
        queue.isPresent());

    if (matchers != null && matchers.length > 0) {
        MatchResult result = isMatch(queue.get(), matchers);
        assertTrue(result.getReason(), result.isMatch());
    }
    return this;
} ...

```

Figure 60 – Middleware Unit Tests Partial Code

7.1.2 Integration

Integration testing is a systematic technique for constructing the software architecture, while at the same time testing is performed to discover errors associated with the interfaces. The objective is, from unit-level tested components, to build a program structure determined by the project (Pressman, 2011).

Following the unit tests, integration tests were created and it should be noted that there are two types of integration: top-down integration and bottom-up integration. Top-down integration was used because the high-level modules are tested and integrated first, allowing to quickly find high-level logic and data flow errors; In this way, we use stubs, which are considered as false modules/components that always simulate the low-level modules/components.

Figure 61 depicts the partial code of the *Middleware Component* integration test using stubs as the *Middleware Client* component.

```

public class MiddlewareIntegrationTest {

    // middleware configuration
    private static final String URI = "SERVER";
    private static final String QUEUE = "TEST";
    private static final String ROUTING_KEY = "FULL.#";
    // services for 2 producers and 2 consumers
    private static ProducerService producerService1;
    private static ProducerService producerService2;
    private static ConsumerService consumerService1;
    private static ConsumerService consumerService2;

    // class used to compare the sent data received by the producers/consumers
    private static SimpleCache cache = new SimpleCache();

    /**
     * Executed prior any tests cases contained in this implementation unit
     * Initiates the producers/consumers services
     */
    @BeforeClass
    public static void startup() {
        try {
            // factories and channels instantiation

```

```

        producerService1 = new ProducerService.Builder(URI, QUEUE, ROUTING_KEY).build();
        producerService2 = new ProducerService.Builder(URI, QUEUE, ROUTING_KEY).build();
        consumerService1 = new ConsumerService.Builder(URI).build();
        consumerService2 = new ConsumerService.Builder(URI).build();
    } catch (Exception ex) {
        Logger.getLogger(MiddlewareIntegrationTest.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * Executed after any tests cases contained in this implementation unit
 * Shutdown the producers/consumers services
 */
@AfterClass
public static void tearDown() {
    try {
        producerService1.close();
        consumerService1.close();
    } catch (IOException ex) {
        Logger.getLogger(MiddlewareIntegrationTest.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * Assert that the consumers have the SimpleCache content with all the
 * data of numNames array send by the producers.
 * Assert that the consumers receive the messages in the right order.
 */
@Test
public void cacheWithNumNamesArrayData() {
    try {
        //-----Consumer 1-----//
        ConsumerHandler1 consumer = new ConsumerHandler1(consumerService1.getChannel(URI), cache);
        consumerService1.setConsume(consumer);
        //-----Consumer 2-----//
        ConsumerHandler2 consumer2 = new ConsumerHandler2(consumerService2.getChannel(URI), cache);
        consumerService2.setConsume(consumer);

        // populate the queue (QUEUE) with the numNames content using 2 producers
        for (String num : numNames) {
            producerService1.publish(Message.builder().payload(num).build(), null);
            producerService2.publish(Message.builder().payload(num).build(), null);
        }

        Thread.sleep(3000); // This, ensures that consumers receive all messages

        // collect data from consumers
        List<CacheEntry> cacheContent = ConsumerHandler1.cache.getContent();
        List<CacheEntry> cacheContent2 = ConsumerHandler2.cache.getContent();
        // assert that collect data from consumers have the same size as numNames array
        assertEquals(38, cacheContent.size());
        assertEquals(38, cacheContent2.size());

        // assert that both consumers receive the first message in the right order
        assertEquals(new CacheEntry(numNames[0], 0).getText(), cacheContent.get(0).getText());
        assertEquals(new CacheEntry(numNames[0], 0).getText(), cacheContent2.get(0).getText());
    } catch (InterruptedException ex) {
        Logger.getLogger(MiddlewareIntegrationTest.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
...

```

Figure 61 - Middleware Integration Tests Partial Code

7.1.3 Validation

Validation tests begin at the end of the integration test, when individual components have already been drilled, the software is completely assembled as a package and interface errors have been discovered and fixed. In validation or at the system level, the distinction between conventional and object-oriented software disappears. The test focuses on user-visible actions and user-recognized system outputs (Pressman, 2011).

Normally, validation can be defined in multiple ways simultaneously, but it only becomes successful when the software works in a way that can reasonably be close to what the customer expects.

The validation of the proposed solution is described, more precisely through the identification of the quantities that need to be used in order to evaluate the developed work, to test the proposed hypotheses, to evaluate the selected methodology and to apply the statistical tests to those same hypotheses.

- **Accelerometers Accuracy Analysis:** For this test, two sensors are used (Arduino 101) to measure the acceleration and the noise in the machine. So, it is necessary to prove that the sensors are reliable, namely from a precision point of view.

The sensors were configured at 2G (g-force) intervals (-2G maps to a raw value of -32768 and + 2G maps to a raw value of 32767). Furthermore, samples of both sensors, with a size of 4999, were collected, and for calibration reasons, and considering that a 3-axis accelerometer is used in order to minimize external noise, the analyzed values will translate the sum of the value of the 3-axis.

Therefore, one can use the hypothesis test, and considering that the variables are independent both samples are actually random. Since the population variance is Known and that samples are greater than 30, it can be concluded that the use of Z-Test of two samples for Mean (Figure 62) is the most appropriate test.

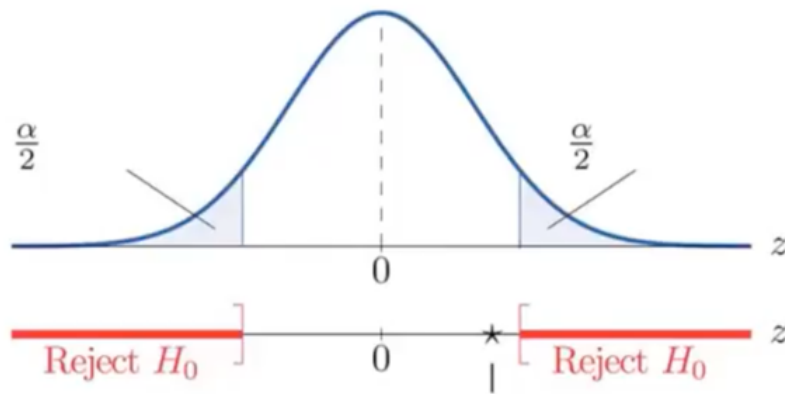


Figure 62 - Z-Test - Two-tail interval

Is there enough evidence (at $\alpha = 0,05$) to conclude that accelerometers mean precision does not differ significantly?

- 1) **Hypothesis:** $H_0: \mu_1(\text{sensor1}) = \mu_2(\text{sensor2})$ or $H_1: \mu_1 \neq \mu_2$
- 2) **Rejection Region:** $\alpha = 0,05$ and reject $Z:H_0$ if $Z < -1,96$ or $Z > 1,96$
- 3) **Test Statistic:** $z = 0,33$
- 4) **P-value** $= 0,75$

Decision/Conclusion:

- $z = 0,33$ is in the rejection region, the H_0 is not rejected
- P-value $> \alpha$, the H_0 is not rejected
-

There is not enough evidence to conclude that accelerometers mean precision does not differ significantly.

The Z-Test of two samples for Mean values are depicted in Table 11.

Table 11 - Z Test: Two samples for Mean

Z Test: Two samples for Mean		
	<i>Node 1</i>	<i>Node2</i>
Mean	1,0142	1,0006
Known Variance	0,0001	8,456
Observation	4999	4999
Hypothesized Mean	0	
z	0,3286	
P(Z<=z) one-tail	0,3711	
z critical one-tail	1,6448	
P(Z<=z) two-tail	0,7423	
z critical two-tail	1,9599	

- **Database storage analysis:** For this particular analysis, it is necessary to define the limit of the database storage, since there is Big data, and the ones that present limited hardware resources.

There is a relational database with 80 Gigabyte (GB) of storage limit, being necessary to analyze the data limits that need to be saved in the database, which represents the data history period. We use samples of the disk space consumption within a period of 8 days and from a single machine.

Hence, a descriptive analysis will be firstly conducted (Table 12), followed by an analysis of the worst-case limit.

Table 12 - Descriptive analysis of the Database storage

DESCRIPTIVE ANALYSIS	
MEAN	243,375 MB
MEDIAN	264 MB
RANGE	330 MB
MINIMUM	69 MB
MAXIMUM	399 MB
SUM	1947 MB
COUNT	8 Samples

Considering that the worst case presents a value of ~399 Megabyte (MB), this will be the value used to calculate the maximum history period without compromising the

integrity of the system. Given the fact that the database stores other smaller data, 10% of the total size will be safeguarded.

$$80 \times 0,90 = 72GB$$

The available total space has been reduced to 72GB. Furthermore, and for purely indicative purposes, it can be verified, and within a period of 8 days, that approximately 1947 MB of space was used.

However, and according to the analysis of the worst case:

$$399(MB) \times 8(d) = 3192 MB$$

With the sample of only one machine, the difference is not very significant. Nonetheless, and if we considered 1000 or more machines, the deviation would be very high.

Conclusion:

$$Limit = \frac{72 GB}{399MB} \cong 180 days$$

The maximum historical data period with the available resources for a machine will be of 180 days.

- **Middleware delay Analysis:** For this test, the RabbitMQ middleware, which is located on the **Edge Server** Subsystem, is used. Although the used infrastructures guarantee bandwidth(β), it is necessary to prove that stability exists, in order to guarantee the quality of service (QoS). Even though the system is made up of more Subsystems, these tests will focus on the middleware, considering that it is the central point and that it is responsible for receiving and delivering all the messages of the system.

Hence, samples were collected from ten consumers, and with a size of 7200MB. The samples collected were sent by ten producers, with a frequency of 1 s and with the same volume of data. It is essential to mention that there were no sending / receiving failures. To obtain more reliable results, the Network Time Protocol service was used, more precisely to synchronize the clocks of all the computers.

As an example, the used data in all tests represent a variable number of data type object, each object presenting values from 50 sensors with a single timestamp:

```
1:37:41.579 [main] INFO eu.mantis.tests.ConsumerTest - [x] Received '
[{"file":"10052016.log","id":""},
```

```
{"data":"12:25:01:062;;250039;250031;199988;0;44;2107465;2292519;0;-256;256;0;0;0;0;0;1;0;0;0;1;1;1;0;0;1;0;1;1;0;0;0;0;0;286;66;59;34;16;32;29;66;59;1;0;2022;2046;2037;2026"},
```

