



**Renato Jorge Marçal Martins**

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

## **Data Analysis in Automotive Industrial Cells**

Dissertação para obtenção do Grau de Mestre em  
**Engenharia Eletrotécnica e de Computadores**

Orientador: José António Barata de Oliveira,  
Professor Doutor, Universidade Nova de Lisboa  
Co-orientador: André Dionísio Bettencourt da Silva Parreira Rocha,  
Doutor, CTS-UNINOVA

Júri

Presidente: Doutor André Teixeira Bento Damas Mora  
Arguente: Doutora Sanaz Nikghadam Hojjati  
Vogal: Doutor José António Barata de Oliveira



FACULDADE DE  
CIÊNCIAS E TECNOLOGIA  
UNIVERSIDADE NOVA DE LISBOA

**Março, 2019**



## **Data Analysis in Automotive Industrial Cells**

Copyright © Renato Jorge Marçal Martins, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.



*To my family*



## ACKNOWLEDGEMENTS

I would like to express my gratitude and recognition to PhD Professor José Barata who gave me the opportunity to develop the work presented in this thesis. This opportunity allowed me to develop myself as both person and academically speaking.

A special thank goes to my co-supervisor PhD André Rocha, for all the support, interest, dedication showed and sharing his experience and scientific knowledge as well as friendship.

Also a special thank to Raquel Caldeira, Nuno Flores and Luís Flores for allowing me to develop and test this thesis at Introsys. Their support was fundamental to this work.

Furthermore, I would like to thank my Introsys colleagues, namely, Nelson Barreira, Magno Guedes, Diogo Correia, Nelson Alves, Daniel Deppen, Pedro Deusdado, Jorge Alcalde, Fábio Miranda, Tiago Cunha, David Henriques, José Gonçalves and André Silva, for sharing their insights and unending support.

I would also like to thank Pedro Barroca for all the friendship throughout the years, a friend in the true sense of the word.

My deepest gratitude to my family, especially to my parents, my brother and grandmothers, whose sacrifice and utterly dedication throughout these years made it possible for me to come this far. Everything I am today is a reflection of them.

To all of you,

My deepest gratitude and a great Thank You!





## ABSTRACT

---

The manufacturing industry always has been one of the leading energy consumers, so companies in this area are always trying to use the best tools provided by the evolution of the technology, to analyse and lower the production costs. Many known studies don't mind inconveniences such as stopping the production of the factory to perform studies or deep architecture improvements in the transport system.

The proposed solution offers two different sets of tools. A device adapter, that targets the gather and storage of data, from industrial robotic cells devices, being the main requirement for a data analysis application, and a data analysis system, that analyses the stored data, without changing the existing production model. The analysis procedure aims the energy usage of a cell and its robot, and the duration of the executed processes.

This solution was tested in two different robotic cells, that execute the same process. Multiple executions with different robot velocities were performed in order to gather the required data to provide an analysis and the conclusion was that, for both cells, the energy usage for each executed product was lower when the robot speed was higher, and that one of the cells is more efficient than other cell when executing at high speed but less efficient on lower velocities.

**Keywords:** manufacturing, energy, data analysis, optimisation

---



## RESUMO

---

A indústria de manufatura sempre foi um dos principais consumidores de energia, pelo que as empresas nesta área estão sempre a tentar usar as melhores ferramentas fornecidas pela evolução da tecnologia, para analisar e reduzir os custos de produção. Muitos estudos conhecidos não se importam com inconvenientes como interromper a produção da fábrica para realizar estudos ou melhorias profundas na arquitetura do sistema de transporte.

A solução proposta oferece dois conjuntos diferentes de ferramentas. Um adaptador de dispositivo, que visa a coleta e armazenamento de dados, de dispositivos de células robóticas industriais, sendo estes dados o requisito principal numa aplicação de tratamento de dados, e um sistema de análise de dados que analisa os dados armazenados sem alterar o modelo de produção existente. O procedimento de análise visa o consumo energético de uma célula robótica e a duração dos processos executados.

Esta solução foi testada em duas células robóticas diferentes, que executam o mesmo processo. Múltiplas execuções com diferentes velocidades do robô foram realizadas para reunir os dados necessários para fornecer uma análise e a conclusão foi que, para ambas as células, o uso de energia para cada produto executado foi menor quando a velocidade do robô foi maior, e que uma das células foi mais eficiente do que a outra célula quando executadas em maior velocidade, mas menos eficientes em velocidades mais baixas.

**Palavras-chave:** manufatura, energia, análise de dados, otimização

---



# CONTENTS

<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvii</b>
<b>Acronyms</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Description . . . . .	1
1.2 Goals . . . . .	2
1.3 Accomplished Work . . . . .	2
1.4 Dissertation Outline . . . . .	3
<b>2 State of the Art</b>	<b>5</b>
2.1 Production and Manufacturing . . . . .	5
2.1.1 Manufacturing and Industry 4.0 . . . . .	6
2.2 Data Analysis . . . . .	8
2.2.1 Data Analysis applied to Manufacturing . . . . .	9
2.2.2 Data Analysis applied to Energy Management in Manufacturing . . . . .	9
2.2.3 Predictive Manufacturing . . . . .	12
2.3 Global conclusions . . . . .	13
<b>3 Supporting Concepts</b>	<b>15</b>
3.1 Communication Protocols . . . . .	15
3.1.1 Open Platform Communications Classic . . . . .	15
3.1.2 Open Platform Communications Unified Architecture . . . . .	16
3.2 Databases . . . . .	17
3.2.1 SQL vs NoSQL . . . . .	18
3.2.2 Non-relational databases . . . . .	19
<b>4 Architecture</b>	<b>21</b>
4.1 Data gathering (Device Adapter) . . . . .	23
4.2 Database . . . . .	26
4.3 Data analysis (Energy Manager) . . . . .	27

<b>5</b>	<b>Implementation</b>	<b>31</b>
5.1	Device Adapter . . . . .	31
5.1.1	Connection and data gathering with the devices . . . . .	31
5.1.2	Connection and storage with the database . . . . .	34
5.2	Database . . . . .	36
5.3	Energy Manager . . . . .	36
5.3.1	Configuration . . . . .	36
5.3.2	Connection and search from the database . . . . .	37
5.3.3	Data analysis . . . . .	38
5.3.4	Alarms . . . . .	43
5.3.5	Charts . . . . .	43
5.3.6	Tasks . . . . .	45
5.3.7	Suggestions . . . . .	48
<b>6</b>	<b>Validation</b>	<b>49</b>
6.1	Introsys Robotic Cells . . . . .	49
6.1.1	Robotic cell - Ford standard . . . . .	49
6.1.2	Robotic cell - Volkswagen standard . . . . .	51
6.1.3	Robotic cells process . . . . .	53
6.2	Robotic Cell Changes . . . . .	54
6.2.1	Data to retrieve . . . . .	54
6.3	Trials . . . . .	55
6.3.1	Access and search into the database . . . . .	55
6.3.2	Analysis and display of results . . . . .	60
6.3.3	Error identification . . . . .	64
6.3.4	Tasks detection and analysis . . . . .	66
6.3.5	Suggestions analysis . . . . .	69
6.4	Other results . . . . .	71
6.4.1	Impact in the current system . . . . .	71
6.4.2	Deliverables and Publications . . . . .	71
<b>7</b>	<b>Conclusion and future work</b>	<b>73</b>
	<b>Bibliography</b>	<b>75</b>

## LIST OF FIGURES

2.1	Example for inter-dependencies of a supply chain in the context of Industry 4.0, from (Lasi et al. 2014).	8
2.2	Three data sets required in the feature-based approach, from (Peng and Xu 2015).	11
2.3	Support energy consumption evaluation in sustainable manufacturing, from (Peng and Xu 2015).	12
3.1	Mapping of the CIM data structure onto the OPC-UA address space, from (Lehnhoff et al. 2012).	17
4.1	Global architecture.	22
4.2	Device Adapter use cases.	24
4.3	Device Adapter connection with devices flow chart.	25
4.4	Device Adapter connection with the database flow chart.	26
4.5	Energy Manager use cases.	28
4.6	Energy Manager actions flow chart.	29
5.1	Sequence diagram of Device Adapter connecting to a device.	32
5.2	Example of the variables file of the Device Adapter.	33
5.3	Sequence diagram of variables update.	34
5.4	Sequence diagram of Device Adapter connection to the database.	35
5.5	Example of an entry from the database.	36
5.6	Example of the configuration file.	37
5.7	Sequence diagram of Energy Manager connection to the database.	38
5.8	Sequence diagram of Energy Manager search into the database.	39
5.9	Class diagram of Energy Manager structure.	42
5.10	Sequence diagram of Energy Manager graphics creation.	45
5.11	Sequence diagram of adding tasks.	46
5.12	Sequence diagram of edit, remove and graphic creation for tasks.	47
5.13	Sequence diagram of Energy Manager suggestions.	48
6.1	Front image of the Ford robotic cell.	50
6.2	Aerial view of the Ford robotic cell.	51

6.3	Image of the Volkswagen robotic cell. . . . .	52
6.4	Loading station with spar. . . . .	53
6.5	Recipes retrieved from the database search, without applying any filter. . . .	56
6.6	Date filter example. . . . .	56
6.7	Recipes retrieved from the database search, after applying the date filter. . .	57
6.8	Date and Recipe filter example. . . . .	57
6.9	Recipes retrieved from the database search, after joining the date and recipe filters. . . . .	57
6.10	Speed filter example. . . . .	58
6.11	Recipes retrieved from the database search, after applying a speed filter. . .	58
6.12	Cycle time filter example. . . . .	58
6.13	Recipes retrieved from the database search, after applying a cycle time filter.	59
6.14	Example of the results tab. . . . .	61
6.15	Results chart. . . . .	62
6.16	Results data of chart table. . . . .	62
6.17	Results table of Volkswagen cell. . . . .	62
6.18	Results table of Ford cell. . . . .	63
6.19	Recipe "4"results. . . . .	64
6.20	Cycle 35 raw data, regarding recipe "4". . . . .	65
6.21	Other results with errors. . . . .	66
6.22	Task list and "Pick"configuration. . . . .	67
6.23	Results table of the complete process. . . . .	68
6.24	Results table of the [Pick] task. . . . .	68
6.25	Results table of the [Weld] task. . . . .	68
6.26	Results table of the [Drop] task. . . . .	68
6.27	Suggestion example, considering the errors occurred during the data gathering.	70
6.28	Suggestion example, removing the errors occurred during the data gathering.	70



## LIST OF TABLES

3.1	Relation databases pros and cons table, data from (Foote 2016). . . . .	18
3.2	Non-relation databases pros and cons table, data from (Foote 2016). . . . .	18
6.1	Ford robotic cell components . . . . .	50
6.2	Volkswagen robotic cell components . . . . .	52
6.3	List of the variables to be extracted from the devices. . . . .	54



## ACRONYMS

**ACID** Atomicity, Consistency, Isolation, Durability.

**AM** Addictive Manufacturing.

**AML** Automation Markup Language.

**BASE** Basically Available, Soft-state, Eventually consistent.

**CEP** Complex Event Processing.

**CNC** Computer Numerically Controlled.

**COM** Component Object Model.

**CPS** Cyber-Physical Systems.

**DCOM** Distributed Component Object Model.

**FMS** Flexible Manufacturing Systems.

**HMI** Human-Machine Interface.

**IPE** Intelligent Predictive Engine.

**LDS** Local Discovery Service.

**MSB** Manufacturing Service Bus.

**OEM** Original Equipment Manufacturer.

**OPC** Open Platform Communication.

**OPC-AE** OPC Alarms and Events.

**OPC-HDA** OPC Historical Data Access.

**OPC-UA** OPC Unified Architecture.

**OPC-DA** OPC Data Access.

**PLC** Programmable Logic Controller.

## ACRONYMS

---

**RMS** Reconfigurable Manufacturing System.

**SQL** Structured Query Language.

**TCP/IP** Transmission Control Protocol/Internet Protocol.

**XML** Extensible Markup Language.

## INTRODUCTION

### 1.1 Problem Description

In the latest years, energy consumption has been one of the greatest concerns in almost every industry. Energy efficiency has been recognised as one of the key issues in achieving sustainable manufacturing. Most factories and their processes were designed and constructed with the cost as the most important economic factor (Karnouskos et al. 2009). The energy consumed by a manufacturing process is a major direct measure of its impact on the environment. The energy consumed usually translates to the amount of energy that has been produced from fossil-fired plants or captive generators, having a strong link with the depletion and degradation of the environment and also to climate change issues as a fallout of the resulting CO<sub>2</sub> emissions (Krishnan et al. 2009).

Continuous changeover leaves little space for optimisation with traditional automation technology. The demand for reduced time to market when deploying new products causes system integrators to rush the machinery without fine tuning their parameters in an optimal way, causing the equipments to consume more energy than actually needed, leading to additional costs and waste.

With the evolution of the technology, most companies are trying to create new solutions to address this problem, by applying new methods of management and data analysis. In the manufacturing industry, the optimisation of processes can lead to a lower execution time and better use of its resources which by consequence will lower production costs.

Accordingly with the European Commission in (Buchholz 2011), process monitoring and control can provide support in reducing the consumption of energy, for optimising

the performance and resource consumption on machine level and factory and supply chain level, where decision- support systems consider energy consumption globally.

For new factories, that are build accordingly with the new industrial revolution, most of the new devices are already equipped with tools that provide data access, and by being able to access data from the devices it is possible to achieve energy monitorization. But this does not happen in older factories, which mainly contains outdated devices, since these devices do not contain the needed features to communicate with more recent systems that could access and use their data. In most of these cases, it is too expensive to buy and install solutions to achieve data monitorization in a complete production factory, since this would lead to a need of reconfiguration of devices, which can only be done when they are not executing, consequently leading to delays in the productions orders.

In the manufacturing industry, every phase of a product execution is programmed and scheduled in order to achieve a targeted production goal. This means that solutions that bring data access cannot have any impact in the execution time of any of the processes.

### 1.2 Goals

The goal is to extract and analyse data from industrial robotic cells in order to monitor the status of the cell during executions, keep track of error that occur and offer suggestions to reduce energy consumption. To achieve this goal, a few questions are placed:

- How is it possible to retrieve and monitor data from an industrial robotic cell without compromising its standard behaviour and performance?
- Which parameters can be changed in order to optimise its energy usage?

It is proposed the implementation of two different applications, where one targets data retrieval and storage, and the other one the analysis of that data. Both applications must be completely decoupled, allowing them to be used together or apart, in other solutions.

### 1.3 Accomplished Work

The architecture presented in chapter 4 is divided in two applications. The first application concerns about data extraction and storage, allowing it to be used by the second application, which regards data analysis and the display of the results.

The data extraction application is meant to connect to an industrial robotic cell, comprised by multiple devices, being one of them an industrial robotic arm. After a connection to the devices is established, an user should be able to select any number of variables

to be stored in a local database, in pre-defined cycle time, once the industrial robotic cell starts producing.

With the database filled with some data, a user shall connect to it, using the data analysis application read its data. If the data stored allows it, this application shall calculate the energy used by the robotic cell, for each product, for each phase of a product and its duration. The velocity of the robot for every execution will also be stored, allowing for a comparison between executions of different velocities to be made. Some of the errors that happen during the data storage should also be detected during the analysis. All results are presented in tables and/or charts.

After the analysis is performed, it is possible to generate suggestions, indicating the robot velocity to achieve lower energy usage, but still meeting the requirements, which are a number of executed products in a given time.

## 1.4 Dissertation Outline

This dissertation is composed by seven main chapters: *Introduction*, *State of the Art*, *Supporting Concepts*, *Architecture*, *Implementation*, *Validation* and *Conclusion and Future Work*.

The first and current chapter, *Introduction*, provides a short introduction of the research problem, the main goals to be achieved and the activities developed to achieve them.

The *State of the Art* briefs the context of this work. Starts by an overview of the evolution in the manufacturing industry and the introduction of Industry 4.0, and it is followed by a brief summary on data analysis and its applications in manufacturing for energy management.

The third chapter, *Supporting Concepts*, is comprised by a description of the OPC communication protocols and its characteristics. A comparison between SQL and NoSQL databases is also presented.

In the *Architecture* chapter, the architecture for a system capable of extracting and analysing data is presented, detailing its different layers and requirements, including the data analysis procedure.

The *Implementation* chapter presents the implementation of the architecture detailed in the previous chapter, from the data extraction application to the data analysis.

In the sixth chapter, *Validation*, the validation environment is presented, comprised by two industrial robotic cells, their working process and the charges performed in order to extract all the required information. It is also described five test cases, each one of them targeting different features of the developed work. The deliverables resulted from this research are also presented.

Finally, the chapter *Conclusion and Future Work* is a critic overview of the work developed for this dissertation and its potential future research directions.



## STATE OF THE ART

In the beginning of production, a small number of models and construction variants made mass production result in an optimisation of the production lines, through the specialisation of labour. Changes in society organisation and structure as well as in markets and economy conditions trigger new and more challenging requirements for manufacturing industry. These changes cause increase demand for highly customised and customised models, as well as to the high number of different products available to the customer. To handle these changes, production lines with new structures and features were developed, on which the standard optimisation methods implemented so far were not enough. Data monitoring and analysis processes started being developed targeting optimisation. In order to answer these new requirements, the industry needed to evolve.

### 2.1 Production and Manufacturing

After World War 1, the demand for manufactured goods and products drastically increased. Henry Ford devised a methodology that consisted in mass production assembly lines to generate large volumes of standardised units. This method brings high advantages for low-mix and high-volume productions, since it minimises the production costs by using using interchangeable parts where one part could readily replace another instead of building an entire product from the beginning. In 1971, the Toyota Production System (TPS) was introduced. The main goal of TPS is to reduce production costs by minimising sources of wastes, which were commonplace in mass production systems, including overproduction, waiting, transport, processing, inventory, motion and defects. To be able to understand what needed to be changed, all production processes started being measured and assessed to determine and eliminate sources of waste while still maintaining high quality standards that are expected of the product. Unlike mass production, TPS is more

suitable for low-mix and low-volume applications, typically triggered by customer-pull or producing only when an order has been initiated to reduce inventory costs. This new methodology triggered new similar philosophies that aim on reducing waste, such as lean principles and six sigma techniques (Lee et al. 2013a).

In order to deal with changeable volume and mix productions, in the manufacturing industry there is a concept called Flexible Manufacturing Systems (FMS), that allows for a variety of products to be produced in the same system. It consists of expensive, general-purpose Computer Numerically Controlled (CNC) machines and other programmable automation. Because of the single-tool operation of the CNC machines, the FMS throughput is lower than that of dedicated lines. This combination makes the cost per part relatively high. Therefore, the FMS production capacity is usually lower than that of dedicated lines and their initial cost is higher, which causes this typology to have a low level of acceptance and satisfaction (Koren et al. 1999).

Given the downsides of FMS, a new paradigm appear, called Reconfigurable Manufacturing System (RMS). The RMS consists in using modular equipment as building blocks to realise the required system functionality for the production of a part family. Instead of providing a general flexibility through the use of equipment with built-in high functionality, as in FMS, it provides customised flexibility through scalability and reconfiguration as needed when needed to meet market requirements. The main goal of RMS is reducing lead time for launching new systems and reconfiguring existing systems, and rapid manufacturing modification and quick integration of new technologies and new functions into existing systems using basic process modules that would be rearranged quickly and reliably (El Maraghy 2006).

### **2.1.1 Manufacturing and Industry 4.0**

The manufacturing industry has been faced with a need that refers to the consumption of highly customised products. For several years the concept of mass production, characterised by the production of the same product on a large scale, has been widely implemented, but today it is imperative to treat variations of product types and can no longer respond to the challenges of modernity and dynamism. Large production lots, production lines with identical machines and processes, and standardisation of products no longer exist.

From the first industrial revolution (mechanisation through water and steam power) to the mass production and assembly lines using electricity in the second, the fourth industrial revolution, also known as industry 4.0, takes what was started in the third with the adoption of computers and automation and enhance it with smart and autonomous

systems with the aid of data and machine learning.

As stated in (Rüßmann et al. 2015), the Industry 4.0 is powered by nine foundational technologies advances: the cloud, additive manufacturing, augmented reality, big data and analytics, autonomous robots, simulation, horizontal and vertical system integration, the industrial internet of things, and cyber-security. The merging and cooperation between these technologies can bring great benefits like integrating production and logistics processes, enhancing cooperation among machines and humans and increasing efficiency on the factory floor. In the future, the car-making process will be overseen by automatic job-control systems. These systems will use data integration to modify the manufacturing process automatically, making multiple order systems obsolete. Car component suppliers will automatically adjust their processes on the basis of new orders from the automaker, maximising just-in-time logistics. This change will reduce the costs of logistics and operations.

The industry 4.0 focuses heavily on inter-connectivity, automation, machine learning, and real-time data. It represents a new paradigm shift in industrial production, with the digitisation within factories, the combination of Internet technologies and future-oriented technologies related with “smart” machines and products, targeting a modular and efficient manufacturing system and scenarios in which products control their own manufacturing process, allowing the production of individual products in a batch size of one while maintaining the economic conditions of mass production. This industrial revolution is related to a range of fundamental concepts, such as: smart factories, cyber-physical Systems, self-organisation, new systems in distribution and procurement, new systems in the development of products and services, adaptation to human needs and corporate Social Responsibility (Lasi et al. 2014).

In figure 2.1 is displayed an example of an extensive integration of different components into the supply chain regarding Industry 4.0. The cyber-physical production network is characterised by autonomous actions independent from the location, widespread integration, automated services, and by its ability to react context-specifically to customers needs and requirements. Among the different protagonists, manifold informational interrelations and inter-dependencies exist.

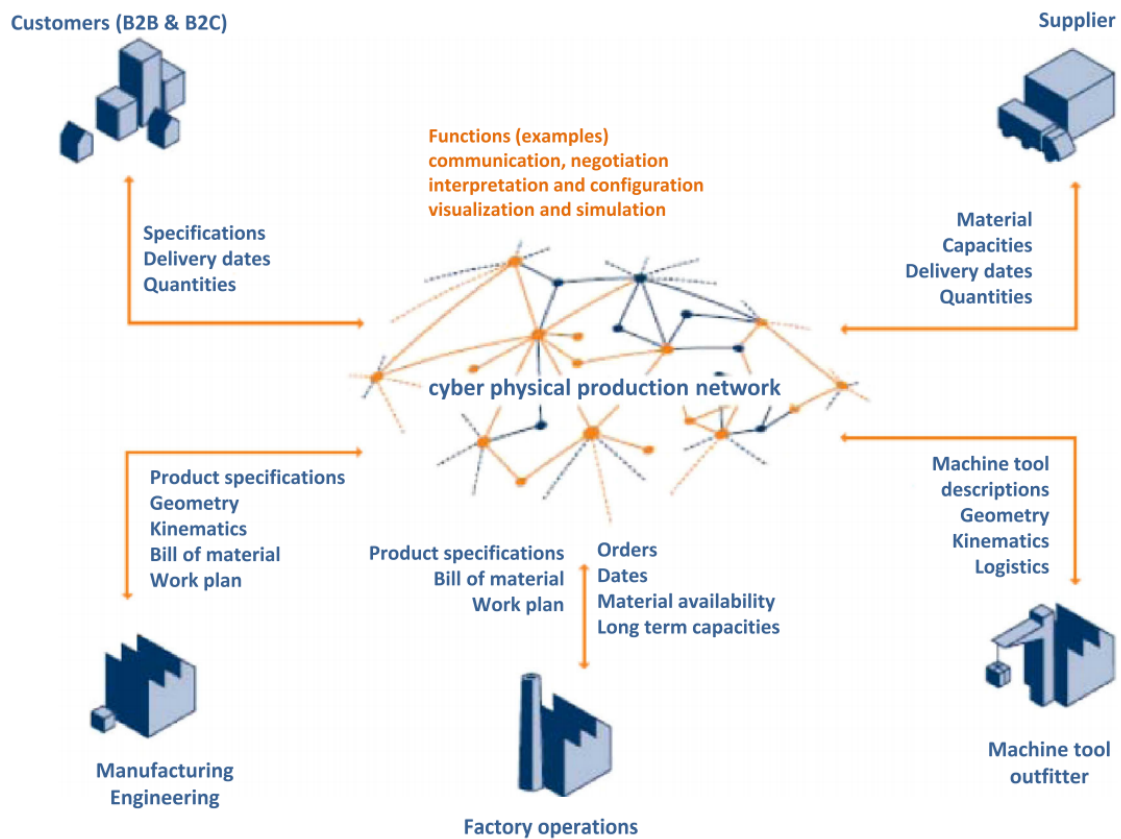


Figure 2.1: Example for inter-dependencies of a supply chain in the context of Industry 4.0, from (Lasi et al. 2014).

