FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulation and Intelligent Visualisation Tool for an Investment Casting Manufacturing Process

Ana Beatriz Campos Cruz

MASTER'S DEGREE IN ELECTRICAL AND COMPUTER ENGINEERING

Supervisor: Armando Jorge Miranda de Sousa

Co-Supervisors: Ângela Cardoso and Bernardo Valente

September 4, 2019

Abstract

With present day industries pressing for retrofitting of current machinery into Industry 4.0 ideas, a large effort is put into data production, storage and analysis. However, not all industries are ready for these changes.

This dissertation case studies a foundry company, *Zollern & Comandita*, that uses the lost wax method to produce metal parts. This company is an example where it is difficult to apply the Industry 4.0 ideas. As retrofitting is underway, modelling, simulation and smart data visualisation are proposed as methods to overcome data shortage in quantity and quality.

When this project began, only a low quantity of data was being stored, so it is necessary to study the entire process in order to understand which is the most relevant and critical data to monitor in each phase.

With the mentioned case study, a simulator was developed to generate sample data until the real data from the machines is being stored. To be able to use such real data, it is fundamental to create intelligent software for analysis and visualisation of a growing, but frequently faulty, amount of data, without the quality and quantity adequate for full blown data mining techniques.

The developed data visualisation system is demonstrated to be adapted to the requirements and needs of this company in order to approach full automation ideas. It allows workers and supervisors to know in real time what is happening in the factory, or study the passage of manufacturing orders for a specific area. Data Analysts will, in near future, be able to predict machinery problems, correct issues with slow changing deviations and gather additional knowledge on the implementation of the process itself.

Since the lost wax investment casting process is composed by many phases, during the analysis of the case study company, the most critical phase of the process was identified. Thus, the work was developed with the focus where the company will get more advantage to meet their needs.

With such challenges in mind, this dissertation focuses on the study of the *Zollern & Comandita*, to develop a proper data visualisation tool properly for this company. The recent approaches on the development of applications were taken in count, to make the application as scalable as possible.

Resumo

Com a pressão existente nas indústrias atuais para modernizar os seus processos de produção com as ideias da Industria 4.0, tem sido colocado um grande esforço na produção, armazenamento e análise de dados. No entanto, nem todas as indústrias estão preparadas para estas mudanças.

O caso de estudo desta dissertação é uma empresa de fundição que usa o método de cera perdida para produzir peças de metal precisas. Esta empresa é um exemplo de indústrias onde é difícil aplicar as ideias da Indústria 4.0. À medida que a remodelação está a decorrer, modelagem, simulação e visualização inteligente de dados são propostas como métodos para superar a escassez de dados em quantidade e qualidade.

Neste moemnto são poucos os dados a serem monitorizados, por isso é necessário estudar todo o processo de fabrico a fim de se perceber quais são os dados relevantes e criticos de cada etapa. Com este estudo, é possivel desenvolver um simulador do processo de fabrico a fim de gerar dados, até que os reais estejam a ser armazenados. Para se poder usar esses dados, é fundamental criar *software* inteligente para análise e visualização de uma quantidade crescente, mas frequentemente defeituosa, de dados, sem a qualidade e quantidade adequadas para técnicas de *data mining*.

O sistema de visualização de dados desenvolvido é apresentado de forma a corresponder aos requisitos e necessidades desta empresa, a fim de abordar as ideias de automação completa. Esse sistema de visualização de dados permite que os trabalhadores e supervisores saibam em tempo real o que está a acontecer na fábrica ou que possam estudar a passagem de ordens de produção por uma área específica. Os interpretadores de dados também podem prever problemas mecânicos, corrigir problemas com desvios lentos e coletar conhecimento adicional sobre a implementação do próprio processo.

Uma vez que o método de fundição por cera perdida é constituido por várias etapas, durante a análise da empresa deste caso de estudo, foi identificada a étapa mais critica no processo. Assim, o trabalho foi desenvolvido com foco a onde a empresa obtivesse o melhor proveito face às suas necessidades.

Com este tipo de desafios como foco principal, esta dissertação centra-se no estudo da empresa *Zollern & Comandita*, para desenvolver uma ferramenta de visualização de dados adequada à empresa. Recentes métodos sobre desenvolvimento de aplicações foram tidas em consideração, para fazer com que a aplicação seja o mais escalável possível.

Agradecimentos

Ao meu orientador, Prof. Dr. Armando Sousa, um grande obrigada por todo o apoio e motivação, não só na fase da dissertação, mas também ao longo de todo o meu percurso académico. Obrigada por ter sempre acreditado em mim e por todas as oportunidades que me proporcionou. À minha co-orientadora, Ângela Cardoso, por toda a paciência, pelas horas dedicadas a este trabalho e pelas palavras de motivação nos momentos mais difíceis. Obrigada também pelas boas gargalhadas. Ao meu co-orientador, Bernardo Valente, por todas as coisas que me ensinou com muita paciência. Obrigada por toda a simpatia e por estar sempre disponível para ajudar. Quero agradecer também ao INEGI e à *Zollern & Comandita* a oportunidade para desenvolver este trabalho.