```
{"data":"12:25:01:250;;250010;250000;199988;0;44;2107465;2292519;0;-128;-128;0;0;0;0;0;1;0;0;0;1;1;1;0;0;1;0;1;1;1;0;0;0;0;0;286;67;56;38;17;31;29;67;56;0;0;2023;2043;2037;2030"},....]
```

Therefore, a descriptive analysis will be firstly conducted, which can be visualized in the table 13.

Table 13 - Descriptive analysis of the Middleware delay in seconds (s)

DESCRIPTIVE ANALYSIS	
MEAN	1,0430s
MEDIAN	1,043s
MODE	1,06s
STANDARD DEVIATION	0,0249s
SAMPLE VARIANCE	0,0006s
RANGE	0,086s
MINIMUM	1s
MAXIMUM	1,086s
SUM	75086,983s
COUNT	71990 Samples

The worst case presents a value of 1.086s, considering that the delay is quite relevant when providing Soft real-time data on the Cloud environment. However, it will be important to verify, namely through a Hypothesis test, whether the system behaves linearly in a real-time environment.

In this context, one can use a hypothesis test, and due to the fact that the variable is independent, the sample is random, the population variance is Known and the sample is greater than 30. Thus, it can be concluded that the use of Z-Test of one sample for Mean is the most appropriate test.

Is there enough evidence (at $\alpha = 0,1$) to conclude that middleware mean delay does not differ significantly from 1s?

- 1) **Hypothesis:** $H_0: \mu_1(\text{delay}) = 1,05$ or $H_1: \mu_1 \neq 1,05$
- 2) **Rejection Region:** $\alpha = 1$ and reject $Z:H_0$ if $Z < -1,65$ or $Z > 1,65$
- 3) Test Statistic: $z = -0,69$
- 4) P-value = 0,49

Decision/Conclusion:

- $z = -0,69$ is in the rejection region, the H_0 is not rejected
- P-value $> \alpha$, the H_0 is not rejected

There is no enough evidence to conclude that middleware mean delay does not differ significantly from 1s. The Z-Test of one sample for Mean values are depicted in Table 14.

Table 14 - Z Test: One samples for Mean

Z Test: One samples for Mean	
	<i>Consumers</i>
Mean	1,0430
Known Variance	0,0006
Observation	71990
Hypothesized Mean	1
z	-0,6980
P(Z<=z) one-tail	0,2425
z critical one-tail	1,2815
P(Z<=z) two-tail	0,4851
z critical two-tail	1,6448

- **Edge Local Subsystem scalability analysis:** The Edge Local Subsystem is located within each company, functioning as an entry-point between the local and the cloud elements of the system. Even though the infrastructures that are used within the companies are able to ensure a stable bandwidth, it is also necessary to define a machine's limit per factory.

There is a single point of exit (router), with 1 megabit per second of fixed upload rate. Since there may be some network noise, as well as other interferences, the upload rate will be reduced to 5% (margin safety) of the total size.

It is possible to conclude that there may be up to 67 machines to send data at the same time, with QoS guarantees at the **Edge Local** Subsystem delivery level.

Hence, samples with a size of 6000MB and from one producer were collected. The collected samples were sent, and at the same time, with a frequency of 1s and without any sending and receiving failure.

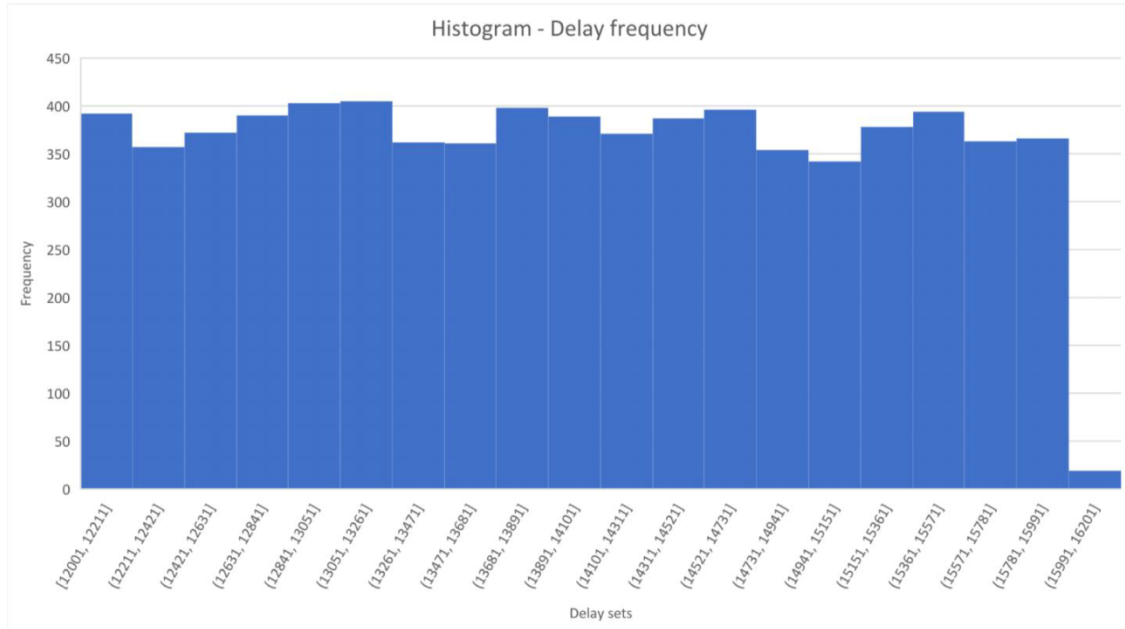


Figure 63 – Histogram of Edge Local Subsystem - Data packets delay frequency

Afterwards, the size of the sent packets will be analyzed, namely by a Histogram (Figure 63), which depicts the frequency of occurrence of the values of a particular set of data. It can be concluded that there is no normal distribution of frequency of data packages with groups of data that present a size of 200MB. There are frequencies between 19 (group 15991-16201) and 405 (group 1351-13261). The use of an intermediate value would constitute a risk, since the range of the frequencies presents a very high value. Therefore, we assume the worst case (16100 bytes) in order to ensure the maximum reliability.

Measurement calculations :

$$0.95 \text{ Megabit/s} \times 1\,000\,000 \approx 950\,000 \text{ bits}$$

$$950\,000 \text{ bits} \times \frac{1 \text{ byte}}{8 \text{ bits}} = 118\,750 \text{ bytes}$$

$$118\,750 \text{ bytes} \times \frac{1 \text{ Kilobyte}}{1024 \text{ bytes}} = 115.9667 \text{ Kilobytes/s}$$

$$16100 \text{ bytes(data packet)} \times \frac{1 \text{ Kilobyte}}{1024 \text{ bytes}} = 1.7226 \text{ Kilobytes/s}$$

$$\frac{115.966 \text{ Kilobytes/s (total upload bandwidth)}}{1.7226 \text{ Kilobytes/s (each Data packet)}} = 67 \text{ (machines)}$$

It is possible to conclude that there may be up to 67 machines to send data at the same time, with QoS guarantees at the Edge Local Subsystem delivery level.

- **Acceptance Tests:** To test the software and find errors, acceptance tests will be performed to different components of the framework. These tests focus on visible actions with user inputs and system outputs. All acceptance tests performed are available in **Appendix-A**.

It should be noted that, since several components have been developed containing many features, the documented tests focus on the most relevant features in the context of the framework. However, all functionalities have been properly tested in order to validate or even to avoid errors on nonlinear scenarios. Finally, the acceptance tests were recorded in video format and shown at one of the MANTIS project meetings in the city of Helsinki in Finland.

7.1.4 System

System tests are actually a series of different tests whose main purpose is to thoroughly exercise the computer-based system. Although each test has a distinct purpose, everyone works to verify that the elements of the system have been properly integrated and perform the functions assigned to them (Pressman, 2011).

These tests are divided into different sections: recovery, safety, performance and stress tests.

Since the performance and the security of the solution are important aspects of the non-functional requirements that the application must support, and considering that it is expected to have a large volume of data, tests will be presented in order to prove that they fulfil the requirements.

Since statistical tests are not applied, the tests will be carried out through certified tools, namely w3af⁴⁰ and Apache JMeter⁴¹.

- Security Testing of the REST components:

To perform security tests, the web application Security Scanner w3af (Figure 64)) was used, namely to audit the platform's *Manager Component* and through its "OWASP TOP 10" analysis tools. The goal was to find security issues and errors. The w3af application offers the following tests in the "OWASP TOP 10"⁴² section:

- 1) A1 Injection;
- 2) A2 Broken Authentication and Session Management;
- 3) A3 Cross-Site Scripting (XSS);
- 4) A4 Insecure Direct Object References;
- 5) A5 Security Misconfiguration;
- 6) A6 Sensitive Data Exposure;
- 7) A7 Missing Function Level Access Control;
- 8) A8 Cross-Site Request Forgery (CSRF);
- 9) A9 Using Components with Known Vulnerabilities;
- 10) A10 Unvalidated Redirects and Forwards.

The w3af application is mostly used to attack platforms that are in the Cloud. Therefore, and after the discovery of the existing vulnerabilities through audit processes, plugins are integrated to develop attacks. Thus, this is the main functionality that justifies why the w3af application is considered as an added value to the development of security tests.

The w3af has been configured through the OWASPTOP10 menu. However, Figure 64 depicts that the only change to the base configuration was the change of the auth (basic access authentication) to false, which means that this test type is not performed. The change occurred since the authentication is not used, considering that we applied an API-Key.

The obtained results revealed some vulnerabilities, more precisely at the level of permissions of files and folders, which is why they have been properly corrected. It should

⁴⁰ <http://w3af.org/>

⁴¹ <http://jmeter.apache.org/>

⁴² https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

be noted that the success of the results is due to the use of the stable version of grizzly and jersey, as well as to good programming practices, which were combined with the use of validation techniques in order to minimize security vulnerabilities.

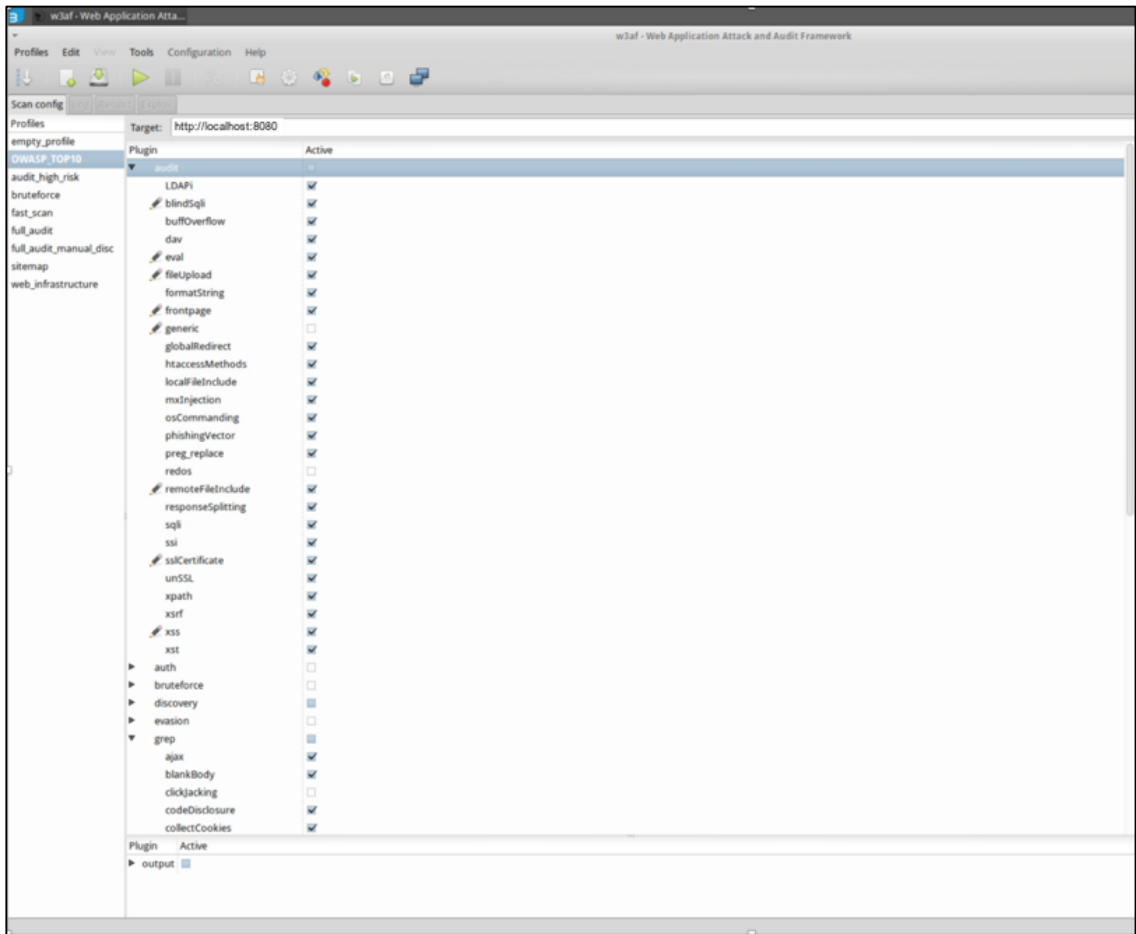


Figure 64 - W3AF OWASP TOP 10 - Audit

Finally, it is important to add that the tests will also take place in the final stage, more precisely to guarantee the exact same results.

Performance and stress tests of the REST components: In the performance and stress tests, the Apache JMeter tool was used, specifically in its "3.2" version. Figure 65 e 66 depict that both tests were performed on different *Manager Component* routes, as well as analyzed by the use of graphs and tables.

The tests were performed with 150 and 300 requests per second, for a period of 10 minutes and a 100 Megabit/s upload/download. It is important to specify that a fiber optic connection was used.

The Graph Results generated a simple graph that combines all the sample's times. Along the bottom of the graph, the current sample (black), the current average of all samples

(blue), the current standard deviation (red), and the current throughput rate (green) are displayed, more precisely in milliseconds.

However, it is crucial to emphasize that the throughput number represents the actual number of requests/minute handled by the server. This calculation includes any delays that are added to the test, as well as JMeter's own internal processing time.

In the graphs of the figures is depicted the following information:

- Data - plot the actual data values (Not active to avoid noise);
- Average - plot the Average;
- Median - plot the Median (midway value);
- Deviation - plot the Standard Deviation (a measure of the variation);
- Throughput - plot the number of samples per unit of time;
- The value displayed on the top left of graph is the max of 90th percentile of response time.

The tested *Manager Component* routes were the following: Create Enterprise, Delete Enterprise, Create Machine, Delete Machine, Create Sensor, Init Cloud System, Close Cloud System, Get System Status and Reload Factory.

Thus, through the graphs, as well as through the summary report, we can observe the values that are represented in Table 15

Table 15 - *Manager Component* performance and stress tests

Requests Seconds	Samples	Average Response Time Milliseconds	Min Response Time Milliseconds	Max Response Time Milliseconds	Std. Deviation Milliseconds	Error %	Throughput bit/sec
150	42182	1885	1837	1941	15	0	4.757,314
300	48026	3756	1992	3836	190	8,3	4.772,494

We can conclude that, with 150 requests, the component does not present failures, meeting an acceptable response time (average 1885ms). However, and by increasing the value up to 300 requests, the component actually presents ~8,3% of errors and a double response time (average 3756ms). Another important fact is related to the conclusion that the samples only increased by 13.9% when the requests were doubled.

Hence, this component presents a problem for the system, in terms of performance, being highly suggested to update the contracted hosting server (Azure) in order to have

a scalable solution, or even to purchase a Virtual Server (VPS), with at least 4 CPUs, 4 GB ram and a port speed > 100 Megabit/s to achieve better results.

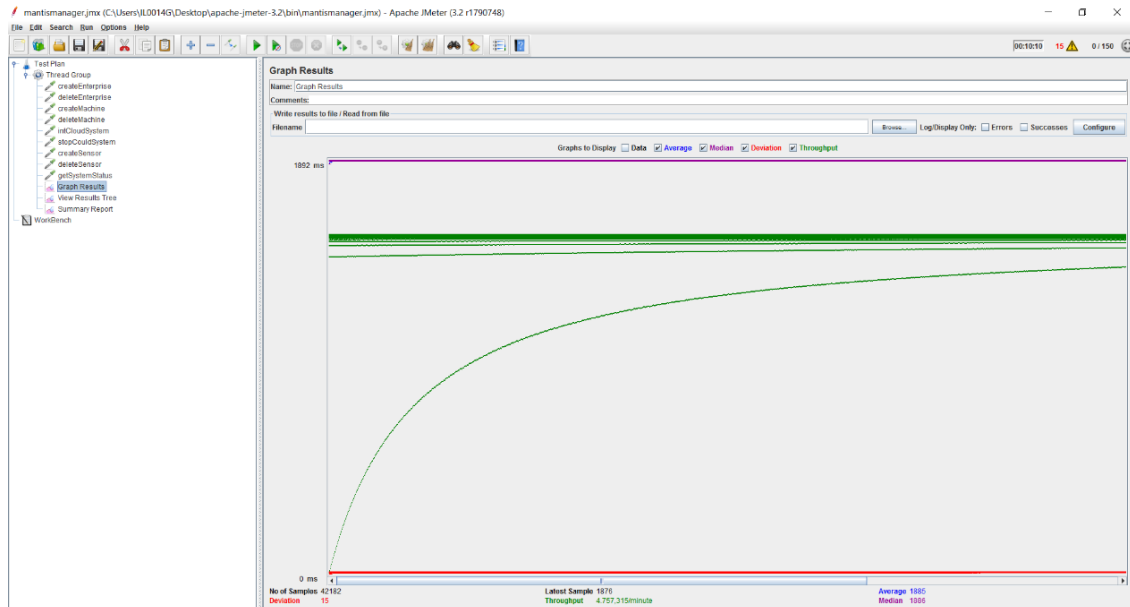


Figure 65 - Apache Jmeter - Graph Report 150 request per second

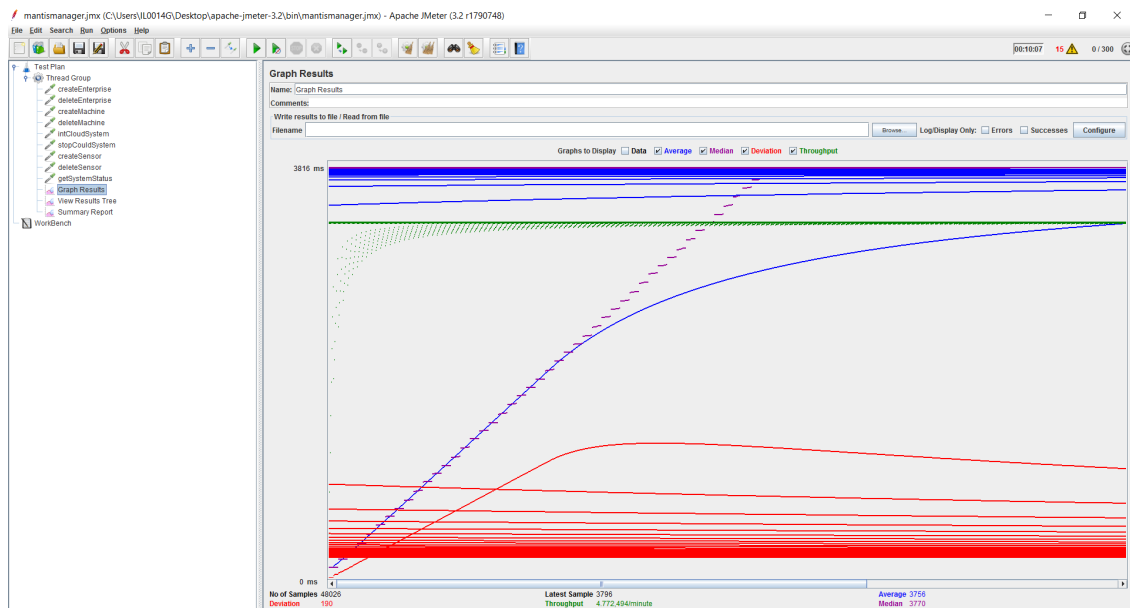


Figure 66 - Apache Jmeter - Tests Graph Report 300 request per second

The code related to the tests of the implemented components can be consulted in the Bitbucket repository of the present project, and it is under constant updates.

7.2 Conclusions

The software testing referred to verification and validation applied to a set of activities ensuring that the software correctly implements the specific functionalities and that meets the stakeholder's requirements. Thus, several tests were performed in order to test the main functionalities of the system with successful results.

Finally, since several components have been developed, it will be necessary to evaluate the need to perform additional tests, validation and to guarantee more code coverage.

8 Conclusions

This chapter summarizes the work that has been highlighting its main contributions. In this section, we provide answers to the questions that are presented in the beginning of the present work and describe the future work to be accomplished regarding the implementation and delivery of the solution to cover all its requirements, which, ultimately, meet the principles of Industry 4.0.

8.1 Accomplished objectives

The present work has been very appealing, since it was possible to explore a new way of attesting the acquired competences and to perceive the that is being followed by the Industry in its path to digitalization in a real context.

At the start of this work it was necessary to understand the general architecture of the existing ecosystem, the processes that are associated with it, the development methodology which would allow the integration of this part with the overall MANTIS project and with the existing legacy infrastructure existing on the machines.

After analysing the state of the art, it was required to evaluate the possible solutions for the middleware, also to provide a Value Analysis on the different hypothesis and to elaborate a Business Model for the exploitation of the solution.

Overall, we can argue that the present project groups a set of functionalities related with the context of the Industry 4.0 which completely fulfils all the objectives defined in Chapter 1. The following list resumes the results achieved for the framework in relation with the objectives 04 and 05:

- **O4.1:** A OPC-UA component, running on the machine, that has the responsibility of collecting machine sensors and actuators data and places it on the middleware through the Edge Local;
- **O4.2:** The *Sensor* Component uses a wireless accelerometers which are able to monitor the machine's moving parts. It transmits that information via Bluetooth to the Mantis-PC, which, by its turns connects with the middleware through the **Edge Local**;

- **O4.3:** The **Edge Local** a component has the responsibility of collecting, pre-processing and sending data to cloud through a middleware, securely isolating a machine inside the factory from the Internet. This component can also provide some other advanced functionalities (e.g. conversion between protocols) and an interface (**HMI**) with data visualization;
- **O4.4:** The *Middleware Component* on the **Edge Server** subsystem, which has the responsibility of enabling communication and management of data between distributed components, namely the *Middleware Client* component from Data Analysis, the History and *Manager Components* from **Edge Server** subsystem and *Middleware Client* component from **Edge Local** subsystem;
- **O4.5:** The *Manager Component* is an interface that works as a gateway between the Component **HMI**, with the *Middleware Component* and the *History Component*;
- **O4.6:** The *History Component* which enables the storing of historical data on a database. Interfaces that have the responsibility of consuming and producing data to and from the middleware, which can be integrated by the **HMI** and **Data Analysis** modules;
- **O4.7:** The *Middleware Client* component that has the responsibility of consuming and producing data to and from the middleware.

Furthermore, to integrate the **HMI** to with the middleware, to maintain and administer the system and to support the display of graphical data related with the accelerometers (which required a high-performance **HMI**). Support and implement some of the framework's external modules were developed:

- **O5.1:** The graphical libraries for **HMI** with soft real-time;
- **O5.2:** The library in the **HMI** subsystem with direct access to the middleware in order to collect with soft real-time;
- **O5.3:** The configuration the Edge sever with FTP, IIS, middleware and services security (SSL/HTTPS), etc.

In sum, we can conclude that the implemented solution was based on a modular and flexible architecture, which makes it possible to only integrate part of the components or services. This work had an important component in the analysis of the solutions (mostly focusing on open source and free tools), which can be available, in the context of Industry 4.0 and allowed, the identification of existing limitations regarding the scalability of the solution:

- Relational database should be switched to a document database;
- The *History Component* must be provided with load balancer, which allows it to run on multiples machines;

- The **Edge Server** subsystem of the solution should be deployed a web server that provides maximum quality of service, for example Azure.

Finally, some papers were published aiming to report the results of the work that has been developed during this dissertation and some other which are not directly related with this thesis, but were performed in collaboration with the authors.

- **The Industrial Internet of Things** (Albano, Silva & Ferreira, 2017)
 - AUTHORS: Michele Albano, José Silva, Luís Lino Ferreira
 - CONFERENCE: Demo in 22º Seminário da Rede Temática Comunicações Móveis (RTCM 2017). 18, Jan, 2017, Session III. Lisboa.
- **A Pilot for Proactive Maintenance in Industry 4.0** (Ferreira, *et al.*, 2017)
 - AUTHORS: Luis Lino Ferreira, Michele Albano, José Silva, Diogo Martinho, Goreti Marreiros, Giovanni di Orio, Pedro Maló, Hugo Ferreira
 - CONFERENCE: Accepted in 13th IEEE International Workshop on Factory Communication Systems (WFCS 2017). 31, May to 2, Jun, 2017. Trondheim, Norway.
- **Interoperable and Interconnected CPS-populated Systems for Proactive Maintenance (Appendix-J)**
 - AUTHORS: Giovanni Di Orio, Pedro Maló, Csaba Hegedus, Michele Albano, Luis Lino Ferreira, José Silva, Pal Varga and Istvan Moldovan.
 - CONFERENCE: 22nd IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2017.
- **Application of Sensors for Proactive Maintenance in the Real World (Appendix-L)**
 - AUTHORS: Michele Albano, Luis Lino Ferreira, José Silva, Edgar M. Silva, Pedro Maló, Godfried Webers, Jarno Junnola, Erkki Jantunen, Luis Miguel Vega.
 - CONFERENCE: 22nd IEEE International Conference on Emerging Technologies And Factory Automation - ETFA 2017.
- **Quality of Service on the Arrowhead Framework** (Albano, *et al.*, 2017)
 - AUTHORS: Michele Albano, Paulo Barbosa, José Silva, Roberto Duarte, Luis Lino Ferreira, Jerker Delsing.

- CONFERENCE: 13th IEEE International Workshop on Factory Communication Systems (WFCS 2017). 31, May to 2, Jun, 2017. Trondheim, Norway.
- **FlexHousing: Flexoffer concept for the energy manager (Appendix-N)**
 - AUTHORS: Joss Santos, Michele Albano, Luis Lino Ferreira, José Silva, Petur Olsen and Luisa Matos
 - CONFERENCE: 22nd IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2017.
- **Maintenance Supported by Cyber-Physical Systems and Cloud Technology (Appendix-O)**
 - AUTHORS: Michele Albano, Luis Lino Ferreira, José Silva, Erkki Jantunen, Jarno JunnolaUnai Gorostegui.
 - CONFERENCE: 22nd IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2017.

In the next subchapter, we present some ideas of what might be done in the future, more precisely within the context of this thesis and regarding other projects that are quite similar to the developed work.

8.2 Future work

It may be necessary to adjust several theoretical aspects, more precisely to better match the needs of stakeholders and allow to add more features. One of the main objectives will also be related to providing the source code that is available on a project sharing platform, such as Github, especially collaboration for further development and testing of this solution.

There is little work ahead in order to complete and make available the solution developed during this dissertation. The architecture and the developed prototype present good results, considering that they are being tested in a real context. Nonetheless, some adjustments to the software might be necessary. Additionally, and since several components have been solely developed by the author of the present project, it will be necessary to evaluate the need to perform additional tests, validation and to guarantee more code coverage.

It should also be noted that the **HMI** should be more flexible, specifically in the implementation of the interoperability models and in the guarantee of the integration's resilience. As it was previously described, performance issues, optimizations and adjustments in services must be considered.

Finally, and since there are several components to deploy, it is essential to provide support in order to create a stable solution. Hence, the ideal would be to take a corrective and evolutionary approach to maintenance, in order to further evolve the solution.

9 Bibliography

- Douglas, B., David, Dick. (2013) Web Services, Service-Oriented Architectures, and Cloud Computing, Second Edition: The Savvy Manager's Guide (The Savvy Manager's Guides), CA: Morgan Kaufmann Publishers
- Åkeson, L. (2016). Industry 4.0: Cyber-Physical Systems and their impact on business models. Master thesis presented at the Karlstads Universitet, Sweden.
- Albano, M., Barbosa, P., Silva, J., Duarte, R., Ferreira, L., Jerker, D. (2017). Quality of Service on the Arrowhead Framework. 13th IEEE International Workshop on Factory Communication Systems, WFCs
- Albano, M., Ferreira, L. L., Pinho, L. M., & Alkhawaja, A. R. (2015). Message-oriented middleware for smart grids. *Computer Standards & Interfaces*, 38, 133-143.
- Allenhof, S. (2015). Implementation of an example application facilitating Industrie 4.0. Sweden: Chalmers University of Technology.
- Alves, P. G. (2014). A distributed security event correlation platform for SCADA. Coimbra: University of Coimbra.
- AMQP (2017). Advanced Message Queueing Protocol. Available at: <https://www.amqp.org/>. Accessed on 27/04/2017.
- Anderl, R. (2014). Industrie 4.0- Advanced engineering of Smart Products and Smart Production. Technological Innovations in the Product Development, 19th International Seminar on High Technology, Brazil.
- Ardichvili, A., Cardozo, R., & Ray, S. (2003). A theory of entrepreneurial opportunity identification and development. *Journal of Business Venturing*, 18(1), 105-123.
- Armbruster, H., Bikfalvi, A., Kinkel, S., & Lay, G. (2008). Organizational innovation: The challenge of measuring non-technical innovation in large-scale surveys. *Technovation*, 28(10), 644-657.
- Baldassarre, C., Daga, E., Gangemi, A., Gliozzo, A., Salvati, A., & Troiani, G. (2010). Semantic scout: Making sense of organizational knowledge. *Knowledge Engineering and Management by the Masses*, 272-286.
- Barquet, A. P. B., Oliveira, M. G., Amigo, C.R., Cunha, V. P., & Rozenfeld, H. (2013). Employing the business model concept to support the adoption of product-service systems (PSS). *Industrial Marketing Management*, 42(5), 693-704.
- Bauer, W., Schlund, S., Marrenbach, D. & Ganschar, O. (2014). Industrie 4.0 – Volkswirtschaftliches Potenzial für Deutschland. Available at: <https://www.bitkom.org/Themen/Digitale-Transformation-Branchen/Industrie-40/index.jsp>. Accessed on 27/04/2017.
- Bauernhansl, T., ten Hompel, M. & Vogel-Heuser, B. (2014). Industrie 4.0 in produktion, automatisierung und logistik : Anwendung, technologien und migration. Berlin: Springer Vieweg.
- Bhuiyan, N. (2011). A framework for successful new product development. *Journal of Industrial Engineering and Management*, 4(4), 746-770.
- Boukouchi, Y., Marzak, A., Benlahmer, H., & Moutachauik, H. (2013). Comparative study of software quality models. *International Journal of Computer Science Issues*, 10(6), 309-314.
- Brettel, M., Friederichsen, N., Keller, M. & Rosenberg, M. (2014). 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*, 8(1), 37-44.

- Brisk Insights (2016). Operational predictive maintenance market by component (solutions, services), by deployment type (cloud deployment, by on premises deployment), by application (automotive, energy and utilities, healthcare, manufacturing, government & defense, transport and logistics), industry size, growth, share and forecast to 2022. Available at: <http://www.briskinsights.com/report/operational-predictive-maintenance-market>. Accessed on 5/05/2017.
- Broy, M., Gleirscher, M., Kluge, P., Krenzer, W., Merenda, S., & Wild, D. (2009). Automotive architecture framework: Towards a holistic and standardized system architecture description. Munich: Technische Universität München.
- Bundesministerium für Bildung und Forschung (2017). Industrie 4.0. Available at: <https://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html>. Accessed on 27/04/2017.
- Bürger, T. & Tragl, K. (2014). SPS-Automatisierung mit den technologien der IT-Welt verbinden. In T. Bauernhansl, M. ten Hompel & B. Vogel-Heuser, Industrie 4.0 in produktion, automatisierung und logistik : Anwendung, technologien und migration. Berlin: Springer Vieweg, p. 559-569.
- Buxmann, P, Hess, T. & Ruggaber, R. (2009). Internet of services. Business & Information Systems Engineering, pp. 341-342.
- Chen, Y. (2017). Performance of message queue telemetry transport protocol and constrained application protocol in wireless sensor networks. Oxford: University of Mississippi.
- Chung, J.-Y. & Chao, K.-M. (2007). A view on service-oriented architecture. SOCA: Service Oriented Computing and Applications, 1(2), 93-95.
- Clarke III, I. (2001). Emerging value propositions for M-commerce. Journal of Business Strategies, 18(2), 133-148.
- Correia, M. A. S. (2014). Industrie 4.0: Framework, challenges and perspectives. Master Thesis presented to the Faculty of Engineering, University of Applied Science, Rüsselsheim.
- Dais, S. (2014). Industrie 4.0 – Anstob, vision, vorgehen. In T. Bauernhansl, M. ten Hompel & B. Vogel-Heuser, Industrie 4.0 in produktion, automatisierung und logistik : Anwendung, technologien und migration. Berlin: Springer Vieweg, p. 625-634.
- DDS (2017). Data Distribution Service. Available at: <http://www.omg.org/spec/DDS/>. Accessed on 27/04/2017.
- DKE (2014). The German standardization roadmap Industrie 4.0. Available at: https://www.dke.de/de/std/documents/rz_roadmap%20industrie%204-0_engl_web.pdf. Accessed on 27/04/2017.
- Drath, R. & Horsch, A. (2014). Industrie 4.0: Hit or hype? IEEE Industrial Electronics Magazine, 8(2), 56-58.
- Edvardsson, B. & Olsson, J. (1996). Key concepts for new service development. The Service Industries Journal, 16(2), 140-164.
- Evans, P. C. & Annunziata, M. (2012). Industrial Internet: Pushing the boundaries of minds and machines. Available at: http://www.ge.com/docs/chapters/Industrial_Internet.pdf. Accessed on 27/04/2017.
- Faisal, I. K. & Mahmoud, M. H. (2003). Risk-based maintenance (RBM): a quantitative approach for maintenance/inspection scheduling and planning. Journal of Loss Prevention in the Process Industries, 16(6), 561-573.
- Fast-Berglund, A., Åkerman, M., Karlsson, M. Hernández, V. G. & Stahre, J. (2014). Cognitive automation strategies. Variety Management in Manufacturing. Proceedings of the 47th CIRP Conference on Manufacturing Systems.

Fernandes, J. L. (2011). Web services approach for ambient assisted living in mobile environments. Master thesis presented at the University of Beira Interior, Portugal.

Ferreira, L., Dam P., Loopstra K., Tijsma B., Zurutuza U., Pradera O., Bergmann A., Sprong H. (2017). D3.6 Report on sensors selected and to be developed. Derivable related to Work Package 3 - Smart sensing and data acquisition technologies of the MANTIS European project.

Ferreira, L., Topan, E., Tijsma, B., Dam, P., Terwee, Daan. (2016). D3.1 Report on sensors selected and to be developed. Derivable related to Work Package 3 - Smart sensing and data acquisition technologies of the MANTIS European project.

Ferreira, L., Albano, M., Silva, J., Duarte, R., Martinho, D., Marreiros, G., Orio, G., Maló, P., Ferreira, H. (2017). A Pilot for Proactive Maintenance in Industry 4.0. 13th IEEE International Workshop on Factory Communication Systems, WFCS

Fettke, P. & Loos, P. (2007). Reference modeling for business systems analysis. USA: Idea Group Publishing.

Fong, J., Wong, H. K., & Cheng, Z. (2003). Converting relational database into XML documents with DOM. *Info. & Soft. Tech.*, 45, 335-355.

Förderschwerpunkte Industrie 4.0 (2017). Federal Ministry of Education and Research in Germany. Available at: <http://www.softwaresysteme.pt-dlr.de/de/industrie-4-0.php>. Accessed on 27/04/2017.

Giusto, D., Iera, A., Morabito, G. & Atzori, L. (2010). The internet of things. Berlin: Springer Vieweg.

Gordijn, J. & Akkermans, J. (2003). Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2), 114-134.

Hermann, M., Pentek, T. & Boris, O. (2015). Design principles for Industrie 4.0 scenarios: a literature review. Dortmund: Technical University Dortmund.

Holmberg, K., Jantunen, E., Bellew, J., Albarbar, A., Starr, A., & Al-Najjar, B. (2010). Maintenance today and future trends. *E-maintenance*, 5-37. DOI: 10.1007/978-1-84996-205-6_2.

Hoyer, W. D., Chandy, R., Dorotic, M., Krafft, M., & Singh, S. S. (2010). Consumer cocreation in new product development. *Journal of Service Research*, 13(3), 283-296.

Jantunen, E., Zurutuza, U., Ferreira, L., & Varga, P. (2016). Optimising maintenance: what are the expectations for Cyber-physical Systems. 3rd International IFIP Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems, Vienna, Austria.

Jantunen, E., Barbella, Michele., Sannino, P., Garcia, A., Jantunen, E., Herman, M., Pal, V., Maló, P., Stefano, V., Mario, R., Bence, P., Anja, V., Silva, E., Hegedüs, C. (2016). D1.6 MANTIS Platform Requirements. Derivable related to Work Package 3 - WP1 - Service platform architecture requirement definition. JMS (2017).

Kagermann, H. (2014). Chancen von Industrie 4.0 nutzen. In T. Bauernhansl, T. ten Hompel & B. Vogel-Heuser, *Industrie 4.0 in produktion, automatisierung und Logistik. Anwendung, Technologien und Migration*. Berlin: Springer Vieweg, p. 603-614.

Kagermann, H., Lukas, W.-D. & Wahlster, W. (2011). *Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. Industriellen revolution*. Available at : <http://www.vdi-nachrichten.com/Technik-Gesellschaft/Industrie-40-Mit-Internet-Dinge-Weg-4-industriellen-Revolution>. Accessed on 27/04/2017.

Kagermann, H., Lukas, W.-D. & Wahlster, W. (2013). Final report of the Industrie 4.0 working group. Recommendations for implementing the strategic initiative Industrie 4.0.