## 2.2 Data Analysis

With the development of computing, there is an increase in the data flow generated by machines and systems. Knowing that these systems can not be represented by simple models of known physical principles, the data they generate becomes a great help when analysed. Data analysis is the process of inspecting, cleansing, transforming, and modelling data with the goal of discovering useful information, reveal connections between system variables, informing conclusions, and supporting decision-making. Data analysis has multiple approaches, encompassing diverse techniques, while being used in different business, science, and social science domains. In today's business, data analysis is playing a role in making decisions more scientific and helping the business achieve effective operation (Xia and Gong 2014).

Now there is more data than ever before with high variety, its volume keeps increasing continuously, to a level that traditional tools are not able to process, making useful real time information harder to get, which creates great advantages for those capable of handling it (Fan and Bifet 2013).

A data analysis process can be divided into multiple phases, where results of later phases may become new inputs for early phases. These phases are: data requirements, data collection, data processing, data cleaning, exploratory data analysis, modelling data analysis, modelling and algorithms, data product, and communication (Schutt and O'Neil 2013).

### **2.2.1 Data Analysis applied to Manufacturing**

In the latest years, almost everyone in the manufacturing industry is talking about the Industry 4.0 and how smart factories are the future of manufacturing. One of the key aspects of a smart factory is having multiple devices capable of data transfer between them, such as sensors. With the technological evolution, smaller sensors are produced capable of being integrated almost anywhere, in any device. Having these sensors scattered across a factory can provide a great volume and diversity of data that can be used to create overviews of system status (Akbar et al. 2009). With the right sensors installed, these overviews can also contain energy data. In (Sung and Hsu 2011) is discussed the development of a smart sensor capable of measuring not only the energy used but also detect the ground vibration and more. Having access to different sort of data may provide a more focused analysis results. Energy efficiency is one of the targets for many future factories. As stated in (Feng et al. 2015), "Energy consumption in the automotive manufacturing plant is an important topic due to its implications on total plant operational costs and therefore the cost of the output product.", and is considered a critical requirement for accelerated economic growth (Krishnan et al. 2009).

In manufacturing, the data retrieved by sensors and actuators are interpreted as events. Any parameter change that have an impact in the system state is considered an event. When there are events that are triggered by other events, they are treated as complex events, which means that powerful data analysis tools are required to understand these patterns (Babiceanu and Seker 2016).

Some authors defend that there are three main stages that enable factory optimisation: monitoring, analysis and management (Cannata et al. 2009). Monitoring is having processes described in terms of data, including energy values for each machine, allowing its stages to be registered. Using the data acquired from monitoring, an analysis method can be developed and applied to better understand the behaviour and resource usage of each machine. The management phase is acting on the results achieved in the other stages.

### **2.2.2 Data Analysis applied to Energy Management in Manufacturing**

Data analysis in manufacturing is considered a really important step by (Gamarra et al. 2016), as it should be for anyone interested in smart factories. It goes similar as

stated in (Cannata et al. 2009) previously, to achieve any optimisation, first the data needs to be selected, analysed to produce some results and then validated. But still there is no defined standard in manufacturing to handle and analyse data regarding optimisation, so in different factories, diverse methods are applied trying to achieve the same goal. In (Boselli et al. 2004) is stated that the most cost efficient and effective way to achieve an optimal operating point of machines and entire production systems is to have a decentralised architecture, where each machine is responsible for its own optimisation, by offering the capability to learn models of their energy consumption behaviour, during a normal operation. It is also defended that a generic energy monitoring and control solution could be developed, combining it with simple frameworks with access mechanisms allowing the capability of connecting to different components and controls, being OPC one of the referred mechanisms, for being supported by many manufactures.

In (Cupek et al. 2014) refers to energy consumption in machines that use compressed air, that are used in almost every factory. His study took place in a laboratory with pneumatic devices, targeting the transport and assembly of parts between stations. The monitoring of the air usage in these machines is of great interest in manufacturing, being its optimisation a really important aspect.

As stated by (Peng and Xu 2015), "The first step towards energy efficiency is to develop an effective approach in understanding and characterising energy consumption of a manufacturing system." Defends that a feature-based approach can be adopted to support a wider range of applications in the manufacturing domain, rather than a operation-based approach. And also that to achieve a better optimisation, only the machine data is not enough but other aspects should be taken into consideration, as shown in figure 2.2.

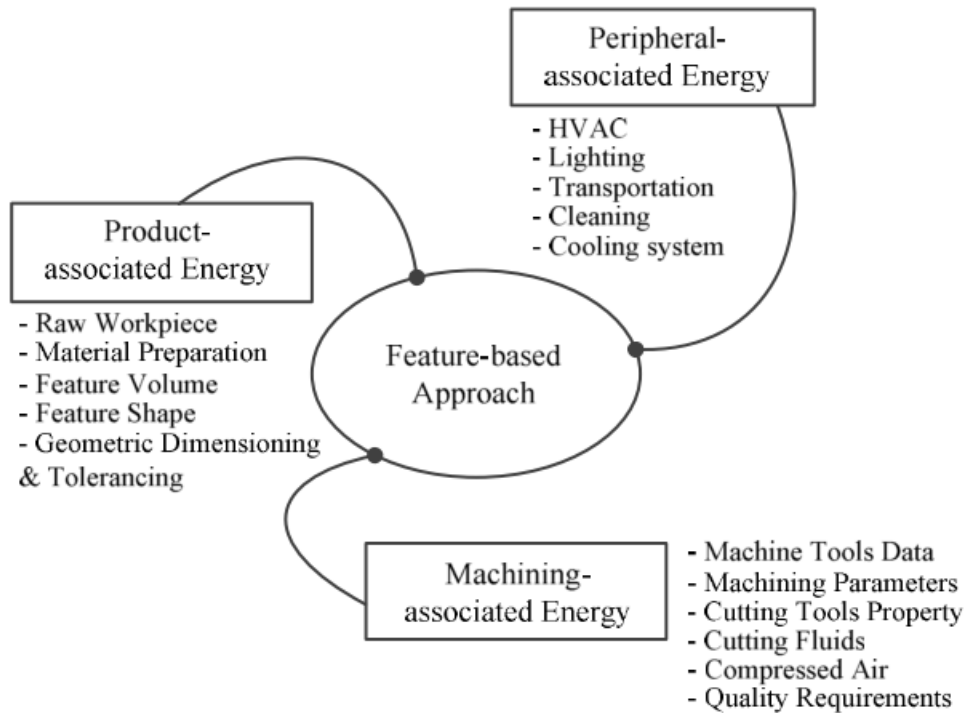


Figure 2.2: Three data sets required in the feature-based approach, from (Peng and Xu 2015).

However, the relationship between the two approaches is equally important. Feature-based approach is to provide a widely applicable energy reference across different machine tool systems, while operation-based approach offers actual energy estimation based on a specific machining system. The former can be treated as energy footprint of a part, and the latter is energy footprint of a machine tool and certain operations performed using this machine tool. The figure 2.3 displays some of the aspects related to the energy consumption evaluation for each approach.

There are not many energy models that can be directly applied in a feature-based approach, which results in them being considered as significant contributors in an operation-based approach, but these models are not satisfactory in supporting energy evaluation in different manufacturing stages. Based on various data required in a machining system, a Confidence-Level-ASSociated (CLASS) energy rating schema is proposed to help users in understanding energy evaluation, by turning energy consumption figures into useful indicators.

In (Feng et al. 2015), a systematic energy classification in the final assembly department is suggested in order to provide a more transparent understanding of the energy consumption in different categories, like:

- Lighting: responsible, in average, for 15% of the total energy consumption in a

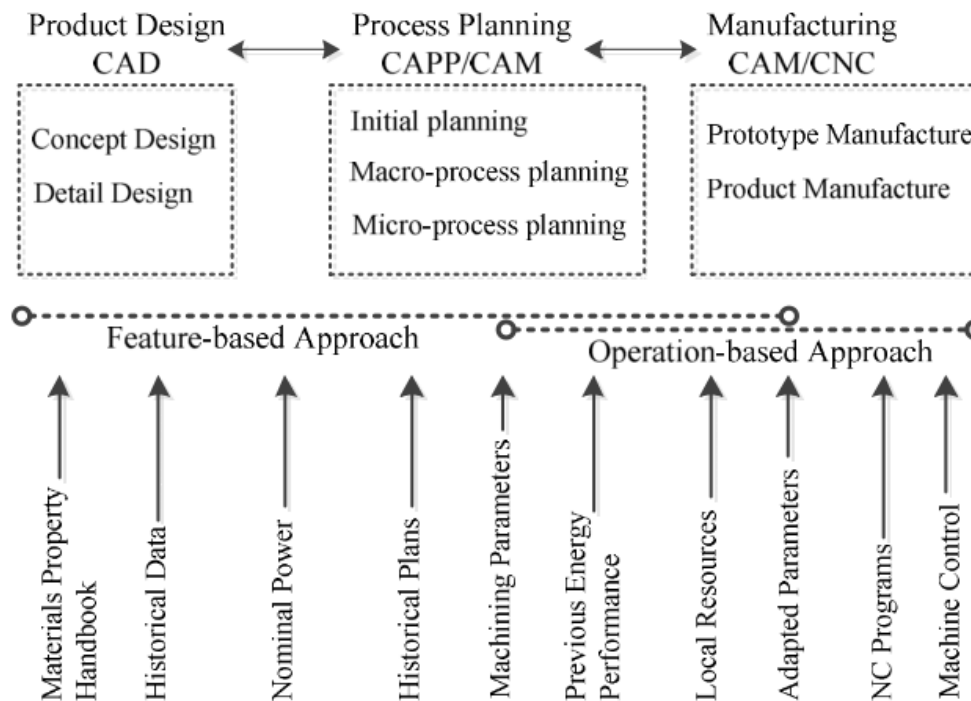


Figure 2.3: Support energy consumption evaluation in sustainable manufacturing, from (Peng and Xu 2015).

factory;

- Heating, ventilation and air conditioning (HVAC): energy used can originate from electricity, as well as natural gas, hot water, and chilled water, which are used to maintain a good working environment;
- Sub-assemblies: most of the energy consumption is electricity and compressed air;
- Main line: this process involves multiple robots to handle the body;
- In-plant transportation: energy consumption does not rely only on the transportation tool design, but also on the in-plant transportation planning and scheduling;
- Conveyor: the energy consumption is highly related to its power and time of use;

### 2.2.3 Predictive Manufacturing

The globalisation of the world's economies is a major challenge to local industries and it is pushing the manufacturing sector to its next transformation – predictive manufacturing.

Transparency is the ability of an organisation to unravel and quantify uncertainties to determine an objective estimation of its manufacturing capability and readiness. In order to achieve transparency, the manufacturing industry has to transform itself into



predictive manufacturing, which requires using advanced prediction tools enabling systematic data analysis resulting into information that can explain the uncertainties and help the responsible personnel to make more informed decisions. Prognostics and health management is a critical research domain that leverages on advanced predictive tools. By being able to estimate when an equipment is going start failing, it is possible to reduce the impact by allowing the users to prepare alternatives or solutions to prevent performance loss of the manufacturing system (Lee et al. 2013b).

The goal of a predictive manufacturing system is to provide machines and systems with “self-aware” capabilities. The core technology is the smart computational agent that contains smart software to conduct predictive modeling functionalities (Lee et al. 2013a).

In order to be able to provide assertive forecasts, the data must be well organised within the system. The Cyber-Physical System (CPS) is the representation of the physical components into computational capabilities. There are two main functional components that comprises a CPS: the advanced connectivity that ensures real-time data acquisition from machines and devices and the actual feedback from the system; and intelligent data management, analytics and computational capability (Lee et al. 2015). These components can be splitted into five levels: smart connection; data-to-information conversion; cyber; cognition; configuration. Predictive analytics techniques have been applied to improve manufacturing system control (Lechevalier et al. 2015).

Prevention and early identifying of faults can provide significant savings for manufacturing enterprises. Predictive techniques have been used for fault diagnosis in manufacturing applications (Lechevalier et al. 2015).

## 2.3 Global conclusions

The manufacturing industry have been creating multiple paradigms through the years, following the market requirements and trying to achieve an optimal manufacturing state, where the wasted resources are at a minimum.

In short, it is stated in this chapter the importance and beneficial impact that a paradigm like the one imposed by Industry 4.0 can have in the manufacturing industry, not only at the shop-floor level but in the complete supply chain, as explained in (Lasi et al. 2014).

In (Boselli et al. 2004) a decentralised approach is defended to be the best way to achieve an optimal operating point for each machine, while in (Peng and Xu 2015) a feature-based approach is taken, claiming that the machine data is important but not

enough for a more successful optimisation.

There are multiple approaches regarding energy management and optimisation since there is no defined standard yet. There is a great limitation regarding the process optimisation in many cases, since the range of inputs to reconfigure to achieve an optimisation is usually small. Nevertheless, the system performance must be analysed so that optimisation possibilities may be found.

## SUPPORTING CONCEPTS

In this chapter are described some of the technologies used in the development of this thesis and concepts that helped selecting those technologies. These concepts are divided in two categories: communication protocols and databases.

### 3.1 Communication Protocols

Open Platform Communications (OPC) protocols are greatly used across multiple industry environments, to interact with different devices. Two different sets of specifications for these protocols are described bellow.

#### 3.1.1 Open Platform Communications Classic

The OPC Classic specifications are based on Microsoft Windows technology using the COM/DCOM (Distributed Component Object Model) for the exchange of data between software components (OPC Foundation 2019a). The specifications provide separate definitions for accessing process data, alarms and historical data:

- OPC Data Access (OPC DA) - The OPC DA specification defines the exchange of data including values, time and quality information;
- OPC Alarms and Events (OPC AE) - The OPC AE specification defines the exchange of alarm and event type message information, as well as variable states and state management;
- OPC Historical Data Access (OPC HDA) - The OPC HDA specification defines query methods and analytics that may be applied to historical, time-stamped data.

Being all these specifications based on Microsoft Windows, they are only compatible with Microsoft systems.

### 3.1.2 Open Platform Communications Unified Architecture

With the introduction of service-oriented architectures in manufacturing systems came new challenges in security and data modelling. The OPC Foundation developed the OPC Unified Architecture (UA) specifications to address these needs and at the same time provided a feature-rich technology open-platform architecture that was future-proof, scalable and extensible. The OPC-UA is a platform agnostic service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework (OPC Foundation 2019b). This multi-layered approach accomplishes the original design specification goals of:

- Functional equivalence: all COM OPC Classic specifications are mapped to UA;
- Platform independence: from an embedded micro-controller to cloud-based infrastructure;
- Secure: encryption, authentication, and auditing;
- Extensible: ability to add new features without affecting existing applications;
- Comprehensive information modelling: for defining complex information.

In (Lehnhoff et al. 2012) are stated the advantages of using OPC-UA as a communication medium between a system and its devices, regarding smart grids. Since it offers information modelling, allows for enriching data with meta-data and thus exchanging information with known semantic rather than just exchanging pure data. It offers a client-server based connection-oriented communication, that with mechanisms like heartbeat and acknowledgements a reliable communication is guaranteed. A secure communication infrastructure is provided, handling authenticity, authorisation, confidentiality, completeness, availability, and traceability. The client can browse or query the server structure to access its data, including information about the nodes and references between them. It is also possible to set subscriptions of variables, where the changes occurred in the server are automatically transmitted to the client. In figure 3.1 is an example of the mapping of the Common Information Model (CIM; IEC 61970/61968) onto the OPC-UA address space is shown.

It is stated in (Palm et al. 2015) that currently there are no open reference implementation to be used in research for free but efforts are being made to achieve that, mainly by the open62541 project, which consists in developing an open source platform of the OPC-UA protocol. Examples of multiple application scenarios are also given in the paper.

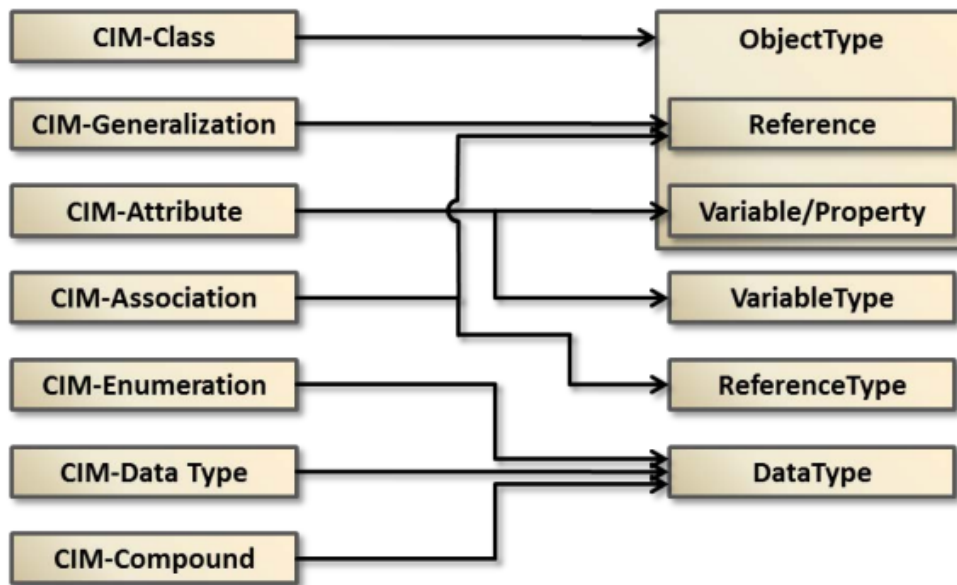


Figure 3.1: Mapping of the CIM data structure onto the OPC-UA address space, from (Lehnhoff et al. 2012).

In (Dorofeev et al. 2018) a concept is presented, to enable Plug-and-Produce, where an OPC-UA server is configured, by parsing an Automation Markup Language (AML) file that holds the description of a workstation, that contains the functionalities and structure of a workstation. Then the server is registered in a known Local Discovery Service (LDS) that is being monitored by a higher level called Manufacturing Service Bus (MSB). The MSB gets a notification that a new server was registered, then it creates a new OPC-UA client that is used to read the address space of the discovered OPC-UA server. Once the parsing is completed, the MSB knows all of the capabilities that the workstation holds, which processes it can execute and how to trigger them, information that is then shared with the rest of the system. With multiple OPC-UA servers configured using this concept, it is possible to propagate all individual and shared capabilities, of complete manufacturing lines, to higher levels to be managed, granting a new access level to those lines and devices properties.

## 3.2 Databases

The existing database technologies can be divided into two groups: relational and non-relational databases. As explained in (Foote 2016), relational databases, also known as Structured Query Language (SQL) databases, are a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganise the database tables. The standard user and application programming interface (API) of a relational database is SQL. SQL statements are used both for interactive queries for information from a relational database and for gathering data for

reports. Relational databases use the ACID (Atomicity, Consistency, Isolation, Durability) system, which ensures consistency of data in all situations of data management but obviously takes longer to process because of all those relations and its branching nature. Non-relational or NoSQL databases use the BASE system (basically available, soft-state, eventually consistent) that provides a mechanism for storage and retrieval of data. Non-relational databases forgo the table form of rows and columns relational databases use in favour of specialised frameworks to store data, which can be accessed by special query APIs. To enable fast throughput of vast amounts of data the best option for performance is "in memory", rather than reading and writing from disks.

### 3.2.1 SQL vs NoSQL

Both types of databases have different characteristics on which, depending on the solution or system where they are or will be implemented, their selection may vary.

A table with the pros and cons of a relational database can be found below:

Table 3.1: Relation databases pros and cons table, data from (Foote 2016).

SQL databases	
Pros	Cons
Relational databases work with structured data	Relational Databases do not scale out horizontally very well (concurrency and data size), only vertically
They support ACID transactional consistency and support "joins"	Data is normalised, meaning lots of joins, which affects speed
They come with built-in data integrity and a large eco-system	They have problems working with semi-structured data
Relationships in this system have constraints	
There is limitless indexing.	

A table with the pros and cons of a non-relational database can be found below:

Table 3.2: Non-relation databases pros and cons table, data from (Foote 2016).

NoSQL databases	
Pros	Cons
They scale out horizontally and work with unstructured and semi-structured data. Some support ACID transactional consistency	Weaker or eventual consistency (BASE) instead of ACID
Schema-free or Schema-on-read options	Limited support for joins
High availability	Data is denormalised, requiring mass updates
Many NoSQL databases are open source, but there are considerable training, setup, and developments costs. There are now also numerous commercial products available	Does not have built-in data integrity (must do in code)
	Limited indexing

In (Gyorodi et al. 2015), a study was made comparing some aspects and performance of a relational database and a non-relational database, MySQL and MongoDB respectively. As it is said in the same paper, "Relational databases are widely used in most of the applications and they have good performance when they handle a limited amount of data. To handle a huge volume of data like internet, multimedia and social media the use of traditional relational databases are inefficient." That said, when someone thinks about the amount of data to retrieve from industrial devices for status monitoring and energy optimisation, the data volume must be huge. In MySQL, data is stored in tables, which contains rows and columns. In MongoDB, data is stored in collections, which contains documents and fields. Still referring to (Gyorodi et al. 2015), a trial was made where the same data, with great volume, was inserted in both databases. MySQL took 440 seconds to insert all data, while MongoDB did it in 0.29 seconds. Testing query operations, two examples were made to select data, where MySQL took 0.0018 and 0.6478 seconds, and MongoDB did it in 0.0011 and 0.0052 seconds. Again, two new examples were created to update some data from the databases and MySQL needed 0.0987 and 0.0428 seconds and MongoDB only 0.0021 and 0.0013 seconds. These trials allowed to evaluate the advantages of using a NoSQL database when dealing with large volumes of data, which is almost every case when handling data analysis in the manufacturing area.

### 3.2.2 Non-relational databases

There are multiple different implementations of non-relational databases. In (Swaminathan and Elmasri 2016), a comparison is made between MongoDB, Cassandra and HBase, mainly taking into consideration the volume of data being managed. For this evaluation, 4 different types of trials were created: read and write, read-only, write-only

and scan-only, with different data volumes. The results for read and write showed that Cassandra have the best performance, followed by HBase. For read-only, MongoDB have the best performance, with Cassandra as second best. Write-only best performer was HBase, followed by Cassandra. For searching data, Cassandra performed better than HBase and MongoDB. In (Mahajan and Zong 2018), trials were made to prove the impact of query optimisation on both, Cassandra and MongoDB. A volume data of 100GB of data was used and the conclusion was that query optimisation can lead to better performance with lower energy usage, being the best results achieved with MongoDB.



## ARCHITECTURE

One of the topics around Industry 4.0 is data monitoring and process optimisation. The proposed architecture aims the gathering and analysis of data from industrial robotic cells, capable of providing a dynamic data gathering method and display of analysis results of processes and sub-processes executions.