Esta dissertação não se resume só a um semestre, mas sim ao caminho percorrido até chegar aqui. À Márcia, a minha grande companheira de todos os momentos, um grande e muito especial obrigada. Ela foi, sem dúvida, o mais importante que este percurso me deu. Ao Fábio por todo o carinho e apoio ao longo deste percurso. Obrigada por me teres ensinado a dar os primeiros passos no mundo engenharia. Sem a tua ajuda, não teria chegado onde cheguei. E porque também foram muito importantes nesta jornada e vão ficar sempre no meu coração, para o Baltasar, o João e o Renato, um xiii muito apertadinho por todas as brincadeiras, por toda a paciência e por estarem sempre lá.

Por último, mas não menos importante, à minha família. Aos meus pais, por todas as oportunidades que me têm proporcionado, por todo o apoio e dedicação, por estarem sempre lá nos bons momentos, mas também nos menos bons. Sem vocês nada disto tinha sido possível. À Xana e ao Afonso, obrigada por serem quem são. Aquilo que eu sou hoje, devo-o muito a vocês.

Beatriz Cruz

“How wonderful it is that nobody need wait a single moment
before starting to improve the world.”

Anne Frank

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
HTTP	HyperText Transfer Protocol
MTV	Model Template View
MVC	Model View Controller
PO	Production Order
PWA	Progressive Web Application
RWD	Responsive Web Design
SDK	Software Development Kit
SPA	Single Page Application

Chapter 1

Introduction

This chapter introduces the work developed in this dissertation. It presents the context, motivation and the goals that this work consists of. It also describes the structure of this document.

1.1 Context

Nowadays, we are going through the fourth industrial revolution, also known as Industry 4.0. This is a digital revolution, whose main objectives are to increase the efficiency of operation and productivity, as well as the level of automation, thus making companies more competitive, as concluded in [1].

This digital revolution is driving the implementation of tools and intelligent platforms that produce a greater amount of data and information for analysis, as explained in [2]. The storage of large amounts of data allows an analysis of the conditions in which a product was created, as well as the use of machine learning techniques, to make an early prediction of the occurrence of product defects or machine failures.

However, not all companies are prepared for this type of revolution, because many operate on rather primitive processes, where human work and control predominate. An example is investment casting companies that use the lost wax casting method, where at least the last part of the process is mostly manual. In this type of manufacturing, the occurrence of failures in some sections, more specific in the casting department, is quite frequent and difficult to control. Generally, data acquisition and monitoring are also very scarce, due to the existing manufacturing processes.

This project belongs to the final dissertation from the master degree in Electrical and Computer Engineering, and it was developed in an entrepreneurial environment in partnership with the Institute of Science and Innovation in Mechanical and Industrial Engineering (INEGI). INEGI is a Research and Technology Organisation, focused on applied Research and Development, Innovation and Technology Transfer activities between university and industry, [3].

For this project, the investment casting international company *ZOLLERN & Comandita*, with a portuguese branch located in Porto, Portugal, that it is working in collaboration with INEGI, was used as a case study. This company is adapting to the Industry 4.0 ideas. After studying the company and its processes, a joint decision was made that is able to conclude that the section that needs more attention, due of its defects, is the foundry department. Since this section is the most manual, the acquisition of data and management control is harder to achieve. Such makes this company a good case study as a representation of the companies of investment casting that want to grow for the Industry 4.0 ideas.

1.2 Motivation

With the ideas of the Industry 4.0, a big quantity of data has been stored in clouds. Without systems capable of filtering and present this data, it is hard to take some advantage of that.

At the time of this work, the portuguese branch of *ZOLLERN & Comandita* has few data stored in the cloud. Since data collection is at its early stages, it was necessary to develop a simulator, which produces relevant data, very close to what would be real. Some work concerning the data structure was developed by INEGI collaborators before this project began. However, the structure of data was improved during the development of this dissertation.

After the development of the simulator, a data visualisation tool could allow *ZOLLERN & Comandita* to see the data stored in cloud and, with that, it has more control over what is being produced in the foundry department. In the case of the manager, who is not on the shop floor, he could know what is being produced with a data visualisation tool that shows data storages in cloud in real time.

Sometimes, a large quantity of parts with defects are produced, however it can be difficult to identify the fault. With this tool, it is easier to know the conditions of the production and identify if it was a human or a machine malfunction. Cause-effect relations help the company to be more profitable and efficient, avoiding faulty parts.

1.3 Goals

The main objective of this dissertation is to develop a data visualisation tool, *SmartVis4.0*, to be applied in *ZOLLERN & Comandita*, that would help it to see collected data in a database. This tool allows to:

- Monitor, in real time, the collected data as well as the whole operation on the shop floor;
- Obtain alerts when an error is detected, by studying the deviations from the standard;
- Visualise data from a finished manufacturing order, which makes it possible to understand the conditions of its production, as well as to make the cause-effect study of its defects or qualities.