- Koch, V., Kuge, S., Geissbauer, D.R. & Schrauf, S. (2015). Opportunities and challenges of the industrial internet. Available at: <http://www.strategyand.pwc.com/reports/industry-4-0> Accessed on 28/04/2017
- Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'Amore, R., et al. (2001). Providing clarity and a common language to the "Fuzzy Front End". *Research-Technology Management*, 44(2), 46-55.
- Kolberg, D. & Zühlke, D. (2015). Lean automation enabled by Industry 4.0 technologies. *IFAC-PapersOnLine*, 48(3), 1870-1875.
- Lasi, H., Peter, F., Kemper, H.-G., Feld, T. & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, 6(4), 239-242.
- Lee, J., Bagheri, B. & Kao, H.-A. (2015). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3, 18-23.
- Lee, J., Kao, H.-A. & Yang, S. (2014). Service innovation and Smart analytics for Industry 4.0 and big data environment. *Procedia CIRP*, 16, 3-8.
- Meertens, L. O., Iacob, M. E., Nieuwenhuis, L. J. M., Sinderen, M. J., Jonkers, H., & Quartel, D. (2012). Mapping the business model canvas to ArchiMate. *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, Italy.
- Mikusz, M. (2014). Towards an understanding of cyber-physical systems as industrial software-product-service systems. *Procedia CIRP*, 16, 385-389.
- Mittermair, M. (2015). Industry 4.0 Initiatives. *SMT: Surface Mount Technology*, 30(3), 58-63.
- Mobley, R. K. (2002). *An introduction to predictive maintenance*. (2nd ed). St.Louis: Butterworth-Heinemann.
- Moffitt, J. (2010). *Professional XMPP programming with JavaScript and jQuery*. New Jersey: John Wiley & Sons.
- Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, 17, 9-13.
- MQTT (2017).
- Nathan, S. (2015). Getting to grips with Industry 4.0. *Engineer* (00137758), 296(7867), 30-34.
- Osterwalder, A., Pigneur, Y., Bernarda, G., & Smith, A. (2014). *Value proposition design. How to create products and services customers want*. Hoboken, NJ: Wiley.
- Platform Industrie 4.0 (2017). Platform Industrie 4.0. Available at <http://www.plattform-i40.de/I40/Navigation/DE/Home/home.html>. Accessed on 27/04/2017.
- Posada, J., Toro, C., Barandiaran, I., Oyarzun, D., Stricker, D., Amicis, R., Pinto, E. B. et al. (2015). Visual computing as a key enabling technology for Industrie 4.0 and industrial internet. *IEEE Computer Graphics and Applications*, 35(2), 26-40.
- Pressman, Roger S. (2011) *Engenharia de software: uma abordagem profissional*. 7. ed. Porto Alegre: AMGH
- Pujari, P. (2016). Enabling Omron's IPC for Industry 4.0, M2M and IIoT. Master thesis presented at the Eindhoven University of Technology, Eindhoven.
- Robert, A., Bradford, W. (2015). Technical Report of Industrial Internet Reference Architecture. DOI: 10.13140/RG.2.2.18076.90249.
- Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *Int. J. Services Sciences*, 1(1), 83-98.
- Sadeghi, A.-R., Wachsmann, C. & Waidner, M. (2015). Security and privacy challenges in industrial internet of things. *Design Automation Conference DAC*.

Saint-Andre, P., Smith, K., & Tronçon, R. (2009). XMPP: The definitive guide – building real-time applications with Jabber technologies. Farnham: O'Reilly.

Saint-Exupery, A. (2009). Internet of Things. Strategic Research Roadmap. Available at: http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf. Accessed on 27/04/2017.

Scheer, A.-W. (2013). Industrie 4.0. Wie sehen Produktionsprozesse im Jahr 2020 aus? IMC AG: Saarbrücken.

Schlechtendahl, J., Keinert, M., Kretschmer, F., Lechler, A. & Verl, A. (2015). Making existing production systems Industry 4.0-ready. *Production Engineering*, 9(1), 143-148.

Schlick, J., Stephan, P., Loskyll, M. & Lappe, D. (2014). Industrie 4.0 in der praktischen Anwendung. In T. Bauernhansl, T. ten Hompel & B. Vogel-Heuser, *Industrie 4.0 in produktion, automatisierung und Logistik. Anwendung, Technologien und Migration*. Berlin: Springer Vieweg, p. 57-84.

Seebacher, R. T. (2013). Messaging challenges in a globally distributed network. Master thesis presented at Open Systems, Switzerland.

Shafiq, S. I., Sanin, C., Szczerbicki, E. & Toro, C. (2015). Virtual engineering object/virtual engineering process: A specialized form of Cyber-physical System for Industrie. *Procedia Computer Science*, 60, 1146-1155.

Shin, S.-J., Woo, J. & Rachuri, S. (2014). Predictive analytics model for power consumption in manufacturing. *Procedia CIRP*, 15, 153-158.

Smith, J. B. & Colgate, M. (2007). Customer value creation: A practical framework. *Journal of Marketing Theory and Practice*, 15(1), 7-23.

Sommerville, I. (2011). *Software Engineering* (9 Ed.), Pearson

STOMP (2017).

Sweeney, J. C. & Soutar, G. N. (2001). Consumer perceived value: The development of a multiple item scale. *Journal of Retailing*, 77(2), 203-220.

Ta, X. (2006). A quality of service monitoring system for service level agreement verification. Australia: University of Sydney.

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Babar, M. A. (2010). A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, 83(3), 352-370.

Teixeira, C. R. S. (2015). Middleware for large-scale distributed systems. Master thesis presented at the Instituto Superior de Engenharia do Porto, Portugal.

Traa, J. W. A. (n.d.). Rational Unified Process vs. Microsoft solutions framework: A comparative study. The Netherlands: Erasmus University Rotterdam.

Trimi, S. & Berbegal-Mirabent, J. (2012). Business model innovation in entrepreneurship. *International Entrepreneurship and Management Journal*, 8(4), 449-465.

Varga, P., Blomstedt, F., Ferreira, L. L., Eliasson, J., Johansson, M., Delsing, J., & Soria, I. M. (2016). Making system of systems interoperable – the core components of the Arrowhead Framework. *Journal of Network and Computer Applications*, 1-26.

Wang, S., Wan, J. W., Li, D. & Zhang, C. (2016). Implementing Smart Factory of Industrie 4.0: An outlook. *International Journal of Distributed Sensor Networks*, 1-10.

Weiser, M. (1991). The computer for the 21st century. Available at: <https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf>. Accessed on 27/04/2017.

- Weyer, S., Schmitt, M., Ohmer, M. & Gorecky, D. (2015). Towards Industry 4.0 – Standardization as the crucial challenge for highly modular, multi-vendor production systems. *IFAC-PapersOnLine*, 48(3), 579-584.
- Woodruff, R. B. (1997). Customer value: The next source for competitive advantage. *Journal of the Academy of Marketing Science*, 25, 139.
- Xiong, R. (2008). *Leadership in project management*. Georgia: Georgia Institute of Technology.
- XMPP (2017).
- Yunpeng, X. (2010). A message queue based event notification system: Football lottery system. Master thesis presented at the Department of Electrical and Computer Engineering of the University of Stavanger, Norway.
- Zeithaml, V. A. (1988). Consumer perceptions of price, quality, and value: A means-end model and synthesis of evidence. *Journal of Marketing*, 52(3), 2-22.
- Zezulka, F., Marcon, P., Vesely, I., Sajdl, O. (2016). Industry 4.0 – An Introduction in the phenomenon. *IFAC-PapersOnLine* 49-25, 008–012

10 Appendixes

10.1 Appendix-A- Acceptance Tests

Table 16 – Start Machine Internal Sensors Data - Acceptance Tests

Features:	11 <i>OPC-UA server</i> Component; <i>CNCBender</i> Component		
Objective:	Test OPC-UA server start/stop and data acquisition from the machine sensors using the shared memory of the CNCBender component.		
Test Method:	Manual		
Scenario	Test	Expected result	Validation
The employee initiates the <i>OPC-UA server</i>	Start the executable and click the “start server” option menu	Receives an event message with information about the server endpoint	True
The employee checks the clients connected to the server	Click the “check connected clients” option menu	Receives an event message with information about the connected clients	True
The employee stops the <i>OPC-UA server</i>	Click the “stop server” option menu	Receives an event message with information about disconnected clients and turns off the server	True

Table 17 – Start Machine External Sensors Data - Acceptance Tests

Features:	Sensors Component; <i>BLE server</i> Component; <i>OPC-UA server</i> Component of the Mantis-PC Subsystem		
Objective:	Test sensors start/stop and data acquisition from the BLE Server. Next, the that then sends the data to the OPC-UA Server of the Mantis-PC Subsystem.		
Test Method:	Manual		
Scenario	Test	Expected result	Validation
The employee places the sensors in the machine and go to the Mantis-PC starts the BLE Serve and the <i>OPC-UA server</i>	Starts the executables on the Mantis-PC and click the “Pair Devices” option menu	Receives an event message with information about the Pairing success	True
The employee removes the sensors in the machine and go to the Mantis-PC	On the Mantis-PC Click the “re-calibrate” option menu	Receives an event message with information about the calibration success	True
The employee removes the sensors in the machine and go to the Mantis-PC stops the BLE Serve and the <i>OPC-UA server</i>	Stops the executables on the Mantis-PC	Receives an event message with information about the unpairing devices	True

Table 18 – HMI Local - Acceptance Tests

Features:	Sensors Component; <i>BLE server</i> Component; <i>OPC-UA server</i> Component of the Mantis-PC Subsystem; <i>OPC-UA server</i> Component of the Machine Subsystem; OPC-UA Client Component; <i>Middleware Component</i> ; <i>Middleware Client</i> Component		
Objective:	Test data visualization and data acquisition for files in the HMI. The data is received from the internal and external sensors of the machine. Next, configure the HMI to send data for the cloud.		
Test Method:	Manual		
Scenario	Test	Expected result	Validation
The employee views the machine internal sensors data in Real-Time	Accesses the Local Subsystem and through the <i>Node-Red</i> dashboard click on the "view webpage machine" option	Receives data in real-time graphs with adjustable cache in the browser	True
The employee views the machine external sensors data in Real-Time	Accesses the Local Subsystem and through the <i>Node-Red</i> dashboard click on the "view webpage sensors" option	Receives data in real-time graphs with adjustable cache in the browser	True
The employee views the data in files	Accesses the Local Subsystem and through the <i>Node-Red</i> dashboard click on the "view file" option	Receives a green light on the file icon	True
The employee sets up the flow in the dashboard in order to send the data to the cloud	Accesses the Local Subsystem and through the <i>Node-Red</i> dashboard Resets the flow to connect the incoming data input to the <i>Middleware Client</i> output	Receives a green light on the <i>Middleware Client</i> icon	True (The middleware was started before the tests)

Table 19 – HMI Web - Acceptance Tests

Features:	Sensors Component; <i>BLE server</i> Component; <i>OPC-UA server</i> Component of the Mantis-PC Subsystem; <i>OPC-UA server</i> Component of the Machine Subsystem; OPC-UA Client Component; <i>Middleware Component</i> ; <i>Middleware Client</i> Component; <i>History Component</i> ; <i>Manager Component</i> ; HMI API; E-mail Component		
Objective:	Test all the HMI features provided by the Framework (Start Cloud-System, Stop Cloud-System, Create Factory, Delete Factory, Create Machine, Delete Machine, View Soft-Real-time Internal Machine Sensor Data, Create Sensor, Delete Sensor, View Soft-Real-time External Machine Sensor Data, View History Data, View Analysis Data). It is assumed that the employee is using a browser on a computer and already insert the IP/DomainName of the HMI Web.		
Test Method:	Manual		
Scenario	Test	Expected result	Validation
The employee starts the cloud system	In the HMI click on the "start system" option of the System menu	Receives an event message with information about the starting cloud-system success and the property configuration of the entire system base (e.g., user, permissions, queues and history)	True
The employee stops the cloud system	In the HMI click on the "stop system" option of the System menu	Receives an event message with information about the stopping cloud-system success and the elimination of the basic system configuration (e.g., user, permissions, queues, history)	True
The employee creates a factory	In the HMI click on the "add factory " option of the Factories menu	Receives an event message with information about the factory creation success and the configuration the system for that factory (e.g. user, permissions, queue, history)	True
The employee deletes a factory	In the HMI click on the "delete factory " option of the Factories menu	Receives an event message with information about the machine deletion success and the elimination of the configuration the system for that factory (e.g. user, permissions, queue, history)	True
The employee creates a machine	In the HMI click on the "add machine" option of the Machines menu	Receives an event message with information about the machine creation success and the configuration of the system for that machine (e.g. queues and history)	True
The employee deletes a machine	In the HMI click on the "delete machine" option of the Machines menu	Receives an event message with information about the factory deletion success and the elimination of the system for that machine (e.g. queues and history)	True
The employee creates a sensor	In the HMI click on the "add sensor" option of the Sensors menu	Receives an event message with information about the sensor creation success and the configuration the history system for this sensor	True
The employee deletes a sensor	In the HMI click on the "delete sensor" option of the Sensors menu	Receives an event message with information about the sensor deletion success and the elimination of the history system for this sensor	True

The employee visualizes the internal machine sensor data in Soft-Real-time	In the HMI click on the "machine live data" option of the MachineActions menu	Receives data in soft-real-time graphs with adjustable cache in the browser and other option	True
The employee visualizes the external machine sensor data in Soft-Real-time	In the HMI click on the "sensors live data" option of the MachineActions menu	Receives data in soft-real-time graphs with adjustable cache in the browser and other option	True
The employee visualizes the history data	In the HMI click on the "history data" option of the MachineActions menu and insert an interval	Receives data in a graph within a defined time interval	True
The employee visualizes the analysis data	In the HMI click on the "data analysis" option of the MachineActions menu and choose an error/alarm	Receives data in a graph of error/alarm detected by the Data Analysis subsystem	True

11.1 Appendix-B- Machine Component (Preliminary Solution)

This component was implemented Java language, a project application was created using the NetBeans IDE. It was developed in the operating system Windows 10 using JDK 8⁴³, being platform independent. This project used the Java swing library⁴⁴ to create the User Interface. Because it was necessary to read files with a lot of data (300Mb), With concurrent access, it was necessary to use Java Streams⁴⁵ in order to resolve the problem.

Since access to share memory was done via DLL, Java Native Access (JNA)⁴⁶ was required to access the functions provided by the DLL. In addition, it was necessary to use the UCanAccess⁴⁷ library to access the Microsoft Access Database that stores the information of the Machine.

The main features and responsibility of the Preliminary Solution are:

- Configuring properties in file, namely, Server IP, Log file and database paths, queues, exchanges, routing keys, etc.;
- Gathering the values from start and stop bottom in the shared memory and activate log file executable;
- Reading the values from log files with a periodicity of 300 milliseconds and catch N lines;
- Analysing data and creating new Comma Separated Values (CSV) files using a pattern to erase dump data;
- Creating a Data Transfer Object (DTO), with data from log files, using JSONArray⁴⁸ and sending it over the internet using AMPQ Server;
- Collecting all inserts in the AlarmDB Database on table Alerts and send it over the internet using AMPQ Server;
- Providing a user-friendly and responsive Graphical User Interface (Figure 6767) to define the state of the Internet connection and access shared memory, configure properties, erase data, etc.

⁴³ <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

⁴⁴ <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

⁴⁵ <http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html>

⁴⁶ <https://github.com/java-native-access/jna>

⁴⁷ <http://ucanaccess.sourceforge.net/site.html>

⁴⁸ <http://docs.oracle.com/javase/7/api/javax/json/JsonArray.html>

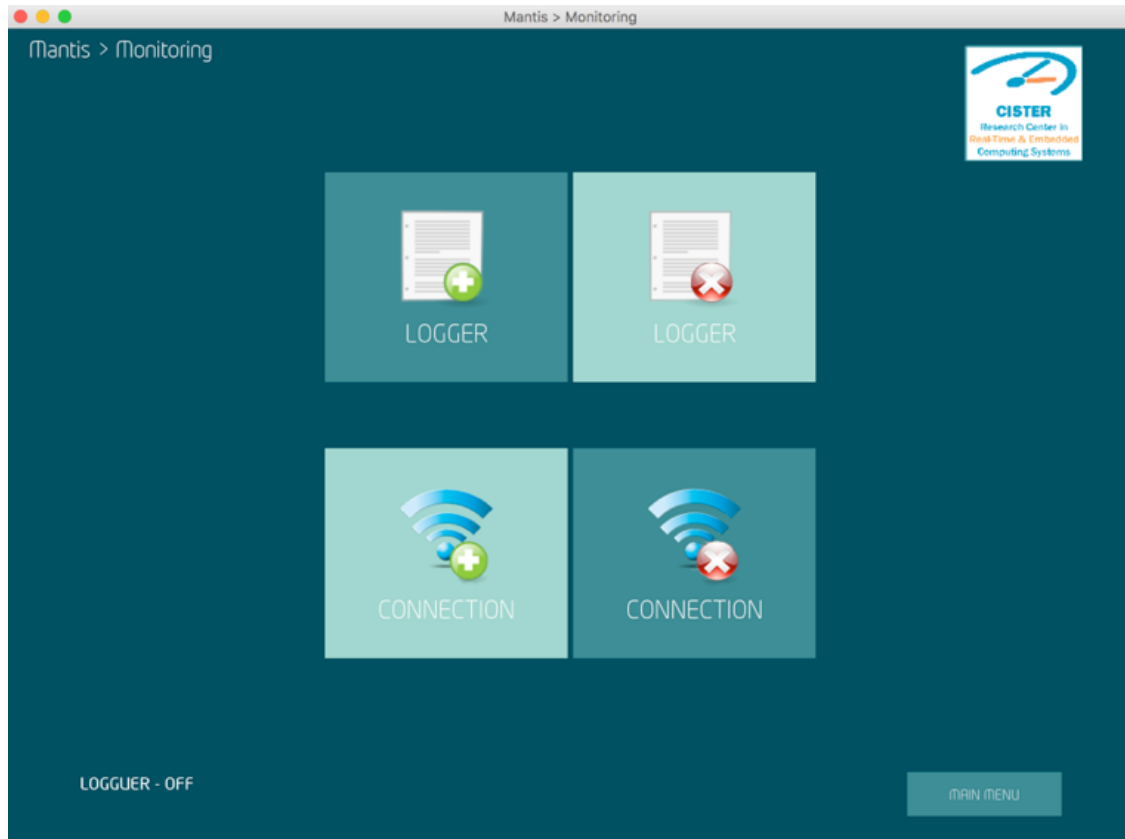
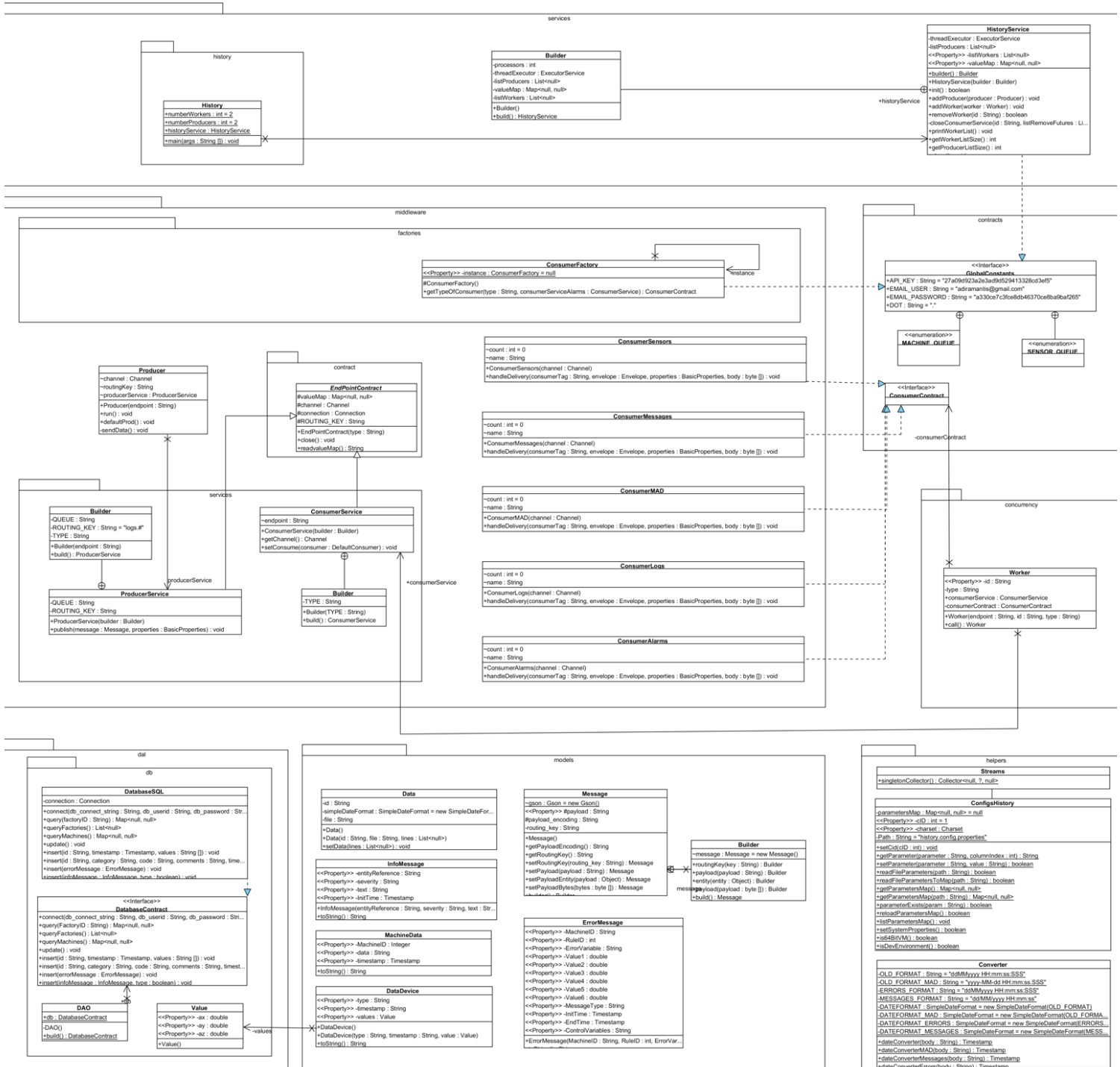


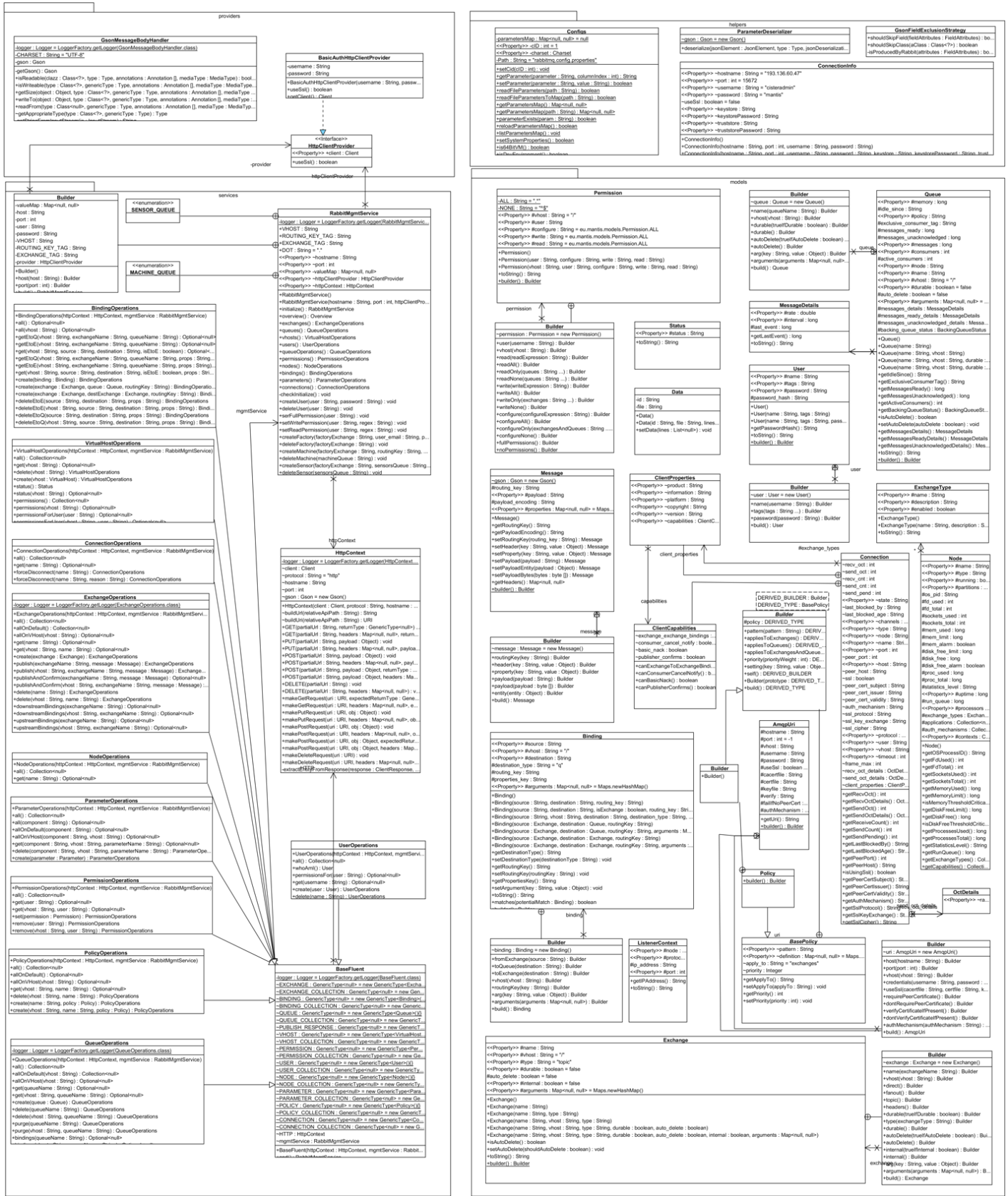
Figure 67 - Graphical User Interface – Main Page

For the point of view of the objects arranged in the detailed time sequence of the CNC Machine Preliminary Solution, which is a data producer, was designed the diagram of Figure 6868 that shows the main features of this solution in a high-level view.

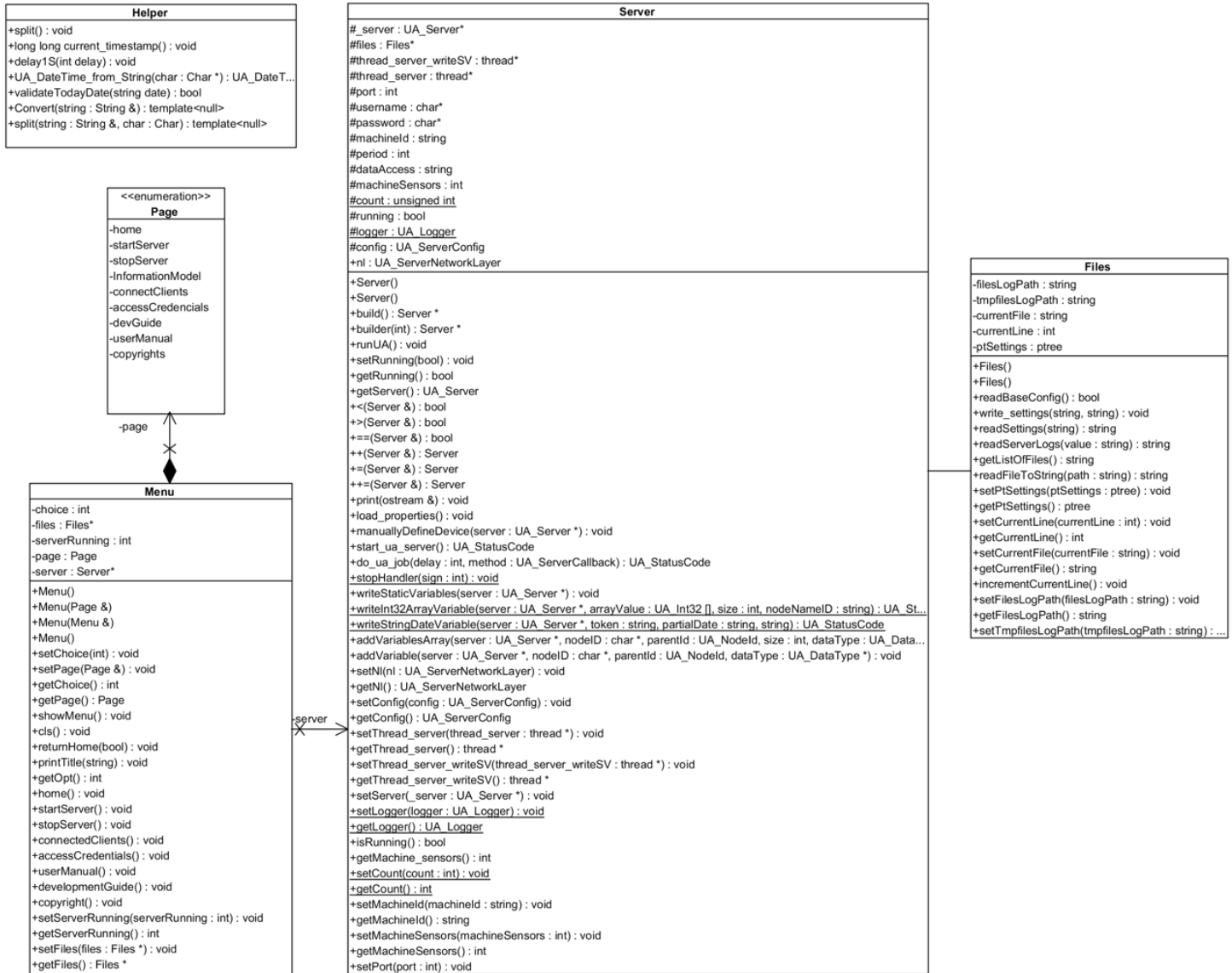
11.2 Appendix-C- UML class diagram for History Component



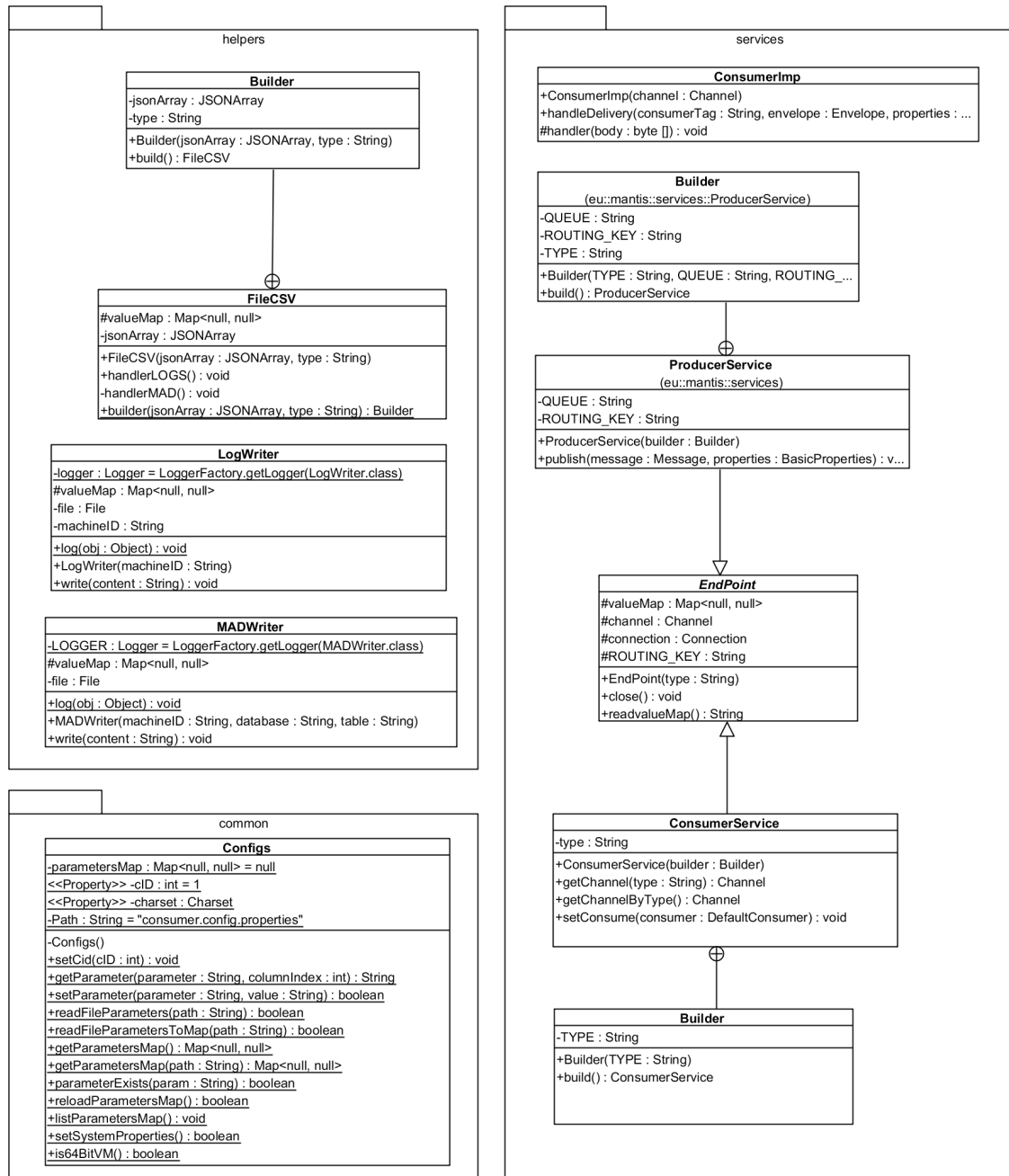
11.3 Appendix-D- UML class diagram for RabbitMQ-Core



11.4 Appendix-E- UML Class Diagram for OPC-UA Server



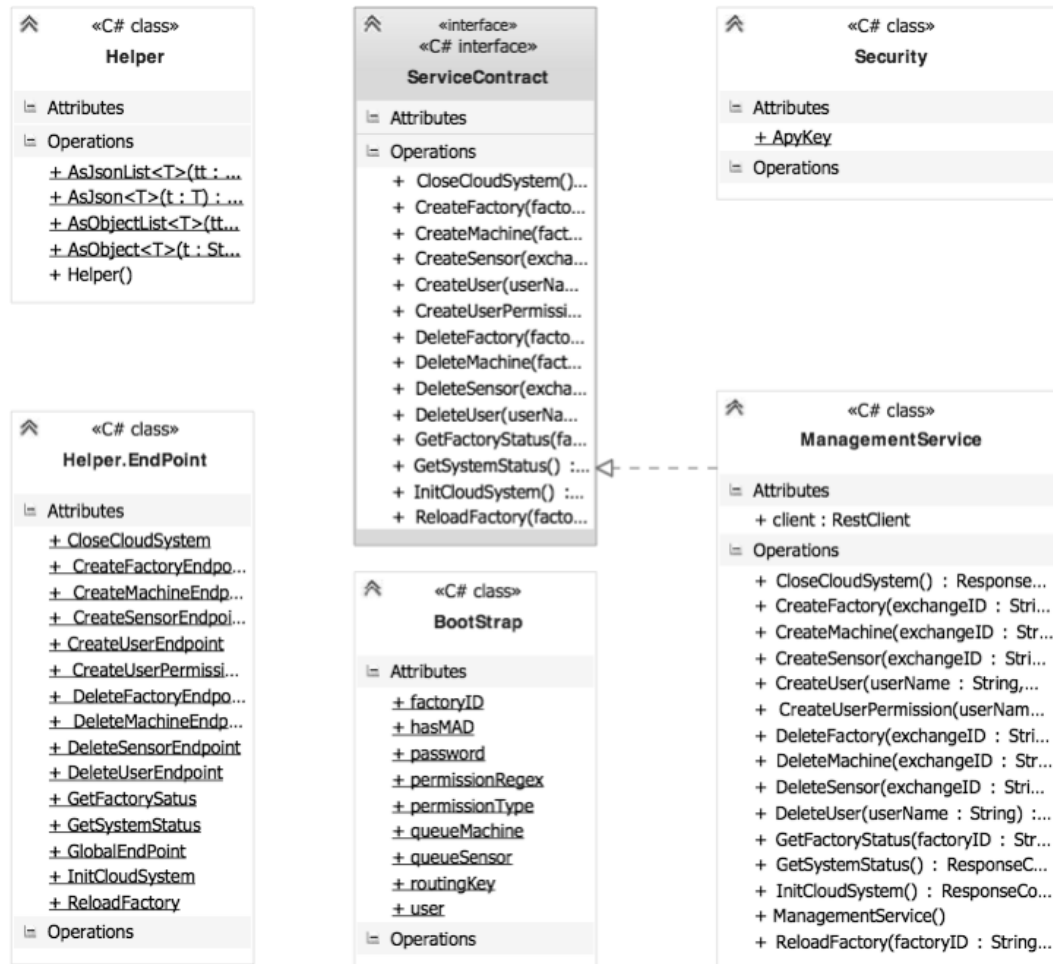
11.5 Appendix-F- UML class diagram for full Middleware Client (Producer/Consumer)



174



11.7 Appendix-H- UML class diagram for API Component



11.8 Appendix-I- Deployment setup and information

Table 20 - OPC-UA server Component Setup and Information

Type	Details
Project IDE	Netbeans Version: 8.2
Dependencies	Boost; Cppunit
Operating system (target)	Windows XP Pro SP3
OPC-UA implementation	Open62541- Version: 0.2
Compiler	Cygwin - Version: 2.5.2
Software (<i>OPC-UA server</i>)	Version: 0.5
Software (<i>OPC-UA server</i>) - Size	7,1 MB
Programming Languages	C and C++
Protocol	OPC-UA

Table 21 - CNC Bender Component Setup and Information

Type	Details
CNC Bender	Version 6.00.30h CN
Kvara (Shared memory module)	Version: 5.2.29.23
Operating system	Windows XP Pro SP3
Processor	Intel Atom N270 1.60GHz
Memory	1GB RAM
Hard Disk	500GB

Table 22 - Sensor Component Setup and Information

Type	Details
Project IDE	ARDUINO 1.8.3
Sensor	Arduino 101
Microcontroller	Intel Curie
Input Voltage (recommended)	7-12V
Flash Memory	196 kB
Features	Bluetooth LE, 6-axis accelerometer/gyro
Box	Arduino 06RBARD16 Box
Fixationility	Double-sided 1.5 mm
Software (<i>Sensor Component</i>)	Version: 0.2
Programming Languages	C and C++
Protocol	BLE

Table 23 - Mantis-PC Subsystem Setup and Information

Type	Details
Mantis-PC	Raspberry PI 3 Model B
SD Card Size	8GB
Operation System	Raspbian Jessie Lite - based on Debian
Dependencies	npm; bluetoothctl

Table 24 - BLE server Component Setup and Information

Type	Details
Project IDE	IntelliJ IDEA 2017
<i>BLE server</i> Component	Server-side JavaScript - NodeJS
<i>Dependencies</i>	npm; noble; async; buffer-dataview; net
Software (<i>Sensor</i> Component)	Version: 0.3
Programming Language	JavaScript
Protocol	BLE

Table 25 - **Edge Local** Subsystem Setup and Information

Type	Details
Operating system	Windows Server 2012 Essentials
Processor	i7 CPU 4th generation (suggestion)
Memory	16GB RAM
Hard Disk	1TB
Dependencies	npm; node-red; node-red-contrib-opcua; node-red-dashboard; node-red-contrib-amqp

Table 26 - Node-Red component Setup and Information

Type	Details
<i>Node-Red</i>	Version 0.16
Dependencies	npm; node-red; node-red-contrib-opcua; node-red-dashboard; node-red-contrib-amqp
Programming Language	JavaScript
Protocols	OPC-UA and AMQP

Table 27 - Middleware Client component Setup and Information

Type	Details
<i>Project IDE</i>	Netbeans 8.2
<i>Middleware Client (producer/consumer)</i>	Version 1.2
Dependency manager	Maven
Development Kit (SDK)	Version 8
Programming Language	Java
Protocol	AMQP

Table 28 - Edge Server Subsystem Setup and Information

Type	Version
Operating system	Windows Server 2012 Essentials
Processor	i7 CPU 4th generation (suggestion)
Memory	16GB RAM
Hard Disk	1TB
Dependencies	IIS;FTP; RabbitMQ;SQL
Protocols	AMQP; HTTPS; STOMP; MQTT;TPC/IP

Table 29 - Manager (with modules) and History and components setup and Information

Type	Version
Manager	Version 0.7
<i>Project IDE</i>	Netbeans 8.2
Dependency manager	Maven
Development Kit (SDK)	Version 8
Programming Language	Java
Dependencies	Mavens; grizzly; jersey; junit; gson; sqlserver; rabbitmq; slf4j; mockito; <i>History Component</i> ; RabbitMQ-Core Module
Protocols	AMQP; HTTPS;TPC/IP
Database	Microsoft SQL Server 2012
Database Space	80 GB

Table 30 - API Component Setup and Information

Type	Details
Project IDE	Visual Studio 2015
<i>API</i>	Version 0.1
Dependency manager	Nuget
Architecture	REST
Protocol	HTTPS
Programming Language	C#

Table 31 - API Component Setup and Information

Type	Details
Project IDE	IntelliJ IDEA 2017
<i>Middleware-Web Client</i>	Version 0.1
Dependency manager	npm
Dependencies	npm;stompjs;highcharts
Protocol	STOMP over Websocket
Programming Language	JavaScript

11.9 Appendix-J- Interoperable and Interconnected CPS-populated Systems for Proactive Maintenance

Interoperable and Interconnected CPS-populated Systems for Proactive Maintenance

Giovanni Di Orio, Pedro Maló
Dep. De Eng. Eletrotécnica, FCT-UNL
UNINOVA-CTS
Lisboa, Portugal
{gido, pmm}@uninova.pt

Csaba Hegedűs
Telecommunications Division
AITIA International Inc.
Budapest, Hungary
hegeduscs@aitia.ai

Michele Albano, Luis Lino Ferreira, José Silva
ISEP/INESC-TEC, CISTER
Polytechnic Institute of Porto
Porto, Portugal
{mialb, llf, jbmds}@isep.ipp.pt

Pál Varga, István Moldován
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
{pvarga, moldovan}@tmit.bme.hu

Abstract— Cyber-Physical Systems (CPS) are creating new market opportunities and business models for all kind of European Industries. CPS-based platforms are increasing in their size and target application areas in a steady manner. However, even if progress is made every day supported by continuous technological advancements, CPS application and deployment is still challenging. Many solutions have been made available or is currently under development in several research projects/initiatives. Typically, these solutions show no interoperability between each other and are tailored to a specific application context. Thus, there is an urgent need for a clear definition of what a CPS-populated system actually is. This will provide a common ground for designing and building interoperable CPS-populated systems. Interoperability represents one of the most challenging problems for such systems essentially due to their intrinsic characteristics: heterogeneity, distribution and networked. These must be addressed to allow the cooperation and collaboration between all the actors of the system. In this landscape, the MANTIS project is aimed to provide a reference model for interoperable and interconnected CPS-populated systems for maintenance-related ecosystems, which is the focus of this paper.

Keywords—Cyber-Physical Systems; Service Orientation; OPC-UA; MIMOSA; Interoperability

I. INTRODUCTION

The wider dissemination of intelligent devices in all aspects of human life is producing extremely complex environments that are, in turn, characterised by heterogeneity and distribution. The next wave in this era of computing envisions the active presence of physical objects/entities on the network [1]. As explained in [2][3], current technological advances are radically changing the way systems in different domains are designed and deployed, monitored and controlled. Especially the CPS approach is opening the doors to a new generation of systems, where information transparency, efficient and effective management, high availability, adaptability and (re-)configurability of assets and resources are the key characteristics.

As stated in [4], three main converging streams are pushing enterprises into the world of “smartness & connectivity”. One of them – smart embedded systems, mobile services and pervasive computing – is contributing to the establishment of a cyber space [5] where data coming from assets and resources can be potentially used to: i) enable more and more exclusive, efficient and sustainable systems to assure a more efficient and effective management of the resources; and ii) create new and powerful business opportunities through the tighter integration of all the steps/stages of the product lifecycle management (PLM), and the provisioning to the customer of new product-service solutions [6].

As stated in [7]: “Through CPS, the development of new business models, new services are expected which may change many aspects of our life”. Nevertheless, CPS are becoming critical to the business success of many enterprises as confirmed in [8]: “In transportation, manufacturing, telecommunications, consumer electronics, health and medical equipment, and intelligent buildings the value share of electronics, computing, communications, sensing, and actuation is expected to exceed 50% the end of the decade”. However, CPS application and deployment is still challenged by set of technical, institutional, and societal issues.

This paper presents how the MANTIS project is building interoperable and interconnected CPS-populated systems for proactive maintenance for enabling service-based business model and improved asset availability at lower costs through continuous process and equipment monitoring and analysis.

II. THE MANTIS PROJECT

A. The Project Background

Nowadays, conventional systems and processes are evolving into CPS. As stated in [9], the term “Cyber-Physical Systems” have been coined in 2006 and specifies a system consisting of computational, communication and control components combined with physical processes [10]. This definition

indirectly points out the core elements and/or characteristics of a CPS, extended from [11], [12]:

- Enhancement of physical entities with Cyber capabilities;
- Networked at multiple and extreme scale;
- Dynamic behaviour (plug and unplug during operation);
- High degrees of automation, the control loops are typically closed;
- High degree of autonomy and collaboration to achieve a higher goal; and
- Tight integration between devices, processes, machines, humans and other software applications.

The elements and/or features of a CPS intrinsically identify a set of research challenges that need to be addressed to accelerate the progress and deployment of CPS in real application context. The research challenges here summarized are the ones related with the MANTIS project ambition and clustered according to [13]:

- Science and engineering foundations: a reference architecture for interoperable and interconnected CPS-populated systems in cross-sector applications;
- System performance, quality and acceptance: to create large, adaptive and resilient networked systems that are capable to operate in the specific environments where the physical entities are installed while delivering the required functionality in a reliable way; and
- Applied development and deployment: to provide methodologies for virtualization of physical entities and integration of heterogeneous systems. To deliver technology foundation for building interconnected and interoperable CPS-populated systems.

B. The Project Vision

The overall aim of the MANTIS¹ project [14] is to develop platform for interoperable and interconnected CPS-populated systems for proactive maintenance ecosystems, i.e. for facilitating the implementation of predictive and proactive maintenance strategies. The MANTIS platform will provide a practical mean for implementing collaborative maintenance by taking advantage from:

- the omnipresence of intelligent devices – that combine physical entities with computational and communication capabilities – in modern processes, machines and other distinct application domains; and
- the maturity level reached by cloud-based infrastructure, as well as, the huge amount of computational and storage resources that are available and usable “on-demand”.

Intelligent devices are the ones directly connected and/or installed to the physical resources and assets. They can potentially optimize and improve current maintenance activities and their related management systems by providing (often live) data – gathered during operation – that can be analysed (low level data analysis) to understand the behaviour of the related physical resources and assets. Furthermore, the data gathered from physical resources and assets can be also combined and analysed globally (high level data analysis) by using

computational resources and complex algorithms running over the cloud (high level) to understand the collective behaviour of group of resources and assets. Therefore, within the MANTIS platform the data extraction, transforming, loading and pattern analysis will take place at different levels, namely (see Fig. 1):

- Low level: extraction, transforming, loading and analysis of simple signals to model and understand the behaviour of selected physical resources and assets. The following algorithms, methods and methodologies are considered: sensor data fusion (feature fusion, decision fusion), noise elimination and erroneous data; this can be implemented by stochastic methods, Kalman filter, fuzzy logic, logical links or rule-based methods. Additionally, it will also reduce bandwidth requirements.
- High level: extraction, transforming, loading and analysis of complex data – typically the results of the low level – to model and understand the global behaviour of physical resources and assets. The following algorithms, methods and methodologies are considered: data mining (classification, cluster analysis, associations and regression analyses), intelligent assessment of data; includes recognition of outliers, k-means-algorithms and machine learning.

Since data sources are typically characterized by distribution, heterogeneity, and a high degree of dynamicity (e.g. data sources like sensors can be plugged and unplugged any time), then the design of the MANTIS architecture has been driven by the following main requirements:

- Integration of complex and heterogeneous large-scale distributed systems from different application domain; and
- the design of CPS-populated systems to enable collaborative proactive maintenance strategies.