The proposed architecture is divided in two different stages: data gathering/storage, which represents the data flow from the robotic cell devices to the database, and data analysis, where the data is read from the database and analysed. A simplified view of this architecture can be seen in figure 4.1.

An industrial robotic cell provides different sorts of data which can describe the process being executed, by analysing energy consumption, robot speed, task in execution and errors. Two applications will be developed targeting the different stages, where the first one will be responsible for data availability and storage into a database, while the second one will retrieve the data stored into the database, analyse it and provide the user with the results. Both applications shall be independent from each other, allowing them to be used in different solutions. The connection link between both applications will be the database. By using both applications in an industrial system, it will be possible to monitor not only the energy consumption of the industrial cell and robot in the different production stages, but also the duration on each process and which errors occurred.

Taking a closer look at figure 4.1, it is expected that the data to be retrieved from the system is not generated only by one device, but multiple devices, which comprise a robotic cell. The data from these devices shall be made accessible using the OPC-UA technology. The Device Adapter application, to be developed, will be able to contain

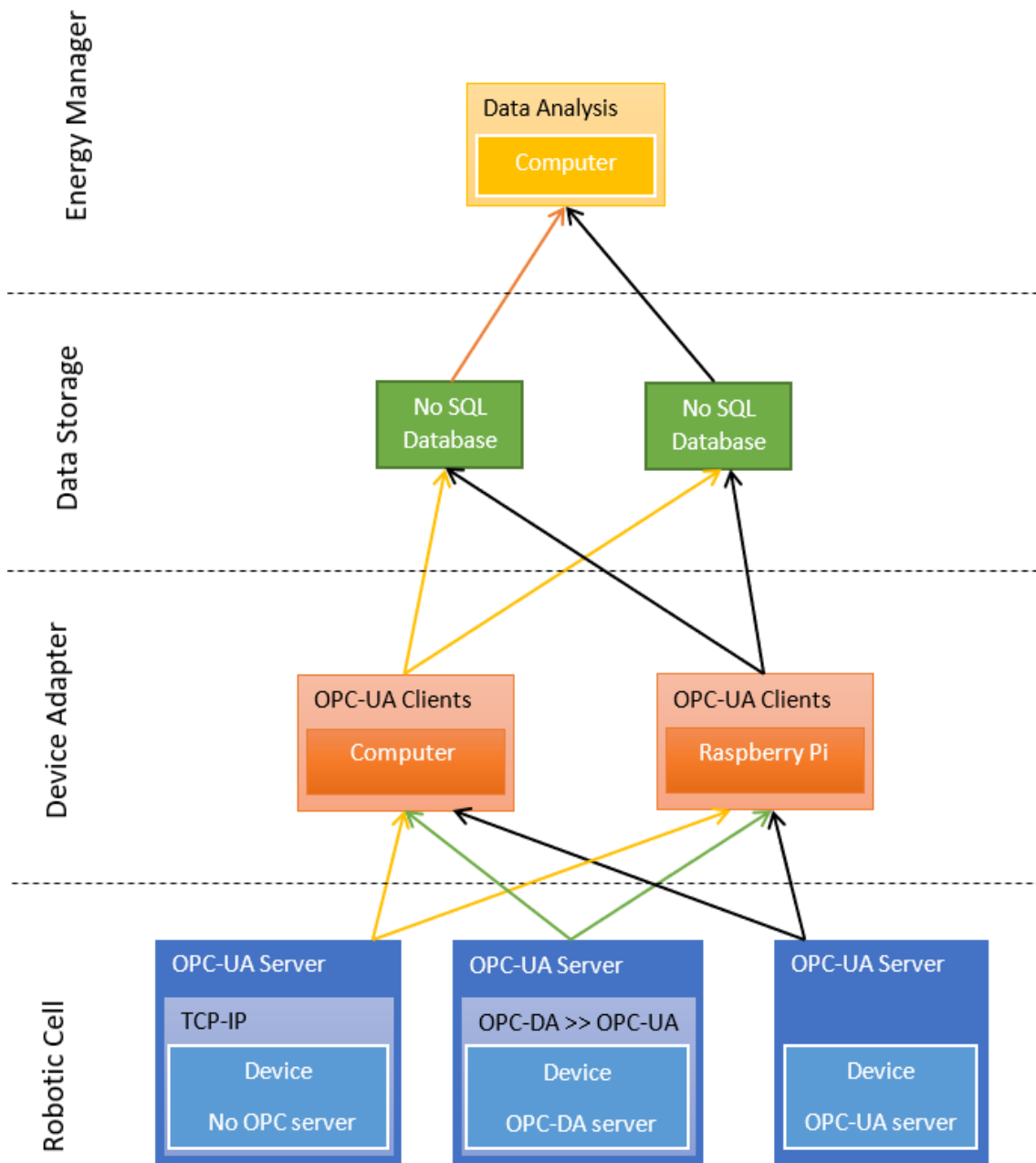


Figure 4.1: Global architecture.

multiple OPC-UA clients, to connect to each device, and then forwarding the data to the selected database. This application will be able to be used from a computer or a simple Raspberry Pi. The application will also implement the required interfaces to connect to the targeted databases. The second application, Energy Manager, will also implement the required interfaces to read the data from the selected databases, and it will display the results of the data analysis of the gathered data.

### 4.1 Data gathering (Device Adapter)

Industrial equipments/devices are built to last lots and lots of years, meaning that, older devices have limited communication capabilities when dealing with newer systems, including databases. To surpass this limitation, it was decided that a middle-ware was needed in order to ease the data gathering from the devices of the robotic cell and storage into the database. The perfect scenario would be for all devices to have an OPC-UA server integrated, but since that is not the reality, we need to prepare for the other cases. As shown in the figure 4.1, three types of devices can be found: devices with an OPC-UA server already integrated, devices with an OPC-DA server and devices with no OPC server.

The OPC-DA communication protocol uses COM/DCOM as communication medium, it is operating system dependent, creating limitations when communicating between different devices, a protocol upgrade was needed. The OPC UA protocol uses TCP/IP (Transmission Control Protocol/Internet Protocol) as communication medium, solving many OPC-DA limitations, making it the perfect candidate for this solution. So, for devices with an OPC-DA Server integrated, an OPC-UA wrapper needs to be installed, in order to translate the OPC-DA configurations and variables and expose them as an OPC-UA server. For devices with no OPC server, an OPC-UA server needs to be installed, communicating with each other through TCP/IP.

Since the data to be retrieved can be different from robotic cell to robotic cell, this middle-ware needs a HMI (Human-Machine Interface), allowing the user to select the devices to connect to, select the data to save and also allow the user to view it in real time. There were identified the use cases shown in figure 4.2.

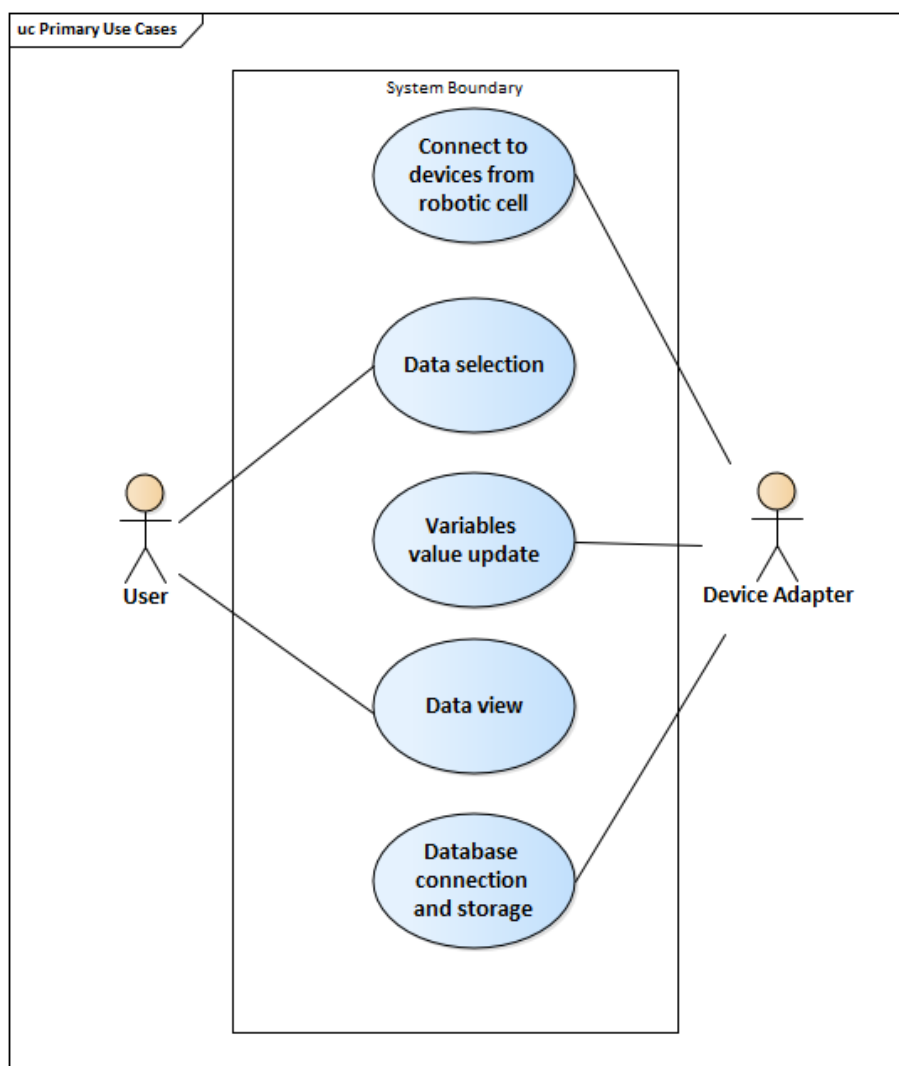


Figure 4.2: Device Adapter use cases.

This middleware is called Device Adapter. In the figure 4.3 is shown the data flow between the Device Adapter and one of the devices. After establishing a connection with a device, using the OPC-UA protocol, the Device Adapter gets all variable from the server, allowing the user to select the ones to monitor. For each selected variables, an OPC-UA subscription event, running in a different java thread, is created, where the variable value is read every cycle of a defined period and stored within the Device Adapter, in a hashmap. If any error occurs while reading a variable, a default error value must be set in the Device Adapter. This errors includes device shut-down or connection interrupted. After the connection with the device is re-established, the variables subscriptions are automatically set again, updating the values locally.

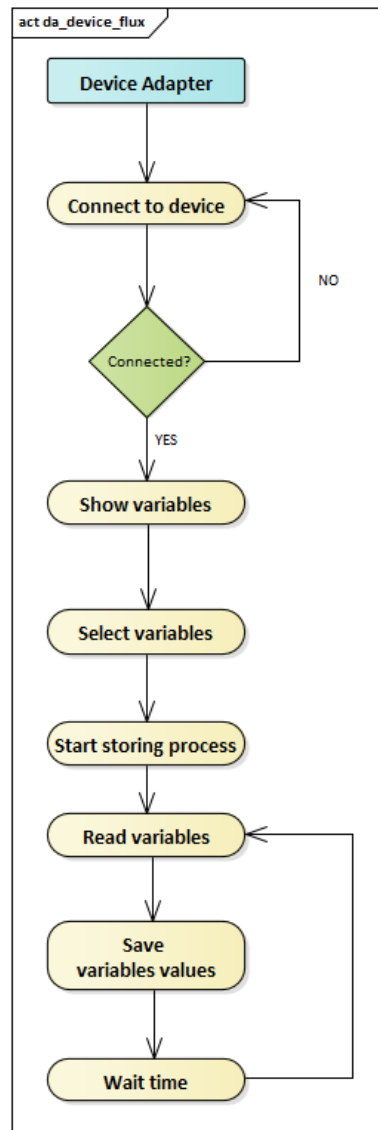


Figure 4.3: Device Adapter connection with devices flow chart.

The figure 4.4 shows the data flow between the Device Adapter and a database. As represented in the figure 4.4, after the Device Adapter successfully connects with a database, the data storage process can start, in a separate java thread, where in pre-defined time cycles, the data from the hashmap will be saved into the database. As explained in the previous flow chat (figure 4.3), variable reading errors are dealt by setting a default error value to the variables involved. This value will be also stored into the database, allowing the user to detect that something wrong happened with one or more of the devices.

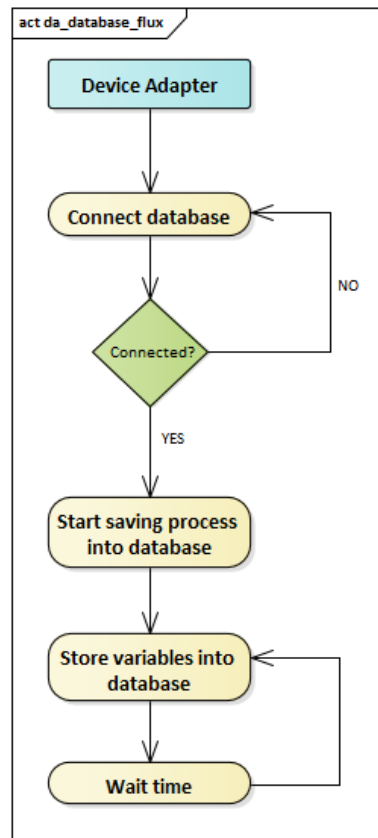


Figure 4.4: Device Adapter connection with the database flow chart.

## 4.2 Database

In order to provide a more detailed and assertive analysis to the robotic cell data, it is required a large volume of data. This data must be stored somewhere in order to be consulted and used by data analysis applications. For that, a database is required. The data to be stored is originated from a robotic cell and it contains information capable of characterise the status of the cell, like the energy being used by the robot, by the entire cell or which process is being executed.

As explained in the previous section (4.1), the user can select any number of variables from devices to be monitored and stored into the database, this means that the database must be capable of dealing this semi-structured data. Considering this, a not relational database is a must, since these databases have no problem receiving different data structures, after all there is no model. Taking not only this into consideration but also what was described in section 3.2, the database to be implement must be a NoSQL database.

The database is the middle layer between the Device Adapter and the Energy Manager. It will be connected to the Device Adapter, which has implemented the required interface and it will be the only input source of all data into the database. As described in the

previous section (section 4.1), the database will receive all variables values every cycle, with a pre-defined cycle time. Once in the database, the data can be used by any other user or application, including the Energy Manager, which also will have the required interface implemented to communicate with the database.

### 4.3 Data analysis (Energy Manager)

After the data is stored into the database, it needs to be analysed. Data analysis is the main goal of the second application to be developed for this solution. It shall have a modular architecture, allowing it to be compatible with other databases implemented with different technologies, with only minor adjustments.

This application shall allow the user to filter the data to be analysed, in order to provide a more precise analysis. A date filter allows to get data only from the selected date interval, for example to analyse the system behaviour in a specific day. In the manufacturing industry, the duration of a process is one of the most important aspects, since a delay in one process may result in delays in the product plans, which leads to production orders and schedules not been achieved, that can turn out to be expensive. A cycle time filter may be helpful identifying these cases. But of course that any other filters for different data fields existing in the database may be developed, to meet any requirements established.

The application should also be equipped with a raw data view, in order to visualise the data without any analysis. To show the results, this application will provide tables and linear and bar graphics, targeting the energy usage, allowing to compare multiple processes, finding out the most efficient. It should also be possible, not only to access historical data but also real time data, meaning that the results must keep being updated every time there is new data. It is also important to compare the time in which the robot is operating in manual and automatic modes. The considered use cases can be found in figure 4.5.

In the figure 4.6 is displayed the data flow of the Energy Manager. Since it is not possible to do data analysis without data, the first step within the Energy Manager is connecting to a database. After the connection is established, the user will be able to search its data, with or without setting any of the filters. The process is then followed by the data analysis and the display of its raw data in the the Energy Manager HMI. After the data analysis is completed, the results should be presented in charts. If the database contains the data regarding the status of the executed processes, it should be possible to split the data analysis for each phase of a process, meaning, split the data by tasks, allow a more detailed analysis for different tasks.

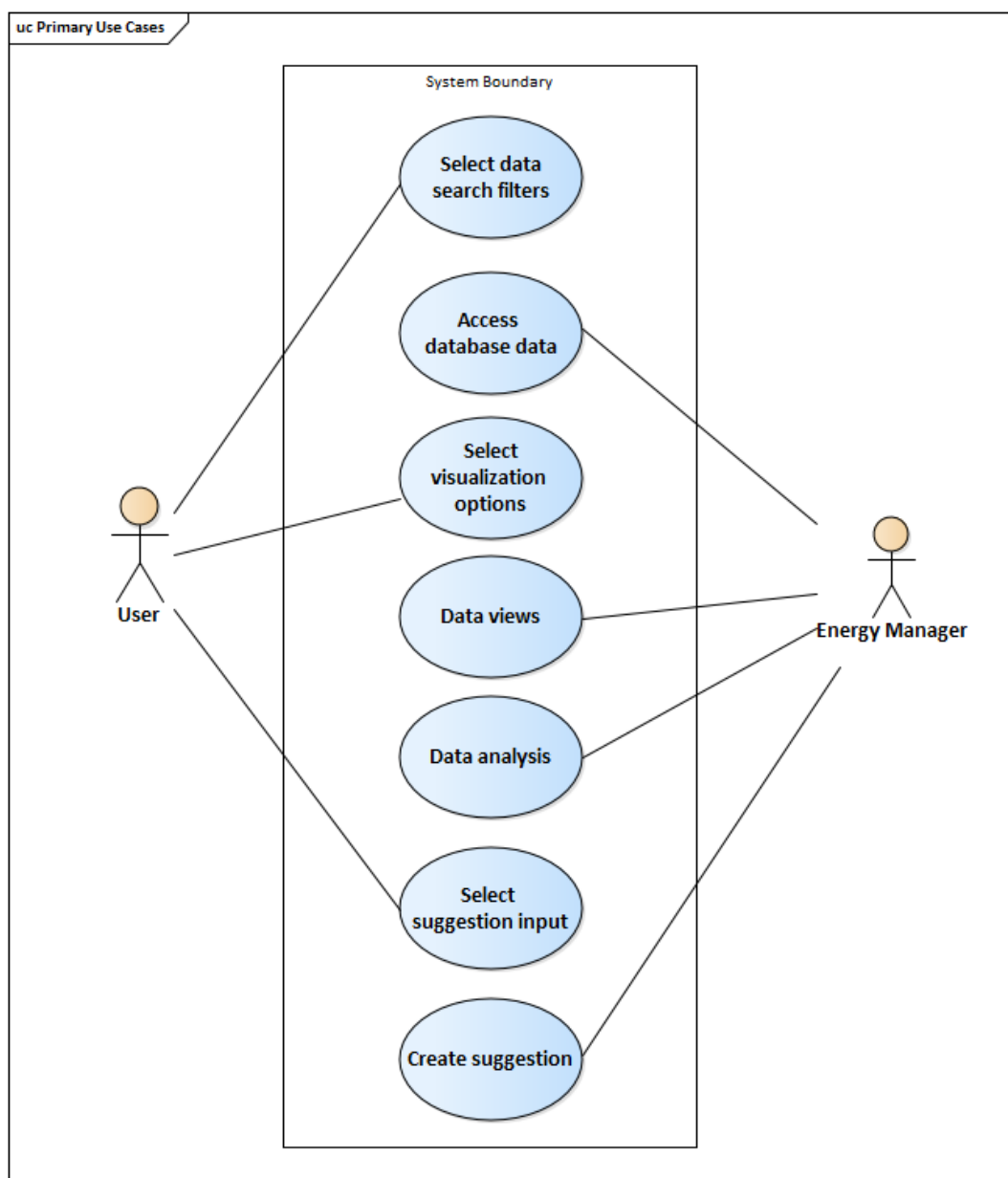


Figure 4.5: Energy Manager use cases.

A recipe is the aggregation of different tasks and inputs of the executed process, meaning that, a new recipe is set if any of tasks changes (new robot programs for pick, weld or drop) or if the execution speed changes. So when a process is executed multiple times but with different velocities, for each velocity a new recipe is created. The recipe identifier is created in the Device Adapter. If multiple Recipes are stored in the database, it should be possible create a suggestion, highlighting the more efficient recipe for execution.



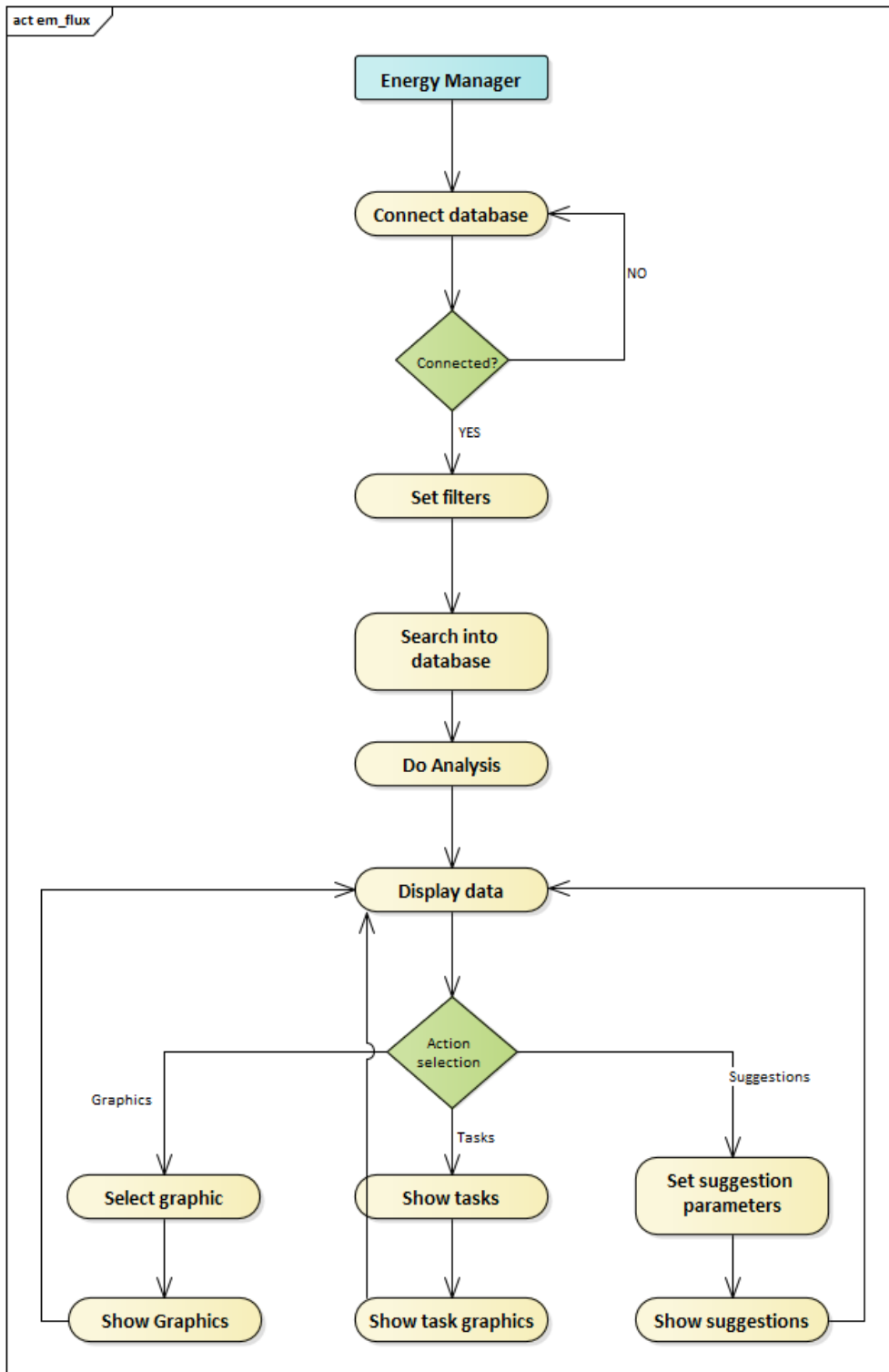


Figure 4.6: Energy Manager actions flow chart.