To make this tool more flexible, it should be a web page to be accessible in any device, wherever the user is. To have a good view in small screens, like in smart-phones, it should be a Responsive Web Design (RWD).

Since this work is being developed in an early stage of the industrial revolution in *ZOLLERN & Comandita*, it was necessary to develop:

- An architecture and structure for a database;
- A simulator that produces relevant data, similar to what would be real.

While real data collected does not exist in a database, the *SmartVis4.0* will show the data generated by the simulator. When the company has real data stored in database, the *SmartVis4.0* should be able to work correctly.

1.4 Contributions

The main contributions of this work are:

- The strategy adopted for the development of a data visualisation tool for the specific case study of an industry with the investment casting process;
- The case study for similar problems where the Industry 4.0 ideas are difficult to be applied when big quantity of data does not exist;
- The developed tool allows the user to see intelligent data, since the data mining ideas do not have statistical validity.
- The study of the investment casting process in *ZOLLERN & Comandita* allows the development of a systematised list with the known and expected problems in the foundry department.

From the development of this work originated the following scientific paper:

- Beatriz Cruz, Armando Sousa, Ângela Cardoso, Bernardo Valente and Ana Reis, "*Smart Data Visualisation as a Stepping Stone for Industry 4.0 - a case study in investment casting industry*".

This paper was accepted in ROBOT2019 conference, that will be happening in Porto, Portugal, in November of 2019.

1.5 Document Structure

Chapter 1 focuses on the context, motivation and goals for the proposed work, followed by this scientific contributions of the work and resulting publications.

Chapter 2 begins with a state of the art (Section 2.1) of data visualisation systems, followed by some fundamentals (Section 2.2) for the work developed and, in the end, the important selected tools (Section 2.3).

Chapter 3 presents the case study for the development of this work with the explanation of the investment casting process. The highlight stage is described with more detail and a list of known problems is presented.

Chapter 4 explains with some detail the data visualisation tool developed. This chapter begins with the identification of the requirements, followed by the architecture of the system. After this the data structure followed by the smart data dynamics with the explanation of the data flow. At the end of the chapter, some relevant implementation details will be explained.

Chapter 5 presents the obtained results and Chapter 6 concludes this work and addresses future research directions.

Chapter 2

State of the Art, Fundamentals and Tools

2.1 State of the Art

This chapter will present some ideas and fundamental concepts that will help to understand the work developed. Beginning with the state of the art about Industry 4.0 and some data visualisation tools that exist at this moment. After that, the expert system concept will be introduced, with a connection to some smart visualisation tools.

2.1.1 Industry 4.0

Since the 18th century, it is possible to follow the evolution of industry. The first revolution was represented by the mechanical production plants based on water and steam power. In the 20th century, the second industrial revolution was a symbol of mass labour production based on electrical energy. The third industrial revolution began in the 1970s with the characteristic of automatic production based on electronics and internet technology. Nowadays, we are living through the fourth industrial revolution, named "Industry 4.0", with the characteristics of cyber-physical systems production, based on heterogeneous data and knowledge integration, as cited in [4]. Figure 2.1 shows the evolution of the industry since its inception.

Scholarly, there have been several different approaches to define Industry 4.0. Article [1] presents a survey on technologies and applications in Industry 4.0. It shows some of the definitions that exist about this revolution. As an example, at the Consortium II Fact Sheet [6], Industry 4.0 is *“the integration of complex physical machinery and devices with networked sensors and software, used to predict, control and plan for better business and societal outcomes.”*. However, it concluded that there is no unanimity.

The different applications of Industry 4.0 are numerous, from smart factory and manufacturing to smart product and finally to smart city [7]. In this dissertation, the main goal is to apply the ideas of Industry 4.0 regarding smart factories with control-centric optimisation and intelligence.