It is easy to understand that the design of interoperable and interconnected CPS-populated systems become a key element of the MANTIS implementations to allow to dynamical and on-demand addition or removal of data sources in/from the MANTIS platform to gather most of the maintenance relevant information automatically from the environment.

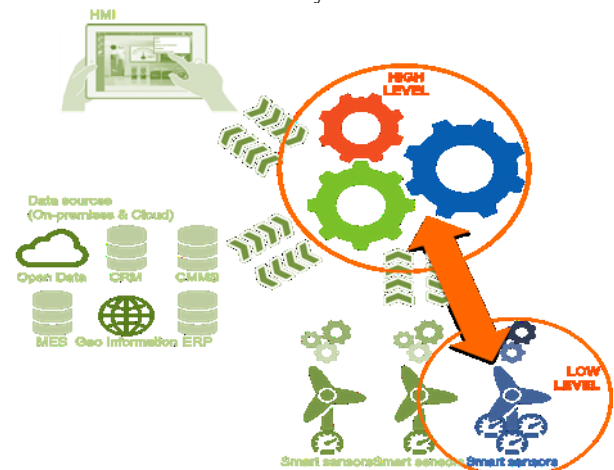


Fig. 1 MANTIS overall concept idea and data processing levels

¹ <http://www.mantis-project.eu>

III. SUPPORTING CONCEPTS

A. Relevant Paradigms and Technologies

1) Edge and Cloud Computing

The architectures supporting CPS-populated systems has been subject to strong research activities, to converge to an approach that allows interoperability, performance and resilience. CPS is having disruptive effect on the way enterprise are structured while motivating the need to find a standard, well-structured and defined concept for modelling and describing implementation of CPS-populated systems.

Nowadays, cloud technologies are creating the foundations for breeding CPS-populated systems through:

1. an elastic infrastructure for CPS integration, i.e. services and/or atomic functionalities provided by CPS can potentially be accessed/used over the internet by other CPS or applications; and
2. a huge amount of computational and storage resources that are available within the cloud and can be used “on-demand”.

Specifically, the cloud computing architectural model enables ubiquitous, on-demand network access to a shared pool of resources that are – thus – available and provided as services. This model typically relies on:

- processing offerings: provides all the required functionalities to execute the work load;
- storage offerings: provide all the necessary mechanisms to store data into the cloud; and
- communication offerings: provide all the necessary mechanisms to enable the information exchange between cloud applications.

Even if cloud environments are by nature distributed the architectural model is highly centralized.

Next to cloud computing, fog/edge computing architectural model has been introduced. This model is aimed to extend the cloud computing paradigm to the “edge” of the network for those applications that are latency-sensitive and – thus – have strict delay requirements [15]. Therefore, fog/edge computing is about pushing intelligence, data analytics and knowledge generation in smaller clouds, between physical devices and traditional cloud computing data centres, and – thus – closer to source of the data [16] while supporting glocalisation, i.e. location awareness and distribution. Fog/edge computing architectural model reflects better than cloud the complexity, heterogeneity and distribution of CPS-populated systems and their ecosystems.

2) Service Oriented Architecture for Cyber Physical Systems

Firstly referred in [17], Service-oriented Architecture (SOA) paradigm has emerged and rapidly grown as a standard solution for publishing and accessing information in an increasingly Internet-ubiquitous world. SOA establishes an architectural model that aims to enhance the efficiency, interoperability, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals [18]. The increasingly interest in SOA has been stimulated by an influential industry trend: Web Services technology [19]. In this landscape, Web Services are helping to

bring SOA to a wider audience, while SOA concepts and principles will contribute to more successful web services initiatives [20].

Nowadays, there are several technologies and standards that can be used to develop web services for networked-based software architectures. Regardless to the specific technology and standard used, web services can be classified into two main categories: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). On the top of these several SOA solutions have been created that are suitable for hardware virtualization, i.e. for cyber-physical integration, the most prominent ones include: DPWS, OPC-UA and Arrowhead.

a) OPC-UA

The OPC UA (Unified Architecture) is the new version of the vastly deployed OPC architecture originally designed by the OPC Foundation to connect industrial devices to control and supervision applications as explained by [21], [22]. Considering the application context, the focus of OPC is on getting access to large amounts of real-time data while ensuring performance constraints without disrupting the normal operation of the devices. The original OPC specifications were based on Microsoft COM/DCOM meaning that they are becoming obsolete and are rapidly being replaced by the newer standards on interoperability (e.g. Web services). Thus, the main visible transformation when looking at OPC and its newer architecture OPC-UA is the shift to cross-platform capable web services [23]. From a technological point of view, OPC-UA can be mapped using widespread Web standards, including XML, WSDL, SOAP and other WS-* specifications, along with other OPC UA defined specifications for UA native communications. Moreover, the criticality of the industrial process imposes security as a fundamental requirement. In this, scenario OPC-UA provides configurable security mechanisms to allow security not to impact too much on the system performance. Finally, OPC UA provides a homogeneous and generic meta-model and defines a set of web services interfaces to represent and access both structure information and state information in a wide range of devices.

b) Arrowhead Framework

The Arrowhead Framework [24] is the result of a large European effort that aimed at normalizing the interaction between industrial IoT applications by the means of Service Oriented Architectures. The approach targeted at number of application domains comprising industrial production, smart buildings, electro mobility, and energy production. Services are exposed and consumed by (software) systems, which are themselves executed on devices. These devices are grouped into local automation clouds, that are self-contained, geographically co-located, independent from one another, and mostly protected from external access through security measures.

Services are considered either application services (when implementing a use case), or core services (that provide support actions such as service discovery, security, service orchestration, and Quality of Service). The approach considers the interoperability of the involved devices as one of the most important issues, and thus trust all these core services to a common Arrowhead Framework, which operates using

registries containing formal descriptions of the devices, systems and services that are present in that local cloud instance.

The technical stance of Arrowhead involves either the deployment of a set of core services at the industrial site, or the secure connection (e.g.: through a VPN) to a set of existing core services. Particular support is given to the orchestration of services, allowing the creation of distributed applications that build over existing ones, for example to create building automation applications that connect to the Virtual Market of Energy [25]. To this aim, the Orchestration core service makes use of an engine that is able to match the requirements on the orchestrated service with the formal description of the available services, while taking into account non-functional requirements such as Quality of Service, and geographical localization of the devices offering the services.

B. Focusing on Standards: MIMOSA

To achieve useful maintenance procedures and strategies information from large numbers of smart devices and systems needs to be collected and analysed. In this scenario, standards provide a set of terms, concepts, data formats, document styles and techniques so that the information collected can be easily processed by data analytics tasks, routines, algorithms within different computer programs. Thus, the provisioning and usage of standard models is a fundamental step to achieve interoperability by assuring that products and services – that comply with them – can communicate and exchange information.

One challenge in designing and developing integrated system health monitoring and service maintenance platforms is the strong presence of vast amounts of data from heterogeneous resources which are exchanged over a heterogeneous collection of communication channels at different levels ranging from local-nodes to cloud applications. This is clearly the case of any CPS-populated system. As stated in [33], the success of these systems strictly depends on an open, uniform, and performance-optimized solution for data management. A solution that includes: data definition, data communication, and data storage.

The Machinery Information Management Open Systems Alliance (MIMOSA) is a not-for-profit trade association composed of industrial asset management system providers and industrial asset end-users [28], [29]. The goal is to develop information integration specifications to enable open, integrated solutions for managing complex high-value assets. MIMOSA's open standards enable collaborative asset lifecycle management in both commercial and military applications. Its primary domains are registry, condition monitoring, reliability, maintenance and work management functions. MIMOSA provides two open standards:

- Open System Architecture for Condition Based Maintenance (OSA-CBM): focused on facilitating the information acquisition processes, i.e. to support interoperability between different components.
- Open System Architecture for Enterprise Application Integration (OSA-EAI): focused on supporting integration between applications at enterprise level.

If from one side the MIMOSA standards support a large range of asset management data types that allow them to be used in many asset management integration processes. From the other side they are complex, intricate and not well-documented. The MIMOSA standards are still immature. This lack of maturity makes the process of building and maintaining MIMOSA-based solutions too expensive for any organization. Moreover, the documentation is sparse and incomplete.

IV. ENGINEERING OF CPS FOR MAINTENANCE: THE MANTIS APPROACH

A. The MANTIS-ARM

1) Edge devices according to MANTIS

The MANTIS platform aims to provide all previously discussed capabilities both on the low and high level analytic planes. To this end, various components, technologies and concepts should appear in the MANTIS-ARM that was presented in the previous sections. Fig. 2 depicts the overview of the architecture.

All edge devices are connected to the MANTIS platform. They are generally connected to the physical world – and can be categorized based on their capabilities and resources as:

- A constrained device,
- A smart and intelligent sensor node,
- A complex cyber-physical system, or
- A local automation cloud on its own.

Constrained devices are merely sensors with restrained processing and communicational capabilities. This can involve a cellular or other wireless connection, and that the device itself is an embedded system. Hence, it is possibly striving for low energy consumption or for staying in sleep mode. However, it must communicate directly with the MANTIS platform, primarily to send in their read outs. Since the connection might be metered based on traffic, the implemented edge-cloud interface must use the shortest possible messages.

Smart and intelligent sensor nodes not just possess significantly higher processing power, but feature low-level implemented algorithms and analytics on the sensory data. In here, high frequency readouts might be extracted into smaller descriptors (or fingerprints) and local storage of the measurements might also be implemented. Therefore, the edge-cloud interface must feature two-way communications, and be suitable for data streaming and bulk transmissions as well.

Complex cyber-physical systems differ from sensor nodes in all aspects. They might feature high level analytic capabilities on their own, and are not restrained in any sense.

If there are multiple CPS-s and a whole local automation environment is to be handled as one entity (from the platform's point of view). This could be achieved using a MANTIS-capable gateway that makes use of the proper ontologies and provides interoperability between local CPS-s and the analytics platform.

2) Cloud modules

MANTIS cloud analytic platform features the modules:

- Edge servers – brokers,
- Stream processors for real time analytics,
- Central database,
- Batch processors for offline analytics.

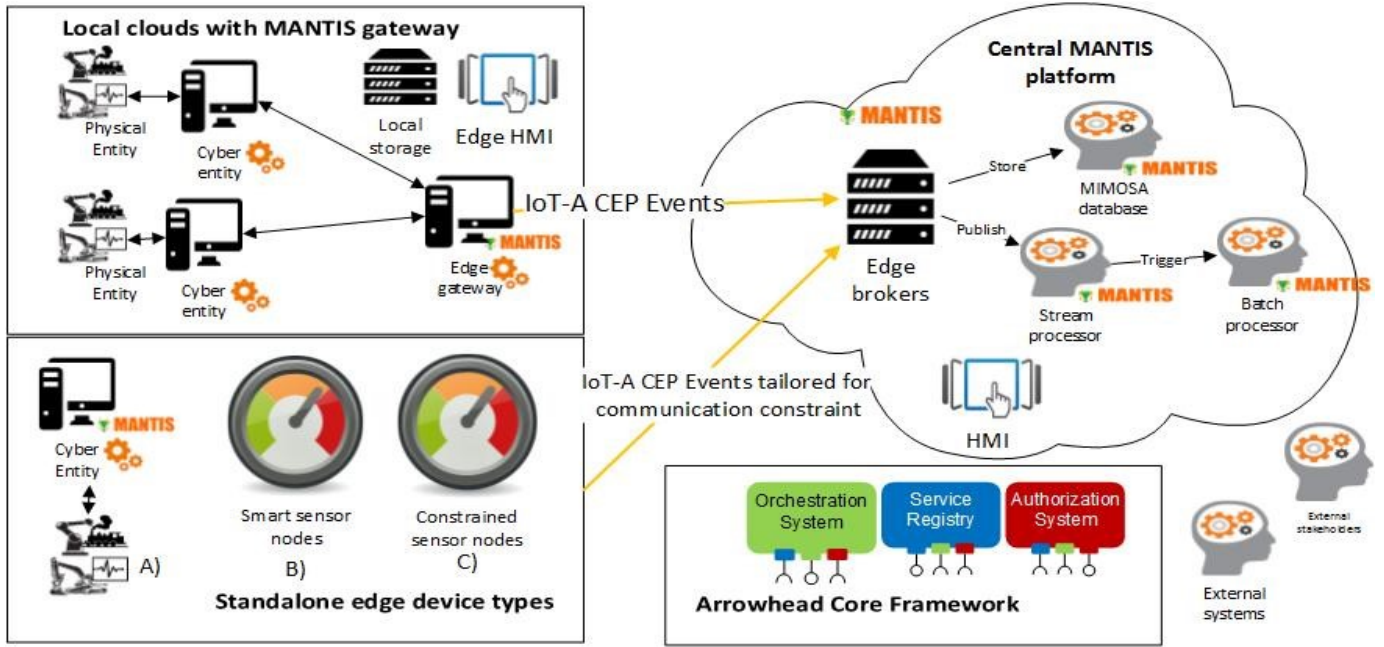


Fig. 2 – Overview of the MANTIS platform

The purpose of the edge servers (or brokers if message-oriented middleware is used for implementation) is to collect, aggregate and queue up the inbound traffic from the edge level. This entity must be highly scalable and therefore might be implemented using load balancers and redirectors. Its output is forwarded to the central storage (with the proper pre-processing per the storing policy), and to the online stream processor unit.

The task of the stream processor is to receive the inbound traffic from the edge and process, detect or predict events in the data stream (e.g. as a time series).

These events are connected to the primary objectives of the platform: i) tracking the remaining useful life (RUL) of assets, ii) early detection and prediction of failures and iii) conduction of root cause analysis (RCA) in case of a failure.

The output of the stream processor is also related to these maintenance-related objectives and runtime. These can include the notification of the appropriate personnel through HMI, or the triggering of the offline analytics engine. For example, if the stream processor is detecting a failure, it can trigger the RCA.

The offline batch processors can execute more complex and demanding algorithms (e.g. machine learning – training phase), that are required for the stream processor's runtime (e.g. parameter estimation and model creation for RUL). This entity is then triggered by the online analytics engine, for example when a maintenance related decision cannot be made by the stream processor. This entity has the time to collect historical data from the database, collect additional data from external systems and make decisions based on them.

3) Arrowhead integration

As per the Service Oriented Architecture (SOA) design pattern, these interfaces and communication can and are abstracted as services. To utilize the i) loose coupling, ii) late binding and iii) the dynamic discovery of services provided by the Arrowhead Framework, the MANTIS platform is organized

as a collection of services provided by each of the above-mentioned modules. Therefore, Arrowhead acts as a system integrator, and provides run-time (re-)configurability of the networked devices. For example, edge devices can look up the appropriate edge broker they are supposed to connect to.

B. MANTIS Interoperability Perspective

Interoperability represents a perspective of the MANTIS-ARM reference architecture building block. A perspective defines a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views [30]. Therefore, interoperability perspective is something orthogonal to the several views defined within the MANTIS-ARM reference architecture building block. Fulfilling qualitative requirements through the architecting process inevitably leads to design challenges, related design decisions and design choices. Since there is usually more than one solution to each of the design challenges, the MANTIS ARM cannot guarantee interoperability between any two concrete architectures, even if they have been derived from the same requirement set. Nevertheless, the MANTIS ARM is an important tool in helping to achieve interoperability between MANTIS-compliant systems and within the MANTIS concrete platform itself.

In MANTIS, interoperability issues are framed in two levels:

- **Edge level:** focuses on a set of physical entities belonging to the same local system. At this level the data extracted from physical entities is used to model and analyse the behaviour of the local system. The edge level also includes a sublevel that is the component level where physical entities are analysed singularly.
- **Cloud level:** focuses on the information exchange and data integration in the cyberspace. At this level the data coming from the edge level is organized in order

to be processed to analyse the overall system behaviour.

Furthermore, there are several interoperability issues that are orthogonal to the considered interoperability levels, i.e. models, guidelines and specifications that can be applied to all the interoperability levels without any restriction, namely:

1. The definition of the communication protocol and message exchange pattern to use in both edge and cloud levels;
2. The definition of an ontology of events to support systems interactions at both edge and cloud levels.

The above topics reveal interoperability issues that are related on how CPS are connected and, thus, on how CPS can exchange data/information. To enable interoperability between CPS a standard-based approach is the way to follow. However, to apply this approach it is necessary to reduce drastically the cost of creating and establishing standard-based solutions. To promote and take advantage from the usage of MIMOSA open standards, in CPS-based applications, a set of meta-models have been designed to reduce the MIMOSA complexity while hiding its own intricacies. The provided models are here presented in sections B.1) and B.2).

1) *Semantic Data Representation and Exchange Model for MANTIS CPS*

The semantic data representation and exchange (see Fig. 3 left) is aimed to describe the structure of all the data and/or information handled by cyber entities at a network level. It means that once a physical entity is virtualized, the related cyber entity should be designed and developed to cope with the semantic data representation and exchange model. As a matter of fact, once a link between physical and cyber entity is created the data extracted from the environment should be organized and structured within the cyber entity to be transmitted to other cyber entities within the MANTIS platform. Thus, the provided model details the way information should be modelled to guarantee that all the data circulating within the MANTIS platform cyberspace satisfies a well-defined structure to assure interoperability. The model is inspired from the IoT information model that has been designed and developed in the scope of the IoT-A project [31]. The obvious similarities between IoT and CPS-based systems are always pushing a merging of the two research streams. The main aspects are represented by the elements Cyber Entity, Functionality Description and Association. A Cyber Entity models a Physical Entity and Functionality Description describes a service that serves information about the Physical Entity itself or the environment. Through an Association, the connection between an Attribute of a Cyber Entity and the Functionality Description is modelled, e.g. the Functionality could act as a “get” function for an Attribute value. Every Cyber Entity needs to have a unique identifier (identifier). Furthermore, a Cyber Entity can have zero to many different attributes (Attribute). Each Attribute has a name (attributeName), a type (attributeType), and one to many values (Value Container). The attributeType specifies the semantic type of an attribute, for example, that the value represents temperature. It can reference some ontology concepts. This way, one can for instance, model an attribute,

e.g. a list of values, which itself has several values. Each Value Container groups one Value and zero to many metadata information that belong to the given Value. The metadata can, for instance, be used to save the timestamp of the Value, or other quality parameters, such as accuracy or the unit of measurement. The Cyber Entity is also connected to the Functionality Description via the Functionality Description – Cyber Entity association. A Functionality Description describes the relevant aspects of the functionality provided, including the interface (e.g. description of the service endpoint interface). Additionally, it may contain one (or more) Resource Description(s) describing, in this case, the Native Communication Library that is exposed by the Functionality. The Resource Description might contain information about the Physical Entity on which the Resource is hosted, i.e. the Physical Entity Description

2) *Event Model for System Interactions between MANTIS CPS*

Significant actions, incidents or episodes need to be registered and stored. The MANTIS platform – by promoting monitoring and data analysis for performance improvement – demands an event model to define event-based interactions between CPSs at both edge and platform levels.

The event model for system interactions between MANTIS CPS (see Fig. 3 right) is inspired from the IoT event model that has been designed and developed in the scope of the IoT-A project [31]. It relies on the Abstract CEP Event base type of all events (where CEP stands for Complex Event Processing) which is abstract and defines some basic information that must be contained in any instance of any other event type or object (e.g. DateTime, EventDescription, MeasurementLocation). The classes DateTime and MeasurementLocation are defined to adhere to the OSA-CBM standard (the same is for the UUID type). There are only two other event types in the model that are directly derived from Abstract CEP Event. These are Simple CEP Event, that contains atomic event information and Complex CEP Event that contains information derived by a complex event processing application. This model has been considered generic enough to enable the description of any event within the MANTIS platform. Therefore, within MANTIS we decided to adhere to the IoT-A event reference model which allow the creation of two types of events. The Value Container, Value and MetaData classes are used to model the content of each event that in turn can be as simple as a temperature value to complex string with serialized java objects.

V. HOW IT IS DONE

A. *Deployment in Real Application Scenario*

1) *Application Scenario Description*

The efforts of the MANTIS project were driven by its pilots, which on its part facilitated the early adoption of the techniques under study. One of the pilots is centred on the continuous monitoring of a press brake machine. The model targeted by the pilot is a hybrid system powered hydraulically and electrically, and controlled via a fluid pumping sub-system. A blade is moved vertically, and presses on the metal workpiece to be bent.

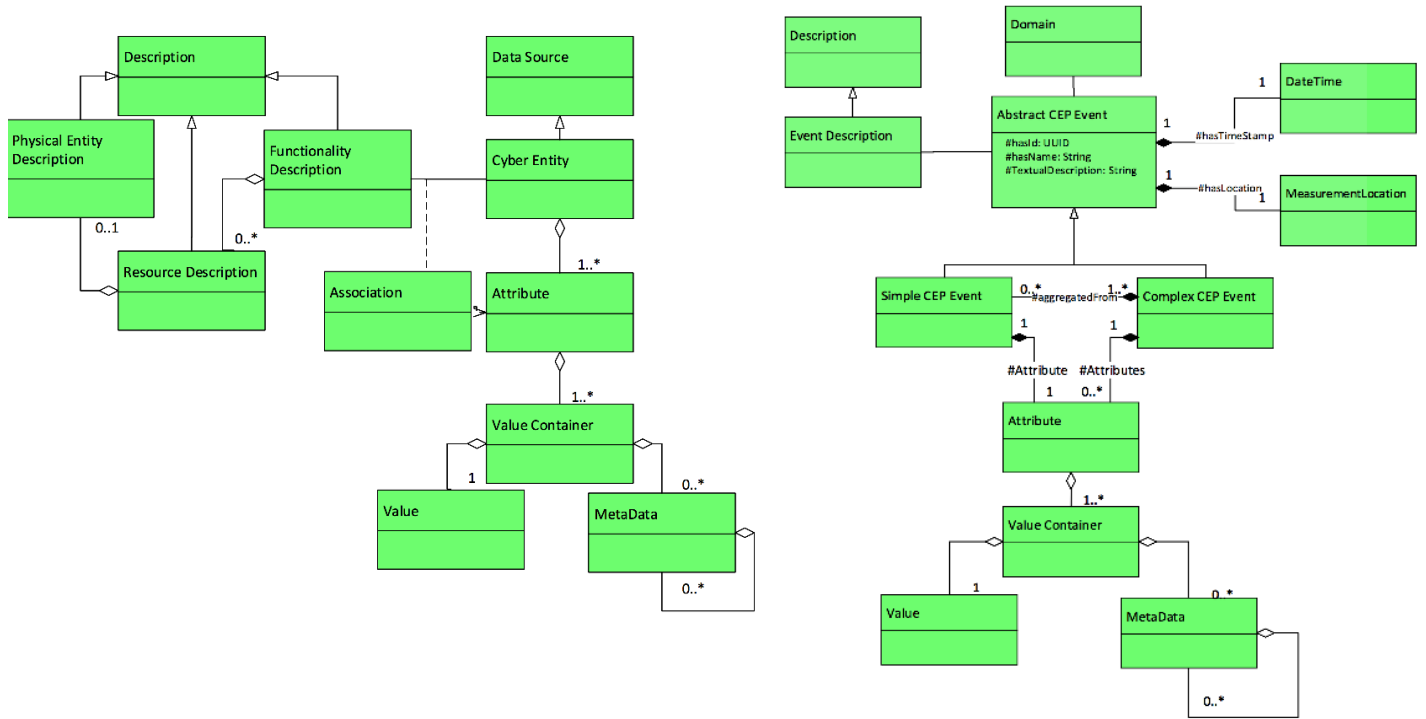


Fig. 3 – Left: MANTIS Semantic Data Representation and Exchange. Right: MANTIS CPS Event Information Model

The process is very complex since the type of the deformation, and the forces to be applied, depend on the material of the workpiece, on its initial and desired shapes, and on the physical phenomena that can verify and which must be compensated for. For example, it is possible that a metal part provides spring-back effects, and uneven loads on the length of the blade.

The machine ends up being quite expensive, and its market cannot be considered local to the factory building it. Thus, upon a machine fault, it is necessary to ship spare parts from the factory to the site where the machine is operated, leading to a downtime of the machine that can be measured in terms of days and that causes substantial economic losses. The benefit of a proactive monitoring platform would reside in the capability of

shipping the spare parts in advance, and thus dramatically reduce the downtime of the machine.

2) Proposed Approach

This scenario provides for obvious interoperability issues. To identify such issues and – thus – extract the main requirements for interoperability approach depicted in Fig. 4. has been applied.

By looking Fig. 4, the approach can be divided into two main phases. The phase 1 is about requirements analysis and is intended to characterize the concrete system with the objective of identifying interoperability needs and location. During this phase, the following steps are performed:

- Use Case Analysis: characterization of the use case concrete architecture in which the MANTIS platform will be integrated;
- Cloud Interoperability needs: identification of the interoperability issues at cloud level;
- Edge Interoperability needs: identification of the interoperability issues at edge level;
- Component Interoperability needs: identification of the interoperability issues at component level; and
- Base technology: identification of the base technologies.

The phase 2 is about interoperability models' application and is intended to apply the provided interoperability specifications to respond to the interoperability requirements gathered during phase 1. During this phase, the following steps are performed:

- New Tools and technology: to identify tools and technologies that could potentially help/facilitate the integration of the MANTIS platform within the pilot ecosystem;

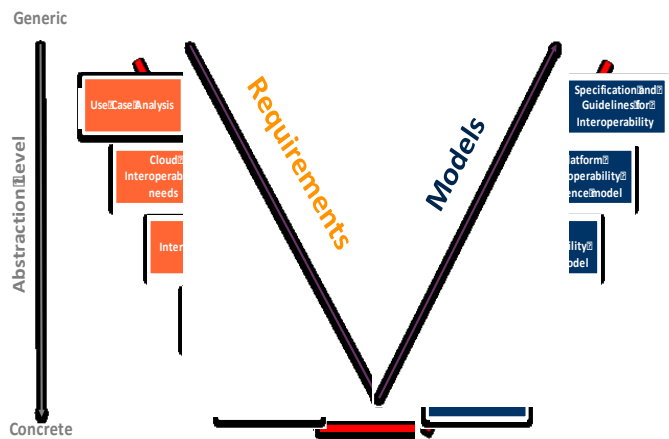


Fig. 4 - MANTIS Interoperability proposed Approach

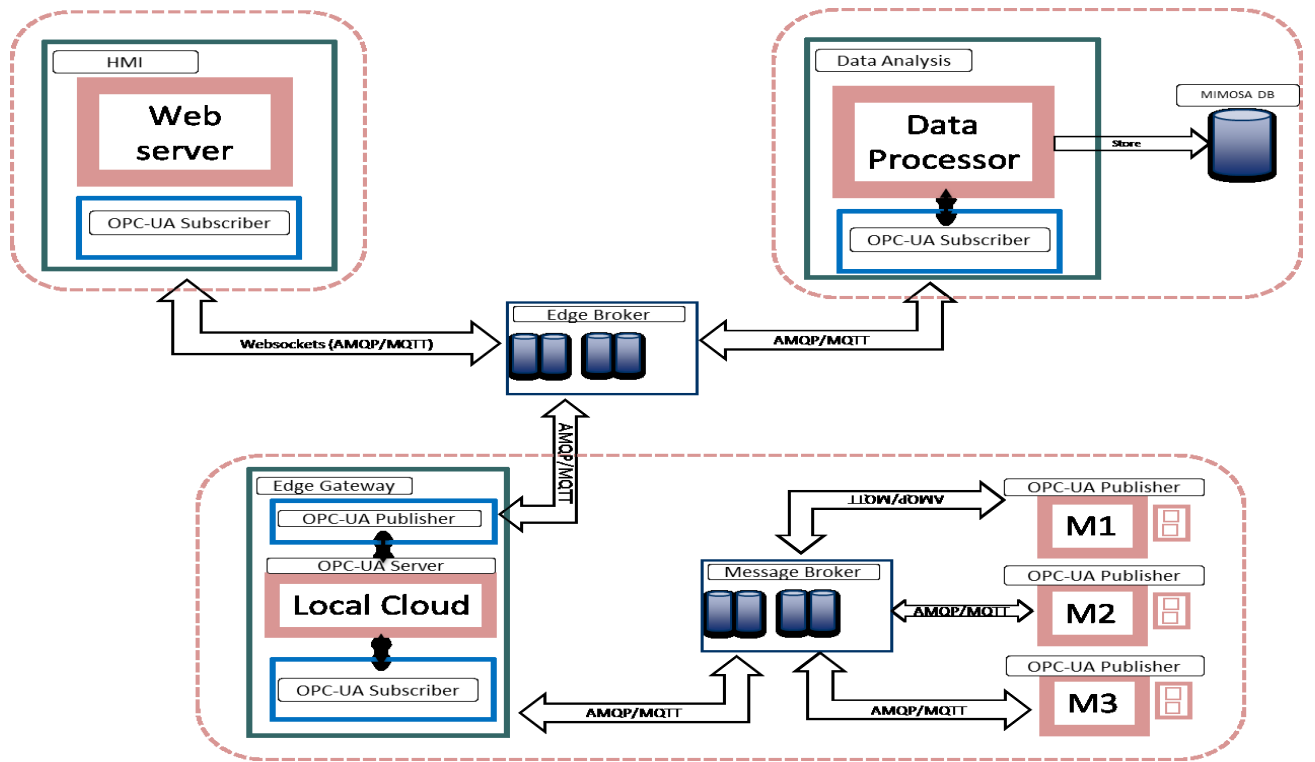


Fig. 5 – Pilot Architecture enhanced with MANTIS platform

- ii. Component Interoperability model: to apply the provided models at the component level to enable integration between physical entities and cyber entities to create CPS;
- iii. Edge Interoperability model: to apply the provided models at the edge level, i.e. between several components within the same local network
- iv. Cloud Interoperability model: to apply the provided models at the cloud level, i.e. integration between cloud digital artefacts that are responsible to process the data provided by CPS;
- v. Specification and guidelines for interoperability: the results of the previous steps have been used to compile a set of specifications and guidelines and guidance for facilitating interoperability between the pilot ecosystems and the MANTIS platform.

3) Pilot Existing Architecture

The considered press brake machine is an industrial asset by the main Programmable Logic Controller (PLC) that controls all the machine operations, the safety PLC, several Input/Output modules (connected to the main PLC) to sense and actuate from/on the real world, and the Computer Numerical Controller (CNC). Above the main controller there are no more monitoring platforms. In the case under study, many sensors were already present in the machine, and their data were stored in a file on the CNC. Anyway, data was used for online control of the machine, for example for fine-tuning of the machine operations and for safety, and for basic monitoring of the machine.

4) Pilot Architecture enhanced with MANTIS platform

The MANTIS project enhanced the existing architecture with both new sensors, and with capabilities based on cloud processing of the data.

To promote the interoperability of the solution, a well-defined architecture was composed with three of the mechanisms described in this paper: the OPC-UA protocol, the AMQP/MQTT communication mechanisms, and the meta-models for integrating MIMOSA-based semantics. The resulting architecture is aligned with the reference architecture presented in section IV.A and can be divided into logical blocks: the Factory, the Edge Broker, the Human Machine Interface (HMI), and the Data Analysis.

The machines under analysis, together with their PLCs and CNCs, existing and newly installed sensors, a MANTIS PC gateway for each machine, and the edge gateway computer, are part of the Factory logical block. The PLCs communicate with the CNC by means of 4-20 mA analogue communication, and the CNCs interact with the MANTIS PC using RS-485 technology. Thus, the MANTIS-PC is the interface of a machine to the rest of the MANTIS architecture.

Newly installed sensors comprise elements to monitor the oil quality and condition, and accelerometers for measuring the bending blade vibrations. The oil monitoring sensors focus on evaluating the temperature of the oil, and the presence of wear particle that would give away the presence of fractures in the machine. The accelerometers detect the vibration pattern of the bending blade while it is performing its function, since the vibration patterns can be related to the health status of the actuators of the machine.

Multiple machines provide, through MANTIS-PCs, their data to a local message broker that enables the concentration of the data into a edge gateway computer. From here on, interoperability with other machines is supported by the solutions adopted by the MANTIS project. In fact, the mechanisms used for the interaction through the message broker

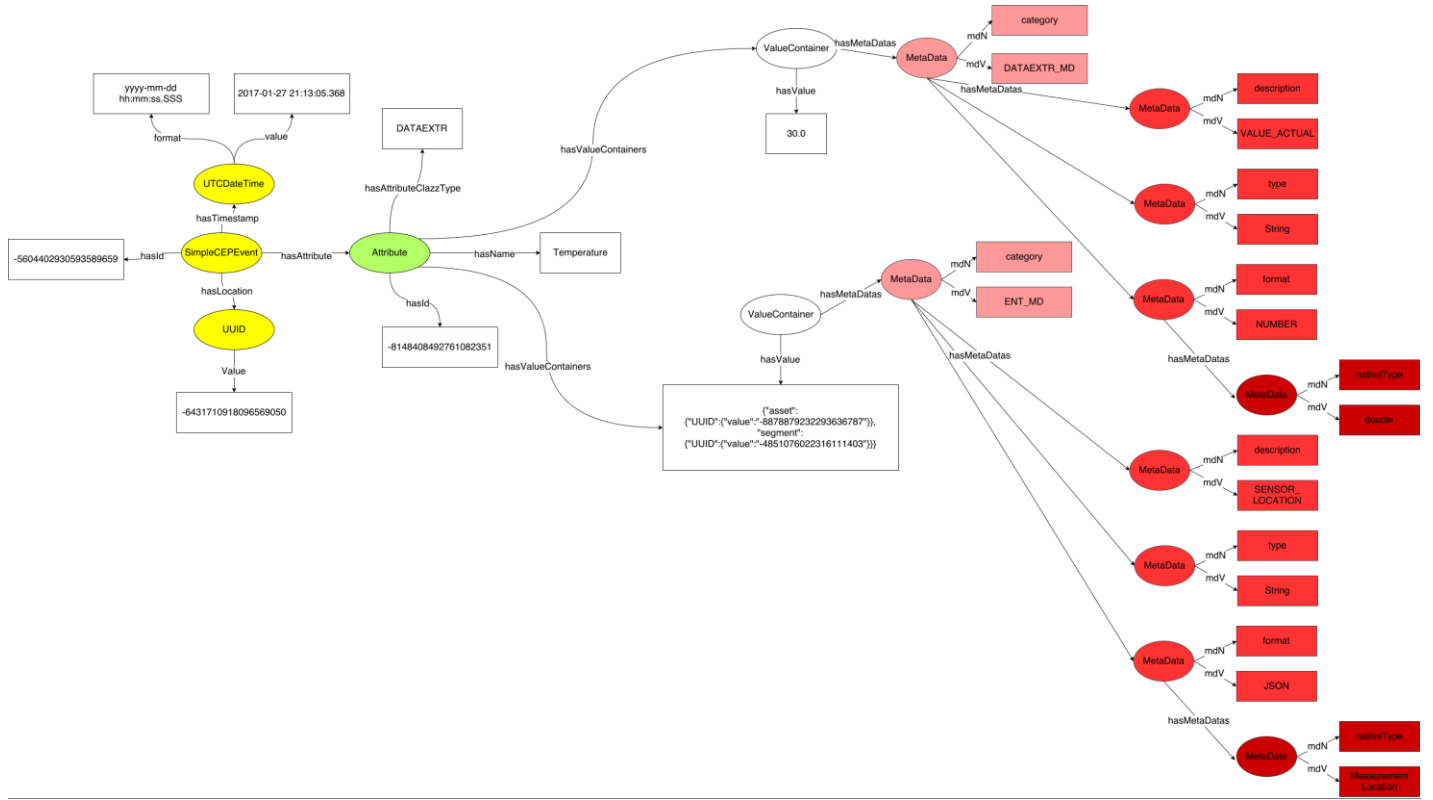


Fig. 6 – Applying the event model for sensor reading: integrating the reading location according to the OSA-CBM

are based on MQTT/AMQP, and the data is exchanged as per the OPC-UA protocol, which is also used to grant to edge gateway the ability to discover and configure the machines, through their MANTIS-PCs. The edge gateway performs pre-processing operations on the data, and it is connected to the logical block called Edge Broker.

The Edge Broker comprises a central messaging bus, where the edge gateway registers itself as OPC-UA Publishers, and the other two logical blocks (HMI and Data Analysis) act as OPC-UA Subscribers. While the Factory (edge gateway) and the Data Analysis interact by means of full AMQP/MQTT channels, the HMI interacts with the central Edge Broker using AMQP/MQTT over websockets.

The HMI acts as OPC-UA subscriber to provide online visualization of the data exchanged in the system, and subscribes to data from both the Factory, and the Data Analysis logical blocks. While in the first case the HMI uses the data to provide online access to either raw sensor data or simply pre-processed data pertaining to a single Factory, in the second case the HMI allows to visualize the results of Data Analysis operations.

The Data Analysis block is located on the cloud, and performs advanced analysis of the data. Machine learning algorithms are applied to compute profiles of the data, with the goal of characterizing automatically the behaviour of machines as either healthy, or leading to a fault in the close future. Moreover, the Data Analysis contains a database that hosts both the parameters of the models used to profile machine behaviours, data from the Factory logical block, and the results of the analysis. The database is based on MIMOSA-compliant