C H A P T E R



# 5

## IMPLEMENTATION

The implementation described in this chapter follows the architecture described in chapter 4. It was developed using the programming language JAVA.

### 5.1 Device Adapter

As explained in the previous chapter, the device adapter is the connecting bridge between the robotic cell and the database. Since all the data to be gathered is exposed in different OPC-UA servers, the middleware would need multiple clients to connect to them. So the first step was searching for an OPC-UA library that allowed OPC-UA client implementation. An open-source library was found, MILO. This library allowed for multiple clients to be developed, one for each device. After a successful connection with each device, the Device Adapter connects with the database, by using the "MongoDB Java Driver", and the storage process starts. To be able to execute all these processes, the computer where the middle-ware is being executed must be able to connect with the robotic cell and the database.

#### 5.1.1 Connection and data gathering with the devices

An OPC-UA client connects to an OPC-UA server, accessing all its exposed variables. As it is explained in figure 5.1, the user must know the OPC-UA server endpoint/URL (IP, port and server name) in order to connect to it. The communication session stays open until there is not data exchange for two minutes. The endpoint of the last server is saved in a configuration file, so at the next application start up, the server URL will be filled in the Device Adapter. There are different fields for each device server.

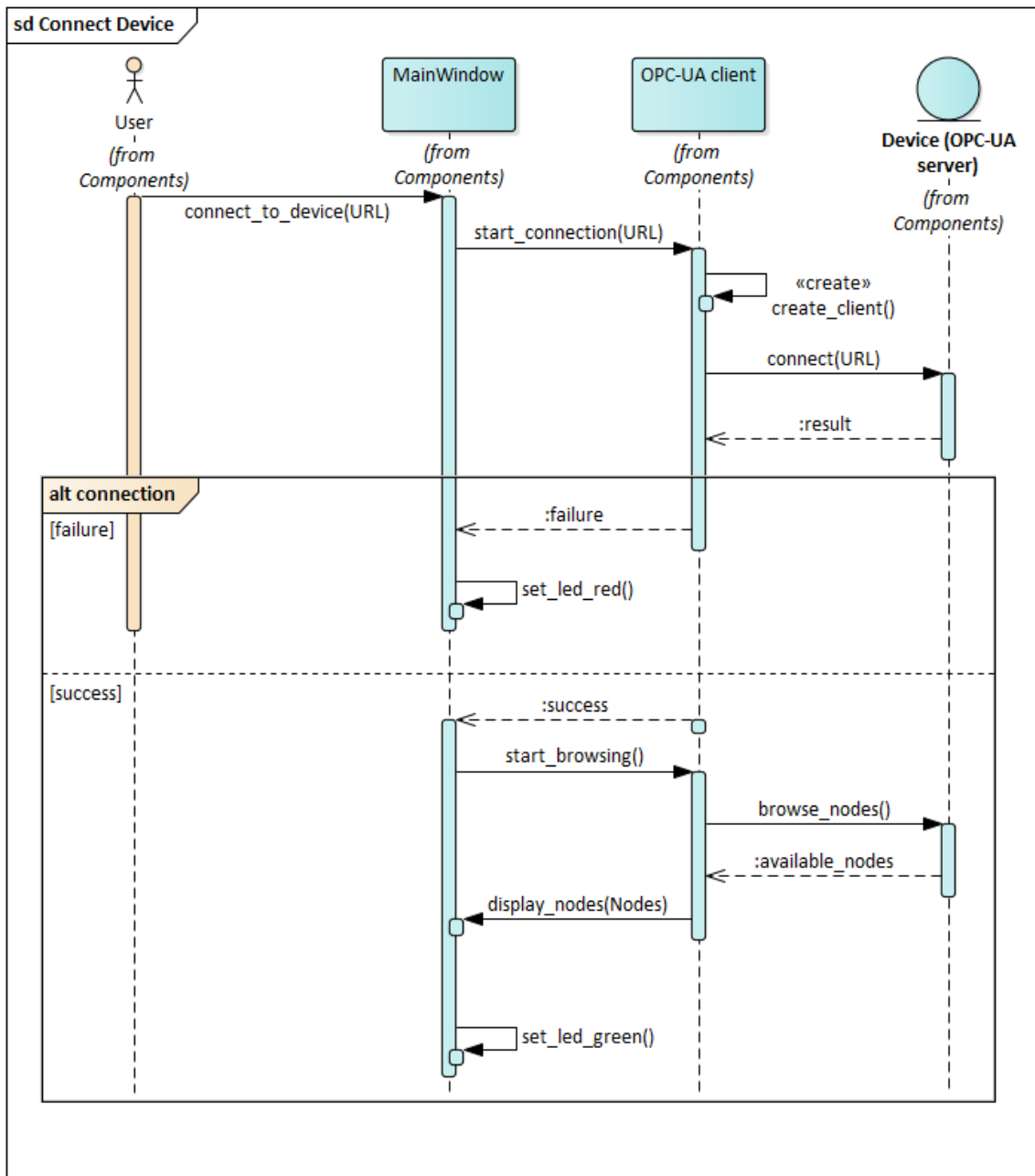


Figure 5.1: Sequence diagram of Device Adapter connecting to a device.

After a successful connection with a device, a list of all variables available in its OPC-UA server is loaded into the Device Adapter HMI (Human-Machine Interface), where the user is able to select the ones that he wants to subscribe. Once the subscription is done, the variables are saved into an hashMap, with the OPC-UA node identifier as key. The hashMap variables are then stored into a XML (Extensible Markup Language) file, an example of that file can be seen in 5.2.

If this file already exists when the Device Adapter gets the list of variables from the device, the variables that match between the list and the file will automatically be tagged for subscription. The variables in the hashMap will be display in the HMI, inside a table,

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Device>
  <Nodes>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_door}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.R_manual_1}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_R_fault}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_barrier}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Home}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.EB_station}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.ReadOK}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.Manual}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Job3}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Job2}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Job1}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.EB_door}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.Ready}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.ProgramCode}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_scanner}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.R_manual_2}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.InPosition}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.Auto}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Job5}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Execute.Job4}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.R_auto}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_CMFault}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_R_DCP}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_air}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_global}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.R_auto_1}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.EB_hmi}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.R_manual}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.E_R_path}</Node>
    <Node>NodeId{ns=2, id=Ford.PLC.Energy.EB_robot}</Node>
  </Nodes>
</Device>

```

Figure 5.2: Example of the variables file of the Device Adapter.

with the last updated values. The variable update process is shown in figure 5.3. The list of subscribed variables can be changed at any time by simply (un)selecting variables from the list and requesting the subscription again. If by any reason the link between the Device Adapter and the device gets interrupted, the variables of that device that were subscribed will be set with the value "bad".

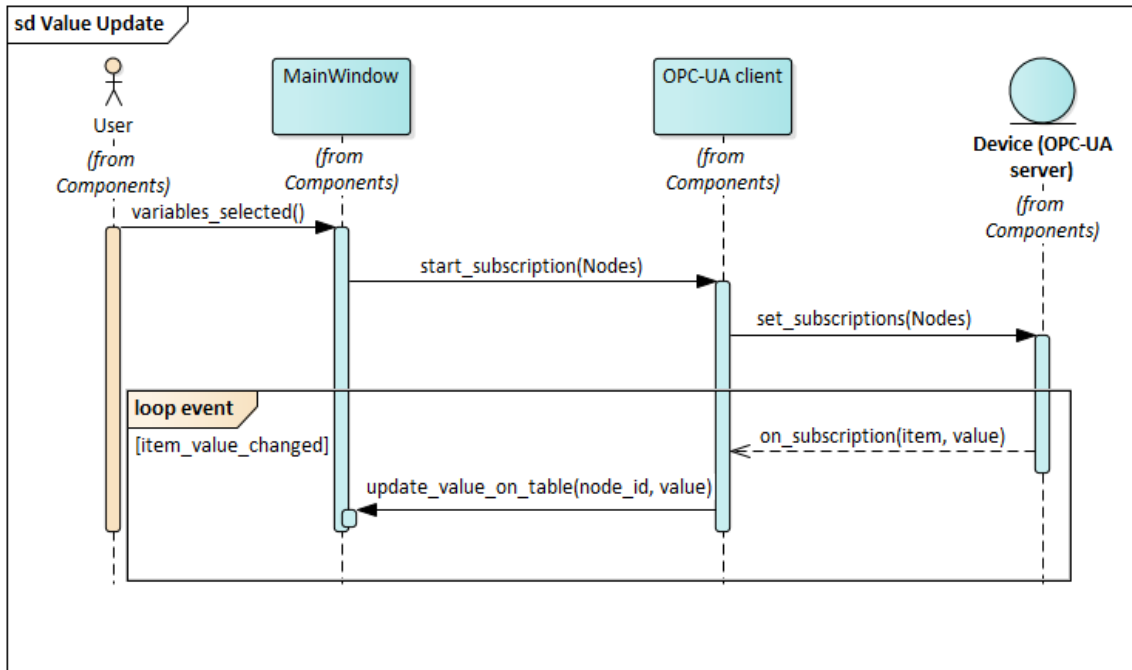


Figure 5.3: Sequence diagram of variables update.

### 5.1.2 Connection and storage with the database

After the Device Adapter and the database are connected, by setting the ip address and port number of where the database is running and by selecting the collection name to store the data, the user can request to start storing all of the subscribed variables values. The storing process happens, in a different java thread, every 200 milliseconds, because it is the same refresh rate as the OPC-UA subscription event, so a lower value would not make sense at this point. In each cycle of this thread, a BSON document is created where the name and value of all variables in the hashmap are added to, alongside with a date. Then, this document is sent to the database by using one of the functions available from the "MongoDB Java Driver", `insertOne(document)`, which means that all subscribed variables are stored into the database with each thread cycle. In figure 5.4 is described the connecting and storage processes.

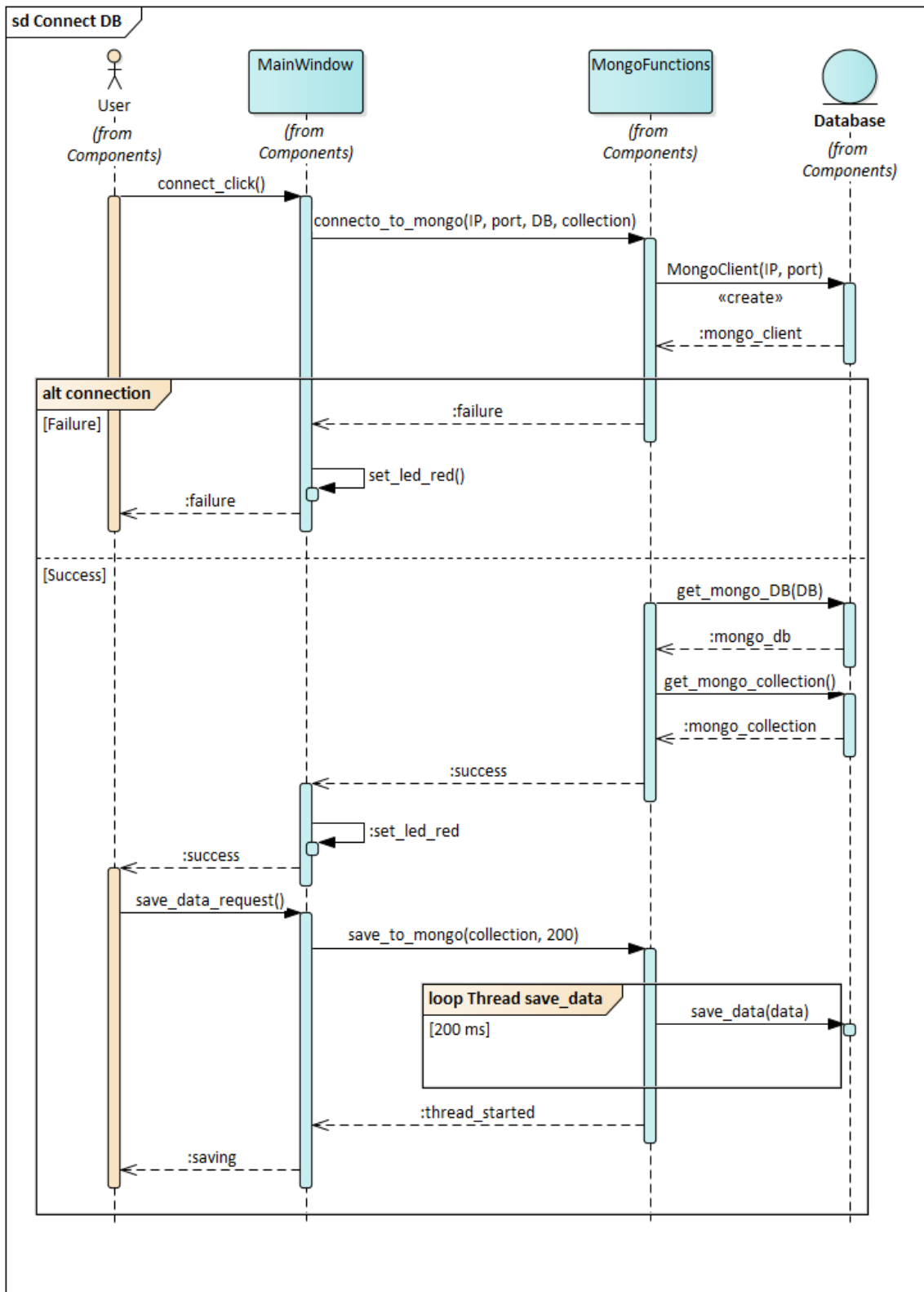


Figure 5.4: Sequence diagram of Device Adapter connection to the database.

## 5.2 Database

The NoSQL database selected for this solution was MongoDB. The MongoDB database stores the incoming data through a table with multiple entries. As a non-relational database, no model was developed. Being a NoSQL database, there is no problem storing data with different structures, which is requirement, since the data to be stored depends completely on the user selection, while using the Device Adapter, described in the previous section (5.1). Every interaction with this database in order to read or write data into it, was done using mongoDB queries, using the "MongoDB Java Driver". The data is stored in mongoDB documents, which are created in the Device Adapter, with each document containing all selected variables (names and values), and them is stored in the database. An example of an entry from the database can be seen in figure 5.5.

Key	Value
> (13) ObjectId("5b7d3310dd3c6715b097497f")	{ 36 fields }
▼ (14) ObjectId("5b7d3311dd3c6715b0974980")	{ 36 fields }
_id	ObjectId("5b7d3311dd3c6715b0974980")
RecipelD	1
ProductID	A
Date	22/08/2018 10:55:29:076
Cycle Time	4238
ProgramCode	1
InPosition	false
E_air_ok	true
E_barrier_ok	true
Home	false
POWER	19209
All_em_ok	true
E_door_ok	true
E_scanner_ok	true
Job3	false
Job2	false
Job5	false
EP_door	false

Figure 5.5: Example of an entry from the database.

The database was installed using a docker, in one of the servers at the Introsys facilities, allowing anyone connected to the Introsys internal network to have access to its data.

## 5.3 Energy Manager

### 5.3.1 Configuration

In order to make the Energy Manager able to read data from different industrial robotic cells a XML configuration file was created. Different robotic cells and different



variables, but by mapping them in the configuration file, a connection is made between the real variables and the Energy Manager variables. An example of a configuration file is shown in figure 5.6.