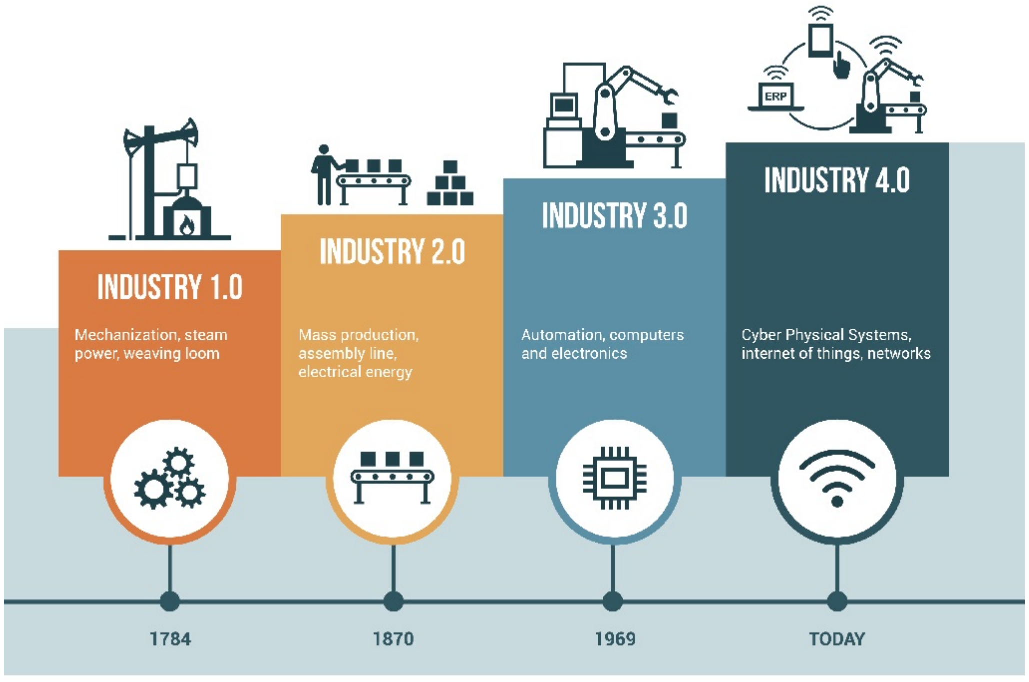

INDUSTRY 1.0
Mechanization, steam power, weaving loom
INDUSTRY 2.0
Mass production, assembly line, electrical energy
INDUSTRY 3.0
Automation, computers and electronics
INDUSTRY 4.0
Cyber Physical Systems, internet of things, networks
ERP
1784
1870
1969
TODAY

Figure 2.1: Evolution of Industry [5].

The strategy is to make industries more intelligent, adding dynamics with sensors, actors and autonomous systems. With the ideas of smart factories, software tools have been developed in order to help the improvement of the factory processes. Intelligent systems that have knowledge about the processes will then be used to optimise product service needs and predict product performance degradation.

2.1.2 Data Visualisation Systems

The usage of data visualisation systems has been increasing due to the big amount of data that has been collected. Without these tools, the data becomes useless, because it is hard to interpret and make conclusions with just big tables.

2.1.2.1 Q-DAS

The ***Q-DAS*** [8] software is specialised in the computerisation of statistical procedures with a focus on quality management applications including Statistical Processes. This software has several tools, such as *QS-STAT*, which allows the user to make statistical analyses of the collected data by producing analysis reports, that refer to the values collected by sensors at certain time intervals.

2.1.2.2 Grafana

Grafana [9], is an open-source platform that allows visualisation of data. It is a dashboard that allows the user to query, visualise and understand data metrics, regardless of where they are stored.

The dashboard gives the users the possibility to chose the type of the graphs which they want to see. The users can add thresholds, to be alerted when the values deviates from that threshold.

2.2 Fundamentals

With the evolution of technology, personal computers and mobile phones became essential tools in daily lives of people. The computer is very useful because of its size, however, the user cannot always bring the computer with them. The mobile phone does not have this constraint as the user always carries it. Because of that, more and more, the applications for the most different purposes are developed to work on mobile phones to allow the users to use the tools everywhere they are.

This evolution raises questions of the user experience when the trend is to use more the mobile phone than the computer, and to develop applications that are normally used in a computer, to be used in a mobile phone.

The following section aims to explain some relevant questions about this topic for the work developed. The respective decisions and conclusions will also be presented.

2.2.1 Native App vs. Web App

As it usually is, when a new application is developed for a specific device, the choice is the native Software Development Kit (SDK). This platform is usually provided by the vendor and each operating system has its specific languages, thus making these applications very restricted to the device and its operating system. To develop an application that is universal and simultaneously compatible with different environments is more expensive. To solve the limitations of the SDK, the concept of cross-platform software was created.

Cross-platform is a computer software, that is implemented on multiple computing platforms. This could solve the problem of compatibility of operating systems. There is one basic characteristic that is shared by all cross-platform app development frameworks.

Web applications are considered cross-platform, since they just need a browser to run the application, and all devices, independently of their operating systems, have a browser. Actually, the quantity of frameworks available to develop web applications, guarantees that the apps could be used in multiple environments [10].

Since one of the objectives for the developed work is that the user of the platform could use it in any device or place inside the company, a web app was chosen to be developed. Then, the same platform could be used in any device without the need of a software installation, just a network connection. The application will read data in real time from the database, so the network connection will not be an issue, because the platform needs this connection to fulfil its main objective, [11].

2.2.2 Recent trends in web programming

With the growing usage of mobile phones, the trends in web programming have been more discussed. There are a lot of different ideas that allow the utilisation of web apps in mobile phones to become more user friendly.

After a study of the most recent trends in web programming, the most relevant for the application to be developed was identified. They are:

- Responsive Web Design (RWD);
- Push Notifications;
- Progressive Web App (PWA).