ontologies responding to its OSA-CBM standard. Being based on this standard, all operations related to integration between the MANTIS system and other existing and future platforms can be based on the application of the approach depicted in Fig. 4 to the MIMOSA ontologies and to the data mapped onto them, thus easing interoperability efforts.

5) Pilot Architecture enhanced with MANTIS platform: CPS Message Structure

Within the pilot architecture all the messages are structured according to the models in Fig. 3. These models are used as meta models for enabling the easy and smooth utilization of MIMOSA standard. The Fig. 6 shows the instantiation of the MANTIS CPS Event Information Model. The concepts, relations and hierarchies, presented in Fig. 3, are included in the instantiation but it adds additional model levels and attributes to better cope with the specific application domain.

A Simple CEP Event is presented that encapsulates the value of a sensor reading as well as its location mapped according to the MIMOSA standard. The sensor reading is represented by the Attribute concept. The Attribute concept has two values (encapsulated in the Value Container concept) that are used to encapsulate the information from the environment and the location (SENSOR_LOCATION) of the asset/resource. In particular, the location (structured according to the OSA-CBM standard) contains ids that are used to allow the mapping of the information within the MIMOSA database. Finally, the Value Container concepts are characterized by Metadata concepts that in turn define the class of the value as well as all the information used to describe the value itself.

VI. CONCLUSIONS AND FUTURE DEVELOPMENTS

CPS provide the necessary technological background and approach to facilitate the design and implementation of distributed networked complex systems. CPS-populated systems could potentially have a tremendous impact in all the application domain. In the industrial context (the one depicted in this paper), they could support the optimization of all the activities associated to the production process as well as implement new customer-centric business models. However, this is true only if sophisticated and efficient information models and exchange mechanisms are in place to guarantee that all the actors of a CPS-populated system are capable to exchange and use the information exchanged or in other words are interoperable. Nowadays, the interoperability problem is far to be solved and the dissemination and proliferation of new technologies and devices is growing more and more its complexity. One way to address this problem is to standardize and homogenize the way data are represented and structured to cope with the problem of integrating data from multiple vendor-based systems for the sake of data and information exchange.

In this paper, the MANTIS approach has been described. It is based on the notion that open standards for exchanging maintenance data about assets and resources can lead to a raft of possibilities for implementing advanced maintenance paradigms and strategies in enterprises. However, open standards (like MIMOSA) are not fully mature and require the design and development of meta-models to facilitate the implementation of interoperable solutions.

Future developments include the refinement of the proposed interoperability models, the instantiation in other application domain as well as the definition of a methodology on how data needs to be described within the models to easily and seemly enable the mapping with the MIMOSA standards.

ACKNOWLEDGMENT

This work has been developed with the support of funds made available provided by the European Commission in the scope of ECSEL/H2020 MANTIS Research and Innovation Action (Project ID: 662189) and by the Portuguese Fundação para a Ciência e a Tecnologia (FCT, I.P.) in the framework of project UID/EEA/00066/2013 PEST (Strategic Plan for Science and Technology) for the Centre of Technology and Systems (CTS).

REFERENCES

- [1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *ArXiv E-Prints*, vol. 1207, p. arXiv:1207.0203, Jul. 2012.
- [2] G. Di Orio, G. Cândido, and J. Barata, "The Adapter module: A building block for Self-Learning Production Systems," *Robot. Comput.-Integr. Manuf.*, vol. 36, pp. 25–35, Dec. 2015.
- [3] G. D. Orio, D. Barata, A. Rocha, and J. Barata, "A Cloud-Based Infrastructure to Support Manufacturing Resources Composition," in *Technological Innovation for Cloud-Based Engineering Systems*, L. M. Camarinha-Matos, T. A. Baldissera, G. D. Orio, and F. Marques, Eds. Springer International Publishing, 2015, pp. 82–89.
- [4] *Cyber-Physical Systems - Driving force for innovations in mobility*, | acatech | Springer. 2011.

- [5] J. Soldatos, S. Gusmeroli, P. Malo, and G. Di Orio, "Internet of Things Applications in Future Manufacturing," in *Digitising Industry - Internet of Things Connecting the Physical, Digital and Virtual Worlds*, River Publishers, 2016.
- [6] S. Cavalieri and G. Pezzotta, "Product-Service Systems Engineering: State of the art and research challenges," *Comput. Ind.*, vol. 63, no. 4, pp. 278–288, May 2012.
- [7] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, Jan. 2014.
- [8] S. Bensalem, M. Cengarle, R. Passerone, A. Sangiovanni-Vincetelli, and M. Tornngren, "CPS Technologies," Public D4.2, Sep. 2014.
- [9] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Comput. Ind.*, vol. 81, pp. 11–25, Sep. 2016.
- [10] M. Cengarle, S. Bensalem, J. McDermid, R. Passerone, A. Sangiovanni-Vincetelli, and M. Tornngren, "Characteristics, capabilities, potential applications of Cyber-Physical Systems: a preliminary analysis," D2.1, Nov. 2013.
- [11] B. X. Huang, "Cyber physical systems: a survey," Jun-2008.
- [12] T. Sanislav and L. Miclea, "Cyber-Physical Systems - Concept, Challenges and Research Areas," *J. Control Eng. Appl. Inform.*, vol. 14, no. 2, pp. 28–33, Jun. 2012.
- [13] Steering Committee, "Strategic R&D Opportunities for 21st Century Cyber-Physical Systems, Connecting computer and Information systems with physical world," 2013.
- [14] E. Jantunen, U. Zurutuza, L. L. Ferreira, and P. Varga, "Optimising maintenance: What are the expectations for Cyber Physical Systems," in *2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC)*, 2016, pp. 53–58.
- [15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, New York, NY, USA, 2012, pp. 13–16.
- [16] R. LaMothe, "Edge Computing." Pacific Northwest National Laboratory, Jan-2013.
- [17] "Service Oriented' Architectures, Part 1." [Online]. Available: <https://www.gartner.com/doc/302868/service-oriented-architectures-> [Accessed: 12-Nov-2015].
- [18] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.
- [19] W3C, "Web Services Activity," *Web Services Activity*, 2002. [Online]. Available: <http://www.w3.org/2002/ws/>.
- [20] N. Josuttis, *Soa in Practice*, First. O'Reilly, 2007.
- [21] T. Hadlich, "Providing device integration with OPC UA," in *Industrial Informatics, 2006 IEEE International Conference on*, 2006, pp. 263–268.
- [22] S.-H. Leitner and W. Mahnke, "OPC UA-service-oriented architecture for industrial applications," *ABB Corp. Res. Cent.*, 2006.
- [23] G. Candido, F. Jammes, J. B. de Oliveira, and A. W. Colombo, "SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications," in *2010 8th IEEE International Conference on Industrial Informatics (INDIN)*, 2010, pp. 598–603.
- [24] J. Delsing *et al.*, "The Arrowhead Framework architecture," book chapter of *IoT Automation: Arrowhead Framework*, pp. 45–91, 2017.
- [25] L. L. Ferreira *et al.*, "Arrowhead compliant virtual market of energy," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.
- [26] E. Gilabert, E. Jantunen, C. Emmanouilidis, A. Starr, and A. Arnaiz, "Optimizing E-Maintenance Through Intelligent Data Processing Systems," in *Engineering Asset Management 2011*, J. Lee, J. Ni, J. Sarangapani, and J. Mathew, Eds. Springer London, 2014, pp. 1–9.
- [27] J. Delsing, *IoT Automation: Arrowhead Framework*. CRC Press, 2017.
- [28] "MIMOSA - An Operation and Maintenance Information Open System Alliance." [Online]. Available: <http://www.mimosa.org>. [Accessed: 03-Feb-2017].
- [29] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2012.
- [30] *Enabling Things to Talk - Designing IoT solutions with the | Alessandro Bassi | Springer*.

11.10 Appendix-L- Application of Sensors for Proactive Maintenance in the Real World

Application of Sensors for Proactive Maintenance in the Real World

Michele Albano*, Luis Lino Ferreira*, José Silva*, Edgar M. Silva[†], Pedro Maló[†],
Godfried Webers[‡], Jarno Junnola[§], Erkki Jantunen[§], Luis Miguel Vega[¶],
Iosu Gabilondo^{||}, Mikel Viguera^{**}, Gregor Papa^{††} and Franc Novak^{††}

*CISTER/INESC-TEC, ISEP, Portugal, [†]UNINOVA-CTS, Universidade Nova de Lisboa, Portugal,

[‡]Philips Healthcare, Netherlands, [§]VTT Technical Research Centre of Finland, [¶]InnoTecUK, UK,

^{||}Ikerlan, Arrasate-Mondragón, Spain, ^{**}Koniker, Arrasate-Mondragón, Spain, ^{††}Jožef Stefan Institute, Slovenia

Abstract—Nowadays, collecting complex information regarding a machine status is the enabler for advanced maintenance activities, and one the main players in this process is the sensor. This paper provides an analysis of the sensors that are currently in use for advanced maintenance of industrial machines. It considers different maintenance strategies, and discusses the sensors that can be used to support them. In this context, the paper categorizes different kinds of sensors spanning from common off-the-shelf sensors, to specialized sensors monitoring very specific components and also to virtual sensors. Later on, the paper provides real world examples of project pilots that make use of the described sensors in the context of reactive, preventive, predictive and proactive maintenance.

Index Terms—advanced maintenance, predictive maintenance, virtual sensors, use cases, pilots

I. INTRODUCTION

The advances in industrial electronics are the leading forces for the fourth industrial revolution. In fact, while most factories have traditionally made heavy use of electronics and information technology to automate production (third industrial revolution), the novel paradigm aims at maximizing the benefits of information by the integration between multiple data sources, and by the ubiquitous fruition of the information itself [1].

A field that has gained momentum is the monitoring of industrial systems, since it is on the verge of profound changes. In the close future, maintenance of industrial systems will feature the revolution from traditional monitoring, based on the reaction to malfunctions, to advanced techniques that allow to greatly reduce response time even to zero by predicting faults. The most advanced maintenance paradigm is Proactive Maintenance (PM), which leverages information collected on the machines, and historical data, to infer the proper time to apply each maintenance action.

Building a proactive maintenance service platform is the goal of the MANTIS Project [2], which is a European initiative that aims to enable novel maintenance strategies of industrial machines pertaining to different industries. The project is focused on real life application of the developed techniques, and its pilots are centred on machines/installations that are based on existing designs, but got adapted for the inclusion of novel maintenance techniques. In this sense, the pilots are the testing grounds for the innovative functionalities of the

proactive maintenance service platform architecture, and for its future exploitation in the industrial world.

This paper focus on the sensors which have been identified with interest for MANTIS project pilots and how they are being used in this project.

In Section II, this paper focuses on defining what PM is, and then proceeds to describe the first step of its workflow: the collection of data through advanced sensing techniques. Section III delves into describing hardware and virtual sensors, and Section IV showcases the application of sensing techniques to maintenance in real pilots of the MANTIS project. Section V draws some conclusions on the topic at hand.

II. MAINTENANCE OF INDUSTRIAL MACHINES

Major industries around the world are investing into new machine maintenance techniques. They think that it is time to throw out old ideas, such as reactive maintenance a.k.a. breakdown maintenance [1].

The right maintenance program can improve efficiency, reduce downtime and save money. Four different maintenance plans of increasing complexity are used today: 1) reactive maintenance; 2) preventive maintenance; 3) predictive maintenance; and 4) proactive maintenance [3].

Reactive maintenance can be described as a fire-fight approach, meaning that equipment is only replaced or repaired after it breaks. It has the advantage of minimizing manpower to keep things running. Disadvantages reside in large levels of scrap, unpredictable production capacity and high overall maintenance costs.

Preventive maintenance, which includes both time and usage-based maintenance, relies on periodic maintenance execution that can range from equipment lubrication to replacement. Maintenance tasks are performed based on specific periods of time or amount of machine usage (number of working hours). This approach requires production stop for maintenance, but it improves equipment lifetime and it reduces malfunction probability [4].

Predictive maintenance, or condition-based maintenance, relies on physical measurements of the equipment condition (e.g.: temperature, vibration, noise, lubrication, corrosion [5]). When these measurements reach a certain level, preventive

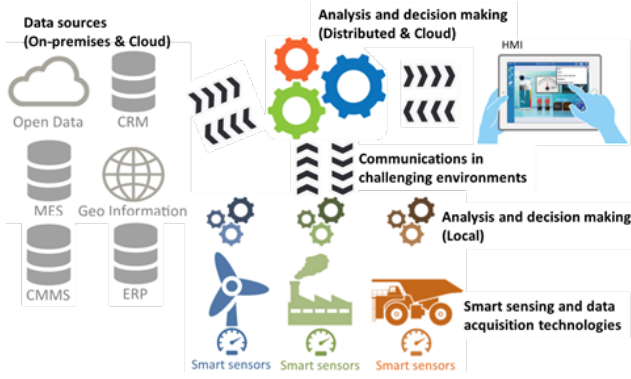


Fig. 1. General Architecture

maintenance is applied. In this sense, maintenance only happens in a need-based when a certain threshold is overcome.

Proactive Maintenance (PM) includes different actions, from system design phase, workmanship, scheduling and maintenance procedures, to the usage of communication technologies, feedback information and optimization techniques [6]. PM benefits from the two previous maintenance strategies, since preventive and predictive methods are also applied. PM goes further by focusing on the causes of the problems, and dealing with them before problems occur.

A PM service platform should have distributed processing chains, which distil raw data into knowledge, while minimizing bandwidth usage. Therefore, need arises for an integrated domain knowledge system that includes key technologies such as (see Figure 1): Smart sensors, actuators and cyber-physical systems; Robust communication systems for harsh environments; Distributed machine learning for data validation and decision-making; Cloud-based processing, analytics and data availability; HMI to visualize information.

The foundation of such service platform is the sensing capability, which is bestowed unto sensors and has the responsibility of nourishing the system with vital information from machinery. In fact, PM relies on item/equipment constant condition monitoring and evaluation to avoid machine failures. Condition monitoring is achieved through sensor data collection and analysis, which enables, in-time, the prediction of equipment failures.

In general, sensors should be robust enough to withstand harsh industrial environments, cheap, and able to sense the wanted phenomenon. There might be also measurement locations where multiple phenomena must be measured at the same time.

Being on the first step of the maintenance process, sensors should be precise, accurate and reliable, and these characteristics cannot be traded off in favour of cheaper sensors. On the other hand, the choice on the measurement locations to monitor is a fundamental part of the trade-off between costs and benefits. Currently, the moderate high cost of the sensors leads to the measurements of few locations that are important, for risks for health or production. In the future, as sensors get cheaper, it will be possible to monitor lower cost-effective

locations with benefits overcoming the losses.

Electronic devices have limitations and real sensors are no exception. An important aspect of sensor application in system maintenance is ageing. Sensor characteristics are degraded with time and for this reason they should be monitored as well. This problem can be mitigated by protecting the sensor from the rigours of industrial environments (see Figure 2 for an example of a protected sensor), still operation in harsh environmental conditions (high temperature, aggressive medium, location, frequency and intensity of loadings) still accelerates the ageing rate of sensors, which manifests in deterioration of material and the defects in the sensing structure. For example, thick-film piezoresistive pressure sensors are prone to accelerated ageing when under mechanical loads [8], and humidity has long-term effects on the sensors' response [9].

Hence, there is often a need for optimization of sensors for specific conditions, particularly for the use in harsh environments. A reasonable solution to this problem can be the right choice of a suitable sensor technology together with the implementation of an appropriate measurement (monitoring) strategy and proper intelligent interpretation of sensor data (e.g., this might be part of the virtual sensor functionalities described in Subsection III-C).

III. SENSORS FOR MACHINE MAINTENANCE

Sensors used in advanced maintenance operations can be classified into two types: hardware and virtual. Whilst the first type concerns physical sensors that are added or already exist on the machine, the second type of sensors derive new results by means of signal processing of data from one or several physical sensors.

The rest of the section starts by discussing physical sensors, and in particular focuses on the most common mass-produced sensors (Subsection III-A), and then briefly describes custom sensors that are created for specific maintenance applications (Subsection III-B). Later on, the section copes with virtual sensors and their current implementation (Subsection III-C).

A. Off-the-shelf Sensors

The work in [7] examined more than 300 devices in 12 different applications, and obtained a distribution of the sensor types by application usability and sensor nodes availability (see Figure 3). In particular, seven sensor types were identified as being the most common sensors, namely temperature, acceleration, light, force, audio, humidity and proximity. The anal-



Fig. 2. Encapsulated wireless sensor node & PCB

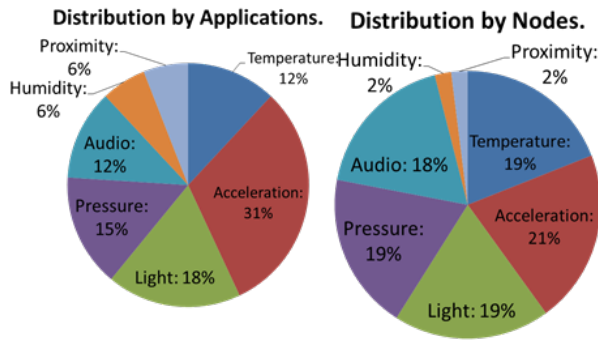


Fig. 3. Distribution of Sensor Types

ysis takes into account that most sensor nodes offer multiple physical data sources (e.g.: pressure, light and temperature).

Temperature effects can take place on materials (solids or fluids) and components. These effects can have a very significant impact on machines operation by causing increased wear, hydraulic systems degradation, materials expansion, etc. Temperature sensing allows for continuous analysis of temperature variation or its stability. For example, scanning bearing housing on motors can prevent major failures. Monitoring fluids' temperature is also useful, since some properties of fluids degrade when temperature increases.

Since mechanical systems are composed by moving parts that deteriorate over time and generate vibration, collecting acceleration data allows early detection of roller elements bearing faults, gear wear, etc.

Measuring pumps pressure can reveal physical changes in the pumps. Operating conditions, such as fluid type, temperature and speed, affect the pressure, and if pressure takes a value outside a given range, there is the possibility of damaging parts. Moreover, pressure variation can lead to cavitation (creation of vapour cavities in a fluid), which can lead to material damage [3]. Cavitation can be sensed by means of pressure, vibration or acoustic emission measurement.

Usage of light sensors may include the detection of material cracks and object detection. By placing an object between a light source and a light sensor, cracks can be detected by the amount of light that goes through the object. Moreover, it is possible to detect an unwanted object in a certain area, for example, a person near a cutting material machine and shutdown the machine safely.

Acoustic (audio) monitoring is strongly related to vibration sensors. While audio sensors listen a component, vibration sensors register the motion of the component they are rigidly attached to. Acoustic sensors are commonly used to monitor bearing and gearboxes, in order to detect any working/movement variation.

Monitoring the percentage of humidity in a certain environment can be useful, since for example, high levels of humidity in an injection molding process line can add moisture to resins, potentially impeding that parts are molded properly. In gearboxes, the accumulation moisture can lead to gearbox corrosion, reduced efficiency and breakdown.

Proximity sensors can be used to measure parts displacement, improper presence of objects, or even vibration in rotational components. Another feature is the non-contact measurements, which makes use of sonar or infrared light emission to detect the presence of objects in a area.

B. Custom sensors

Many other kinds of sensors can be found in specific applications. Usually, these sensors are not mass produced, their structure presents a high degree of customization, and they retrieve very specific environmental data. Among the plethora of the custom sensors, there are sensors capable of performing crack detection, torque measurement, analyse wear of material and retrieve oil status [10].

The early detection of cracks, allows the prevention of fracture failures. These cracks can be produced by an applied stress concentration, excessive stress over time, overload, defective assembly, or environmental conditions. Crack detection (through non-destructive methods) can be performed using different techniques like radiography, ultrasonic, penetrating liquid, magnetic particle inspection, etc.

Several sensing techniques and technologies can be applied to estimate or compute torque measures. Through components speed is possible to calculate torque and torque brake; an alternative method is using pressure sensors to correlate torque brake.

Other custom sensors can target deviation of torque, brake torque and friction values from the normal values, since they can detect shaft misalignment or the presence of wear particles, which in turn are predictors for equipment malfunction.

Another type of custom sensor is the oil sensor. Oil sensors can be divided into different groups based on the data under measurement, such as oil condition, oil temperature and oil pressure. Oil condition sensors have the capability to detect ferrous particles, water, viscosity changes, etc. Oil condition monitoring allows detection of lubricant related engine wear and lubricant quality degradation, among other problems. Early problem detection leads to on-time, preventive adjustments that reduce machinery downtime.

C. Virtual Sensors

The virtual sensor is a technology used to retrieve more effective and accurate information from collected data [11]. Virtual sensors make use of readings collected either by multiple networks, or from a single sensor. Data are combined from multiple sources (e.g. temperature, humidity, CO2) and process models are applied to compute new outputs, based on not only on current sensor values but also on its time series.

The Virtual Sensor Architecture, whose view is represented in Figure 4, can retrieve sensor data in two common ways: in an *event-based* acquisition, meaning that physical sensors will make the data available (generate events) when certain conditions are met, or in a *time-based* fashion, where the virtual sensor will periodically inquire the physical sensors for new data. This step is accomplished in the *Acquisition Method* module.

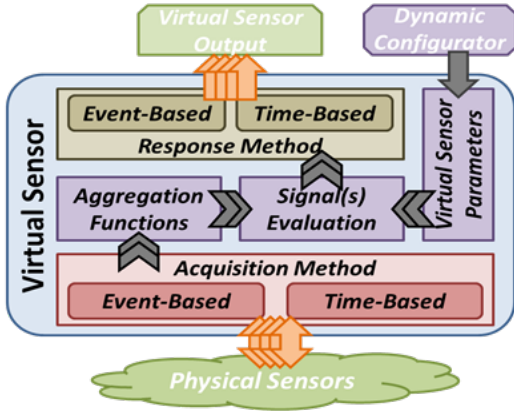


Fig. 4. View on the Virtual Sensor Architecture

The *Aggregation Functions* module has the task of applying common mathematical functions (e.g. temperature average of different sensors in a same room) or complex models (e.g. wear prediction model). The entity/user managing the virtual sensor has the capability (through the *Dynamic Configurator* module) to change threshold parameters used to generate outputs or to change signal evaluation parameters. Configuration parameters are kept in the *Virtual Sensor Parameters* module, and are used by the *Signal(s) Evaluation* module to perform an analysis of the results achieved in the *Aggregation Functions* module. Finally, similar to the *Acquisition Method*, the *Response Method* module is able to generate the virtual sensor output, by the same two common paradigms, i.e. through events or in a time-based manner.

IV. USE CASES

This section presents four pilots in which the usage of PM can facilitate maintenance interventions, cost reduction, equipments lifetime, and in general provide added value to the monitoring process. All use cases feature real world factories and installation, and therefore provide a connection between the role that PM is supposed to hold, and what is actually happening in real installations as technology evolves and our economy and society change with it.

The first pilot (Subsection IV-A) exploits the composition of data from off-the-shelf and custom sensors, the second one (Subsection IV-B) focuses on the use of custom sensors, the third one (Subsection IV-C) features virtual sensors, and the fourth one (Subsection IV-D) uses a set of sensors deployed dynamically by using a robotic platform.

A. Monitoring of a Sheet Metal Bender

The Sheet Metal Bending Process use case, whose architecture is represented in Figure 5, involves detection, prediction and diagnosis of malfunctions in a sheet metal bender machine that pertains to the Greenbender family, manufactured and commercialized by ADIRA (see Figure 6). The machine is able to exert a force up to 2200 kN using 2 electric motors of 7.5 kW each, and it is able to bend metal with high precision.

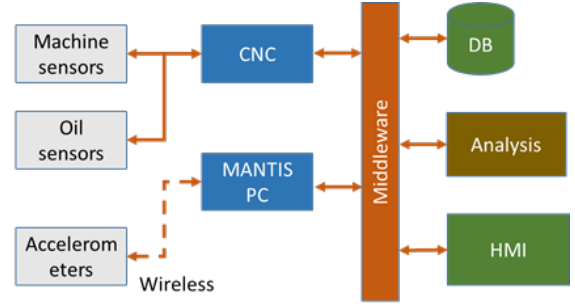


Fig. 5. Monitoring architecture for Sheet Metal Bender

The use case considers two scenarios. In the first scenario, pertaining to reactive maintenance, a malfunction in a component raises the need for the replacement of component(s), and the goal of the work is to allow the monitoring subsystem to detect a potential failure in the industrial process, perform proper analysis, and communicate the replacement operation that must be implemented. The second scenario aims to predict machine failures before they occur, by means of applying machine learning techniques to data collected from the sensors, thus realizing PM for the machine.

Collection of data can either be direct or indirect, the latter involving the access to the data already stored on the machine monitoring subsystem. On the other hand, direct data collection involves receiving data from sensors, which can be either already present in the machine, or deployed purposely for PM roles. Two sets of new sensors were installed.

Sensors responding to the Custom Sensors category (see Subsection III-B) are applied to the oil that lubricates the machine's hydraulic circuits. The sensors monitor the temperature of the oil and its quality, being the latter related with presence of water particles and other impurities in the oil. The system that analyses the oil temperature and condition (contaminations like water, particles, glycol, etc.) consists of two parts. The first part is the sensor unit (Hydac Sensor AS1008) and the second a data acquisition and computation board.

The sensor reads temperature from -25 to 100 Celsius degrees, and saturation from 0 to 100 percent. Both signals are given by the sensor in current using a 4-20 mA interface. The data acquisition/computation module receives the signals, convert them, and can export the data through two interfaces, as an analogic voltage signal with a range from 0 to 10 Volts, or through a RS485 communication protocol after an analog-to-digital conversion process. In the specific use case, the analogic voltage signals can be provided to the machine numeric controller for on-site view or direct analysis by the machine PLC, and the digital data can be sent to the MANTIS-PC.

The second set of sensors is made of two accelerometers, which are off-the-shelf sensors (section III-A) and monitor the blade that performs the bending of the metal sheet. The machine is depicted in Figure 6, where the two sensors are highlighted. The sensors collect data from the movement of the

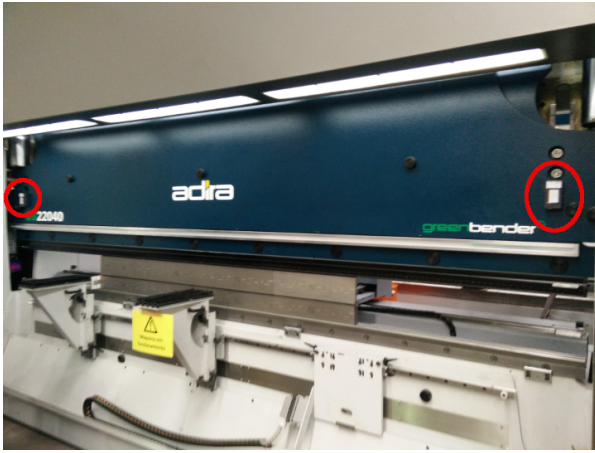


Fig. 6. Frontal view of the machine with two IMU

blade of the press, and from vibration patterns caused by the hydraulics. In fact, the vibratory pattern can be related to the condition of the machine's bending motors, and the collected data can thus be used to perform PM of the machine. For example, if the hydraulic pistons are starting to malfunction, due to the existence of particles in the oil then a different vibration pattern can be detected. These sensors can also be placed on a different machine location for malfunction diagnosis.

The sensors are based on the Arduino 101 platform that provides a 3-axis accelerometer with a maximum amplitude range of $8g$, and are powered by two 9 Volt batteries in order to ease components' installation and testing. In the specific case of this pilot, the sensors were configured for a lower measurement range ($0g$ to $2g$) to attain a better accuracy. The *CurieBLE* library is used to send data from IMU (Inertial Measurement Unit) to the MANTIS-PC wirelessly via Bluetooth Low Energy (BLE) using the Generic Attribute Profile (GATT) [15]. Preliminary experiments have confirmed that the maximum distance for this technology is 30 meters, as stated in the BLE specifications.

The MANTIS-PC is a Raspberry Pi 3 Model B that acts as a BLE server, a data-converter, a simple Human Machine Interface (HMI) and a middleware client. The component uses a server-side JavaScript program built over Node.js and the noble library to collect values from both sensors with a period of 30 milliseconds, and sends them to cloud using a Middleware based on the AMQP [16] protocol. In terms of data conversion, raw values are mapped to the $2g$ range ($-2g$ maps to a raw value of -32768 and $+2g$ maps to a raw value of 32767). The simple HMI presented by the MANTIS-PC uses a server-side/client-side JavaScript based on Node.js to send warnings to management personnel. The interface is based on the *Highcharts* library and it enjoys the "full-responsiveness" capability of the Highcharts.