```

<Field Name="Date">
  <Variable>Date</Variable>
</Field>
<Field Name="CellPower">
  <Variable>POWER</Variable>
</Field>
<Field Name="LastCycle">
  <Variable>Last Cicle</Variable>
</Field>
<Field Name="AutoMode">
  <Variable>Auto Mode</Variable>
</Field>
<Field Name="ManualMode">
  <Variable>Manual Mode1</Variable>
  <Variable>Manual Mode2</Variable>
</Field>
<Field Name="RobotPower">
  <Variable>Actual_Power_1</Variable>
  <Variable>Actual_Power_2</Variable>
  <Variable>Actual_Power_3</Variable>
  <Variable>Actual_Power_4</Variable>
  <Variable>Actual_Power_5</Variable>
  <Variable>Actual_Power_6</Variable>
</Field>
<Field Name="Errors">
  <Variable>Error CMFault;true</Variable>
  <Variable>Error DCP;true</Variable>
  <Variable>Error Fault;true</Variable>
  <Variable>Error IBSOK;false</Variable>
  <Variable>Error Path;false</Variable>
</Field>
:/CONFIG>

```

Figure 5.6: Example of the configuration file.

### 5.3.2 Connection and search from the database

To connect the Energy Manager to the database, it was used the same library as in the Device Adapter, MongoDB Java Driver. The driver contains tools that allow the connection, searching and filtering of data. The connection process is the same as the one explained in the sub-section 5.1.2, without the storing component, as it is shown in figure 5.7.

The data search and filters are achieved by using queries. These queries allows for the data to be filtered by any existing field into the database. Taking into consideration the variables at table 6.3, some default filters were developed:

- Date - searches the database for data given a start date and/or end date ;
- Speed - searches the database for data where the robot speed was lower, higher or equals than the given speed. Multiple speed values can be inserted;

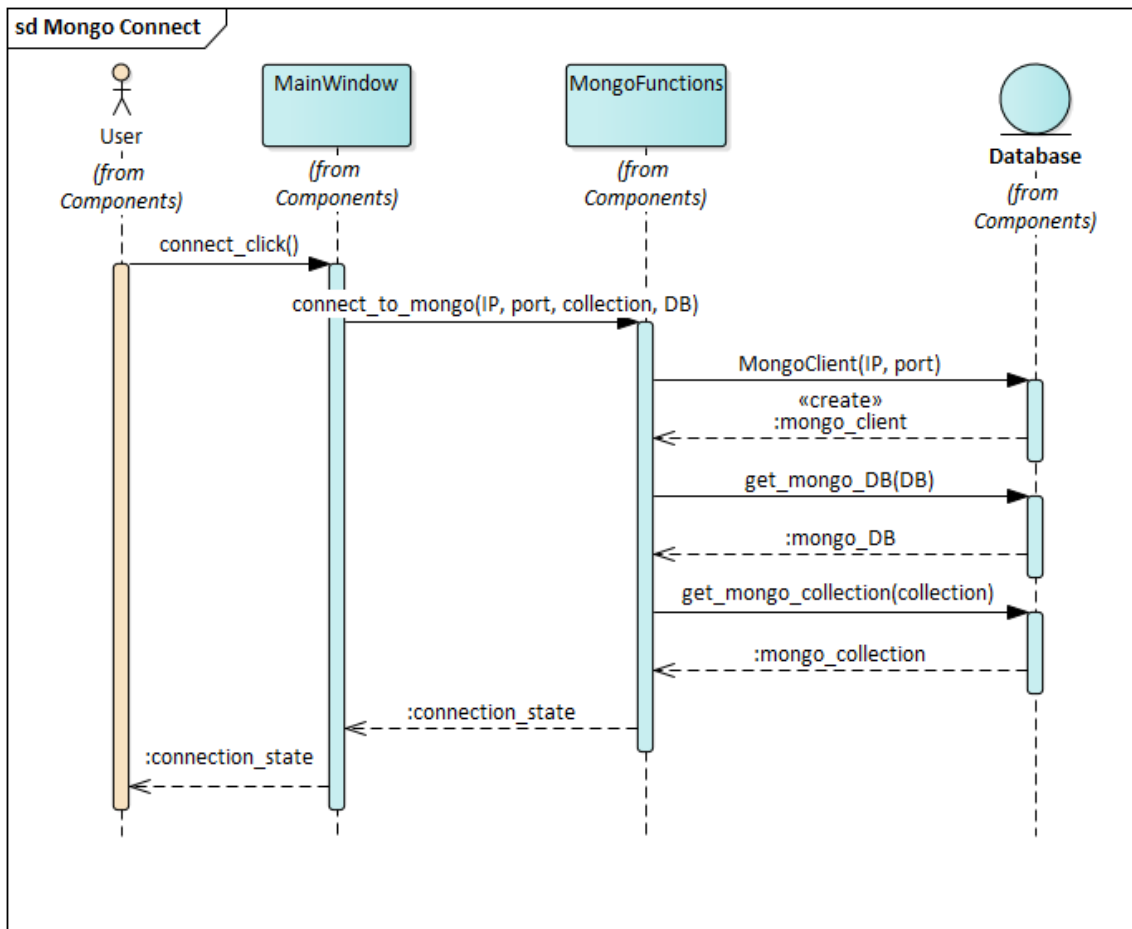


Figure 5.7: Sequence diagram of Energy Manager connection to the database.

- Cycle Time - searches the database for data containing execution cycles higher or lower, depending the selection, than the given time;

After the search is made, the data analysis process starts. This process also occurs in a dedicated java thread. The related diagram is found in 5.8.

### 5.3.3 Data analysis

As explained before, the data stored, including the energy values, is always instant values, meaning that some calculations were needed to find out the true value for the energy usage, for the robot and for the entire robotic cell. The energy data retrieved from the robot is instant data. To calculate the energy usage between two instants, the following equation was used:

$$E_{12} = \frac{E_{i1} + E_{i2}}{2} * (t_2 - t_1)$$

With:

$E_{i1}$  = instant energy value for sample 1;

$E_{i2}$  = instant energy value for sample 2;

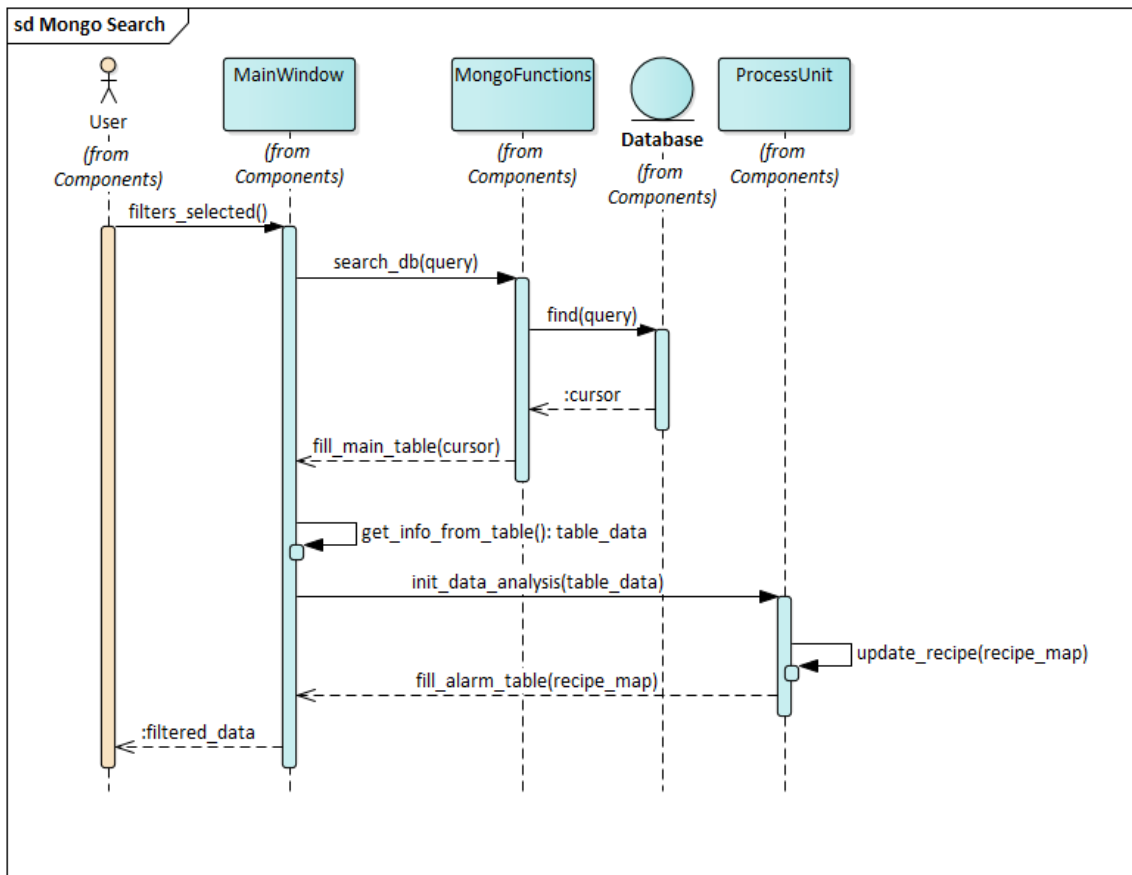


Figure 5.8: Sequence diagram of Energy Manager search into the database.

$t_1$  = time for sample 1;

$t_2$  = time for sample 2;

$E_{12}$  = energy used between sample 1 and sample 2;

The variable Cycle Time allows to the split the data of executions between cycles. Every time that the Cycle Time value is lower than the previous one, it means that previous cycle has ended and a new one just started. A java class was created, called EnergyData to organise the results of the data analysis of a sample, meaning, between two instants. The results stored for a sample are:

- Date - a variable containing the starting date and time of the sample;
- Duration - a variable containing the sample duration, in milliseconds;
- EnergyAxis - an array containing the energy usage, in watts, for each engine of the robot, during a sample;
- EnergyRobot - a variable containing the sum of values of the EnergyAxis array;
- EnergyCell - a variable containing the energy usage, in watts, of the whole cell;

- Cycle - a variable containing the cycle number;
- OtherData - a hashMap containing the other database fields and values of the remaining data for the sample;
- HaveError - a variable indicative if any error was found in the sample.

A list of EnergyData results is saved into the CycleData class, also created to provide a better organisation. A CycleData contains the results of an execution cycle. These results are:

- Cycle - a variable containing the cycle number;
- Duration - a variable containing the cycle duration, in milliseconds;
- EnergyAxis - an array containing the energy usage, in watts, for each engine of the robot, during a cycle;
- EnergyRobot - a variable containing the sum of values of the EnergyAxis array;
- EnergyCell - a variable containing the energy usage, in watts, of the whole cell, during a cycle;
- Faults - an array containing all errors detected during the analysis.

There is a CycleData for each execution cycle detected. Each CycleData is stored into the Recipe class.

For each recipe found into the database, the following structure is stored:

- Name - a variable that contains the recipe identifier;
- CycleData - a list that contains the results explained before;
- EnergyAxis - an array that contains the sum of energy usage for each robot engine, in all cycles;
- EnergyTotalRobot - a variable that contains the sum of EnergyAxis;
- EnergyTotalCell - a variable that contains the sum of energy usage of all cycles;
- TimeTotal - a variable that contains the sum of total duration of all cycles;
- StandardDeviation\_cell - a variable that contains the standard deviation of the cell energy usage, during the cycles;
- StandardDeviation\_robot - a variable that contains the standard deviation of the robot energy usage, during the cycles;

- Energy\_Hour\_cell - a variable that contains the average energy usage of the cell for an hour
- Energy\_Hour\_robot - a variable that contains the average energy usage of the robot for an hour
- Energy\_Average\_cell - a variable that contains the average energy usage of the cell during each cycle
- Energy\_Average\_robot - a variable that contains the average energy usage of the robot during each cycle
- Speed - a variable that contains the robot speed value for the recipe
- Alarm - a variable that indicates if the robot energy usage of any cycle is higher/lower than the standard deviation.

Having the energy usage divided in multiple cycles, it is possible to calculate the standard deviation of the robot energy usage for a recipe, using the following equation:

$$SD = \sqrt{\frac{\sum_{i=0}^{CycleCount} (energy_i - energy_m)^2}{CycleCount - 1}}$$

With:

$energy_m$  = average of the total energy usage;

CycleCount = total number of cycles for the recipe;

In figure 5.9 is shown the organising structure of the Energy Manager. The class "Data" holds all data structures of the system. It owns a "Map" called "recipeMap", that is used to organise the data of each recipe detected in the database. It has as a key the recipe identifier and an instance of the class "Recipe" as a value. The "Recipe" class structure was described previously. It holds a list of "CycleData", where each "CycleData" represents an execution cycle of a process. Its structure was also presented before. It has a list of "EnergyData", where each element represents the analysis between two instants. This structure allows for the data and analysis results to be divided between recipes, execution cycles and instants. The class "Data" also owns a list of "Task" elements, each "Task" contains some key aspects of a sub-process, that will be explained ahead in sub-section 5.3.6.

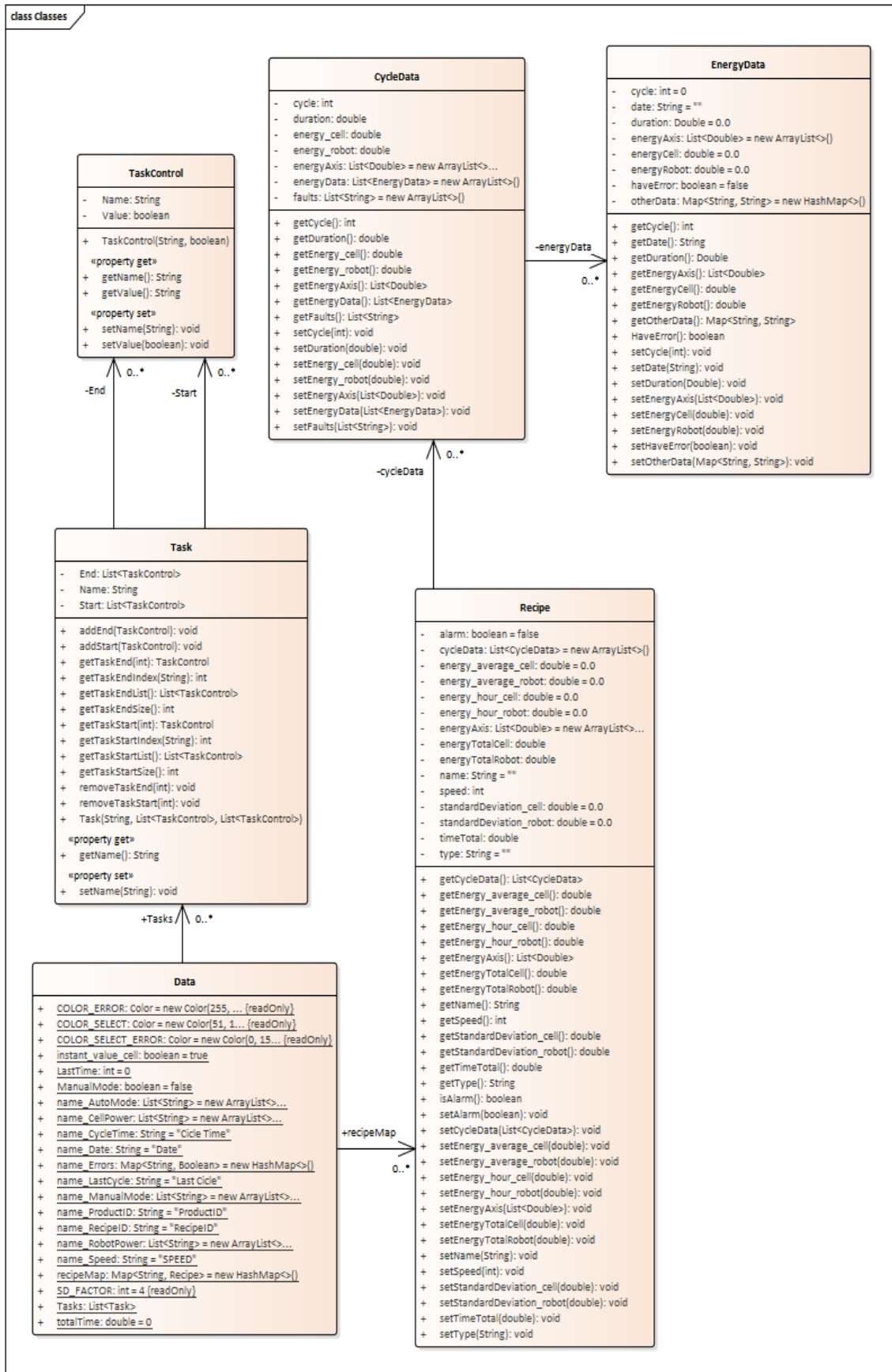


Figure 5.9: Class diagram of Energy Manager structure.

### 5.3.4 Alarms

As explained before, if the robot energy usage value of a cycle is lower or higher than the average of the other cycle after applying the standard deviation, the energy usage was not stable.

$$energy > energy_m + SD$$

or

$$energy < energy_m - SD$$

With:

$energy$  = robot energy used in a cycle;

$energy_m$  = average robot energy used in all cycle, for a recipe;

$SD$  = standard deviation;

If one of these conditions is true, the related cycle will be tagged with an alarm. In the HMI, there is a tab where all cycles, of all recipes, tagged with an alarm are displayed, showing all its raw data from the database, and the rows where one or more of the error variables are active, will have a different colour, red. It is possible to filter the cycles that appear in this tab by (un)select the recipes check-boxes.

### 5.3.5 Charts

There is no better way than using charts to show the energy usage related to multiple cycles or during a time period. The Energy Manager application is equipped with three types of charts: linear chart, bar chart and pie chart.

The pie chart is used to show the time usage between the robot automatic and manual mode. The bar chart is used to show the total energy usage by the robot between all recipes searched. In the bar chart, by selecting one of the recipes, it is possible to view the total energy usage divided by each engine of the robot. There are two possible views for the linear chart: energy per cycle or energy per time. The energy per cycle chart shows the energy usage for each cycle, for all recipes, while the energy per time chart shows the energy usage, gathering the data in ten seconds groups. It is possible to draw the linear chart showing the robot energy or the cell energy. It is also possible to view the following information associated to a linear chart, for each recipe:

- RecipeID - recipe identifier;
- ProductID - product identifier;
- Speed - robots velocity;

- [R] Energy (Wh) - total robot energy;
- [R] Energy/Cycle (Wh) - average robot energy per cycle;
- Standard Deviation - standard deviation of the total robot energy;
- Total Time (hh:mm:ss) - total duration;
- Time/Cycle (hh:mm:ss) - average duration per cycle;
- Energy/Hour (Wh) -
- Total Cycles - total number of cycles;
- [C] Energy (Wh) - total cell energy;
- [C] Energy/Cycle (Wh) - average cell energy per cycle;
- Cycles with error - cycle numbers containing errors;
- Alarms - energy usage anomaly found.

In the figure 5.10 there is the sequence diagram of a chart creation.



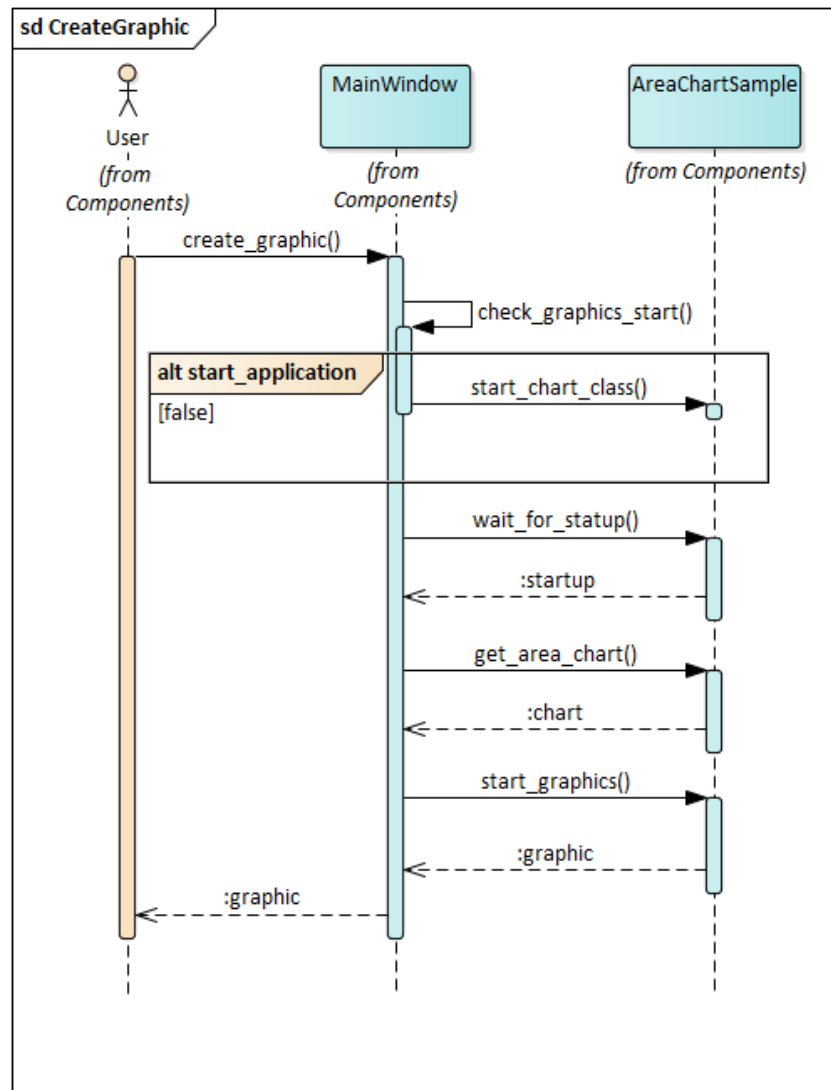


Figure 5.10: Sequence diagram of Energy Manager graphics creation.

### 5.3.6 Tasks

Usually, it is possible to split a robotic cell process into smaller processes. These smaller processes are called tasks. Information regarding the execution state of these tasks are comprised into the industrial robotic cell data. So, if that data is stored into the database, the energy usage of the robot and cell can be analysed for each task. For this, the first step is adding a task. In the figure 5.11 there is the sequence diagram of creating a new task.

To create a task, not only an unique name needs to be set, but also 2 lists of variables from the database: start and end variables. Associated to a variable there is also its value. Once all the start variables values match the entries from the database, all the following data will be associated to that task, until the end variables are met. After the data association is completed, the data analysis, like the one explained in the sub-section 5.3.3, is applied. Upon the data analysis is completed, the results can be display by linear charts,

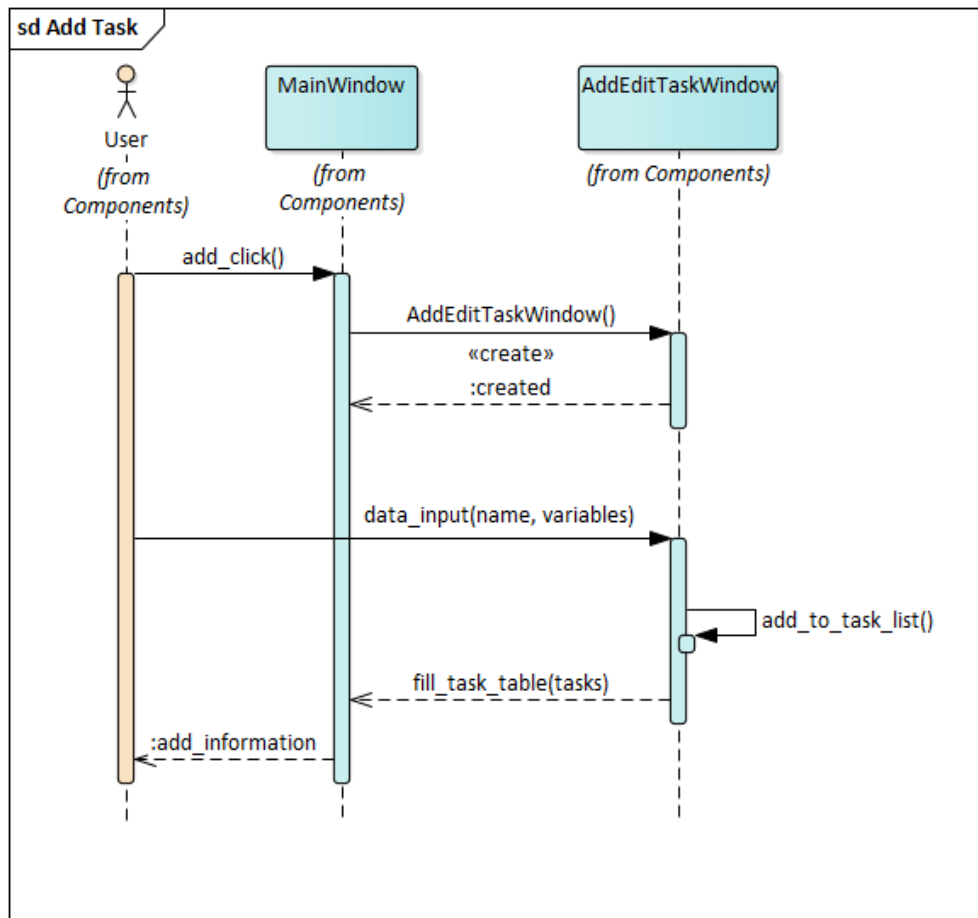


Figure 5.11: Sequence diagram of adding tasks.

like those from 5.3.5. It is also possible to edit and remove existing tasks. In the figure 5.12 there is the sequence diagram of editing, removing or creating a new chart for a select task.

Editing a task allows to change the start and end variables names and values.

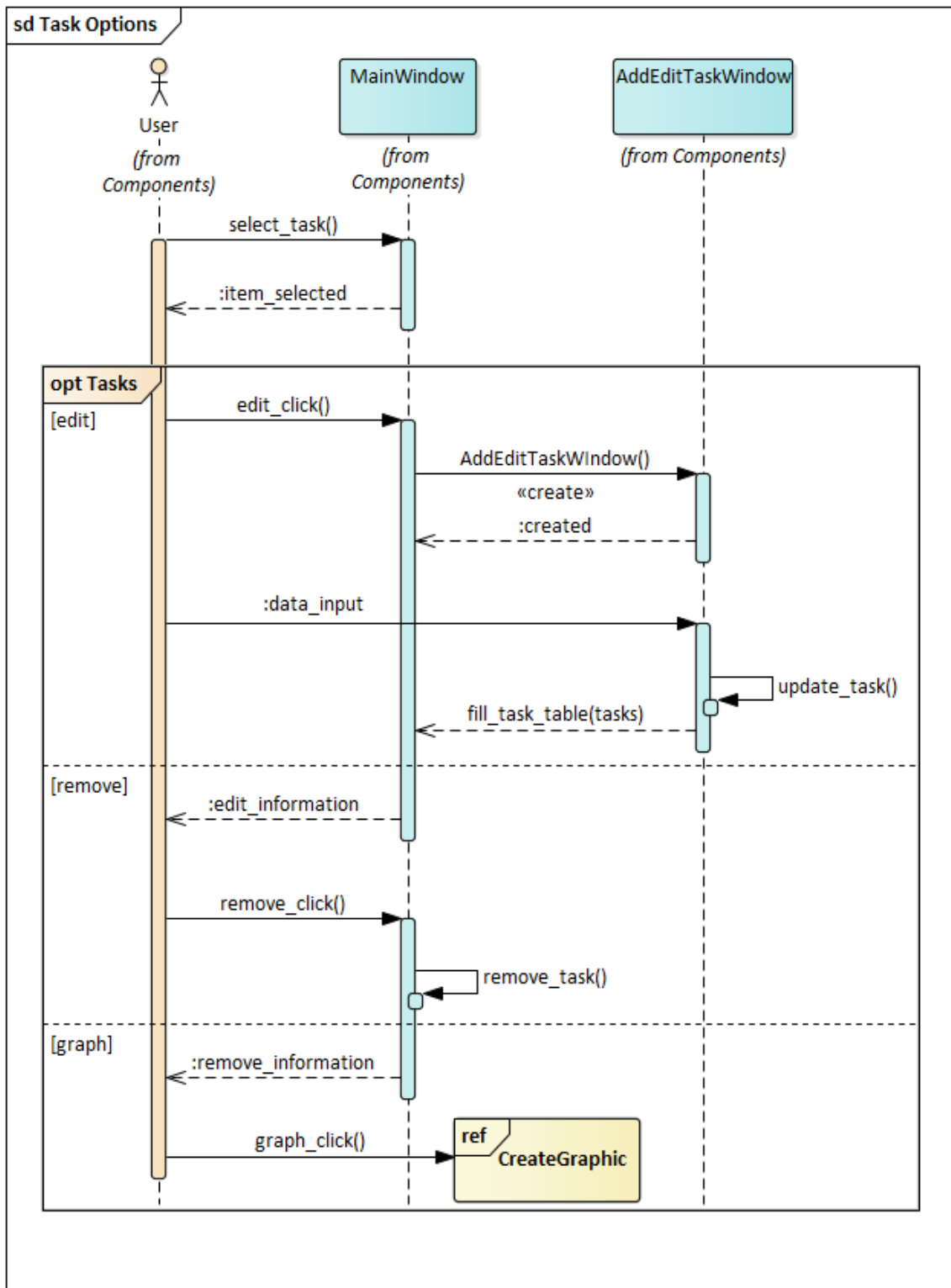


Figure 5.12: Sequence diagram of edit, remove and graphic creation for tasks.

### 5.3.7 Suggestions

A suggestion will need as inputs the number of products to be executed (cycles) and the time period to have those products finished. After this, the Energy Manger will calculate a forecast for the energy usage to execute the all products and the respective duration, using the average values, for each recipe, calculated from the data analysis, previously executed. All recipes shall be listed, sorted by energy usage in ascending order, but only if the execution time can be achieved. In figure 5.13 is shown the sequence diagram of requesting a suggestion and getting the result.

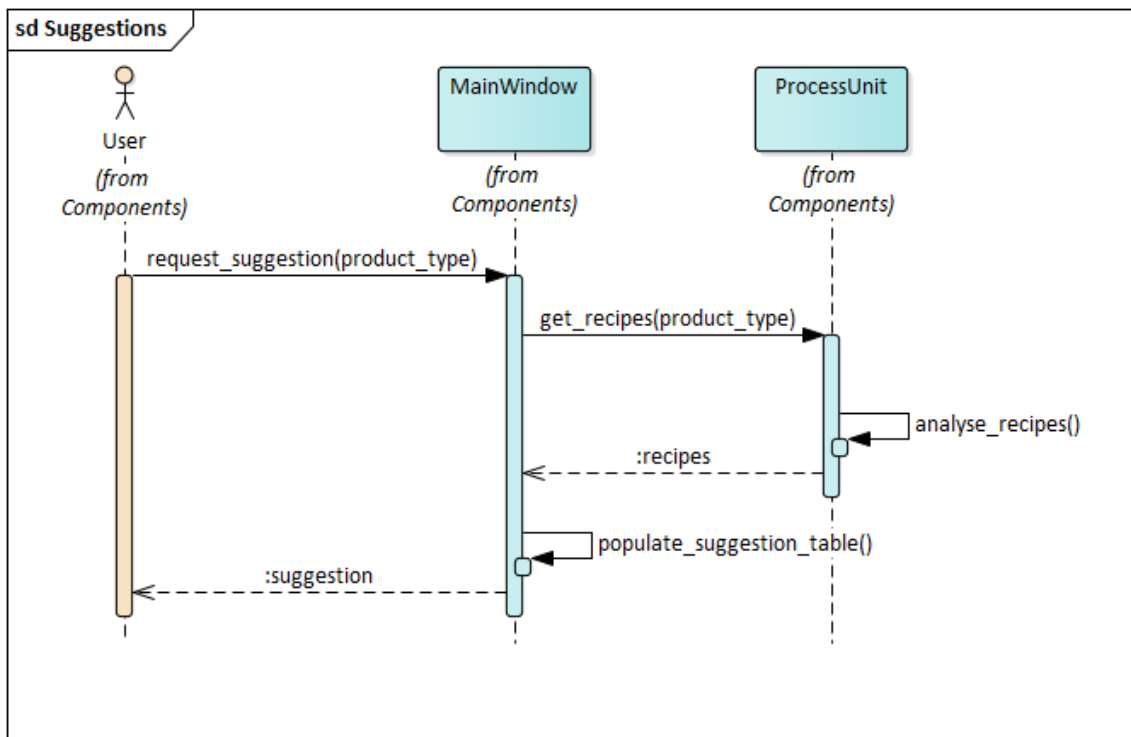


Figure 5.13: Sequence diagram of Energy Manager suggestions.

## VALIDATION

This thesis was tested and validated using two industrial robotic cells, owned by Introsys. Each cell was developed and configured for different OEM (Original Equipment Manufacturer) standards (programming and configurations methods/rules), but both of them execute the same process.

### 6.1 Introsys Robotic Cells

The industrial robotic cells simulate the welding process of a car side spar. Each cell comprises an industrial robot, that is responsible for carrying the spar, from the loading station to the welding station, going through all welding spots pre-programmed, returning to the starting station, to unload the spar. There is also a PLC (Programmable Logic Controller), which is the brain of the whole process, orchestrates the different components of the robotic cell (robot, security, operator). The robot have its own control unit, responsible for executing pre-programmed tasks, when ordered by the PLC. Every robotic cell have security areas, delimited and controlled by security components, such as barriers, scanners and doors. These components ensure that there is no obstacle in the robot working area. If the security area is violated, by the operator, hardware crash or security failure, the robotic cell enters an error state and the robot will stop immediately. After this, and some safety procedures completed, the robot will resume its execution. The last component in this robotic cell is an energy module, that allows the extraction of the energy usage of the entire cell.

#### 6.1.1 Robotic cell - Ford standard

This industrial robotic cell was build and programmed under the Ford OEM standard, so it can be referred as Ford robotic cell.

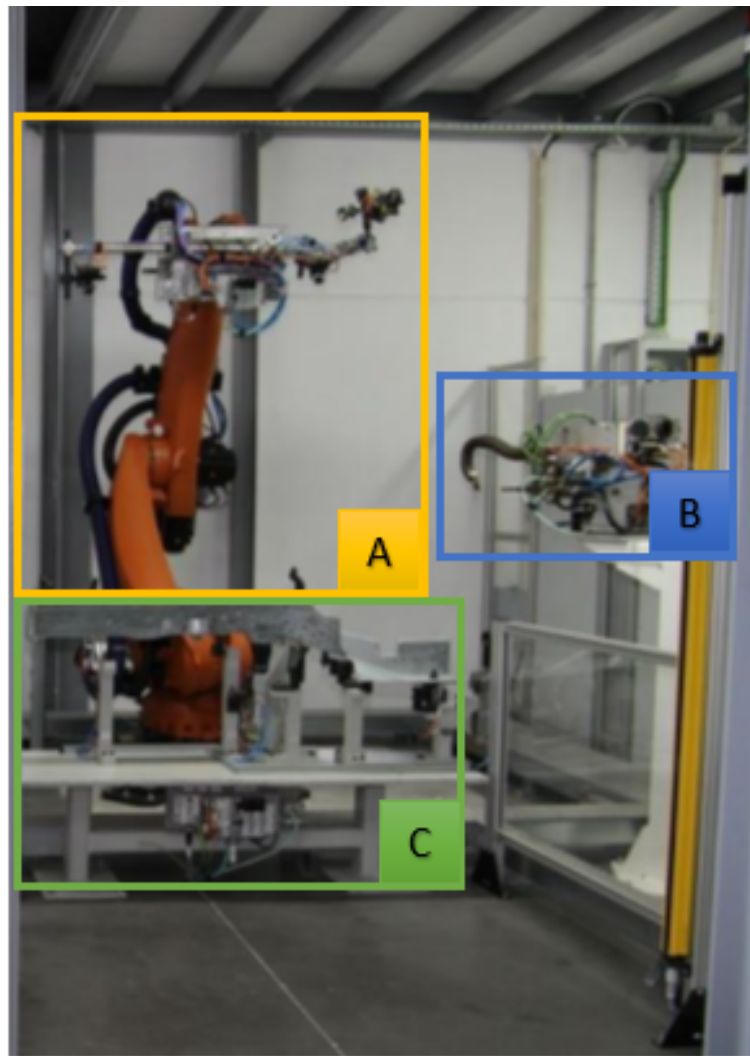


Figure 6.1: Front image of the Ford robotic cell.

The figure 6.1 is picture of the Ford robotic cell. There, it is possible to view the robot (A), equipped with the needed gripper to grab the spar from the loading station (C) and move it until the welding station (B). In the figure 6.2 is an aerial view of the cell.

The table 6.1 shows all the components comprised by the robotic cell compliant with the Ford standard.

Table 6.1: Ford robotic cell components

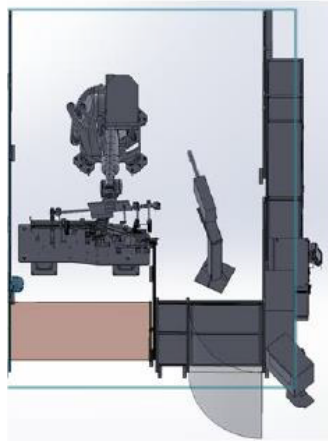


Figure 6.2: Aerial view of the Ford robotic cell.

Component	Manufacturer	Type	Communication Interface
KR C4	KUKA	Robot controller	Proprietary/Ethernet IP
210 R2700 extra	KUKA	Robot	Digital wiring
Gripper	Introsys	Robot gripper	Ethernet IP
PLC	Allen Bradley	Controller	Ethernet IP
AB Safety I/O	Allen Bradley	I/O	Ethernet IP
Panel View Plus 1250	Allen Bradley	HMI	Ethernet IP
Loading Station (Tool)	Introsys	Sub-assembly	Ethernet IP
Welding Gun	ARO	Joining Unit	Ethernet IP
Scanner PLS3000	SICK	Operation safety	Digital wiring
Light Barrier C4000	SICK	Operation safety	Digital wiring
WAGO 750-493	WAGO	Energy measurement	Ethernet IP

### 6.1.2 Robotic cell - Volkswagen standard

This industrial robotic cell was build and programmed under the Volkswagen OEM standard, so it can be referred as Volkswagen robotic cell.

The figure 6.3 is a picture of the Volkswagen robotic cell. There, it is possible to view the robot (A), equipped with the needed gripper to grab the spar from the loading station (C) and move it until the welding station (B).

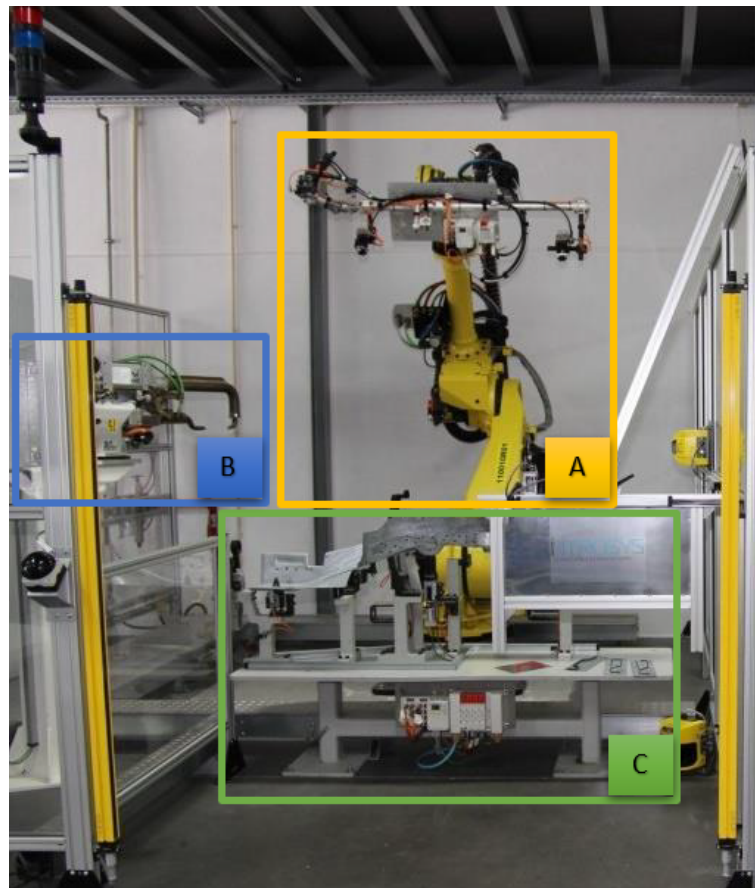


Figure 6.3: Image of the Volkswagen robotic cell.

The table 6.2 shows all the components comprised by the robotic cell compliant with the Volkswagen standard.

Table 6.2: Volkswagen robotic cell components

Component	Manufacturer	Type	Communication Interface
R30-iB	FANUC	Robot controller	Siemens/Profinet IO
R-2000iB 210F	FANUC	Robot	Digital wiring
Gripper	Introsys	Robot gripper	Profinet IO
PLC 319F 3PN DP	Siemens	Controller	Profinet IO
Simatic IPC677C	Siemens	HMI	Profinet IO
Loading Station (Tool)	Introsys	Sub-assembly	Profinet IO
Welding Gun	ARO	Joining Unit	Profinet IO
Scanner PLS3000	SICK	Operation safety	Digital wiring
Light Barrier C4000	SICK	Operation safety	Digital wiring
NZM3-XMC-MB	EATON	Energy measurement	Digital wiring



### 6.1.3 Robotic cells process

In the cells procedure, the spar is placed and removed from the loading station by the operator, as shown in figure 6.4. This station is inside a security area, meaning that the operator can only interact with the station when the robotic cell is not executing, otherwise the it enters in error state. The aid the operator, there are lights indicating if he should or not enter the loading area.

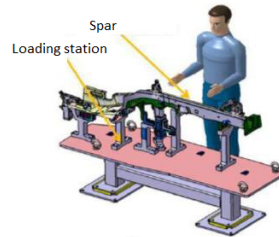


Figure 6.4: Loading station with spar.