The choice of a web application for this work was to allow the user the possibility to use the application in all devices wherever the user is. However, the size of the computer screen and the mobile phone screen are very different. The design of a web page to be assessed in a computer, is not always an attractive design to be visited in a mobile phone. With this in mind, the concept of RWD can be introduced [12]. In [13], three elements are identified in RWD, which are a fluid layout, flexible media and media queries. These features improve the user experience in the web application, with the capacity to adapt the web design in function of the screen size.

The concept of an adaptive web app is also very used when talking about web design, but this is different from the concept of responsive. The adaptive web design uses distinct layouts for multiple screen sizes, the layout largely depends on the screen size being used so with each of these sizes in mind a layout would have to be designed for it, while the responsive web design adapts to the size of the screen no matter what the target device screen size is. The layout is fluid and uses CSS media to change styles, thus enabling the page to resize its width and height to adapt to different screen sizes and display correctly.

Another interesting feature in web development are **push notifications**. Push notifications are capable of alerting the user, even if the application is running in background. These alerts use sounds, vibration and the standard notifications of the device. In native applications, it is common and easy to use this tool, making the interaction between application and user more powerful. Before the growing usage of web apps, there were few frameworks that enabled this feature in web pages. However, this paradigm has been changing and nowadays, it is possible to find a lot of frameworks compatible with multiple devices.

All results in searches made about new trends in web development talk about **Progressive Web Application** (PWA). This concept comes up as a possible unifying technology for web apps and native apps, as mentioned in [14]. While a regular web site requires the user to open a browser, type in a URL and wait for all content to be downloaded on every visit, a PWA only requires these steps for the first visit. After a home screen installation, all necessary static files, including HTML, CSS, JavaScript, images and fonts for the web site, are now stored on the phone of the user, ready to be used offline [15].

Increasingly, the frameworks and paradigms in web development have been created with the rise of web applications usage. There is a lot of other recent trends in this area, however, the enumerated were the most relevant for the work to be developed.

2.2.3 Web Architecture

The architecture of a web application is crucial for supporting its future growth, which may come from increased demand, its future interoperability, and its enhanced reliability requirements.

A web app is typically composed of two main sides: the client side and the server side. As an overview, in this architecture the user sends the command to the server through the Internet, using the browser or the interface of the application. The web server is responsible for forwarding the command to the requested server. The server executes the requested commands (either the data processing or the database querying). The server delivers the processed information to the web application, which provides the user with the requested data. This is the basic of a client-server architecture.

Researching about specific architectures in web development on the client side (or frontend), there is a lot of recent documents that talk about Single Page Application (SPA). This concept is very common when talking about new trends in web development. The SPA is a web application that fully loads all of the resources in initial request and then the page components are replaced by other component depending on user interaction, as cited in [16]. In the same paper, it is mentioned that this architecture is very flexible and elegant when dealing with data.

In Figure 2.2, it is possible to compare the two different architectures mentioned above, the traditional client-server and the SPA client-server. While in traditional architecture, for each request, the page is loaded again, in the SPA architecture, the page is loaded on the first request, and for the following requests only the required data is updated.

2.3 Tools

There are a lot of different tools available to develop web applications. According to the ideas and objectives for the work to be developed, and following the trends mentioned above, the selected frameworks will be described in this section.

2.3.1 Angular

In web development, when talking about the frameworks to develop the frontend, *Angular* is one of the best frameworks referenced in all web pages that talk about this topic. This is an open source *JavaScript* framework, although the most recent versions use *TypeScript*. This is a typed superset of JavaScript that compiles to plain JavaScript. *Angular* allows the implementation of the recent trends in web programming mentioned above. This framework is a fully client sided library.

Angular follows a Model View Controller (MVC) architecture. This segregates user interfaces into 3 parts: The model, which contains the data; The view, to which the data is presented; The

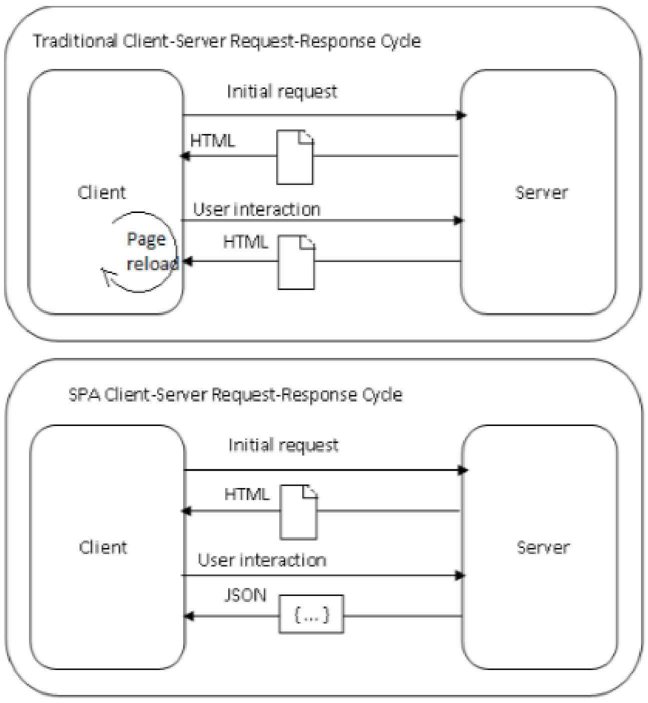


Figure 2.2: Client-Server Request-Response Cycle, from [16].

controller, which acts as a negotiator between user and model. The MVC architecture has the capability to simplify development, maintenance and testing, as it is mentioned in [16].