B. Press Machine Maintenance

A second use case focuses on press machine maintenance, monitored continuously by a broad and diverse range of

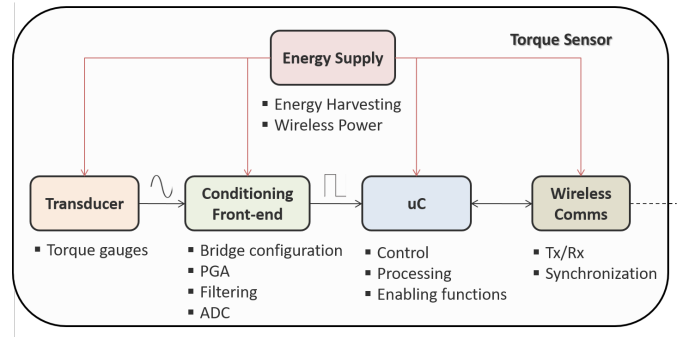


Fig. 7. Wireless torque sensor node block diagram

intelligent sensors that keep track of its operational conditions. A stamping press is a metal working machine used to shape or cut metal by deforming it with a die.

A mechanical press, during its active lifetime, might be capable of giving more than 40 million strokes, insofar as it is used and maintained appropriately. The machine under study belongs to Fagor Arrasate, whose customers demand products with high levels of quality and availability. These latter characteristics are in contrast with the production downtimes caused by maintenance and repair operations. Therefore, it was decided to incorporate cyber-physical systems in the most critical components, to facilitate predictive and proactive maintenance activities.

This pilot applies novel maintenance strategies that are enabled by a cloud service platform, which on the other hand leverages on data captured continuously, monitored, transmitted, stored and analyzed by intelligent sensors responding to the Custom Sensor category (see section III-B). In particular, two applications collect data from multiple data sources related to press structural health, cranks forces and wearing of gears and bushings.

A first application is focused on Structural Health monitoring by means of an early detection of cracks/fissures in the press' head and caps, which enables to prevent damaging fracture failures caused by press' damping and stress concentration in certain parts of the structure. Both crack gauges and conductive inks are being used, the last allowing higher surface measurements based on the change of the conductivity/resistivity of the drawn circuit. In this case, the cracks on the target make the ink that is spread on the surface break and thus increase the resistivity of the circuit. The change in conductivity is measured by collecting the value of the current as an analog input and comparing it with a threshold.

The second application is represented in Figure 7, and it implies the sensorization of a gear shaft. A shaft-adapted wireless sensor node [12] comprises a transducer (torque oriented gauges), a signal conditioning circuit and a signal processing software, the latter allowing a local preprocessing and treatment of the collected data, by means of intelligent functions. In fact, a couple of approaches are implemented. First, a finite iteration based auto-zeroing algorithm is applied, which

setups the proper gain and offset values for the system, taking into account gauge's signal and measured signal feedback, settling the gauge's signal to a desired point (virtual ground), thus enhancing system's dynamic range and avoiding signal saturation. Secondly, digital data is retrieved and preprocessed, reducing its payload by means of averaging. These rough data is transmitted to a gateway based on the Beagle Bone platform via a specific approach to industrial protocols, as standard ones either lack of deterministic features (e.g. IEE802.15.4) or scalability (e.g. IEEE 802.15.1). Moreover, industrial solutions (e.g. ISA 100.11a) do not provide tools for guaranteeing sampling synchronization, which is critical for certain applications (e.g. SHM), therefore, a TDMA MAC has been placed on top of the physical layer and specific synchronization elements have been added for obtaining synchronized ADC conversions in nodes [14]. Finally, the necessary calculations to obtain torque values (Nm) are done in a computer connected to the gateway.

The fact that it has to be applied in a rotatory and shaky shaft (working at approximately 88 rpm) implies, on one hand, the need to develop a robust housing to protect it from vibrations and lubrication oiliness [13]. On the other hand, a power friendly approach must be considered, such that the wireless sensor can work without external grid power. Current design allows a finite duration of the measurement process, as the system is powered with a small Li-Ion battery. Thus, supplementary solutions such as wireless power or energy harvesting can be analysed, in order to provide a parallel power source besides the battery. Additionally, energy optimization techniques can be implemented at microcontroller level to lower down system's power consumption and enhance the duration of the measurement process and sensor's autonomy.

C. Maintenance of Medical Devices

Modern medical devices have a large number of embedded hardware sensors and virtual sensors, which are used for monitoring their condition and contribute to predictive and proactive maintenance of the device. In the context of this pilot, represented in Figure 8, hardware sensors cover the complete range of off-the-shelf sensor type (section III-A), and data are distilled into more advanced information by means of virtual sensors (section III-C) executed on a stand-alone sensor box called *e-Alert controller* and manufactured by Philips.

Sensed data reveal physical phenomena that may lead to part failure or even device failure. The sensor data by itself has limited information, but its combination of sensor data with service data (such as maintenance records) can create the information that is needed to develop PM on-device and off-device. Availability of high quality service data is as essential as the availability of device sensor data.

The e-Alert controller monitors autonomously environmental and operational conditions of the medical device, and generate electronic notifications to the different stakeholders of the medical device. The e-Alert controller is based on a Raspberry Pi platform and can sample connected sensors,

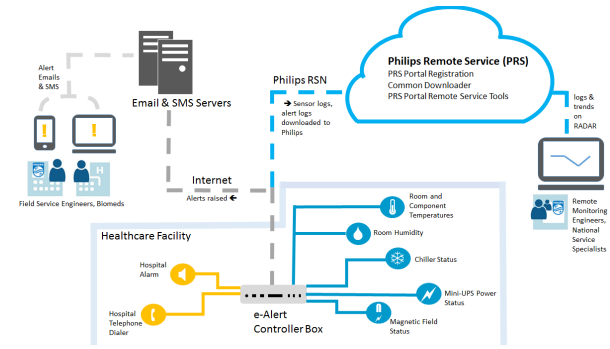


Fig. 8. Sensor box context diagram

for example, temperature sensors, humidity sensors, magnetic field sensors.

The physical sensors are off-the-shelf one-wire sensors and are connected to an interface box that can support up to 8 sensors each. The interface box is connected to one of the inputs of the e-Alert controller box. Multiple interface boxes can be daisy-chained, providing a scalable sensor platform that can be tailored for the specific device under monitoring. A mini-UPS can be used to power the e-Alert controller in case of power failures so that it can continue to monitoring environmental conditions. The mini-UPS provides an electrical interface to indicate that the controller is running on battery power.

The e-Alert controller box acquires sensor values once per minute and checks these values against configured thresholds. To avoid false positives, a sensor value must be out-of-spec for a number of consecutive samples before an alert is raised. If a sensor value remains outside the configured threshold, the e-Alert controller box sends an Email or SMS alert to configured alert receivers.

The e-Alert controller software provides a web-based user interface to configure sensors, thresholds, Email/SMS server, and Email/SMS receivers. The e-Alert controller is connected to the hospital network, and healthcare facility staff can access the user interface of the e-Alert controller, which provides capabilities to view the history of sensor values for root cause analysis. The development of intelligent on-device schemas aims to detect devices operating outside operational limits at the earliest and in an optimized workflow. Specific workflows are required to fulfil the different needs and match the different capabilities of the users of the medical device.

The e-Alert controller also provides a capability to interface with the medical device manufacturer. For this purpose, connectivity to Philips Remote Service (PRS) can be configured. With this interface, sensor values can be aggregated by means of more virtual sensors and analyzed statistically. This enables the manufacturer to determine an operational profile, specific to that medical device. This information can be used to fine-tune the configured alert thresholds for that specific device. Moreover, the aggregated data of the medical device are made available for offline data processing and analysis, to determine the minimal data set that is useful for PM, with the



Fig. 9. Wind Turbine area of interest

goal of saving bandwidth when transferring data for PM.

D. Monitoring of a Wind Turbine

This use case considers monitoring the status of a wind turbine, represented in Figure 9, by means of a set of Acoustic Emission sensors (AE) placed on the tower next to its joint with the nacelle.

The monitoring process is represented in Figure 10. Typically, all rotary equipment produces an acoustic signature (Acoustic Emission) which is propagated through the material. The purpose of the presented technique is, by means of AE sensors, to acquire those signals, process them and compare them over time to verify the structural health of the wind turbine. The wind turbine structural noise is the basis for the degradation analysis. Such noise is composed by the contribution of each wind turbine rotary components, the mechanical forces generated by the blades movement, and wind.

During the normal operation of the turbine, all the components are rotating and producing fairly constant and stable signals. Those are treated as benchmark signals and considered as background or Gaussian white noise. If a malfunction occurs and one of the rotary components is permanently damaged, the acoustic signature would change. It is possible to identify three main different structural changes: a) Degradation of the bearings/gearbox that increases the friction forces applied on the rotary shaft and reduces the power transmission ratio while producing higher floor levels of noise. b) Shaft and/or bearing misalignment, which produces periodic acoustic signal patterns that can be detected and analysed. c) Mechanical cracks, which generate new frequency harmonics, can be visualized as spikes outstanding from the regular noise.

To undertake the signal processing supported by the theory stated below, the signals are amplified and their mean levels are removed before being acquired by an analogue digital converter. The pre-amplifiers are installed as close as possible to the acoustic emission sensors in order to improve the signal noise ratio (SNR). The analogue digital converter (ADC) is

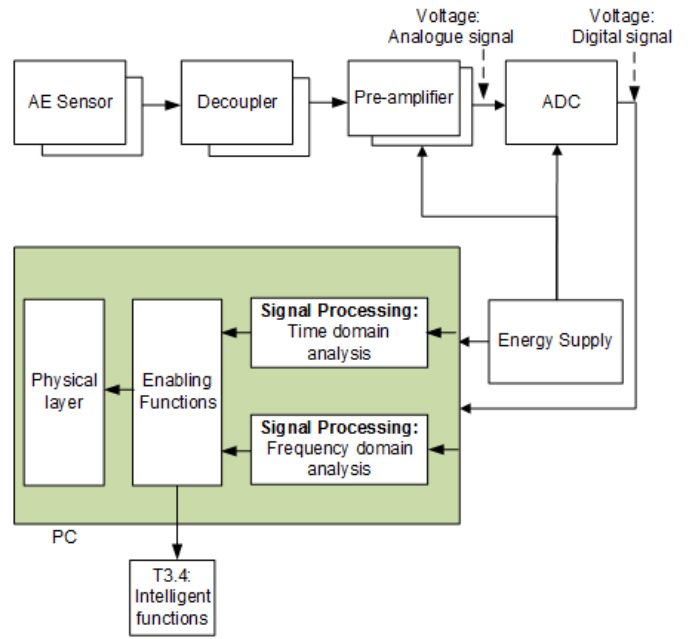


Fig. 10. Wind Turbine Monitoring Process

connected directly to a PC used to run the signal processing algorithms and export the signal to be further processed.

The condition monitoring system implements two different measurement strategies. The Periodic and Automatic Periodic Monitoring (PAPM) strategy considers that the monitoring subsystem is permanently installed and attached on the top part of the wind turbine tower, and the monitoring process is periodically executed according to the schedule previously defined by the end user. The Spot Measurement (SM) strategy considers that the measurement process is executed whenever the end user requests it, disregarding the previously defined schedule; this implies the interaction with the end user, but on the other hand it allows executing multiple measurement when a problem is expected.

The actual inspection is undertaken by means of a robotic platform able to climb up to the area of interest using magnetic adhesion. Once the position has been reached, the AE Sensors, which are installed on board, are deployed and so the signal acquisition begins. The acquisition is done through a Red Pitaya which makes the analogical-digital conversion. The data are transferred in real time to the ground control box where there are two possible options: a) Real time signal processing can be executed to visualise and assess in situ the status of the wind turbine. b) The data can be uploaded into the cloud for post processing.

V. CONCLUSIONS

The paper discussed the different maintenance strategies that are used nowadays, and provided a taxonomy of existing sensors. Later on, it presented four different real world pilots that showcase advanced maintenance operation.

Future work will develop the single use cases, finalize their implementation and compare the benefits obtained by means

of different maintenance strategies, as described under each use case.

ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234); also by FCT/MEC and the EU ECSEL JU under the H2020 Framework Programme, within project ECSEL/0004/2014, JU grant nr. 662189 (MAN-TIS).

REFERENCES

- [1] K. Schwab, *The fourth industrial revolution*, World Economic Forum, Geneva, Switzerland, 2016.
- [2] E. Jantunen, U. Zurutuza, L. L. Ferreira, P. Varga, *Optimising Maintenance: What are the expectations for Cyber Physical Systems*, The 3rd International IFIP Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC' 16). Vienna, Austria on April 11-14, 2016.
- [3] E.C. Fitch, *Proactive Maintenance for Mechanical Systems*, Dr. E.C. Fitch technology transfer series, Elsevier Science, ISBN: 9781856171663, 1992.
- [4] L. Swanson, *Linking maintenance strategies to performance*, International Journal of Production Economics, 18 April 2001, Vol. 70, Issue 3, pp 237-244.
- [5] R. Eade, *The importance of predictive maintenance*, Iron Age New Steel 13 (9), 1997, pp 68-72.
- [6] B. S. Dhillon, *Engineering Maintenance: A Modern Approach*, CRC Press, Boca Raton, 2002.
- [7] M. Beigl, A. Krohn, T. Zimmer, C. Decker, *Typical Sensors needed in Ubiquitous and Pervasive Computing*, in Proceedings of the First International Workshop on Networked Sensing Systems (INSS '04), 2004, pp. 153-158.
- [8] M. S. Zarnik, V. Sedlakova, D. Belavic, J. Sikula, J. Majzner, P. Sedlak, *Estimation of the long-term stability of piezoresistive LTCC pressure sensors by means of low-frequency noise measurements*, Sens. Actuators A Phys. 199, 2013, pp 334-343.
- [9] M. S. Zarnik, D. Belavic, *The effect of humidity on the stability of LTCC pressure sensors*, Metrol. Meas. Syst. XIX(1), 2012, pp 133-140.
- [10] C. W. De Silva, *Sensors and Actuators: Engineering System Instrumentation*, CRC Press, 2015.
- [11] L. Lichuan, S. M. Kuo, M. Zhou, *Virtual sensing techniques and their applications*, Int. Conf. on Networking, Sensing and Control, ICNSC '09, Okayama, 2009, pp. 31-36.
- [12] Z. Herrasti, I. Gabilondo, J. Berganzo, I. Val, F. Martínez, *Wireless Sensor Nodes for acceleration, strain and temperature measurements*, 30th Eurosensors Conference (EUROSENSORS '16), 2016
- [13] M. Tijero, E. Arroyo-Leceta, Z. Herrasti, I. Gabilondo, I. Reinales, J. Anduaga, J. Berganzo, *Wireless Energy-data Transmission and Packaging Solution for Smart Systems to Monitor Industrial Components*, Procedia Engineering 168:1589-1592, December 2016, DOI: 10.1016/j.proeng.2016.11.467
- [14] Z. Herrasti, I. Val, I. Gabilondo, J. Berganzo, A. Arriola, F. Martínez, *Wireless sensor nodes for generic signal conditioning: Application to Structural Health Monitoring of wind turbines*, Original Research Article, Sensors and Actuators A: Physical, Volume 247, 15 August 2016, Pages 604-613
- [15] Kuor-Hsin Chang, *Bluetooth: a viable solution for IoT? [Industry Perspectives]*, IEEE Wireless Communications 21.6 (2014): 6-7.
- [16] Michele Albano, et al, *Message-oriented middleware for smart grids*. Computer Standards & Interfaces 38 (2015): 133-143.

11.11 Appendix-N- FlexHousing: Flexoffer concept for the energy manager

FlexHousing: FlexOffer concept for the energy manager

Joss Santos, Michele Albano,
Luis Lino Ferreira, Jose Silva
CISTER Research Center

Polytechnic Institute of Porto
Rua Dr. António Bernardino de Almeida
4200-072 Porto, Portugal
Email: {jodos, mialb, llf, jbmds}@isep.ipp.pt

Petur Olsen
Center for Embedded Software
Systems (CISS)

Department of Computer Science
Aalborg University, Aalborg, Denmark
Email: petur@cs.aau.dk

Luisa Matos
Virtual Power Solutions
Instituto Pedro Nunes

Rua Pedro Nunes - Edifcio D
3030-199 Coimbra
Portugal
Email: lmatos@vps.energy

Abstract—Energy management in buildings can provide massive benefits in financial and energy saving terms. It is possible to optimize energy usage with smart grid techniques, where the benefits are enhanced when the energy consumer can trade the energy on energy markets, since it forces energy providers to compete with each other on the energy price. Anyway, two hurdles oppose this approach: energy markets limit trading activities to large quantities of energy, thus impeding access for small consumers, and the devices providing control over appliances do not interoperate with each other. This work considers using the FlexOffer (FO) concept to allow the consumer to express its energy needs, and FO-related mechanisms to aggregate energy requests into quantities relevant for energy markets. Moreover, the presented system, called FlexHousing, is based on the Arrowhead framework, and exploits its Service Oriented mechanisms to provide interoperability. The implemented FlexHousing system uses multi-level FO aggregation to empower either the final user, for example the owner of an apartment, to manage his own energy by defining his flexibilities, or to offload this responsibility to an energy manager who takes care of all the apartments in a building or set of buildings.

Index Terms—Arrowhead, Service Oriented, Prosumer, Smart Grid, Interoperability

I. INTRODUCTION

Energy management in buildings can provide massive benefits, with regards to financial gains, pollution reduction, and total energy saving, since the energy consumed in buildings is one of the major factors in global energy expenditures. For example, it is estimated [1] [2] that energy consumption by the residential and commercial sectors cover together 39% of the total energy consumption in the US, and that most of that energy is consumed in buildings.

The application to buildings of techniques from the smart grid paradigm, such as remote control of appliances and autonomous acquisition of energy, can provide dramatic savings [3], especially when energy is bought from energy markets, since this forces energy providers to compete with each other on the energy price. Current advances are contributing to the feasibility of this vision, since communication protocols are converging to a standardized vision of the last mile of the smart grid [4], and there is increased competition between energy utilities, leading to consumers' savings. On the other

hand, two hurdles oppose the application of the smart grid to buildings: energy markets limit their trading activities to large quantities of electricity, thus impeding access for small consumers, and the devices providing control over appliances do not provide interoperability with each other.

This work considers using the FlexOffer (FO) concept to allow the consumer to express its energy needs [5]. FOs aim to balance energy demand and response by synchronizing consumption with production. They permit exposing demand and supply loads with associated flexibilities in time and quantity for energy commerce, load levelling, and different use-cases. To put it in a simple way, a FO specifies an amount of energy, a duration, an earliest begin time, a latest finish time, and a price, e.g., "I want 50 kWh over 3 hours between 5 PM and 12 PM, at a maximum price of 0.25€/ kWh".

In order for the FO to be relevant for the Energy Market, the FO concept considers that FOs can be combined or aggregated together when they are on the same energy grid. A system called Aggregator receives the FO from all the house-holds, combines them, and sends a unique aggregated FO to the Market. Multiple Aggregators can be employed in sequence, thus creating tree-shaped topologies, and producing FOs that are large enough to be sent to the Market. When the Market reply arrives, the responsibility of each Aggregator is to redistribute the energy between the underlying Aggregators, until it is delivered to the house-holds supervised by the bottom-layer Aggregators.

To take care of the second hurdle—the interaction with the appliances—this work made use of the Arrowhead Framework [6], which normalizes the interaction between systems in a distributed system by means of *services*, which are consumed and produced by software *systems* executed on *devices*. In fact, all the systems related to FOs are already Arrowhead-compliant, and past work [7] had already studied how to extend the Arrowhead Framework to custom protocols by means of adapters.

The present work makes use of smart plugs that obey custom protocols, but which can be driven by the interaction with a simple Service Oriented Architecture (SOA) interface, to which the Arrowhead Framework was extended easily. The

employment of these plugs allowed to enforce the energy consumption schedule sent back from the Aggregator, and to collect data regarding energy consumption of the appliances.

Two main types of Energy Markets exist, the day-ahead and the intraday. While the second kind allows to buy energy that is delivered by the grid in close to real-time, the first kind requires the buyer to plan ahead, since energy is delivered a day from the placement of the order. On the other hand, the day-ahead markets provide two important advantages: the price of the energy is lower, and it allows the bulk energy producers to optimize energy production and thus to refrain from using peaking power plants, which are more expensive and pollutant.

The implemented system, named FlexHousing, allows the application of the FO concept to buildings in real settings. The multi-level approach enabled by the Aggregator empowers either the final user, for example the owner of an apartment, to manage his own energy by defining his flexibilities, or to offload this responsibility to an energy manager who takes care of a whole building or set of buildings. Other Aggregator will take care of aggregating the FOs of multiple buildings until they reach a magnitude of interest for the Energy Market. Moreover, FlexOffers can be created at different times, thus allowing the use of the day-ahead markets as much as possible, while still providing the flexibility of the intraday market.

The rest of the paper is structured as follows. Section 2 provides background information, in particular regarding the FlexOffer concept, the Arrowhead platform that is used to provide FlexOffer-related services, and on related work. Section 3 describes the architecture of the FlexHousing system of systems, and Section 4 provides implementation details. Section 5 is devoted to the test results obtained while experimenting with the implemented system, and Section 6 draws some conclusions on the topic at hand.

II. BACKGROUND INFORMATION

A. FlexOffer Concept

A FlexOffer is a data structure for expressing flexibility in energy consumption (or production). A FlexOffer contains a number of slices, each representing a minimum and maximum consumption for a given time period. A set of slices is called a profile, and can be used to express the consumption profile for a device.

A FlexOffer allows the profile to be shifted in time by specifying an earliest start time and a latest end time. This provides two types of flexibility to a FlexOffer: energy flexibility in the bounds of the slice, and time flexibility in the start and end time.

A Schedule can be attached to a FlexOffer. The schedule includes a start time and an assigned energy amount for each slice. Figure 1 shows an example of a FlexOffer. The green area shows the flexible energy, the grey area is minimum required energy. The red lines show the final schedule, sent by the Aggregator.

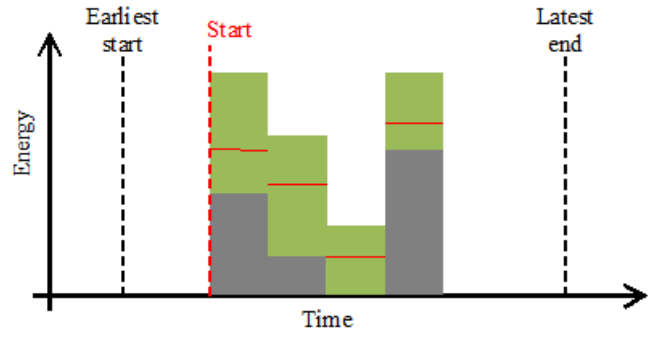


Fig. 1. Example of a FlexOffer

A FlexOffer is intended to be used on a Virtual Market of Energy. The actors on the market are energy sellers and buyers, and flexibility sellers and buyers.

The energy buyers are private homes or companies, which have devices that can be controlled in order to utilize their flexibility. This way, the energy buyers act as flexibility sellers, which generate FlexOffers based on the profiles and possible flexibility in their flexible resources. The FlexOffer is also associated to a default schedule that will be followed in case the FlexOffer is not sold. Along with the FlexOffer, the flexibility seller sends pricing information for the cost of deviating from its default schedule. This price can be calculated from local energy prices or loss in efficiency.

The energy sellers are energy producers that act as buyers with regards to flexibility, in order to shift energy consumption away from a possible grid overload. The pricing information is used to evaluate how much flexibility it is worth to buy. A new schedule is assigned based on the sold flexibility and the buyer compensates the seller based on the price. If a schedule is sold but is not followed, then the flexibility seller is penalized.

Since a single home or small company does not consume much energy compared to the capacity of the grid, the amount of flexibility offered is relatively small and therefore not very interesting to the buyers. For this reason, Aggregators are put in between the sellers and the market. The Aggregator receives several FlexOffers from different sellers and aggregates them into larger FlexOffers, which are of interest to the buyers.

The FlexOffer concept was originally introduced in the MIRABEL project [5]. It has been further developed in the Arrowhead and TotalFlex projects [8]. Some research has been done to quantify the benefits of flexible resources on the energy grid [9].

B. The Arrowhead Framework

The FlexHousing system was developed on top of the Arrowhead framework [6], which facilitates the development of service oriented distributed CPS applications.

The Arrowhead approach simplifies design and implementation of distributed application by means of normalizing communication via services. Every communication of the distributed application is mediated through services, exposed

and consumed by systems. A system can both provide and consume application services, and thus implement the functional requirements of a specific use case, or provide core services, which are diagonal to the use cases and provide support to non-functional requirements such as service registry, service discovery, Quality of Service and security. The set of systems that provide core services are delivered in the form of the Arrowhead Framework.

The systems of an Arrowhead distributed application are organized into a System of Systems (SoS), which is deployed into the form of a local cloud, which is a bounded set of computational resources used by stakeholders to attain a goal. Supported by the Arrowhead Framework, the devices are able to set up protected communication that enable the execution of critical applications. Among them, there is the Virtual Market of Energy [8], which implements a service-oriented interface to trade energy and flexibility on one or multiple energy markets.

The rationale is that the systems of the SoS register themselves, together with the list of services they produce or want to consume. The Orchestrator system collects all the data regarding the SoS and matches systems and services to satisfy both the functional (which services are consumed) and non-functional (QoS, security, geographical localization of the system producing the consumed service) requirements.

The extension of the Arrowhead Framework to non-Arrowhead compliant components is done by means of adapters, which can act at different levels:

- Communication paradigm: since Arrowhead is service-oriented, an adapter can be used to interact with components that for example exchange message through publish/subscribe systems such as XMPP [10];
- Ontologies: Arrowhead considers devices, systems and services are the actor in communication scenarios. Adapters and stubs should be used to hide a group of systems behind a unique Arrowhead system, or to relate a service to consider a system as built by a number of Arrowhead virtual systems, each able to provide services;
- Semantics: some framework associate timing characteristics to the communications, which assume semantics relevance. Since Arrowhead does not use this technique, an adapter can be necessary for mediating interactions;
- Syntax: the format of the messages can be different, thus a translation operation can be needed.

As described in Section IV, the SoS presented in this paper interacts with custom smart plugs and sensors through a service-oriented interface, and the adapter has to perform just a syntax translation between the protocols, easing up the adaption for a large measure.

C. Related Work

The field of energy management in buildings is very active, and many competing solutions have been devised. In fact, it is sufficient to look at the many surveys published on the topic [2], [11]–[13] to acknowledge the research efforts that aim at innovating in the application area.

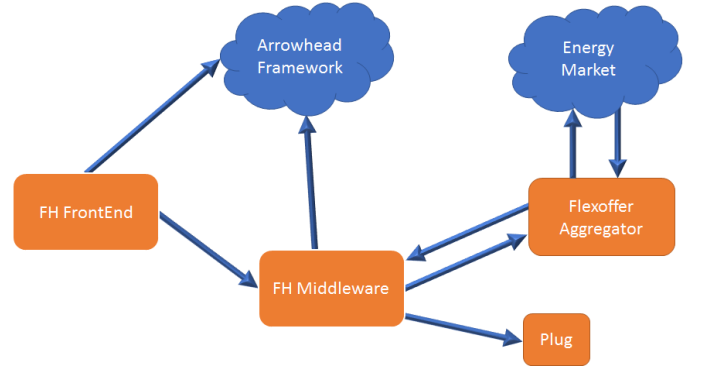


Fig. 2. Architecture of the FlexHousing System of Systems

This paper provides novelty regarding the tools used to improve the state-of-the-art: the FO concept, and SoS based on the Arrowhead Framework.

III. SYSTEM ARCHITECTURE

The system architecture that was devised as support to the FlexHousing functions is a SoS (see Subsection II-B) and it is based on a number of systems, as depicted in Figure 2. In the Arrowhead sense, three of them, FlexHousing Middleware (FHMW), FlexHousing Frontend (FHFE) and FlexOffer Aggregator, are application systems, since they implement the use case. All the systems interact at some point with the Arrowhead Framework, for example to perform service discovery to find the other systems. The FlexOffer Aggregator interacts with other Aggregators, and finally with the Virtual Market of Energy. Moreover, the FHMW interacts with the smart plug housing the sensors and actuators of the house to manage energy consumption.

A. FlexHousing Middleware (FHMW)

The FHMW is responsible for the integration of the systems involved in the FlexHousing environment, and to facilitate the creation of FlexOffers. The FHMW takes care of automating the emission of FOs and the actuation on the appliances depending on the schedule that was retrieved from the Aggregator. The flow of energy will be enabled depending on the Schedule that is active on a plug.