The task to be analysed is described as follows:

1. The operator loads the spar into the loading station;
2. The loading station closes the clamps to grab the spar;
3. the robot goes to the loading station and grabs the spar;
4. The loading station opens the clamps.
5. The robot carries the spar to the welding station;
6. The welding procedure starts;
7. The robot moves between all pre-programmed welding spots;
8. The welding procedure ends;
9. The robot carries the spar to the loading station;
10. The loading station closes the clamps and the robot drops the spar;
11. The loading station opens the clamps;
12. The operator unloads the spar.

The process is similar in both cells.

## 6.2 Robotic Cell Changes

To be able to expose the devices data, the communication protocols were updated. The KUKA robot had already installed the KUKA OPC Server, the allows to expose the selected robot variables through a OPC-DA server. As explained in the previous chapter, to ease the communication between the robot and the computer connected to the robotic cell, it was installed an "OPC-UA Wrapper" in the robot, that, just like the name suggests, translates the robot OPC-DA server into an OPC-UA server exposing its variables through OPC-UA protocol. The energy module was never used before, so it was not yet programmed to fulfil its duty. To program it, it was used a proprietary software, "WAGO I/O PRO CAA", allowing the variable of the instant energy usage of the entire cell to be exposed, also using an OPC-DA server. To convert the OPC-DA server into an OPC-UA server it was used exactly the same procedure as in the robot, meaning, an "OPC-UA Wrapper" was installed. The ladder programming language was used to program the module. To exposed the PLCs variables, was installed an OPC-UA server into the line computer, connecting to the PLC through Ethernet/IP communication protocol.

### 6.2.1 Data to retrieve

To be able to analyse the energy consumption of the robotic cell there is a need, not only have to save energy usage data but also the data that describes the task that the cell is executing and other conditions, like robot velocity and if any error occurred during the execution. All this data is not comprise in a single device, so the extraction goes through three different devices: the robot, the energy module and the PLC.

From the PLC, being the unit that controls the whole process, is the device the more data to be retrieved. Owing lots and lots of data, it would not make sense to analyse list every single variable, but after some research and inputs from experienced colleagues from Introsys, I was lead to a small group of variables that would help describe the different tasks of the robot, like Operating mode (manual/automatic), finished tasks, execution time and errors. Given all the data to be retrieved from the PLC, the only relevant data from the robot was its velocity and the current usage of each engine, where these values are the percentage of their maximum values.

The energy module only exposes the energy usage of the entire robotic cell. These variables only contains its instant value, with a refresh rate of 200 milliseconds.

Table 6.3: List of the variables to be extracted from the devices.

Variable	Meaning
Actual Power1-6	Instant energy usage in watts (one variable for each of the six engines)
Auto Mode	True if the robot is in automatic mode
Cycle Time	Value of the current execution's duration
Errors	Multiple variables indicating errors
Home	True if the robot is at home (starting) position
Job1-5	True when a task is completed
Last Cycle	Value of the last cycle's full duration
Manual Mode1-2	True if the robot is in manual mode
POWER	Instant energy usage of the robotic cell in watts
Ready	True if the robot is ready to execute
RecipeID	Process in execution
Speed	Percentage of the maximum velocity of the robot

## 6.3 Trials

The trials developed and shown in this section were made with the purpose of evaluation and validation of the different aspects of this tool, regarding the data analysis of energy consumption of industrial cells. All data used for these test was originated in both industrial robotic cells, described in section 6.1, it was gathered using the Device Adapter and stored into the database.

### 6.3.1 Access and search into the database

#### 6.3.1.1 Description

The trial of access and search in the database was completed by connecting the Energy Manager application and the database. After a successful connection, multiple filters were applied, allowing for different range of data to be selected.

#### 6.3.1.2 Inputs / Resources

As explained in the section 5.2, the MongoDB database used was deployed at the Introsys facilities. For this test, to connect to MongoDB was used the "DB=Introsys" and the "Collection=energyVW", collection where the Volkswagen cell data is stored. The multiple filters used were:

- Date;
- Recipe;
- Speed;

- Cycle time.

### 6.3.1.3 Performance

With the computer running the Energy Manager in the same local network as the database, a successful connection was established. Then, a search was made, without applying any filter, which return all data from the database. The first entry's date was 22/08/2018, around 10h55, the latest was from the same day, at 16h48. The data retrieved is divided between six different recipes. A reference to these recipes can be found in the figure 6.5.

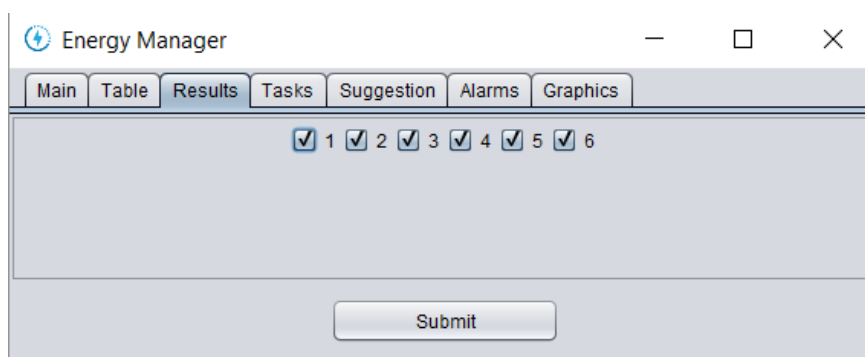


Figure 6.5: Recipes retrieved from the database search, without applying any filter.

In order to validate the date filter, a filter was applied to get the data from the same day, between 12h00 and 14h30, as shown in figure 6.6.

The image shows a "Date" filter configuration panel. It starts with a checked checkbox labeled "Date". Below this, there are two rows of input fields. The first row is labeled "Inicio:" and contains a text box with "22/08/2018", a small "..." button, a time dropdown menu showing "12:00", and a downward arrow. The second row is labeled "Fim:" and contains a text box with "22/08/2018", a small "..." button, a time dropdown menu showing "14:30", and a downward arrow.

Figure 6.6: Date filter example.

With the filter applied, the data retrieved is only divided in three different recipes, as can be seen in figure 6.7.

To verify the recipe filter and in order to validate the usage of multiple filters at the same time, using the same date filter as before, a recipe filter was applied for recipes 3, 4, and 5, as shown in figure 6.8.

Knowing that the previous date filter returned the recipes 2, 3 and 4, like it was shown in figure 6.7, merging it with the recipe filter, the result must be matching recipes, in this

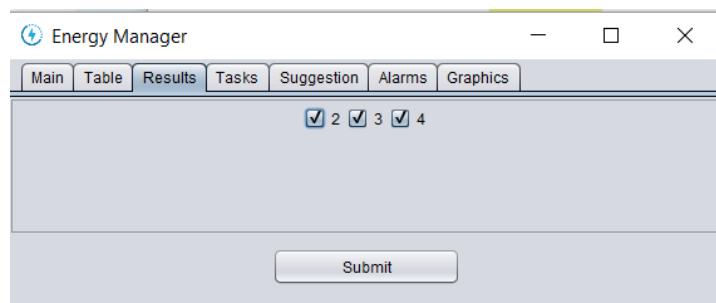


Figure 6.7: Recipes retrieved from the database search, after applying the date filter.

The screenshot shows a filter configuration panel. It has a checked checkbox for "Date". Below it, there are two rows of input fields. The first row is for "Inicio:" with a date field containing "22/08/2018" and a time dropdown menu set to "12:00". The second row is for "Fim:" with a date field containing "22/08/2018" and a time dropdown menu set to "14:30". Below these, there is a checked checkbox for "RecipeID" with a text input field containing "3;4;5".

Figure 6.8: Date and Recipe filter example.

case, the recipes 3 and 4, as it is presented in figure 6.9.

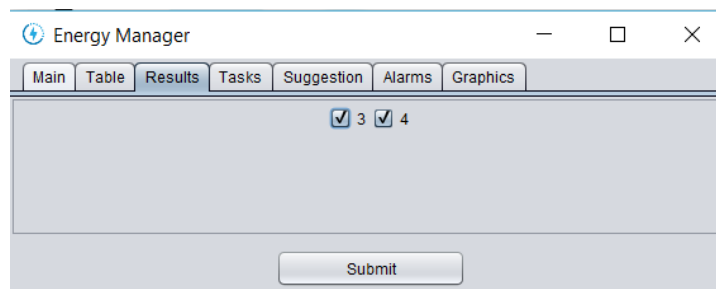


Figure 6.9: Recipes retrieved from the database search, after joining the date and recipe filters.

To validate the speed filter, a search was made for data where the execution velocity was higher than 50, as shown in figure 6.10.

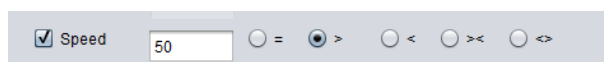
A screenshot of a filter control interface. It features a checked checkbox labeled "Speed", a text input field containing the value "50", and a set of radio buttons for comparison operators: "=", ">", "<", ">>", and "<<". The ">" radio button is selected.

Figure 6.10: Speed filter example.

Since the data stored in the database is comprised with the velocities of 75, 62, 50, 38, 30 and 21, the expected result is the recipes where speed was 75 and 62, as it is represented in figure 6.11.

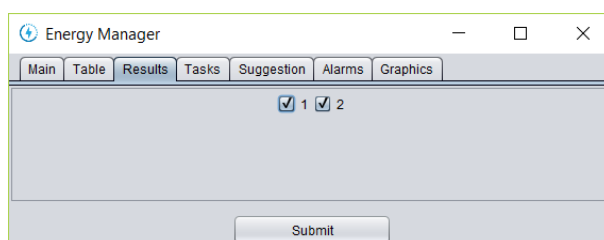


Figure 6.11: Recipes retrieved from the database search, after applying a speed filter.

At last, in order to validate the cycle time filter, a filter for data where the cycle time was lower than 40 000 milliseconds was set, as it is demonstrated in figure 6.12.

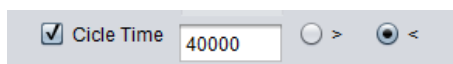
A screenshot of a filter control interface. It features a checked checkbox labeled "Cycle Time", a text input field containing the value "40000", and a set of radio buttons for comparison operators: ">" and "<". The "<" radio button is selected.

Figure 6.12: Cycle time filter example.

This search returned two different recipes, "1" and "2". Regarding recipe "2", the search returned only one cycle, since all other cycle for that recipe had higher cycle time. This result is shown in figure 6.13.

Energy Manager

Main Table Results Tasks Suggestion Alarms Graphics

1  2

Submit

R_ID	Cycle	Speed	[C] En...	[C] SD	[C] Av...	[R] En...	[R] SD	[R] Av...	Duration	Err...
1	23	75	213.53	27.55	210.71	6.19	0.82	6.18	00:00:39	□
1	24	75	214.75	27.55	210.71	6.32	0.82	6.18	00:00:39	□
1	25	75	213.8	27.55	210.71	6.19	0.82	6.18	00:00:39	□
1	26	75	215.03	27.55	210.71	6.32	0.82	6.18	00:00:39	□
1	27	75	214.08	27.55	210.71	6.34	0.82	6.18	00:00:39	□
1	28	75	215.31	27.55	210.71	6.22	0.82	6.18	00:00:39	□
1	29	75	214.36	27.55	210.71	6.27	0.82	6.18	00:00:39	□
1	30	75	214.5	27.55	210.71	6.2	0.82	6.18	00:00:39	□
1	31	75	214.63	27.55	210.71	6.23	0.82	6.18	00:00:39	□
1	32	75	215.87	27.55	210.71	6.31	0.82	6.18	00:00:39	□
1	33	75	216.0	27.55	210.71	6.2	0.82	6.18	00:00:39	□
1	34	75	216.15	27.55	210.71	6.25	0.82	6.18	00:00:39	□
1	35	75	216.28	27.55	210.71	6.34	0.82	6.18	00:00:39	□
1	36	75	215.34	27.55	210.71	6.28	0.82	6.18	00:00:39	□
1	37	75	216.56	27.55	210.71	6.3	0.82	6.18	00:00:39	□
1	38	75	216.71	27.55	210.71	6.21	0.82	6.18	00:00:39	□
1	39	75	215.74	27.55	210.71	6.26	0.82	6.18	00:00:39	□
1	40	75	218.09	27.55	210.71	6.28	0.82	6.18	00:00:39	□
1	41	75	216.02	27.55	210.71	6.24	0.82	6.18	00:00:39	□
1	42	75	217.26	27.55	210.71	6.33	0.82	6.18	00:00:39	□
1	43	75	217.4	27.55	210.71	6.3	0.82	6.18	00:00:39	□
1	44	75	218.64	27.55	210.71	6.54	0.82	6.18	00:00:39	□
1	45	75	216.58	27.55	210.71	6.22	0.82	6.18	00:00:39	□
1	46	75	216.72	27.55	210.71	6.28	0.82	6.18	00:00:39	□
1	47	75	216.87	27.55	210.71	6.32	0.82	6.18	00:00:39	□
1	48	75	218.1	27.55	210.71	6.87	0.82	6.18	00:00:39	□
1	49	75	216.04	27.55	210.71	7.49	0.82	6.18	00:00:39	□
2	0	62	217.27	0.0	217.27	7.4	0.0	7.4	00:00:39	□

Figure 6.13: Recipes retrieved from the database search, after applying a cycle time filter.

## **6.3.2 Analysis and display of results**

### **6.3.2.1 Description**

After accessing the data, it is analysed and organised in multiple cycles and recipes.

### **6.3.2.2 Inputs / Resources**

To test and validate the analysis and display of results, the data used regarding the Volkswagen cell is stored under "DB = Introsys", "Collection = energyVW" and "Collection = vw", and the data regarding the Ford cell is stored under "DB = Introsys", "Collection = ford".

### **6.3.2.3 Performance**

After a search is completed, in the collection "energyVW", the data analysis algorithm starts automatically. This algorithm was explained previously, in subsection 5.3.3. An overview of the result is displayed in figure 6.14.

When the algorithm process finishes, all data is organised in cycles, which will be associated to a recipe. The energy values for the cell and robot are displayed along side with its averages values and standard deviation values, for each recipe. It is also shown the execution time of each cycle and its average.



Energy Manager

Main Table Results Tasks Suggestion Alarms Graphics

1  2  3  4  5  6

Submit

R_ID	Cycle	Speed	[C] Energy	[C] SD	[C] Avera...	[R] Energy	[R] SD	[R] Avera...	Duration	Errors
1	23	75	213.53	2.65	214.34	6.19	0.22	6.28	00:00:39	0
1	24	75	214.75	2.65	214.34	6.32	0.22	6.28	00:00:39	0
1	25	75	213.8	2.65	214.34	6.19	0.22	6.28	00:00:39	0
1	26	75	215.03	2.65	214.34	6.32	0.22	6.28	00:00:39	0
1	27	75	214.08	2.65	214.34	6.34	0.22	6.28	00:00:39	0
1	28	75	215.31	2.65	214.34	6.22	0.22	6.28	00:00:39	0
1	29	75	214.36	2.65	214.34	6.27	0.22	6.28	00:00:39	0
1	30	75	214.5	2.65	214.34	6.2	0.22	6.28	00:00:39	0
1	31	75	214.63	2.65	214.34	6.23	0.22	6.28	00:00:39	0
1	32	75	215.87	2.65	214.34	6.31	0.22	6.28	00:00:39	0
1	33	75	216.0	2.65	214.34	6.2	0.22	6.28	00:00:39	0
1	34	75	216.15	2.65	214.34	6.25	0.22	6.28	00:00:39	0
1	35	75	216.28	2.65	214.34	6.34	0.22	6.28	00:00:39	0
1	36	75	215.34	2.65	214.34	6.28	0.22	6.28	00:00:39	0
1	37	75	216.56	2.65	214.34	6.3	0.22	6.28	00:00:39	0
1	38	75	216.71	2.65	214.34	6.21	0.22	6.28	00:00:39	0
1	39	75	215.74	2.65	214.34	6.26	0.22	6.28	00:00:39	0
1	40	75	218.09	2.65	214.34	6.28	0.22	6.28	00:00:39	0
1	41	75	216.02	2.65	214.34	6.24	0.22	6.28	00:00:39	0
1	42	75	217.26	2.65	214.34	6.33	0.22	6.28	00:00:39	0
1	43	75	217.4	2.65	214.34	6.3	0.22	6.28	00:00:39	0
1	44	75	218.64	2.65	214.34	6.54	0.22	6.28	00:00:39	0
1	45	75	216.58	2.65	214.34	6.22	0.22	6.28	00:00:39	0
1	46	75	216.72	2.65	214.34	6.28	0.22	6.28	00:00:39	0
1	47	75	216.87	2.65	214.34	6.32	0.22	6.28	00:00:39	0
1	48	75	218.1	2.65	214.34	6.87	0.22	6.28	00:00:39	0
1	49	75	216.04	2.65	214.34	7.49	0.22	6.28	00:00:39	0
2	0	62	238.23	2.8	243.03	8.26	0.09	8.31	00:00:43	0
2	1	62	239.49	2.8	243.03	8.15	0.09	8.31	00:00:43	0
2	2	62	238.54	2.8	243.03	8.26	0.09	8.31	00:00:43	0
2	3	62	238.69	2.8	243.03	8.03	0.09	8.31	00:00:43	0
2	4	62	239.95	2.8	243.03	8.31	0.09	8.31	00:00:43	0
2	5	62	240.11	2.8	243.03	8.26	0.09	8.31	00:00:43	0
2	6	62	240.25	2.8	243.03	8.3	0.09	8.31	00:00:43	0
2	7	62	239.31	2.8	243.03	8.23	0.09	8.31	00:00:43	0
2	8	62	239.45	2.8	243.03	8.41	0.09	8.31	00:00:43	0
2	9	62	239.62	2.8	243.03	8.26	0.09	8.31	00:00:43	0
2	10	62	240.87	2.8	243.03	8.18	0.09	8.31	00:00:43	0
2	11	62	239.92	2.8	243.03	8.33	0.09	8.31	00:00:43	0
2	12	62	241.18	2.8	243.03	8.31	0.09	8.31	00:00:43	0
2	13	62	241.33	2.8	243.03	8.3	0.09	8.31	00:00:43	0
2	14	62	240.37	2.8	243.03	8.11	0.09	8.31	00:00:43	0

Figure 6.14: Example of the results tab.

The energy usage data of the recipes can also be visualised using charts. An example can be seen in figure 6.15.

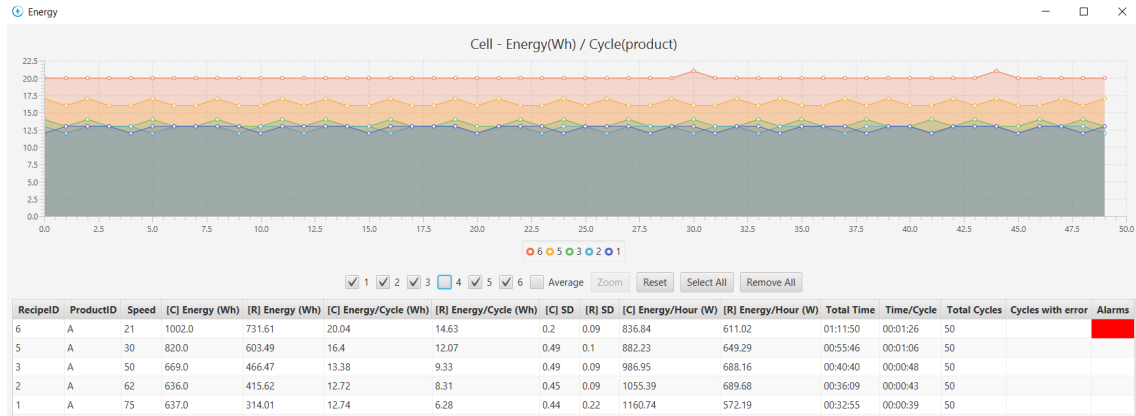


Figure 6.15: Results chart.

In figure 6.16, it is possible to verify that the standard deviation values calculated for the energy usage are low, regarding cell and the robot, in all recipes except for recipe "4", where the standard deviation for the cell energy usage was much higher, but as we can also view, an error was detected in that recipe.

The recipes "4" and "6" were tagged with "Alarms", meaning that at least for one of the cycles the cell energy usage was higher or lower than the average cell energy usage of the recipe adding or removing, respectively, four times the standard deviation calculated. All data regarding the "Alarm" cycles can be viewed in the tab "Alarms". Clicking in the red cell will move the view to the "Alarm" tab.

RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles	Cycles with error	Alarms
6	A	21	1002.0	731.61	20.04	14.63	0.2	0.09	836.84	611.02	01:11:50	00:01:26	50		
5	A	30	820.0	603.49	16.4	12.07	0.49	0.1	882.23	649.29	00:55:46	00:01:06	50		
3	A	50	669.0	466.47	13.38	9.33	0.49	0.09	986.95	688.16	00:40:40	00:00:48	50		
2	A	62	636.0	415.62	12.72	8.31	0.45	0.09	1055.39	689.68	00:36:09	00:00:43	50		
1	A	75	637.0	314.01	12.74	6.28	0.44	0.22	1160.74	572.19	00:32:55	00:00:39	50		
4	A	38	1345.0	539.36	26.9	10.79	86.17	0.08	695.98	279.09	01:55:57	00:02:19	50	35;	

Figure 6.16: Results data of chart table.

In the figures 6.17 and 6.18 there are represented other results obtained from both robotic cells, Volkswagen and Ford respectively. The collection holding this Volkswagen data is collection "vw" and the collection holding the Ford data is collection "ford".

RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles
1	A	75	311.0	132.07	17.28	7.34	0.46	0.07	1217.72	517.13	00:15:19	00:00:51	18
2	A	62	343.0	159.92	17.15	8.0	0.37	0.07	1115.62	520.14	00:18:26	00:00:55	20
3	A	50	338.0	207.87	17.79	10.94	0.42	0.46	1034.85	636.41	00:19:35	00:01:01	19
4	A	38	387.0	253.79	19.35	12.69	0.49	0.09	959.35	629.12	00:24:12	00:01:12	20
5	A	30	409.0	275.77	21.53	14.51	0.51	0.1	915.88	617.53	00:26:47	00:01:24	19
6	A	21	338.0	231.58	26.0	17.81	0.41	0.25	862.47	590.92	00:23:30	00:01:48	13

Figure 6.17: Results table of Volkswagen cell.

Since we are referring to different robotic cells, the internal programs used by each one are not the same, they were programmed using different robot trajectories. However,

RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles
1	A	75	296.06	133.85	17.42	7.87	1.05	0.16	1291.26	583.76	00:13:45	00:00:48	17
2	A	62	353.37	165.91	17.67	8.3	0.41	0.15	1206.94	566.65	00:17:34	00:00:52	20
3	A	50	356.43	178.48	17.82	8.92	0.44	0.11	1108.81	555.22	00:19:17	00:00:57	20