Angular is based on full Bidirectional-Data-Binding. Bidirectional-Data-Binding is an automatic way of updating the view whenever the model changes, and updating the model whenever the view changes, as it is referenced in [16].

SPA is becoming popular in recent web development ideas and the technology *Angular* also aids to create such applications in an easy way. When developing an *Angular* application, the developer creates HTML templates according to *Angular's* template language; the HTML templates include HTML that is embedded with *Angular* scripts and other *Angular* coding constructs, such as directives. At the web browser, *Angular* JavaScript libraries are loaded and interpret the HTML templates, such that the resulting pages look and behave as defined in the templates, as it is explained in [17].

2.3.2 Django

Article [18] provides a methodology to evaluate three open source web development frameworks for the backend. These are *Django*, *Ruby on Rails* and *CakePHP*. Features like maintainability, data management and migration, testability, popularity, marketability, community and maturity were used to evaluate each one. The final result gave the *Django* framework the highest score, with 4.05, while Ruby on Rails and CakePHP got 3.85 and 2.95, respectively.

Other articles agree that *Django* is a good framework in web development. This framework is based on Python, which is a very popular programming language. Python is an open source language and has a large number of libraries and modules available.

Django calls its architecture Model Template View (MTV), which is very similar to MVC. The layer Model is for structuring and manipulating the data of the Web application. The concept of Views is to encapsulate the logic responsible for processing a user request and for returning the response. The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. This information is explained in [19].

2.3.3 Chart.js

The final web application will present data in a dashboard. For that it was necessary to chose a framework compatible with the frameworks and languages chosen above.

Chart.js is an open-source JavaScript library, that has eight different types of charts, each of them animated and customisable. It follows the HTML5 canvas with a great rendering performance across all modern browsers. As an added bonus, *Chart.js* is a responsive tool. All this information is described in the official documentation, in [20].

2.3.4 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, front-end web development. The primary purpose of adding it to a web project is to apply choices of colour, size, font and layout from Bootstrap to that project. There are a lot of free open-source templates available on the internet.

The template used in this dissertation was *AdminBSB - Material Design* [21] because it has all necessary elements to develop an intuitive dashboard and it is is a fully responsive template.

Chapter 3

Case Study - Investment Casting

3.1 Introduction

In this section, the process of investment casting, more specifically in *ZOLLERN & Comandita*, will be explained. The highlight stage for this work will be demonstrated with more detail. The list of known problems and the requirements for the work developed will also be shown.

3.2 The Investment Casting Process

Lost wax casting is a method of producing metal parts with high precision. This process has eight stages of development until the final product is obtained. Table 3.1 demonstrates the different stages of this casting method and their respective explanation.

At *ZOLLERN & Comandita* the production of parts is organised in Production Orders (PO). Each PO has associated a specific part to be produced, the number of parts per tree, the number of trees and all the necessary parameters to correctly produce the parts on each phase of the process.

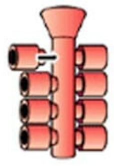
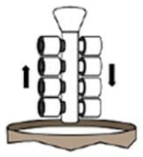
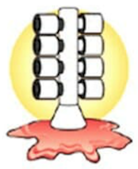
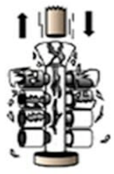
Each phase of the production process has its specific function, as was shown in Table 3.1. In *ZOLLERN & Comandita* some phases are more rudimentary than others and the number of rejected parts in each phase can vary. This case study was focused on a specific phase that will be explained next.

3.3 Highlight Stage

A study was conducted at *ZOLLERN & Comandita* to understand which section has the most issues and the foundry was the one that stood out in the high number of incidences. That is the main reason why the focus of this case study was the foundry section, since this area will benefit the most with this work.

The foundry area is composed of various inner processes such as wax removal, tree sintering, alloy melting and casting. **??** represents a wax tree before the layers of ceramic. The trees, at first, arrive at this area in shelves, with a non-constant number of trees per shelf. This happens mainly

Table 3.1: Phases of Investment Casting Process, adapted from [22].