In particular, as far as FOs are concerned, the FHMW allows the FHFE to create FOs, takes care of providing them to the Aggregator, receives energy schedules, drives energy consumption based on the schedules, and takes care of collecting energy data to be used to verify schedules execution and facilitate the creation of future FOs.

We consider that the FHMW is capable of retrieving the data resulting from the monitoring of energy of the appliances connected to the plugs, as well as online information regarding the actuation on non-Arrowhead compliant appliances that have a plug attached to them. The retrieved data is used for the recognition of usage patterns of the appliance during a certain period. Using clustering mechanisms, such as an unsupervised neural network, certain patterns can be discovered. The net

result is a catalog of scenarios. Each scenario represent the energy behavior of a user depending for example of the weekday (working days vs holidays). Those patterns can then be used to facilitate the creation of FlexOffers, by providing default patterns at FlexOffer creation time, to allow the user to create a FlexOffer corresponding to his normal energy usage in a given scenario, without the effort of defining manually energy consumption profiles.

The timeline of the provisioning of FOs to the Aggregator is studied to maximize the usage of day-ahead energy markets. In fact, the FHMW uses an heuristics to evaluate how much time it needs to send all configured FOs, and starts the process a number of minutes before midnight to be sure to complete it before the end of the day.

B. FlexHousing FrontEnd (FHFE)

The web-based front-end FHFE provides a means to drive the capabilities of the FlexHousing environment, and of the FHMW in particular. This interface supports multiple users (and their respective roles, i.e. home owner, energy manager), allowing them to verify a room's or device's current energy consumption, turn a device on/off, and create and send FOs for an appliance in the simplest way possible.

When it comes to creating a FO, this process is divided into three steps: introducing basic details, choosing what kind of pattern to use, and creating the energy pattern for the FO. The first step requires the user to input the FlexOffer's name and the time period in which it must be applied. The second step requests the user to define the energy consumption pattern, which can be done either manually through the graphical interface, or based on one of the default patterns that are created based on the user profile computed by the FHMW. By choosing to create it manually, in the third step the user can select the duration of the pattern and define the energy consumption for each 15 minute interval, by dragging the bars in the chart with the mouse. When using a default consumption pattern, the user can modify a pattern that is predefined based on the device's consumption data.

The FHFE allows for three roles: the normal user who can visualize his energy consumption, the power user who can configure his own energy usage by setting up FlexOffers, and the energy manager who is entitled with the management of multiple houses in a complex. This latter role is used to relieve the homeowner from the work-load of configuring his energy usage by means of FOs. Using the same options and features the energy manager sets up and customizes the FOs in the same way as the power user, after deciding which of the FHMW, such as apartment complexes, industrial facilities, or condominium, he is actually configuring by means of the same FHFE.

The authentication and authorization module for the manager is handled by the Arrowhead Framework, which also provides the addresses of the FHMWs that a specific FHFE can handle and control.

C. FlexOffer Aggregator

The Aggregators receives FOs from FHMWs, combines them with FOs from other sources into larger FOs and then either sends them to other Aggregators, or to the Virtual Market of Energy. Note that FOs need a significant magnitude to be interesting on the Energy Market.

Afterwards, the aggregator receives a response from the Virtual Market of Energy, which can be a refusal, or an energy schedule. In the latter case, the Aggregator disaggregates the response and sends downwards the consumption schedule. Many types of Aggregators may exist, some may be specific for a use case, such as the management of electrical motors, whereas others may be more generic. In addition, selecting the most adequate aggregator additionally depends on the geographic region.

D. Arrowhead Framework

The Arrowhead Framework provides services to support other systems with respect to security, system and service registration, and service discovery and orchestration (see Subsection II-B). The local cloud providing FlexHousing services requires a number of basic core services that enable fundamental SOA properties like service registration, service discovery, authentication and authorization plus orchestration of SoS. The ServiceRegistry system allows a system to expose a service it is producing to the cloud, and allows consumer systems to discover services they wish to consume. The Authorization system is responsible to controlling which service can consume a certain producer. The Orchestration system allows coordination (orchestration) of which producer will a certain consumer be able to employ/use.

E. Energy Market

The Aggregators are able to schedule energy consumption by allocating it through a virtual market of energy, which communicates with appliances through FOs. The Energy Market allow to trade both energy and flexibility. (see Subsection II-A).

The Energy Market secures the balance in a logical sub-domain within the grid, i.e. ensure that consumption is equal to production. It utilizes the aggregated FOs from Aggregators for an internal energy balancing, and places FOs on the flexibility market for trading with energy producers, which end up being the flexibility consumers. The Energy Market minimizes total costs by scheduling energy loads while respecting the constraints contained in the FOs (minimum/maximum power, earliest/latest start of energy consumption, etc.). In the case of the energy consumer, the net result of the interaction of the Aggregator with the Energy Market is to buy energy while selling the consumer's flexibility, attaining the goal of saving the consumer's money and supporting the energy producer in flattening off consumption peaks.

F. Actuators and Sensors in the User's Home

Common household appliances cannot be controlled remotely, and thus aren't fitted to FO compliance. One of the

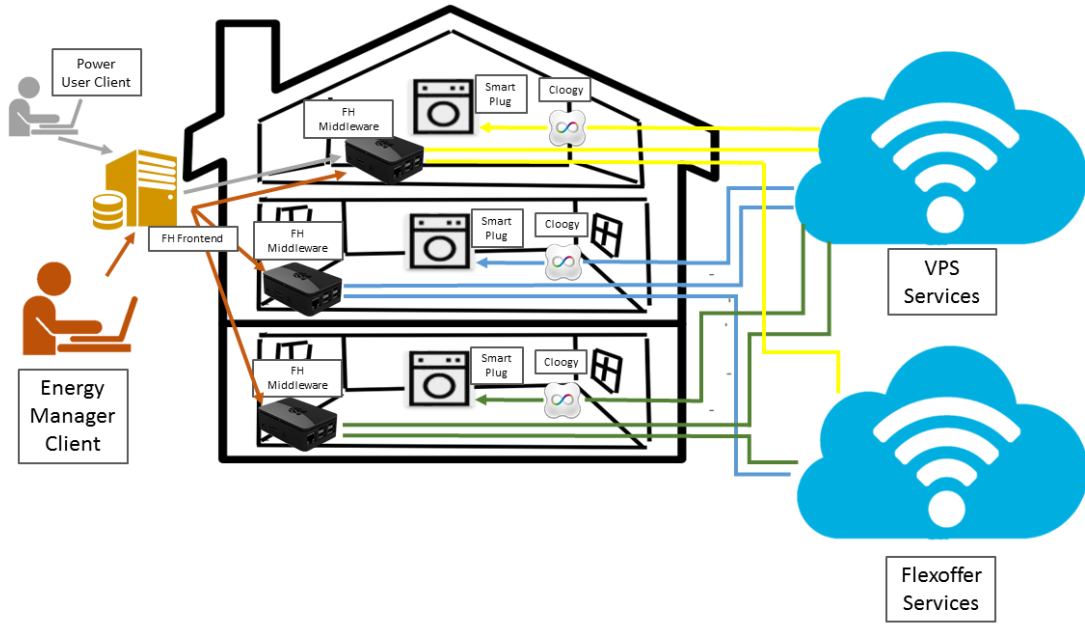


Fig. 3. Implementation of the FlexHousing SoS

solutions to tackle this issue is to attach to the appliances an IoT device called smart plug. The smart plug is placed between the appliance and the electrical outlet, and from there on the plug is able to control the flow of electricity towards the appliance. Sensors are also installed in the plug, capable of collecting data regarding the energy usage and other metrics.

The FlexHousing SoS considers that devices are either smart plugs equipped with both sensors and actuators, or more complex devices that provide also smart plug functionalities. The actuator of the smart plug allows to remotely switch on and off the appliance it is installed onto. Moreover, through its sensor, the smart plug can collect data regarding energy consumption, to inform the user regarding energy consumption, regarding the correct execution of FlexOffers, and to allow for the creation of FlexOffers based on past consumption. The FlexHousing SoS considers that an API is provided, capable of receiving requests and forwarding them back to the plug. The API can be exposed on the plug, or on an external service platform.

In the most common topology, a central hub called gateway is installed in the customer's premises to provide the connectivity required to interact with the smart plugs from the local cloud. The requests are sent to an external service provider, and then received by the gateway that, in return, sends the commands to each specific plug.

IV. THE IMPLEMENTATION

The FlexHousing SoS applies FO concepts to the real-life management of appliances based on FOs. The SoS allows a user to create FOs upon devices, and lets the power usage of the device be dictated by the aforementioned FOs. The house of the user is modeled as a set of devices, organized

into rooms (kitchen, living room, garage ...) that pertain to buildings/houses. Each device is equipped with sensors and actuators to measure and control energy consumption of the appliances, respectively.

The FlexHousing SoS comprises the Arrowhead Framework, which provides core services such as service registry, orchestration, security, the Virtual Market of Energy application service, and two novel systems, the FHMW, responsible for the FOs and devices management, and the FHFE, which is a graphical interface for the setup of user configurations and all-around managing of the building/house. The FHFE is hosted by a web server and interacts with the FHMW through the services the latter provides.

A context diagram of the SoS of the pilot is depicted in Figure 3. All the systems of the SoS are considered to interact with the Arrowhead Framework, and thus these interactions are not represented. The users access the FHFE, which interacts with the FHMW only, since the latter contains the business logic and the information for the management of the FOs and of the smart plug. The FHMW interacts with the Aggregator's services, which are Arrowhead application services. The FHMW also communicates with the VPS API, which is not Arrowhead-compliant. The VPS API is exposed by an external service provider and communicates with the Cloogy gateway (the device with the infinity symbol in Figure 3) to interact with the smart plugs attached to the appliance. Being already a service-oriented interface, the VPS API's adaptation onto the Arrowhead Framework was as easy as the provisioning of a one-to-one mapping to the Arrowhead and the VPS API protocols.

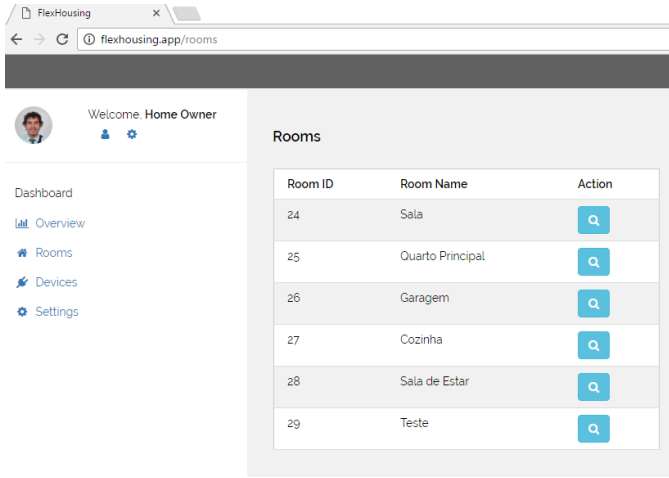


Fig. 4. Overview page of the FHFE.

In the following, the Arrowhead Framework is not described, but details can be found for example in [6].

A. Implementation of the FHMW

The FHMW is built around the components required for communication. As such, it employs 3 different solutions, one for each communication path available.

For the interaction with the Aggregator, a DER agent was implemented [8], which is responsible for the emission of FOs using the XMPP protocols [10] used by the Aggregator, and the retrieval of Schedules through the same mechanism. The application is coded in Java, using the Maven dependency Maven to handle dependency issues from other software components. Derby Apache was selected to host the data due to its easy integration into a Java application. For the services exposure, a mix of Grizzly - for the HttpClient - and Jersey - for the service resources - was implemented.

There is no direct contact between the FHMW and the user, and the MW exposes all the services through an API that is used by the FHFE. Those services are mostly CRUD actions around the rooms, devices, FlexOffers and schedules.

For Aggregator purposes, the FHMW implements all the methods that allow the registration of the middleware on the Aggregator, the emissions of the FOs and the retrieval of the Schedules.

For the interaction with the VPS services, a HTTP client was implemented, which executes all the request involving the query or interaction with the smart plugs and sensors.

Even though most of the data is stored in the database, the FHMW keeps the most used/requested objects in cache to avoid excessive database queries. Any modification on the objects is reflected on the copy in cache but also in the database. This allows for the persistence of the data in the case of a power outage.

The FHMW system and the FHFE system are executed on Windows environments. In the same local network, there are the deployments of the Cloogy gateway (the gateway to VPS) and the VPS smart plug.

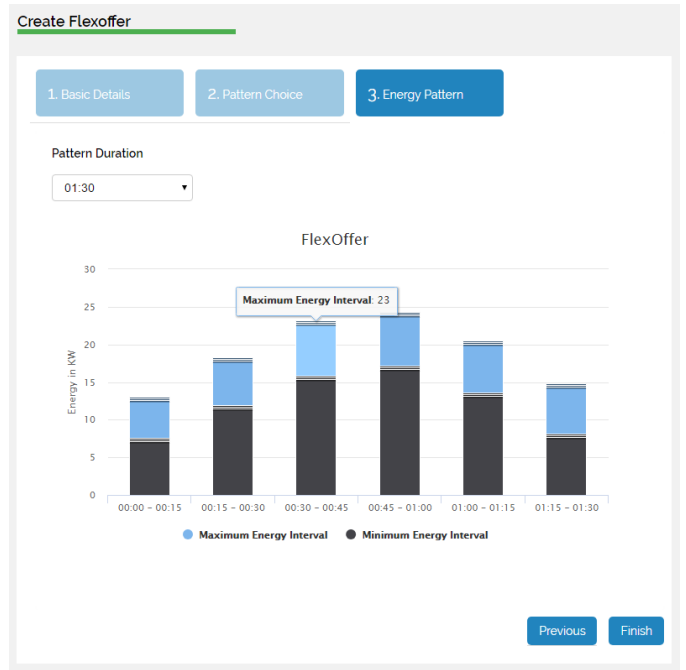


Fig. 5. Creation of a FlexOffer in FHFE: manually creating a consumption pattern.

B. Implementation of the FHFE

The web-based front-end FHFE provides a responsive, cross-browser compatible, graphical user interface to demonstrate the capabilities of the middleware. The FHFE is an MVC application, where its back-end is built in PHP, using Laravel 5.4 as its framework, while its front-end is built with HTML5, CSS3, and JavaScript, using the Bootstrap framework.

The FHFE supports multiple users (and their respective roles, i.e. home owner, energy manager), allowing them to check a rooms or devices current energy consumption, turn a device on/off, and create and send FlexOffers of a device in the simplest way possible. Therefore, the FHFE is composed of several sections: the overview page, the rooms section, and the devices section.

The overview page provides insights on the total used energy, the number of FOs set up for each device, and other global statistics.

The rooms section allows to decide on which room (see Figure 4), and later on on which device, to work on to define involved FOs.

Figure 5 shows the process of creating a FO manually, (see Subsection III-B). In the third step, the user can select the duration of the pattern (whilst respecting the time period defined in the previous step) and define the energy consumption for each 15-minute interval, by dragging the bars in the chart with the mouse. A similar interface is used to tune up a FO created using a pattern based on the device's past consumption data.

In the case of the Energy Manager role, the FHFE allows the same simplicity for the setup as the homeowner but without the constraints of having to constantly having to swap credentials.

This is achieved by enabling a dropdown list at the top of each view. The list contains the name/identification of all the houses the manager can operate on, acquired through the Arrowhead Framework. Swapping house, the FHFE refreshes all the data concerning the objects of the FHMW system (Rooms, plugs, FlexOffers,), updating them to reflect the ones of the current house.

C. Aggregators deployment

The setup at Aalborg University (AAU) in Aalborg, Denmark contains the communication infrastructure, the Aggregator, and the Energy Market implementation. All services are exposed through the Arrowhead framework.

The communication infrastructure consists of an XMPP server [10] to facilitate the HTTP-over-XMPP communication between the actors in the system. The Aggregator receives small FlexOffers from multiple consumers and aggregates them into (possibly multiple) larger FlexOffers. FlexOffers are aggregated using different algorithms e.g. [14] [15]. The aggregated FOs are sent to the Energy Market as selling bids, expressing flexibility being sold on the market.

The Energy Market receives selling bids from Aggregators or directly from energy consumers. It also receives buying bids from buyers, which are energy producers. At set intervals, the market will be cleared. The interval could for instance be 15 minutes for an intra-day market, or daily for a day-ahead market. The clearing algorithm will match buyers and sellers such that highest buyers and lowest sellers will be favored.

Once the Energy Market is cleared, the winning bids are sent back to their owners. The Aggregator will distribute the sold flexibility among the flexible energy consumers and send schedules back.

D. VPS Services

An external API is provided by the manufacturer of the smart plugs and sensors - and gateways - used in the energy consumer's Home Area Network. The API is service-oriented, and adheres to the RESTful principles of the HTTP protocol, thus providing a machine friendly, robust and predictable interface to the system functionalities.

The FHMW has a module to account for the location of each sensor and smart plug, and it uses the information to send messages to the correct component. Communication with the VPS Services is performed using a TCP/IP connection that hosts a HTTPS session, which exchanges data encoded using the JSON data format.

Each controlled device can provide several parameters (current, power, etc.) and send the data that has been read to the VPS Services. The VPS Services store data, which can be queried - and cached - by the FHMW, either when verifying that a consumption Schedule is being respected, or when building the profiles used as default FOs when the FOs are set up.

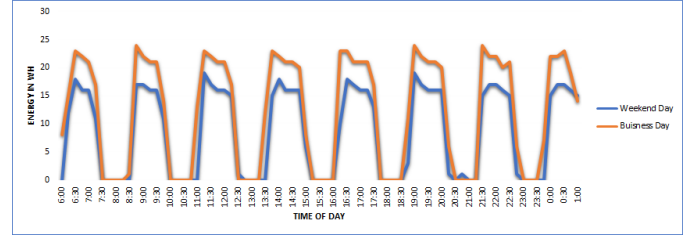


Fig. 6. Energy consumption pattern for a refrigerator

V. TEST RESULTS

Some preliminary tests were executed on the implemented platform, to verify its correctness, and to ensure its performance characteristics.

A. Consumption patterns

The plugs used in the FlexHousing SoS are equipped with sensors, which allow to retrieve the consumption of any particular plug, independently if a FO was applied to it or not. In fact, as soon as a plug is registered in the SoS, it starts collecting data. Data is kept both on the cloud of VPS services, accessible through measurement queries, and cached onto the FHMW after queried.

Figure 6 reports the result of a proof-of-concept data collection performed on a refrigerator. This test verified that the FlexHousing SoS is able to collect data with a reasonable granularity, to use them to build an energy consumption profile. The values were collected during a business day and during a weekend day, allowing to verify that human actions affect the energy consumption pattern for the appliance. In fact, during a business day the refrigerator is used on a regular basis, while during the weekend, especially on Sunday, the refrigerator is kept close, thus requiring less energy to maintain its cold temperature.

The energy usage matches with the theoretical consumption of the refrigerating cycle (Vapor-compression cycle) [16]. The periodicity is the same: between each peak, there's a period of no consumption, matching the idle state of the refrigerator. Aside from small statistical fluctuation in the data, the only difference between the days is the local maxima of each individual peak. While the weekend day averaged around 14.5Wh, the business day averaged at 18.5Wh.

B. Communications tests

This test targeted the latency between the systems during communication. Since the latency inside the Arrowhead Framework was already targeted by analysis [6], this paper studied the latency for an actuation over a smart plug.

This test executed 30 request of actuation over a single plug. Two batches were performed, one by requesting the VPS API directly for the operation, and the other executed through the FHMW. The results of the tests are depicted in Figure 7.

The blue series represents the data collected when the request is directed to the VPS Services platform, and the orange series is related to mediating the interaction through the

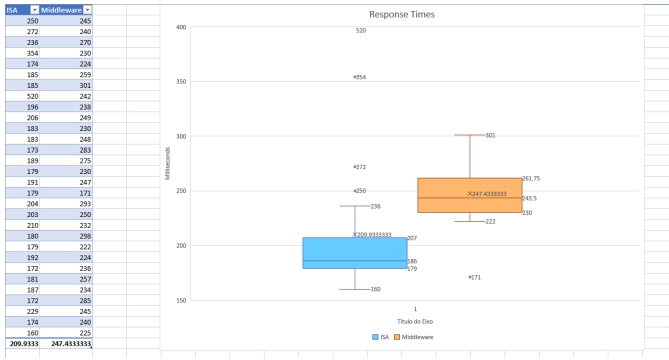


Fig. 7. Latency for actuation on a smart plug.

FHMW. The delay is less than $210ms$ and $270ms$ respectively, and it can be concluded that the FlexHousing internals do not impair the performance of the SoS.

VI. CONCLUSIONS

This paper presented the FlexHousing System of Systems, which is a platform for energy management in buildings. It is based on the Arrowhead Framework and on the FlexOffer concept, and it allow either the fine-grained management of energy by a power user that is knowledgeable regarding Energy Markets, or by the professional services of an energy manager.

The paper provided insights regarding how the systems are implemented, and some results regarding experimental tests.

In the future, the FlexHousing System of Systems will be extended to different smart plugs that obey to different interaction patterns, and data regarding the energy saved in real-world deployments will be collected.

ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234); also by FCT/MEC and the EU ECSEL JU under the H2020 Framework Programme, within project ECSEL/0004/2014, JU grant nr. 662189 (MAN-TIS) also by FCT/MEC and the EU Artemis JU within project ARTEMIS/0001/2012 - JU grant nr. 332987 (ARROWHEAD)

REFERENCES

- [1] EIA, *Annual Energy Review 2015*, <http://www.eia.gov/totalenergy/data/annual/>
- [2] Tuan Anh Nguyen, and Marco Aiello, *Energy intelligent buildings based on user activity: A survey*. Energy and buildings 56 (2013): 244-257.
- [3] Thibaut Le Guilly, et al. *ENCOURAGEing results on ICT for energy efficient buildings*. IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016.
- [4] Michele Albano, Luis Lino Ferreira, and Luis Miguel Pinho, *Convergence of Smart Grid ICT architectures for the last mile*. IEEE Transactions on Industrial Informatics 11.1: 187-197. 2015.
- [5] M. Boehm, et al. *Data management in the MIRABEL smart grid system*. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops. EDBT-ICDT'12, ACM, 95-102. 2012.

- [6] Delsing, Jerker, et al., *The Arrowhead Framework architecture*, chapter 3 of *IoT Automation: Arrowhead Framework*. CRC Press, 2017.
- [7] L. L. Ferreira, M. Albano, and J. Delsing, *QoS-as-a-Service in the Local Cloud*, IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016.
- [8] Luis Lino Ferreira, et al. *Arrowhead compliant virtual market of energy*, Emerging Technology and Factory Automation (ETFA), IEEE, 2014.
- [9] Bijay Neupane, Torben Bach Pedersen, and Bo Thieson. *Evaluating the value of flexibility in energy regulation markets*. Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems. ACM, 2015.
- [10] P. Saint-Andre, K. Smith, and R. Troncon, *XMPP: The Definitive Guide*, O'Reilly, 2009
- [11] Pervez Hameed Shaikh, et al. *A review on optimized control systems for building energy and comfort management of smart sustainable buildings*. Renewable and Sustainable Energy Reviews 34: 409-429. 2014
- [12] Alessandra De Paola, et al. *Intelligent management systems for energy efficiency in buildings: A survey*. ACM Computing Surveys (CSUR) 47.1: 13. 2014
- [13] Douglas Harris. *A guide to energy management in buildings*. Routledge, 2016.
- [14] L. Siksny, et al. *Aggregating and disaggregating flexibility objects*. IEEE Transactions on Knowledge and Data Engineering 27.11: 2893-2906. 2015.
- [15] E. Valsomatzi, et al. *Towards constraint-based aggregation of energy flexibilities*. Proceedings of the Seventh International Conference on Future Energy Systems Poster Sessions. ACM, 2016.
- [16] Piotr Domanski, and David Didion. *Computer modeling of the vapor compression cycle with constant flow area expansion device*. Final Report National Bureau of Standards, Washington, DC. National Engineering Lab. 1983.

11.12Appendix-O- Maintenance Supported by Cyber-Physical Systems and Cloud Technology

Maintenance Supported by Cyber-Physical Systems and Cloud Technology

Erkki Jantunen, Jarno Junnola
VTT Technical Research Centre
of Finland Ltd
Espoo, Finland

Unai Gorostegui,
University of Mondragon,
Mondragon, Spain

Michele Albano, Luis Lino
Ferreira, José Silva
CISTER, ISEP/INESC-TEC
Polytechnic Institute of Porto,
Portugal

Abstract—The paper discusses about the application of Cyber-Physical Systems (CPS) and cloud technology to maintenance. In fact, the heavy utilization of large quantity of data collected on machines, and their processing by means of advanced techniques such as machine learning in the cloud, enable novel techniques such as Condition-Based Maintenance (CBM). Fairly new sensor solutions that could be used in maintenance and in interaction with CPS are also presented. Data models are an important part of these techniques because of the huge amount of data that are produced and that must be processed. The Machinery Information Management Open System Alliance (MIMOSA) Open System Architecture for Condition-Based Maintenance (OSA-CBM) standard architecture supports streamlining the modeling of collected data and transferring information, and in this sense OSA-CBM is discussed in this paper. Finally, current and future directions for application of CPS and cloud technologies to maintenance are discussed.

Keywords—Cyber-Physical Systems; CPS; Cloud; MEM; Piezofilm; MIMOSA; OSA-CBM

I. Introduction

The increase of complexity and cost of current industrial equipment is becoming an important factor that is helping to change the maintenance from a corrective to a preventive maintenance strategy and more specifically to the CBM. This change is given more emphasis also by the increase in the requirements and the availability, performance and quality while trying to reduce the cost of the production equipment during its whole life cycle. A CBM system is based on monitoring the different parameters of an asset to compare to previously gathered data through various mathematical algorithms to do a diagnosis on the health level of the equipment and predict how it will behave in the future. This way, the downtimes of the machines can be programmed so that they affect in the least the production, decreasing the unpredicted production standstills and increasing the availability and consequently the Overall Equipment Effectiveness (OEE). The ultimate goal of a CBM system is to link it with the Computerized Maintenance Management System (CMMS), which is a software package that maintains a computer database of information about an organization's

maintenance operations, which on its part is used to schedule maintenance operations in a factory, and supports decisions regarding maintenance (e.g.: comparing cost of early preventive maintenance vs machine breakdown) in the broadest sense. Thus, CBM takes part in the maintenance process by creating action reports for the technicians or shutting down a machine or reducing the speed if the system predicts it necessary.

The introduction of new technologies such as Micro Electro Mechanical Systems (MEMS) sensors and the constant price drops of these have enabled maintenance strategies such as CBM to become more and more popular and achievable during the last few years. Another aspect to take into account is the Internet of Things (IoT) that has allowed the communication between machines and the components that take part in an industrial plant. The IoT, while connected to the cloud technology, permits the users access the data from the CPS from anywhere and a wide range of devices. The focus of this paper is to reinforce the idea that a maintenance system supported by CPS and Cloud Technology can be beneficial in the increasing of the OEE and reducing the maintenance costs.

II. Cyber-Physical Systems

The main objective of a CBM system is not only to monitor an asset but also to take direct action in its maintenance. If this requirement is fulfilled, the whole system responds to the definition of a Cyber-Physical System (CPS), since it integrates the monitoring of the equipment by computer-based algorithms with the internet and its users. It is hard to see the difference between CPS, machine to machine (M2M) and Wireless Sensors Networks (WSN) under the architecture of Internet of Things (IoT) but CPS is seen as an evolution of M2M systems [10]. Another term that can be associated to the ones mentioned above is e-maintenance [11]. There exists a number of definition for this term see e.g. Karim's doctoral thesis [12]. However, the simplest one is to define e-maintenance as a technology that supports maintenance by means of information collected by IoT. Consequently, e-maintenance can be seen as a sub-category of IoT concentrated in to support maintenance activities, and in

particular CBM. Furthermore, CPS would fit into this framework as the technology where the hardware and software form an intelligent solution that can perform task that are needed to carry out efficient CBM under the e-maintenance umbrella.

According to the U.S. National Science Foundation, “the term cyber-physical systems refers to the tight conjoining of and coordination between computational and physical resources”. The CPS in this case would consist of various components. A transducer or sensor that would measure a meaningful parameter of the asset to be monitored, linked to a data acquisition equipment, where, using mathematical algorithms the gathered signals are processed to obtain important data. Once the data are acquired, a diagnosis is made to evaluate the health level of the asset at that moment.

The aforementioned diagnosis is based both on mathematical models of the real systems and also the previously gathered data that are stored in a database. The system should then be able to predict the future health states and the possible failure modes based on the current health level as well as historical data. It is worth mentioning that some companies are already offering solutions as a software platform that implement the data analysis and pattern recognition algorithms to help the integration of a CBM strategy with their system [16][17]. Typically, these commercial solutions are able to carry out diagnosis of the condition of a certain set of components and fault types associated with them i.e. there is no prognosis element that would predict the future development of these faults.

With the diagnostics and prognostics information, the system could take pertinent actions to optimize the life of the asset. These actions could vary depending on the outcome of the system, from shutting down the engine, to creating an action report with the instructions to change a component and send it to the maintenance technician that is available (closest) at that moment. The following will give an example of CPS:

“A filter was starting to get clogged causing an increase in the power consumption of the motor and an increase on the temperature. If the motor were to keep working at this level, soon the temperature would get to a point that was not permitted or could cause a breakdown or a risky situation. The system realized this and sent the maintenance technician a report with medium priority stating that the filter needed to be changed, gave definition of the location of the machine, reserved the tools needed for the repair and the spare parts that were going to be needed. It also scheduled maintenance for that equipment outside its working hours, so that the downtimes were reduced.”