4	A	38	379.45	205.54	18.97	10.28	0.24	0.11	1018.93	551.93	00:22:20	00:01:07	20
5	A	30	411.69	236.91	20.58	11.85	0.2	0.14	959.99	552.44	00:25:43	00:01:17	20
6	A	21	461.39	286.75	24.28	15.09	0.16	0.11	895.21	556.37	00:30:55	00:01:37	19

Figure 6.18: Results table of Ford cell.

they can be considered similar because both represent a welding process, and in this case, the data in the collections "vw" and "ford" was produced using the same number of welding spots. Analysing the results shown in figures 6.17 and 6.18, it is possible to verify that there are six recipes for each cell and that the same velocities were used. Comparing the cycle time between them, for the different recipes, it is possible to conclude that there is a difference of 3 seconds, for the fastest recipe, and a difference of 1 seconds, for the slowest recipe, where the Ford cell is the fastest one. Comparing the energy usage between both cells it is possible to confirm that the values are identical, for the same execution speed. Just as it was verified previously for another collection of data, the standard deviation values calculated are low, between 0.07 and 1.05, validating the consistency of the data retrieved, using the Device Adapter. Analysing the energy usage per hour calculated for the cells, for the same velocity, the Ford cell have higher values, but the difference is lower than 100 watts. As expected, the energy usage is directly connected with the execution velocity of the robot, meaning that, execution with higher speed will result in higher energy usage per hour.

However, observing the robots energy usage, the variations between the results obtained for the different recipes are not those expected, because, ideally the cell energy usage should mimic differences on the robot energy usage, meaning that, an increase in the robots energy should trigger also an increase the cells energy, which was what happened in the Ford cell, but changes of 10 watts in robot triggered changes higher than 100 watts in the cell. But in the Volkswagen cell, the opposite happened, an increase in the robots energy usage, triggered a decrease in the cells energy usage.

After analysing these results, it is possible to state that lowering the execution velocity triggers a higher energy usage producing each product. This happens mainly because of the increase of the cycle time. Analysing the energy per hour registered in the different recipes, for both cells, it is possible to verify that the energy per hour is lower when executing with lower velocities.

### 6.3.3 Error identification

#### 6.3.3.1 Description

This trial aims to validate the different errors during an execution that the Energy Manager is able to detect.

#### 6.3.3.2 Inputs / Resources

To test and validate the error identification feature, the data used was from the Volkswagen cell and is stored under "DB = Introsys", "Collection = energyVW".

#### 6.3.3.3 Performance

Still using the results from figure 6.16, it is possible to verify that at least one error was detected in recipe "4", more particularly in the cycle 35, which was highly the probable cause of the higher standard deviation value. More details regarding this recipe are shown in the tab "Results", exemplified in figure 6.19, with recipe "4" already selected.

R_ID	Cycle	Speed	[C] Energy	[C] SD	[C] Average	[R] Energy	[R] SD	[R] Average	Duration	Errors
4	11	38	15.0	86.17	26.9	10.61	0.08	10.79	00:00:57	
4	12	38	14.0	86.17	26.9	10.83	0.08	10.79	00:00:57	
4	13	38	15.0	86.17	26.9	10.85	0.08	10.79	00:00:57	
4	14	38	15.0	86.17	26.9	10.73	0.08	10.79	00:00:57	
4	15	38	15.0	86.17	26.9	10.81	0.08	10.79	00:00:57	
4	16	38	14.0	86.17	26.9	10.85	0.08	10.79	00:00:57	
4	17	38	15.0	86.17	26.9	10.86	0.08	10.79	00:00:57	
4	18	38	15.0	86.17	26.9	10.89	0.08	10.79	00:00:57	
4	19	38	15.0	86.17	26.9	10.94	0.08	10.79	00:00:57	
4	20	38	14.0	86.17	26.9	10.71	0.08	10.79	00:00:57	
4	21	38	15.0	86.17	26.9	10.72	0.08	10.79	00:00:57	
4	22	38	15.0	86.17	26.9	10.81	0.08	10.79	00:00:57	
4	23	38	14.0	86.17	26.9	10.76	0.08	10.79	00:00:57	
4	24	38	15.0	86.17	26.9	10.74	0.08	10.79	00:00:57	
4	25	38	15.0	86.17	26.9	10.83	0.08	10.79	00:00:57	
4	26	38	14.0	86.17	26.9	10.7	0.08	10.79	00:00:57	
4	27	38	15.0	86.17	26.9	10.82	0.08	10.79	00:00:57	
4	28	38	15.0	86.17	26.9	10.83	0.08	10.79	00:00:57	
4	29	38	15.0	86.17	26.9	10.81	0.08	10.79	00:00:57	
4	30	38	15.0	86.17	26.9	10.85	0.08	10.79	00:00:57	
4	31	38	14.0	86.17	26.9	10.88	0.08	10.79	00:00:57	
4	32	38	15.0	86.17	26.9	10.8	0.08	10.79	00:00:57	
4	33	38	15.0	86.17	26.9	10.75	0.08	10.79	00:00:57	
4	34	38	14.0	86.17	26.9	10.84	0.08	10.79	00:00:57	
4	35	38	624.0	86.17	26.9	10.93	0.08	10.79	01:09:04	[E_scanner_ok]
4	36	38	15.0	86.17	26.9	10.76	0.08	10.79	00:00:57	
4	37	38	14.0	86.17	26.9	10.83	0.08	10.79	00:00:57	
4	38	38	15.0	86.17	26.9	10.73	0.08	10.79	00:00:57	
4	39	38	15.0	86.17	26.9	10.89	0.08	10.79	00:00:57	
4	40	38	15.0	86.17	26.9	10.59	0.08	10.79	00:00:57	
4	41	38	15.0	86.17	26.9	10.85	0.08	10.79	00:00:57	
4	42	38	14.0	86.17	26.9	10.74	0.08	10.79	00:00:57	
4	43	38	15.0	86.17	26.9	10.76	0.08	10.79	00:00:57	
4	44	38	15.0	86.17	26.9	10.68	0.08	10.79	00:00:57	
4	45	38	15.0	86.17	26.9	10.86	0.08	10.79	00:00:57	
4	46	38	14.0	86.17	26.9	10.68	0.08	10.79	00:00:57	
4	47	38	15.0	86.17	26.9	10.83	0.08	10.79	00:00:57	
4	48	38	15.0	86.17	26.9	10.73	0.08	10.79	00:00:57	

Figure 6.19: Recipe "4" results.

As it is demonstrated in figure 6.19, the robot energy usage was stable in all cycles, including in the cycle 35, where the error was detected. However, the cells energy usage was also stable, except in the cycle 35. Analysing the cycle time, it was 57 seconds in all cycle, but in the cycle 35 drastically increased to 1 hour, 9 minutes and 4 seconds

(01:09:04). Observing the column "Errors" in the line of the cycle 35, we can check that error occurred was "E\_scanner\_ok", which means that one of the security areas of the robotic cell was violated and no-one cleared it for more than one hour.

In figure 6.20 is shown the raw data regarding recipe 35. Looking into the columns regarding the robot axis energy usage (the columns that start with "Actual\_"), it is possible to verify that right after the error occurs, about 800 milliseconds after, the robot energy usage goes to zero, meaning that once the cell is in error state, the axis will not use any energy. This is the reason why the value of the standard deviation regarding the robots energy usage did not suffer any change, even with the extended duration on that cycle. But the same did not happen with the cells energy usage ("POWER" column in figure 6.20), since its value kept increasing even with the robot stop, which is what was expected because every device is still turned on. After the security error was cleared, the execution continued without any problem, since the following cycle did not have any error.

R_auto...	EB_door	All_em...	E_air_ok	E_tem...	Actual_...	SPEED	Actual_...	Actual_...	Actual_...	E_barri...	E_scan...	Actual_...	Actual_...	Recipe...	POWER
true	false	true	true	true	-43	38	50	-36	18	true	true	21	4	4	21677
true	false	true	true	true	-43	38	73	-21	0	true	true	20	4	4	21677
true	false	true	true	true	-48	38	55	-31	-15	true	true	23	1	4	21677
true	false	true	true	true	-65	38	48	-24	-17	true	true	14	-5	4	21678
true	false	true	true	true	-33	38	60	-30	-23	true	false	24	4	4	21678
true	false	true	true	true	-46	38	51	-26	-38	true	false	18	-3	4	21678
true	false	true	true	true	-19	38	56	-52	-22	true	false	34	2	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21678
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21679
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21679
true	false	true	true	true	0	38	0	0	0	true	false	0	0	4	21679

Figure 6.20: Cycle 35 raw data, regarding recipe "4".

Using a different data source to create the results displayed in figure 6.21, it is possible to verify that the Energy Manager can detect multiple errors, in different cycles.

R...	Cy...	Spe...	[C] Ene...	[C] ...	[C] Avera...	[R] Ene...	[R] ...	[R] Average	Duration	Errors
4	13	38	15.0	0.22	14.95	10.35	0.07	10.37	00:00:57	[]
4	14	38	15.0	0.22	14.95	10.32	0.07	10.37	00:00:57	[]
4	15	38	15.0	0.22	14.95	10.46	0.07	10.37	00:00:57	[]
4	16	38	15.0	0.22	14.95	10.32	0.07	10.37	00:00:57	[]
4	17	38	15.0	0.22	14.95	10.37	0.07	10.37	00:00:57	[]
4	18	38	15.0	0.22	14.95	10.32	0.07	10.37	00:00:57	[]
4	19	38	15.0	0.22	14.95	10.46	0.07	10.37	00:00:57	[]
5	0	30	17.0	92....	38.05	11.84	0.33	12.01	00:01:06	[]
5	1	30	16.0	92....	38.05	11.93	0.33	12.01	00:01:07	[]
5	2	30	17.0	92....	38.05	11.94	0.33	12.01	00:01:07	[]
5	3	30	429.0	92....	38.05	12.07	0.33	12.01	00:46:13	[E_barrier_ok, E_scanner_ok]
5	4	30	20.0	92....	38.05	12.82	0.33	12.01	00:01:21	[E_scanner_ok, E_barrier_ok]
5	5	30	16.0	92....	38.05	12.01	0.33	12.01	00:01:07	[]
5	6	30	23.0	92....	38.05	12.96	0.33	12.01	00:01:41	[All_door_ok, Ready, E_door_ok, EB_door, All_em_ok, EB_hmi]
5	7	30	19.0	92....	38.05	12.25	0.33	12.01	00:01:17	[All_em_ok, Ready]
5	8	30	20.0	92....	38.05	12.0	0.33	12.01	00:01:29	[E_air_ok, Ready]
5	9	30	17.0	92....	38.05	11.85	0.33	12.01	00:01:06	[]
5	10	30	17.0	92....	38.05	12.01	0.33	12.01	00:01:07	[]
5	11	30	16.0	92....	38.05	11.7	0.33	12.01	00:01:07	[]
5	12	30	17.0	92....	38.05	11.71	0.33	12.01	00:01:07	[]
5	13	30	17.0	92....	38.05	11.92	0.33	12.01	00:01:06	[]
5	14	30	16.0	92....	38.05	11.93	0.33	12.01	00:01:07	[]
5	15	30	17.0	92....	38.05	11.85	0.33	12.01	00:01:07	[]
5	16	30	17.0	92....	38.05	11.84	0.33	12.01	00:01:07	[]

Figure 6.21: Other results with errors.

### 6.3.4 Tasks detection and analysis

#### 6.3.4.1 Description

This trial consisted on validating task feature of the Energy Manager. To be able to configure this feature, the user must have some insight into the cells procedures and data stored into the database.

#### 6.3.4.2 Inputs / Resources

To test and validate this feature, the data used was from the Ford cell and is stored under "DB = Introsys", "Collection = energyFord". Knowing the cells procedure, as explained in sub-section 6.1.3, the tasks analysed were "Pick", "Weld" and "Drop".

#### 6.3.4.3 Performance

Knowing the process used to generate the data, three tasks were created. The first was "Pick", which is the procedure part the goes from the start, grabs the car part and returns to the starting position. Then there was "Weld", which consists on going to the welding station, welding the spots and returning to the starting position. Finally the "Drop" that represents going to the loading station, releasing the car part and returning to the starting position. For each one of these tasks, some variable needed to be set to be able to detect them. Regarding "Pick", the variables selected can be seen in figure 6.22.

The "Home" variable is true when the robot is at the starting position, "Job1" represents the picking action on the robot, so it goes true after the car part is grabbed, "Job2" is true when the welding is completed and "Job5" is true when the car part is released.

For the "Weld", also four starting variables were set:

- Home as True;

Task Name	Start	Value	End	Value
Pick	Home	True	Job1	True
Weld	Job1	False	Home	True
Drop	Job2	False		
	Job5	False		

Figure 6.22: Task list and "Pick" configuration.

- Job1 as True;
- Job2 as False;
- Job5 as False.

And two ending variables:

- Home as True;
- Job2 as True;

And finally for "Drop":

- Home as True;
- Job1 as True;
- Job2 as True;
- Job5 as False.

And two ending variables:

- Home as True;
- Job5 as True;

After creating a task, it is possible to view the energy usage of the cell and robot by chart. The charts display is exactly the same as those viewed previously, in figure 6.15, where all displayed data is related only to the selected task, including the errors. In the figure 6.23 is presented the data of the whole process. Regarding recipe "1", there were errors in the cycles [9, 10, 15, 16], and in the cycle [5] for the recipe "6", and a standard deviation calculated for the cell energy usage of 29.74 and 38.97, respectively.

## CHAPTER 6. VALIDATION



RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles	Cycles with error	Alarms
1	A	75	534.71	164.12	26.74	8.21	29.74	1.69	744.41	228.48	00:43:05	00:02:09	20	9;10;15;16;	
2	A	62	353.37	165.91	17.67	8.3	0.41	0.15	1206.94	566.65	00:17:34	00:00:52	20		
3	A	50	356.43	178.48	17.82	8.92	0.44	0.11	1108.81	555.22	00:19:17	00:00:57	20		
4	A	38	379.45	205.54	18.97	10.28	0.24	0.11	1018.93	551.93	00:22:20	00:01:07	20		
5	A	30	411.69	236.91	20.58	11.85	0.2	0.14	959.99	552.44	00:25:43	00:01:17	20		
6	A	21	659.99	302.27	33.0	15.11	38.97	0.13	672.37	307.94	00:58:53	00:02:56	20	5;	

Figure 6.23: Results table of the complete process.

In figure 6.24 are displayed the results regarding the task "Pick", where errors were detected in the cycles [10, 16] in the recipe "1". It is also possible to verify that the standard deviation values are lower than those from the complete process, specially for the recipe "6", because the error in the recipe did not occur during the "Pick" task.

RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles	Cycles with error	Alarms
1	A	75	128.97	29.67	6.45	1.48	8.51	0.11	641.44	147.55	00:12:03	00:00:36	20	10;16;	
2	A	62	73.55	32.12	3.68	1.61	0.15	0.03	1075.44	469.66	00:04:06	00:00:12	20		
3	A	50	75.91	34.87	3.8	1.74	0.14	0.02	1011.37	464.61	00:04:30	00:00:13	20		
4	A	38	81.62	39.32	4.08	1.97	0.09	0.03	952.14	458.66	00:05:08	00:00:15	20		
5	A	30	88.41	45.0	4.42	2.25	0.08	0.02	905.67	460.95	00:05:51	00:00:17	20		
6	A	21	104.76	56.83	5.24	2.84	0.06	0.02	857.49	465.21	00:07:19	00:00:21	20		

Figure 6.24: Results table of the [Pick] task.

In figure 6.25 are shown the results of the task "Weld", where the errors detected were only in the cycle [9] for the recipe "1" and in the cycle [5] in the recipe "6". After analysing the values of the standard deviation, it is possible to conclude that the error in the cycle [5] had a great impact in the cells energy usage, in the welding task, causing the standard deviation value to be increase. This case validates that the "Tasks" feature of the Energy Manager allows the detection of errors and the isolation of the affected sub-process.


RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles	Cycles with error	Alarms
1	A	75	194.4	96.09	9.72	4.8	0.61	0.16	1388.57	686.37	00:08:24	00:00:25	20	9;	
2	A	62	198.76	101.68	9.94	5.08	0.51	0.14	1296.23	663.08	00:09:12	00:00:27	20		
3	A	50	197.85	108.67	9.89	5.43	0.32	0.1	1174.18	644.91	00:10:06	00:00:30	20		
4	A	38	211.39	126.09	10.57	6.3	0.22	0.11	1066.38	636.07	00:11:53	00:00:35	20		
5	A	30	230.68	145.95	11.53	7.3	0.17	0.12	998.84	631.93	00:13:51	00:00:41	20		
6	A	21	447.91	187.15	22.4	9.36	38.99	0.14	609.25	254.57	00:44:06	00:02:12	20	5;	

Figure 6.25: Results table of the [Weld] task.

In figure 6.26 are presented the results targeting the "Drop" task, where the errors only occurred in the recipe "1", in the cycle [9]. This error also had a great impact in the energy usage of the cell that can be confirm by checking the standard deviation values.


RecipeID	ProductID	Speed	[C] Energy (Wh)	[R] Energy (Wh)	[C] Energy/Cycle (Wh)	[R] Energy/Cycle (Wh)	[C] SD	[R] SD	[C] Energy/Hour (W)	[R] Energy/Hour (W)	Total Time	Time/Cycle	Total Cycles	Cycles with error	Alarms
1	A	75	188.18	27.15	9.41	1.36	28.98	1.59	532.47	76.83	00:21:12	00:01:03	20	9;	
2	A	62	57.63	20.68	2.88	1.03	0.16	0.03	1221.82	438.37	00:02:49	00:00:08	20		
3	A	50	60.15	23.01	3.01	1.15	0.2	0.03	1112.65	425.61	00:03:14	00:00:09	20		
4	A	38	65.09	27.78	3.25	1.39	0.12	0.02	1008.29	430.27	00:03:52	00:00:11	20		
5	A	30	71.79	33.3	3.59	1.66	0.07	0.02	939.06	435.59	00:04:35	00:00:13	20		
6	A	21	87.46	45.04	4.37	2.25	0.06	0.02	870.71	448.38	00:06:01	00:00:18	20		

Figure 6.26: Results table of the [Drop] task.

Observing the figures 6.24, 6.25 and 6.26 and adding the average duration for each cycle of the different tasks, is achieved a difference of 5-6 seconds, comparing with the average duration of the complete process. This difference is cause by the waiting time between complete processes, since when a drop is concluded, the system waits for a user



to go into the station to swap the car part. But, because the welding procedure is only simulated, if the security barrier of the loading area is not interrupted in 5 seconds after the "Drop", the robotic cell will start the process again. This waiting time is not included in any of the tasks referred previously, which is the reason why the error detected in the cycle [15] for the recipe "1" in the complete process, was not detected in any of the individual tasks.

### 6.3.5 Suggestions analysis

#### 6.3.5.1 Description

This trial was performed in order to validate the quality of the suggestions created using the energy data retrieved from the robotic cells.

#### 6.3.5.2 Inputs / Resources

To test and validate the error identification feature, the data used was from the Volkswagen cell and is stored under "DB = Introsys", "Collection = energyVW". It was also needed to select a product type to select only the recipes executed for that product, in this case, product "A". A product quantity to execute and the targeted deadline also needed to be introduced, which were 10 and 30 minutes, respectively.