Illustration	Description
	1. Wax Injection In the first phase of the process, a model of the final product is created in wax. With the metal mould, liquid wax is injected into the interior. When it is in the solid state, it is removed from inside the mould. It is necessary to make as many models, as many pieces as the costumer wants.
	**2. Pattern Assembly** Once the wax pieces are created, they are welded to a trunk, also made of wax, to create a tree of pieces. Thus, it is possible to create several items at once.
	**3. Shell Making** When a tree of pieces is complete, it is covered by several layers of ceramic. A layer of ceramic is made by dipping the tree into a special glue, then it is subjected to a "sandwash" and is finally dried. This process is repeated several times cyclically, at the beginning with fine sands and at the end with coarser sands.
	**4. Dewaxing in a high pressure autoclave** After the ceramic shell has all its layers complete and well dried it is then placed inside a high pressure autoclave in order to remove the majority of the wax from inside the shell.
	5. Melting and pouring The alloy is melted in an induction furnace and once both the metal is ready and ceramic shells are sintered and heated up the pouring phase may then start. In the step of the process the molten metal is poured to the shell in order to fill all the empty cavities. The metal will then cool down and solidify.
	**6. Ceramic break down** After the solidification phenomena and the temperature is low enough, the shell is subjected to high vibration in order to break down the ceramic material. At the end of this step the metal tree will then be clean of ceramic.
	7. Cutting Off At this stage, the parts are separated of the tree by saw cutting.
	8. Finishing and quality control In the last step of the process the parts are finished by grinding and polishing work. There is visual inspection and some other types of quality checks, such as dimensional control, to verify if the pieces and within the client requirements or not. This final step ends the process.

because their size varies and the workers always place the maximum possible number of trees in each shelf. The trees do not need to belong to the same PO, so one shelf could be composed of more than one PO and the same PO could be divided into multiples shelves. The first step of the foundry department is placing a shelf of trees inside a high pressure autoclave.

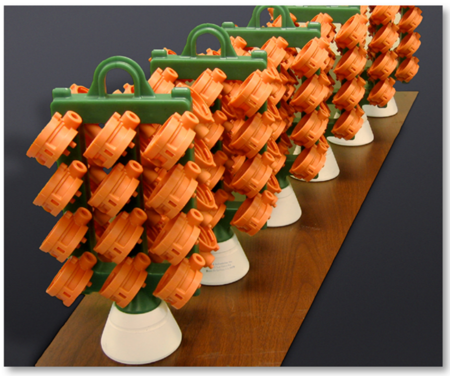


Figure 3.1: Wax tree before layers of ceramic [23].

An autoclave is a pressure chamber used to carry out industrial processes requiring elevated pressure and temperature different from ambient air pressure. In the case of investment casting, its main objective is the removal of the majority of wax from inside the ceramic trees. The autoclave is composed of two zones, the steam generator and the main chamber.

The main objective of the steam generator is to slowly increase its pressure, until it reaches 12 bar. It is crucial that when a new cycle begins in the main chamber, the steam generator is at 12 bar, to guarantee that the process functions correctly. When the autoclave begins a new cycle, with a shelf inside of the main chamber, the existing pressure in the steam generator is quickly transferred to the main chamber, so the pressure in the main space increases the pressure between the environmental pressure (approximately 1 bar) to 9 bar, in less than 14 seconds.

The fast increase of the pressure inside the main chamber of the autoclave is fundamental to maintain the good quality of the ceramic trees, because the wax inside of that will expand with the increase of the temperature. If the pressure grows fast, the first layer of wax will melt first, and that will make space for the expansion of the remaining wax, without damaging the ceramic tree.

After the fast pressure increase, the autoclave maintains the pressure during 15 minutes in the main space where the shelf is, and, at the same time, increases the pressure in the steam generator until the 12 bar. This high pressure removes a big part of the wax inside the tree. At the end of the 15 minutes, the main space begins decreasing the pressure by steps. When the pressure reaches the environmental pressure, the cycle is finished and the shelf can be removed from the autoclave.

After the dewaxing is complete, the empty trees, also known as shells, are prepared, if necessary, and stored where they wait their turn to be sintered and later casted after the rotary furnace process.

In the rotary furnace, the shells are placed in small sections, called buds, with at most three shells per bud, that corresponds to three different positions in each bud, and run through the five different zones until they reach the exit.

Each zone is composed by a different number of buds. Figure 3.2 represents the rotary furnace and its structure. As presented there, zone 1 and zone 3 have five buds, zone 2 has seven buds, zone 4 has four buds and the last one, zone 5, has three buds.