Today, it is possible to build a system that fully automatically carries out the above described tasks. In the Dynamic Decisions in Maintenance project which is documented in the E-Maintenance book [11] all the necessary elements were covered. Naturally, since that project was completed the development of low cost sensors and low cost processors has been very rapid. Even though technically possible and today at a lower cost the above type of fully automatic solutions are not often used for other than very basic fault types for very cheap spare parts. In case of more

sophisticated machines humans are always linked to the process so they can check that everything is going as it should. Also, the total number of solutions where the expected lifetime of components is predicted with reasonable accuracy is still rather limited.

III. Sensors for CBM

MEMS sensors are becoming more and more popular. With different MEMS sensors it is possible to measure a lot of different phenomena from the physical environment, including temperature, acceleration, pressure, inertial forces, chemical species, magnetic fields, radiation, etc. [1]. MEMS sensors can be found from multiple different devices including handheld devices like mobile phones [2]. The name Micro Electro Mechanical System already reveals that MEMS are components that combine microelectronics and micromechanical parts together to a same packet. MEMS sensors are manufactured using the same methods as with Integrated Circuits (IC) and thus the price of individual sensor is relatively low [3] and its size can be extremely reduced.

According to MEMS manufacturers the price difference between, for example, accelerometers can be really significant for MEMS accelerometers in relation with piezoelectric accelerometers (which are more commonly used in condition monitoring). The price of MEMS accelerometers is around tens of dollars and the price of piezoelectric accelerometers is measured in hundreds or even thousands of dollars [4]. MEMS sensors are light in weight, tiny, usually highly integrated devices, have low power consumption and works with low voltage [1]. The tininess of MEMS sensors is a huge advantage when sensors need to be integrated to small devices or measurements have to be made in tiny locations. In figure 1 we show, from left to the right, a MEMS accelerometer ADXL001 from Analog Devices attached to a printed circuit board, a tiny MEMS accelerometer KX122-1037 from Kionix and a piezofilm accelerometer ACH01 from TE Connectivity.

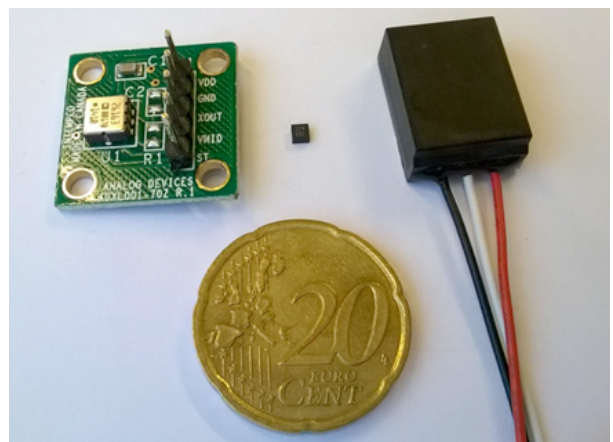


Fig. 1. Accelerometers from left: ADXL001, KX122-1037 and ACH01.

However, it should be noted that when choosing a sensor great care must be taken that the sensor in question fulfils its tasks i.e. that is suitable for the monitoring purpose in question

so that it is technically good enough to detect the early indications of faults, which quite often are hidden under other influencing factors.

Vibration monitoring is very often used in CBM or predictive maintenance and it is especially suitable for monitoring the wear of the components of rotating machinery. This is due to the fact vibration measurements basically offer the opportunity to monitor both the amplitude and the frequency of the dynamic signal thus enabling the opportunity to distinguish between various fault types especially by looking at the frequency they influencing. Examples of such fault types are unbalance (seen at rotational speed), misalignment (seen at the harmonics of the rotational speed), bearing faults (seen at the so-called bearing frequencies), and gear faults (seen the gear tooth frequencies), etc.

MEMS accelerometer seems to be an obvious choice instead of the more common piezoelectric accelerometer because of the price, but MEMS accelerometers has it downsides also: usually MEMS accelerometers suffers from lower bandwidth, lower resonance frequency, poorer “off-the-shelf” protection against harsh and difficult measuring environment and higher noise, when compared against piezoelectric accelerometers. Depending on the demands of measurements, MEMS accelerometers might be more than good enough to replace piezoelectric accelerometers in some vibration measurement events. However, the opposite might also be true i.e. in case a very typical use the monitoring of bearing faults the demands for the sensor are very high because the impact a bearing fault causes in the beginning is very small and thus very difficult to detect [6]. At the same time the demand for early detection might be very demanding e.g. in case of offshore wind turbines the goal is to detect bearing fault a year ahead they would stop the turbine.

Another option for piezoelectric accelerometer in vibration monitoring is the piezofilm accelerometer. Piezofilm accelerometers are usually made of polyvinylidene fluoride (PVDF / PVF2) which is shaped in thin layers and those layers are coated with metal electrodes and plastic which protects the accelerometer [7]. Piezofilms are light, bendable, flexible, deformable, mechanically durable and easy to form for specific measuring location [8]. In figure 1 is shown ACH01 piezofilm accelerometer which is in a cover. Piezofilm accelerometers are valued in few dollars according to a big MEMS manufacturer Analog Devices and so they are even cheaper than MEMS accelerometers [4]. According to some piezofilm accelerometer data sheets it is possible to find quite promising accelerometer options for piezoelectric accelerometers which have high resonance frequencies (up to 35 kHz), wide bandwidth (for example 2 Hz - 20 kHz) and a low noise level that approaches the noise level of conventional piezoelectric accelerometer [9]. Even though these values look very promising, the experience with piezofilm sensor is still limited when compared to the long history with piezoelectric sensors.

As indicated earlier in this paper care should be taken when choosing a sensor. This means that it is important to understand the function of the sensor and what kind of wear phenomena it is expected to pick up. When a clear

understanding of the above exists theoretically and on paper the new sensor types will explode the use of sensors for condition monitoring purposes and this with the concept of CPS.

IV. Cloud Technology

A. Cloud Service Providers

There are many different cloud service providers and among them are the big players Microsoft Azure, Google Cloud and Oracle. Cloud services are internet-based services that provide all the things that computer systems can offer. With cloud services, it is possible to get an access to high performant computing resources without the need of expertise of managing the computer system yourself. Cloud services can be accessed with any device that has an internet connection which is a big advantage. Cloud services typically offer protection against malicious programs which releases companies' resources to other duties.

B. Local Clouds

The concept of local clouds is focused to industry, while cloud is targeted also to the general public, and thus has not gained much publicity. A local cloud is an intranet solution that provides computational services internally to the company itself, and its business partners. In such a system, only the computers that are inside the local cloud have full access to the services, which in principle are similar to the large commercial cloud systems mentioned earlier. In some cases, the local clouds can be accessed also by outsiders, but when allowing this the IT departments of companies tend to have very strict rules and limitations.

From IoT, e-maintenance, and CBM point of this access to data within local clouds is an important question. For example, a machine tool manufacturer would like to provide maintenance services for the machine tools they have sold. In order to carry out maintenance in an efficient way they would like to use CBM strategy and thus follow the machine tools with CPS which in turn means that they need access to the data from the machine tools which as such can be considered as local clouds within the local clouds the end user has at their plant.

A number of collateral issues are related to local cloud, such as the ownership of the data and security, and solutions to these issues are not yet mature for many environments. A great deal of discussions on these issues can be for example found in [13], which presents the results of a recent European project called Arrowhead in terms of theoretical findings and developed solutions, as well as open software that can be used as basis in order to provide a solution to some of the above-named challenges.

C. Communication Protocols

Usually, cloud service providers provide a web-based interface, which provides services that strengthens the security between both the cloud and the web services. Web-based services make it possible to share the data from cloud servers to graphical user interfaces (GUI) for the end-user applications

through the internet. Usually GUIs are used through web browsers. Web-based services integrate Web applications through the internet and they are created using server-side scripts (ASP, PHP, etc.) and client-side scripts (JavaScript, HTML, Flash, etc.). In figure 2 is a diagram of web-based service which shows the basic structure of web-based service.

There are Web services that are made more for industries like Open Platform Communications Unified Architecture (OPC UA). OPC UA is an industrial machine to machine (M2M) communication protocol developed by OPC foundation and as a web service it consists of an OPC client that interacts with an OPC server. OPC UA increases interoperability and is designed to be able to have real-time data access, historical data for analytics and reporting of data or events, and alarms, and conditions to notify when the alarm-trigger goes off.

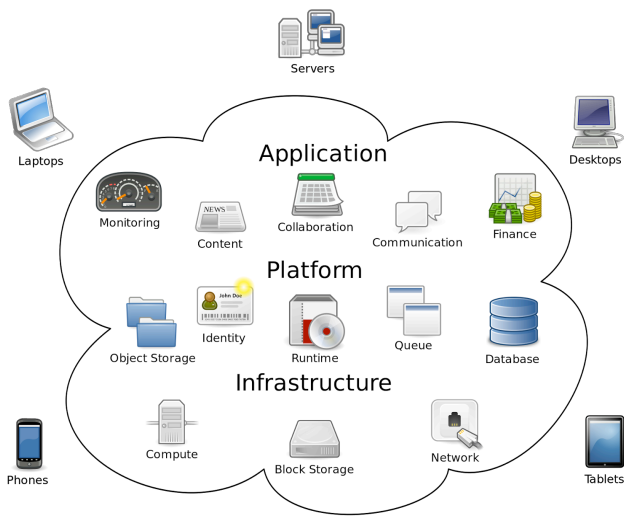


Fig. 2. Cloud computing [14]

OPC UA offers multi-threaded operations, a multi-platform implementation (ANSI C, Java, .NET) and new standard based security among others. High level of security of OPC UA includes, among others, sequencing, the use of end-to-end encryption, auditing and redundancy. The OPC foundation also provides members with Compliance Test Tools (CTTs) for test-case specifications, automated testing or interoperability workshops for tests with different vendors.

Another web service is representational state transfer (REST) which offers interoperability between computer systems on the internet. REST fully relies on the HTTP standard and thus it is usable by any device that support the HTTP standard and makes it easy to connect old and new devices as the protocol keeps the same. As REST fully relies on the HTTP it is compatible with intermediate components like firewalls, proxies and gateways.

D. Meta Data Model

A CBM system is composed of different elements, and each company might use a distinct way of connecting these

elements and transmitting the information through them. This way to interconnect the different elements and pass the information has been called the “Meta Data Model”. The solution to this problem is the standardization of the data models.

However, it can be claimed that standardization and meta-models are not the same way to ensure the interoperability in the exchanges of data during CBM processes. Indeed, three approaches can contribute to the improvement of interoperability:

- the integrated approach based on the use of standard formats
- the unified approach which requires the definition of meta-models
- the federal approach which is based on the definition of ontologies whose implementations enable the dynamical adaptation of the systems

Machinery Information Management Open System Alliance (MIMOSA) produced Open System Architecture for Condition-Based Maintenance (OSA-CBM), represented in figure 3, which is one of the most important open standard for information exchange between the plant and the machinery information systems. One of the advantage of using a standard is that components from different companies become interoperable and the compatibility issues disappear.

Even though MIMOSA is presented as defining standard format for the data exchange, it also provides the Meta Data Model structure together with the definition of the ontologies of the data. In fact, one of the greatest benefits in using MIMOSA is this definition of semantics and ontologies so the party that is developing their CBM solution does not need to worry about how different types of information need to get linked together.

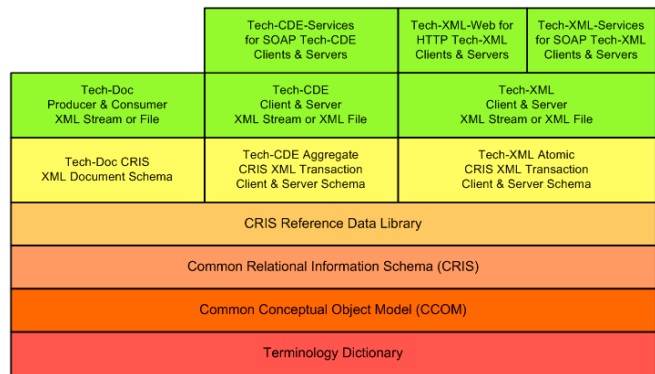


Fig. 3. MIMOSA OSA-EAI Architecture [18]

It can even be claimed that the whole foundations of a Computerized Maintenance Management System (CMMS) can be found defined in MIMOSA. Naturally, the one thing that is missing from MIMOSA so that it cannot be claimed to be a CMMS solution is the user interface which is the key factor in enabling the easy and efficient use of an CMMS

solution in field i.e. the maintenance technicians cannot be expected to be using a database engine in their everyday work.

It should be noted that maintenance and especially condition monitoring data are very hierarchical i.e. when something is measured it needs to be linked to a component of a machine that is monitored. The component in turn needs to be linked into to the machine that is monitored. Also, the component needs to be linked to maintenance history data. Measured condition monitoring signal often needs to be linked with the measured process parameters so that the measuring condition can be defined. Then follow the signal analysis, diagnosis, and prognosis phases with the associated data. Further on the prognosis should lead to actions i.e. management of work orders and spare parts and so on. The UML descriptions that are available in www.mimosa.org give a nice inside view on this. It seems that quite often all of this is actually forgotten when new condition monitoring solutions are developed together with new data structures.

OSA-CBM comes from the words Open System Architecture for Condition-Based Maintenance and it is a standard architecture for transferring information in CBM systems. OSA-CBM was developed in 2001 by an industry led team (participants from Boeing, Caterpillar, Rockwell Automation, different universities etc.) and it was partially funded by the Navy through a Dual Use Science and Technology (DUST) program. OSA-CBM was developed to standardize information exchange specifications within the community of CBM users and through that ideally drive the CBM supplier base to produce interchangeable hardware and software components and thus result to a free market for CBM components.

OSA-CBM has multiple benefits including cost reduction, increased specialization, increased competition and on the other hand it gives also a possibility to increase cooperation. Cost reduction comes through eliminating the need of system integrators and vendors to spend time creating new or proprietary architectures and through increased competition.

The OSA-CBM consists of multiple interoperable functional blocks as shown further and thus the whole CBM system doesn't have to be ordered from a single vendor but instead every block can be competed with different vendors. Also, smaller companies which are not capable to offer the whole CBM can take part of CBM system through providing functional blocks. On the other hand, interoperable functional blocks can increase cooperation. These multiple functional blocks give possibility to concentrate on smaller areas of CBM and so increase the quality of the whole CBM system.

OSA-CBM follows the ISO-13374 Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- -standard from the International Organization for Standardization. Table 1 shows the data-processing and information-flow blocks that are presented in ISO-13374 and that OSA-CBM also follows. The following paragraph will discuss about the individual blocks:

TABLE 1. OSA-CBM Functional blocks [15].

Data Acquisition (DA)
Data Manipulation (DM)
State Detection (SD)
Health Assessment (HA)
Prognostics Assessment (PA)
Advisory Generation (AG)

- Data Acquisition:** The data acquisition is the first step in the different stages. It consists on getting the real world data into an electrical signal that can later be processed in a computer. This is done by transducers or sensors that can measure a wide range physical phenomena such as acceleration, position, temperature, pressure, etc. The signal that comes from the sensor needs to be suited and cleaned for an accurate representation of the physical phenomena, so it goes through different amplification or filtering stages as well as an analog to digital converter when necessary. Data is usually refined in a local server and then sent to the maintenance information center.
- Data Manipulation:** Here the signal analysis is performed, where the meaningful descriptors from the gathered signals are computed and the virtual sensor readings are created from the raw signals from the Data Acquisition block.
- State Detection:** It creates a “baseline” and compares the new data to the previously created profiles to detect if there are any abnormalities, and, if so, which profile the data belong to.
- Health Assessment:** It diagnoses the faults and the current health level. It is usually done by analyzing the previously collected information such as health story trends, operational status or loading and maintenance history.
- Prognostic Assessment:** This stage determines the future health state and the Remaining Useful Life (RUL) of the monitored asset. To be able to apply this stage, a wide range of initial data are needed on the possible failure types of the asset. The prognostic stage can be approached in two different ways: a model that describes the physical phenomena of degradation or a data-driven model where a pattern recognition system is implemented alongside machine learning techniques. Both approaches have their advantages and disadvantages, but often, both methods are combined to get the best result.
- Advisory Generation:** It provides the information on what actions have to be carried out, or takes part in

the actions required to optimize the life cycle of the asset or increase the Overall Equipment Effectiveness (OEE) of the plant by decreasing the downtimes of the equipment or the process.

v. ADIRA case study

This section describes a case study that applies the envisioned approach to CBM. In particular, the case study uses sensors to monitor an industrial machine, collects and transmits data by means of efficient middleware, and process the data in the cloud.

The Greenbender (figure 4) [19] is a metal sheet bending machine commercialized by ADIRA. The press brake model in this case study is a hybrid system that is powered both hydraulically and electrically, and is controlled via a fluid pumping sub-system. The hydraulics drive two pistons located on a pair of beams that serve as actuators. These actuators move a ram vertically up and down onto a die that is fixed on the machine's base. The ram holds a punch. The workpiece is placed between the punch and the die, acting as clamps, so that it can be deformed.

The architecture for the CBM solution of the Greenbender is depicted in figure 5, which first of all divides the solution into 2 main parts, one local to the company hosting the machine, and another in the cloud, and which exposes an HMI accessible via web.

The components that build the local part of the solution are the machine under study, the MANTIS-PC, the edge local, and the local HMI. The machine, apart from being the target of the monitoring actions, comprises a number of data sources, some of them pre-existing such as a CNC and a Safety PLC, and some that were specifically added for the maintenance platform, for example Arduino-based accelerometers that monitor the bending blades for both acceleration, and vibrations.

The pre-existing sensors are accessible through the CNC of the Machine, while the new sensors needed a data concentrator, which is located on the MANTIS-PC. Each machine has got its own MANTIS-PC, and this latter component collects data from the new sensors, and afterwards converts them and sends them to the Edge Local. The MANTIS-PC is implemented on a Raspberry PI 3 Model B platform and communicates via Bluetooth Low Energy with the sensors, and through the OPC-UA protocol with the Edge Local.

The Edge Local component provides mechanisms in order to support communication and management of the data acquired across multiple heterogeneous and distributed data sources using OPC-UA protocol, and acts as the OPC-UA client in the interaction with the OPC-UA servers located on the MANTIS-PCs. The Edge Local is meant to be unique in each factory, and it performs preprocessing of collected data, and it represents the only connection with the Cloud, by acting as an AMQP client for the AMQP server located in the cloud.



Fig. 4. The Greenbender machine, featured in the ADIRA case study

The local HMI is a simple application that allows the user to view and monitor both the raw data collected in the factory, and the preprocessed information computed by the Edge Local.

The Edge Server subsystem is located on the cloud, and comprises a Middleware, a Database module, and the server for the HMI. The Middleware manages the data, more precisely by storing and transporting them between the Edge Local, the Data Analysis and HMI modules. The Middleware is message-oriented and is built over a AMQP messaging bus server. The Edge Local afterwards, saves received data on the Database (DB) Component, which is structured according to the MIMOSA/ IoT-A standard.

The Data Analysis Component comprises three modules. The first one is a set of Prediction Models, which is used for the detection, prognosis and diagnosis of machine failures. The models can be built specifically for one machine family or can be generic and further adapted to different machine families. The second module is the Prediction Application Programming Interface (API), which allows clients to request predictions from the models and provides data to feed and train the models involved. The final module is the Intelligent Maintenance Decision Support System (IMDSS), which is used to manage the models (model generation, selection, training and testing), for example on the reception of training data or when the API is contacted.

The HMI modules is a Human Machine Subsystem that allows for data visualization and management. The HMI allows to view historical and live data as it is received from the Middleware, inspect the results from the Data Analysis subsystem, (alarms for unusual data, warnings of impending failures, etc)

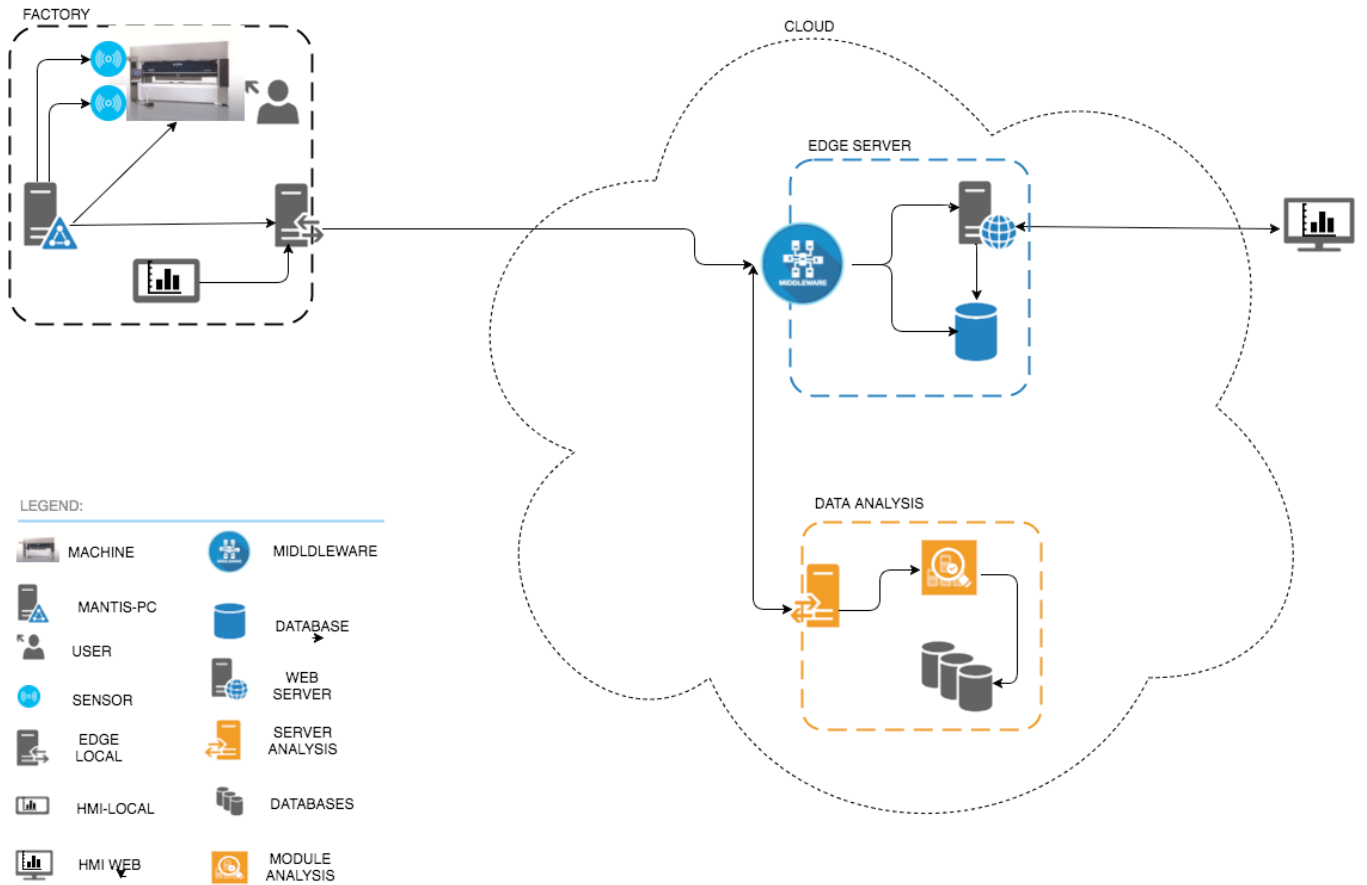


Fig. 5. CBM architecture for the ADIRA case study

VI. Discussion

The application of new technologies such as new sensors, CPS and cloud technology are stimulating an enormous change in how CBM can be taken in everyday use in the industry even in Small and Medium Size Enterprises (SMEs), and in that context their economic potential is huge. In the forefront of this introduction of new technology is the manufacturing industry that produces intelligent production machinery.

The whole service business can, in the future, be based on these new technological solutions. Consequently, the OEE can be raised to a new level i.e. from 60 % to levels around 90 %. For the European industry, this is especially meaningful as there are numerous SME companies that rely on this type of production machinery.

By increasing the OEE to one and a half times what it has been is really a dramatic change which can be benefitted from in the global markets. As explained in the previous chapters the backbone of this change is the fact that with the new technologies all machines can be monitored and the maintenance can be based on the need and not on some

statistics or guessing and at the same time sudden stoppages of production can be avoided.

The new technology can be used by personnel that are not experts of signal analysis or diagnosis and in cases where detailed and sophisticated knowledge is needed expert help can be called through the cloud and web services. It should be noted that the new technologies also remove the adverse influence of poorly done maintenance which today often is the cause failures.

First when following CBM strategy maintenance is not done in vain. With the new technology maintenance is always carried out to the right components using always the correct type of spare parts. The actual maintenance work can also be supported by new technological solutions like Virtual Reality (VR). Again, here it has been interesting to see how quickly the price of the needed hardware such as 3D glasses has dropped into a level where they can be used even at home for e.g. gaming purposes.

When considering what are the challenges in the introduction of the new technologies there are some:

- There will be high competition i.e. who will be the first to really introduce these new solutions in numbers.
- High competition might mean that not always the right technological solutions are made.
- Making the wrong solution might be expensive and a lot of time could be lost.
- When working in a hurry there is always the risk that short cuts are looked for and the results might then be very discouraging.
- If there will be numerous examples where the goals are not reached this will discourage the whole manufacturing industry.
- There are still technical challenges that have to be kept in mind e.g. even if all technology works the physics behind the need for maintenance have to be understood.
- When new technology is introduced companies are anxious to get financial benefit from it and it might be again discouraging if the time is longer than was originally expected.

VII. Conclusion

The paper describes the concept of CBM and CPS and how they are integrated together with the cloud computing in a maintenance system. Implementing a CPS will lead the industry to have more information on the monitored assets and, thus, more control over them. Not only this will help to reduce the downtime and increase the OEE, but it will also help the designers to create better equipment if the weak points are known.

As mentioned before, the new technologies such as MEMS sensors and the drop on the price of high processing power enable this type of maintenance strategy to be growing more popular. This, at the same time, allows the creation of new tools for said strategy, such as more powerful sensors or specific software. Current trends suggest that the price of these devices is going to keep decreasing in the near future. Taking a look at the evolution of the industry, there is a high chance that the CPS are going to become a must in the sector.

Data processing done by means of cloud-based advanced techniques can provide an edge for the implementation of CBM, and this is enabled by means of robust data models, and open standards.

Novel techniques are being experimented with by using demonstrators and pilots applied to real scenarios, speeding up innovation and proving the technological and economic potential of CBM for all the involved.

In summary, the implementation of such a system will help increase the automation of the plant while carrying out the maintenance with as little disruption as possible and improve the design of the equipment.

Acknowledgment

This work has been developed with the support of funds made available provided by the European Commission in the scope of ECSEL/H2020 MANTIS Research and Innovation Action (Project ID: 662189), by the Portuguese Fundação para a Ciência e a Tecnologia (FCT, I.P.) in the framework of project UID/EEA/00066/2013 PEST (Strategic Plan for Science and Technology) for the Centre of Technology and Systems (CTS), by the Finnish Funding Agency for Innovation Tekes, and by Ministerio de Industria, Energía y Turismo (Spain).

References

- [1] MEMSNET, What is MEMS Technology?. Available: <https://www.memsnet.org/about/what-is.html> [01/04, 2017].
- [2] M.J. McGrath & C.N. Scanaill, C.N., Sensor technologies: Healthcare, wellness, and environmental applications. Sensor Technologies: Healthcare, Wellness, and Environmental Applications, Apress, Open Access ISBN 978-1-4302-6013-4, DOI 10.1007/978-1-4302-6014-1 2013.
- [3] R. Frank, Understanding Smart Sensors, Artech House, Boston, ISBN 0-89006-311-7, 2013
- [4] J. Doscher, "Accelerometer Design and Applications", Available: http://elpuig.xeill.net/Members/vcarceler/articulos/jugando-con-el-wiimote-y-gnu-linux/sensor971.pdf/at_download/file [11/13/2016].
- [5] K. Agoston, "Accelerometer characteristics, errors and signal conditioning", The 6th edition of the interdisciplinarity Conference, Editura Universitatii "Petru Maior" of Tirgu Mures, 2012, pp. 276-281.
- [6] I. El-Thalji & E. Jantunen, "Fault analysis of the wear fault development in rolling bearings", Engineering Failure Analysis, Vol 57, 2015, pp. 470-482.
- [7] Piezo Film Sensors Technical Manual, Measurement Specialities Inc., 1999.
- [8] J. Fraden, Handbook of modern sensors: physics, designs, and applications, 5(th) ed., Springer, ISBN 978-3-319-19302-1, 2016.
- [9] TE Connectivity, ACH 01 Sensor Specifications. Available: <http://www.te.com/usa-en/product-CAT-PFS0014.html> [11/7/2016].
- [10] C. Rad, O. Hancu, I. Takacs & G. Olteanu, "Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture", *Agriculture and Agricultural Science Procedia*, 6, 2015, pp. 73-79.
- [11] K. Holmberg, A. Adgar, E. Jantunen, J. Mascolo, A. Arnaiz, & S. Mekid, E-maintenance, Springer, London, ISBN 978-1-84996-204-9, 2010.
- [12] R. Karim, A service-oriented approach to eMaintenance of complex technical systems, Doctoral Thesis, Luleå University of Technology, ISSN: 1402-1544, 2008.
- [13] J. Delsing, IoT Automation: Arrowhead Framework, CRC Press, ISBN 9781498756754, 2017.
- [14] Cloud image: <https://commons.wikimedia.org/wiki/File:Cloud>, [16/7/2016].
- [15] MIMOSA OSA-CBM Available: www.mimosa.org/mimosa-osa-cbm
- [16] Expert Microsystems, Available: <http://expmicrosys.com/>
- [17] National Instruments, Condition Monitoring. Available: <http://www.ni.com/condition-monitoring/>
- [18] MIMOSA OSA-EAI, <http://www.mimosa.org/mimosa-osa-eai>
- [19] L. L. Ferreira, et al. "A Pilot for Proactive Maintenance in Industry 4.0." 13th IEEE International Workshop on Factory Communication Systems (WFCS 2017). 31, May to 2, Jun. 2017.