#### 6.3.5.3 Performance

As observed in the previous cases, six different recipes were analysed. To each one of them is associated a product type and, in this case, all recipes are associated to the same product, "A", since the only difference between them is the execution speed.

When a suggestion is requested, the first component to be verified is the execution time. In this trial, if the execution time for a recipe to execute 10 products is higher than 30 minutes, that recipe is excluded of being a possible option, without analysing its energy usage. If no recipe is able to execute the 10 products in 30 minutes or less, only one option is displayed. That recipe is the one with lower energy usage, but the "bestOption" field will be set as "false". For recipes able to execute 10 products over 30 minutes, in this case, all of them, a forecast for the cell energy usage is calculated and compared between different recipes. The recipe with lower energy usage is tagged with "bestOption". Errors that occur during the data gather and storage can have a great impact in the obtained results, so to deal with this problem, an option was added, allowing for the cycles with errors to be removed from the data analysis results. In figure 6.27 is displayed a suggestion result, where the error cycles were not removed. Observing the column "Time" it is possible to check that all recipes execute the 10 products in less than 30 minutes. In the column "[C]Energy Total" are the values of the forecast for the 10 products execution, values between 176.69 and 329.99 watt-hour, where the lowest value belongs to recipe "2", being tagged with "bestOption".

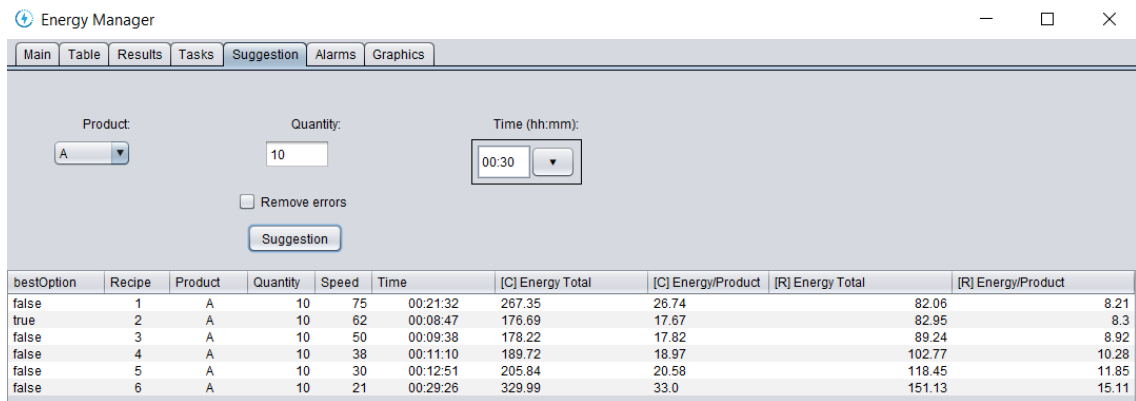


Figure 6.27: Suggestion example, considering the errors occurred during the data gathering.

In figure 6.28 is shown the result of the suggestion, but this time the error cycles were removed from the results. After checking the column "Time" we can verify that execution time for the recipe "1" is lower, comparing with the results of the first suggestion, went from 21 minutes and 32 seconds to 8 minutes and 6 seconds, and the recipe "6" from 29 minutes and 26 seconds to 16 minutes and 16 seconds. A similar impact can be found in the cells energy usage, for the same two recipes, being the recipe "1" suffered the greater difference, since the cells energy usage went from 267.35 to 174.65 watts-hour. This new value for recipe "1" is now the lowest forecast for the execution of 10 products, and it was now tagged with "bestOption", instead of recipe "2". Observing these two examples we can conclude that the errors that occur during the gather and storage phase have a great impact in the energy usage, being its detection extremely important.

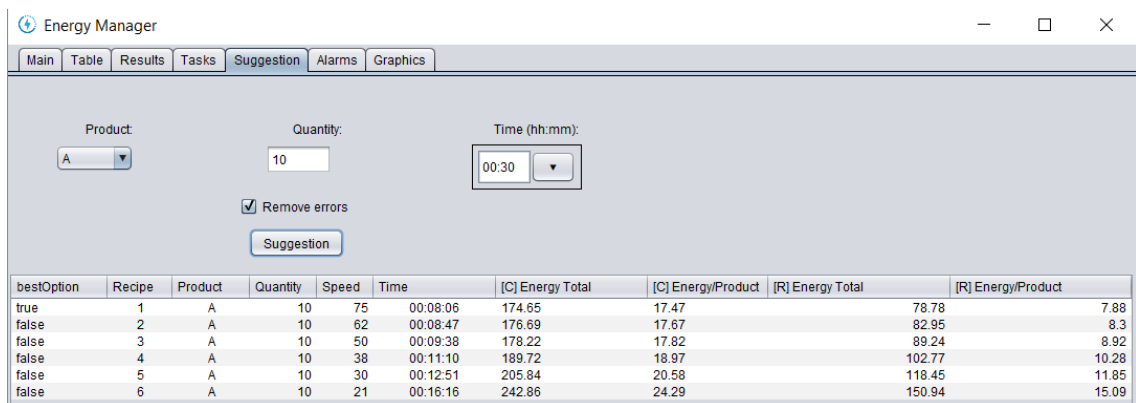


Figure 6.28: Suggestion example, removing the errors occurred during the data gathering.

## 6.4 Other results

### 6.4.1 Impact in the current system

In order to validate if the data gathering process have any impact in the robotic cell behaviour, some executions were performed without using any of the developed components, where the cycle time was registered manually. For the Volkswagen cell, using the same welding process used to generated the data shown in figure 6.17 with a velocity of 75, a cycle time of around 51 seconds was registered, which was the same value calculated by the analysis. In the Ford cell, using the same welding process used to generated the data shown in figure 6.18 also with a velocity of 75, a cycle time of around 48 seconds was registered, the same value calculated by the analysis. These results prove that the Device Adapter data gathering procedure has no impact in the duration of the process.

### 6.4.2 Deliverables and Publications

Regarding the Energy Manager application, it was part of a bigger project, from Portugal2020, that aimed the creation of a RDI Core, at Introsys, targeting the development of several products and services in different areas, including manufacturing, on which, three documents were created, specifying the details and results about the Energy Manager, namely requirements, architecture and trials.

The data and results of the analysis of this solution were used in the writing of the paper (Miranda et al. 2018), related to an European project called OpenMOS, from Horizon2020, where the main goal was developing a common, openly accessible plug-and-produce system. The data presented in that paper associated to "without OpenMOS", in the "Results"section, were retrieved and analyse by the complete solution presented in this thesis. The main goal of that section in the paper was to compare the behaviour of the industrial robotic cells, with and without the OpenMOS technology integrated, since the technology have an executor module implemented making it responsible for the executed processes in the cells. Since the data gathering process presented in this thesis does not have any impact in the execution process, the data retrieved from it is able to translate the standard behaviour of the cells. To clarify, still in (Miranda et al. 2018), there is a section entitle of "Device Adapter"but it is related to a different implementation than the one presented in this thesis.



## CONCLUSION AND FUTURE WORK

The developed solution indicates that it is possible to add monitoring capabilities to existing manufacturing systems without compromising the standard behaviour of a manufacturing robotic cell.

It was also shown that, having access to the required data, it is possible to analyse a manufacturing process and split the process and respective analysis into smaller tasks, being able to determine the energy consumption and errors occurred in each one of them.

With the samples and trials presented, it is possible to state that simply speed variations in the robots execution have a great impact into the duration of a process and the energy used in that process, meaning that, with higher velocities the process executes faster and the energy used to execute that process is lower than executing with lower velocities. The energy variations between velocities is mainly due to changes in the execution time, since the energy consumed per hour does not vary that drastically, for both stations.

The trials performed allowed to conclude that, comparing the energy consumption for each process between the Volkswagen and Ford stations, the Volkswagen station is more energy efficient than the Ford station for higher robot velocities but less efficient when the robot is executing with lower velocities.

By having a modular architecture allows for both applications developed under this thesis to be used separately, which means that if someone already has a data analysis application but does not have the data gathering process implemented, he can simply use the Device Adapter and the database described in the architecture and merge with his

own solution. The same goes if the data storage process is already implemented and the data analysis platform is the one missing, then he can use only the Energy Manager alongside his own applications.

The results of this work were used in the writing of a scientific paper (Miranda et al. 2018) and in the writing of documents containing the details of the development, only regarding the Energy Manager.

For future work, the Device Adapter should be compliant with more communication protocols, besides OPC-UA, allowing it to connect with a greater range of devices, extracting their data. The Energy Manager application should offer more features and capabilities, more related to predictive manufacturing, targeting maintenance. The analysis process should be improved, introducing data mining algorithms. The result handling should also be improved, granting the capability of creating new possible recipes, merging the more efficient sub-procedures (tasks) found in the analysed recipes. Ideally, it should be possible to create completely new tasks to deliver more efficient results for a target process, but for this to be possible, using the industrial robotic cells for welding as an example, the analysis application needed to be aware of the targeted welding spots and their coordinates, in order to provide more efficient robot trajectories. This possible feature would need also to deal with possible collisions, meaning that, the robot trajectories are not always the shorter path between two spots because other component can be in the working area, for example the welding station, and the trajectories are set to avoid those.

## BIBLIOGRAPHY

- Akbar, S. A., N. P. Khawaja, P. R. Brown, R. Tayyeb, J. Bamfo, and K. H. Nicolaidis (2009). "Angiotensin II type 1 and 2 receptors gene polymorphisms in pre-eclampsia and normal pregnancy in three different populations." In: *Acta Obstetrica et Gynecologica Scandinavica* 88.5, pp. 606–611. ISSN: 00016349. DOI: 10.1080/00016340902859307.
- Babiceanu, R. F. and R. Seker (2016). "Big Data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook." In: *Computers in Industry* 81.2015, pp. 128–137. ISSN: 01663615. DOI: 10.1016/j.compind.2016.02.004. URL: <http://dx.doi.org/10.1016/j.compind.2016.02.004>.
- Boselli, A., X. Wang, M. Armenante, L. D'Avino, G. Pisanli, and N. Spinelli (2004). "Study of a multy-layered structure of aerosol vertical distribution over Naples performed during EARLINET project." In: *European Space Agency, (Special Publication) ESA SP 2.561*, pp. 903–906. ISSN: 03796566. DOI: 10.1007/978-3-642-19692-8.
- Buchholz, S. (2011). *Factories of the future*. Vol. 82. 3, pp. 27–29. ISBN: 9789279312380. DOI: 10.2777/29815.
- Cannata, A., S. Karnouskos, and M. Taisch (2009). "Energy efficiency driven process analysis and optimization in discrete manufacturing." In: *IECON Proceedings (Industrial Electronics Conference)*, pp. 4449–4454. ISSN: 1553-572X. DOI: 10.1109/IECON.2009.5414889.
- Cupek, R., M. Drewniak, and D. Zonenberg (2014). "Online energy efficiency assessment in serial production - Statistical and data mining approaches." In: *IEEE International Symposium on Industrial Electronics*, pp. 189–194. DOI: 10.1109/ISIE.2014.6864609.
- Dorofeev, K., C. H. Cheng, M. Guedes, P. Ferreira, S. Profanter, and A. Zoitl (2018). "Device adapter concept towards enabling plug&produce production environments." In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pp. 1–8. ISSN: 19460759. DOI: 10.1109/ETFA.2017.8247570.
- El Maraghy, H. A. (2006). "Flexible and reconfigurable manufacturing systems paradigms." In: *Flexible Services and Manufacturing Journal* 17.4 SPECIAL ISSUE, pp. 261–276. ISSN: 19366590. DOI: 10.1007/s10696-006-9028-7.
- Fan, W. and A. Bifet (2013). "Mining big data." In: *ACM SIGKDD Explorations Newsletter* 14.2, p. 1. ISSN: 19310145. DOI: 10.1145/2481244.2481246. URL: <http://dl.acm.org/citation.cfm?doid=2481244.2481246>.

## BIBLIOGRAPHY

---

- Feng, L., D. Ulutan, and L. Mears (2015). "Energy consumption modeling and analyses in automotive manufacturing final assembly process." In: *2015 IEEE Conference on Technologies for Sustainability, SusTech 2015*, pp. 224–228. DOI: 10.1109/SusTech.2015.7314350.
- Foote, K. D. (2016). *Databases: Relational versus Non-Relational*. URL: <https://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/>.
- Gamarra, C., J. M. Guerrero, and E. Montero (2016). "A knowledge discovery in databases approach for industrial microgrid planning." In: *Renewable and Sustainable Energy Reviews* 60, pp. 615–630. ISSN: 18790690. DOI: 10.1016/j.rser.2016.01.091. URL: <http://dx.doi.org/10.1016/j.rser.2016.01.091>.
- Gyorodi, C., R. Gyorodi, G. Pecherle, and A. Olah (2015). "A comparative study: MongoDB vs. MySQL." In: *2015 13th International Conference on Engineering of Modern Electric Systems, EMES 2015*, pp. 0–5. ISSN: 1051-0117. DOI: 10.1109/EMES.2015.7158433.
- Karnouskos, S., A. W. Colombo, J. L. Lastra, and C. Popescu (2009). "Towards the energy efficient future factory." In: *IEEE International Conference on Industrial Informatics (INDIN)*, pp. 367–371. ISSN: 19354576. DOI: 10.1109/INDIN.2009.5195832.
- Koren, Y., U. Heisel, H. Van Brussel, G. Pritschow, G. Ulsoy, T. Moriwaki, and F. Jovane (1999). "Reconfigurable Manufacturing Systems." In: *CIRP Annals* 48.2, pp. 527–540. ISSN: 00078506. DOI: 10.1016/S0007-8506(07)63232-6.
- Krishnan, S. S., N. Balasubramanian, E. Subrahmanian, V. Arun Kumar, G. Ramakrishna, A. Murali Ramakrishnan, and A. Krishnamurthy (2009). "Machine level energy efficiency analysis in discrete manufacturing for a sustainable energy infrastructure." In: *2009 2nd International Conference on Infrastructure Systems and Services: Developing 21st Century Infrastructure Networks, INFRA 2009*, pp. 1–6. ISSN: 0167-1685. DOI: 10.1109/INFRA.2009.5397871. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5397871>.
- Lasi, H, P Fettke, H.-G. Kemper, T Feld, and M Hoffmann (2014). "Industry 4.0." In: *Business & Information Systems Engineering* 6.4, pp. 239–242. DOI: 10.1007/978-981-13-3384-2\_13.
- Lechevalier, D., A. Narayanan, and S. Rachuri (2015). "Towards a domain-specific framework for predictive analytics in manufacturing." In: *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, pp. 987–995. ISSN: 9781479956661. DOI: 10.1109/BigData.2014.7004332.
- Lee, J., E. Lapira, S. Yang, and A. Kao (2013a). *Predictive manufacturing system - Trends of next-generation production systems*. Vol. 46. 7. IFAC, pp. 150–156. ISBN: 9783902823335. DOI: 10.3182/20130522-3-BR-4036.00107. URL: <http://dx.doi.org/10.3182/20130522-3-BR-4036.00107>.



- Lee, J., E. Lapira, B. Bagheri, and H. a. Kao (2013b). "Recent advances and trends in predictive manufacturing systems in big data environment." In: *Manufacturing Letters* 1.1, pp. 38–41. ISSN: 22138463. DOI: 10.1016/j.mfglet.2013.09.005. URL: <http://dx.doi.org/10.1016/j.mfglet.2013.09.005>.
- Lee, J., B. Bagheri, and H. A. Kao (2015). "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems." In: *Manufacturing Letters* 3, pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001. URL: <http://dx.doi.org/10.1016/j.mfglet.2014.12.001>.
- Lehnhoff, S., S. Rohjans, M. Uslar, and W. Mahnke (2012). "OPC unified architecture: A service-oriented architecture for smart grids." In: *2012 1st International Workshop on Software Engineering Challenges for the Smart Grid, SE-SmartGrids 2012 - Proceedings*, pp. 1–7. ISSN: 1940-4387. DOI: 10.1109/SE4SG.2012.6225723.
- Mahajan, D. and Z. Zong (2018). "Energy efficiency analysis of query optimizations on MongoDB and Cassandra." In: *2017 8th International Green and Sustainable Computing Conference, IGSC 2017*. Vol. 2017-October, pp. 1–6. ISBN: 9781538634707. DOI: 10.1109/IGCC.2017.8323581.
- Miranda, F, R Martins, K Dorofeev, V Gentile, P Ferreira, and M Guedes (2018). "Towards a Common Manufacturing Service Bus to Enable Flexible Plug-and-Produce Automation." In: *ISR 2018; 50th International Symposium on Robotics* Ga 680735, pp. 1–8.
- OPC Foundation (2019a). *OPC Classic*. URL: <https://opcfoundation.org/about/opc-technologies/opc-classic/>.
- (2019b). *OPC Unified Architecture*. DOI: 10.1524/auto.2011.0934. URL: <https://opcfoundation.org/about/opc-technologies/opc-ua/>.
- Palm, F., S. Grüner, J. Pfrommer, M. Graube, and L. Urbas (2015). "Open source as enabler for OPC UA in industrial automation." In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2015-October*. ISSN: 19460759. DOI: 10.1109/ETFA.2015.7301562.
- Peng, T. and X. Xu (2015). "Energy consumption evaluation for sustainable manufacturing: A feature-based approach." In: *Proceedings of the World Congress on Intelligent Control and Automation (WCICA) 2015-March*. March, pp. 2310–2315. DOI: 10.1109/WCICA.2014.7053082.
- Rüßmann, M., M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch (2015). "Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries." In: *Boston Consulting* 62.4, pp. 40–41. ISSN: 00119342. DOI: 10.1007/s12599-014-0334-4.
- Schutt, R. and C. O'Neil (2013). *Doing Data Science*. O'Reilly Media, p. 408. ISBN: 978-1-449-35865-5.
- Sung, W. T. and Y. C. Hsu (2011). "Designing an industrial real-time measurement and monitoring system based on embedded system and ZigBee." In: *Expert Systems with Applications* 38.4, pp. 4522–4529. ISSN: 09574174. DOI: 10.1016/j.eswa.2010.09.126. URL: <http://dx.doi.org/10.1016/j.eswa.2010.09.126>.

## BIBLIOGRAPHY

---

- Swaminathan, S. N. and R. Elmasri (2016). "Quantitative analysis of scalable NoSQL databases." In: *Proceedings - 2016 IEEE International Congress on Big Data, BigData Congress 2016*, pp. 323–326. ISSN: 0160-6689. DOI: 10.1109/BigDataCongress.2016.49.
- Xia, B. S. and P. Gong (2014). "Review of business intelligence through data analysis." In: *Benchmarking* 21.2, pp. 300–311. ISSN: 14635771. DOI: 10.1108/BIJ-08-2012-0050.