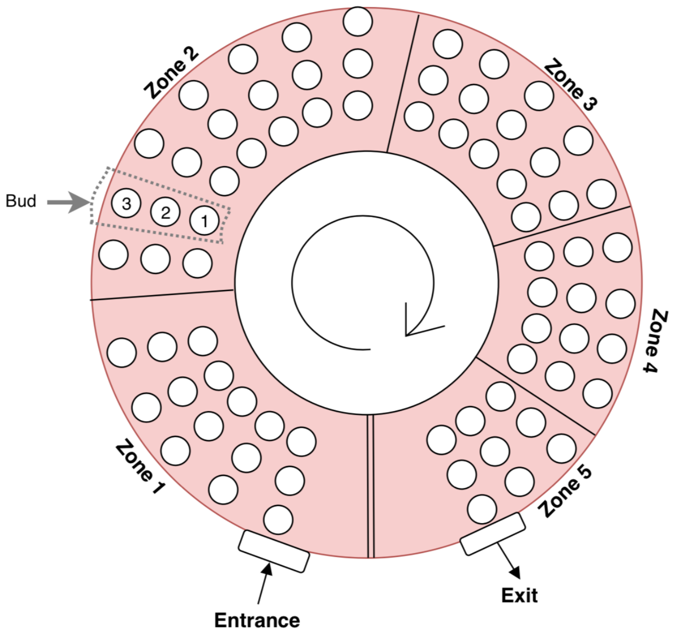


Figure 3.2: Scheme of the rotary furnace.

The shells are placed inside the furnace, one by one, in each position of the bud, starting from the entrance. When that bud is full, the factory operator presses a pedal that rotates all the buds inside the rotary furnace one position in clockwise direction. Since there is a wall between the exit zone and entrance zone, the bud that is in the exit needs to be free before the rotary movement. To remove shells from the interior of the rotary furnace, the bud with the shells to be removed, needs to be at the exit. After that, the shells are removed one by one. When the bud is free, the operator presses a pedal to move the buds forward and continue the unloading process. When this phase is complete and pouring has ended it is important to move the buds back in order to use the empty spaces in a optimised way when loading more shells.

Zone 1 aims to burn the remaining wax from inside the shell. Both zones 1 and 1 rise the temperature of the ceramic material up 1000^{o}C in order to sinter it. Zones 3, 4 and 5 are where the ceramic material achieves the temperature required for the pouring phase. This phase is important to avoid thermal shocks and prevent the metal alloy from solidifying before reaching all the areas in the interior of the shell.

While the shells are inside the rotary furnace the alloy is being melted in an induction furnace. After at least two hours of shell sintering and once the metal is ready, the pouring may start.

The foundry department at *ZOLLERN & Comandita* has two autoclaves, two rotation furnaces and three induction furnaces, as represented in Figure 3.3 . For the development of this work, only one of each furnace will be considered.

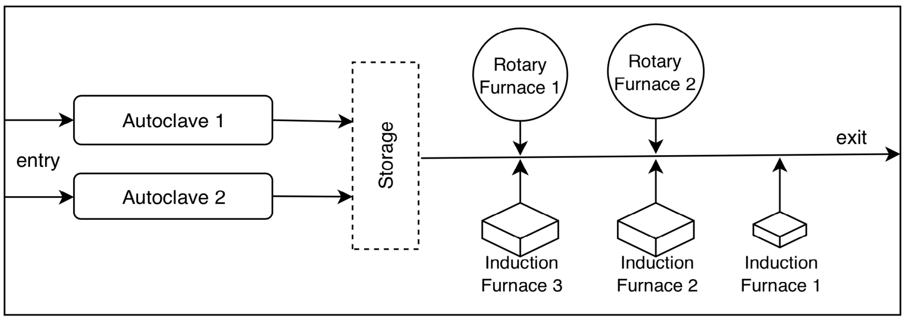


Figure 3.3: Scheme of the main components of the foundry department.

3.4 List of known problems

One of the main objectives of this work is to develop a system that is able to identify and alert the manager of the foundry department when something does not follow the correct procedure. Although, the manager and the operators of the shop floor in the foundry department at *ZOLLERN & Comandita* know the problems that could occur, a list with those possible problems and the respective consequence in a production order did not exist.

In the scope of this dissertation by lit review and interviews with the manager of the foundry department in order to organise the relevant information about all possible failures, a list of the events that could lead to faulty parts in each furnace was created.

Table 3.2 and Table 3.3 list the problems that can occur in each furnace.

3.5 Conclusion

The industry of investment casting is an example of an industry with some rudimentary phases. For that reason, and given that they have room for improvement, *ZOLLERN & Comandita* is good case study for this dissertation.

The foundry department of the *ZOLLERN & Comandita* was identified as the harder phase to control. It is also the one that stood out in the high number of incidences. With the focus on that department, the most relevant problems were identified and how the process should work correctly in detail.

Table 3.2: Problems that could by occur at the autoclave.

1. Delay less or more than 14 seconds to arrive approximately 9 bar In the normal work of the autoclave, the pressure should increase to 9 bar in more or less 8 seconds, but it can take up to 14 seconds. It is considered a failure if the pressure does not get to 9 bar before 14 seconds have elapsed. If this occurs, the ceramic part of the tree, may be damaged with the wax expansion.
2. Pressure in steam generator is not at 12 bar At the beginning of a new cycle of the autoclave, the steam generator should be at 12 bar. If this does not occur, the main space will not reach the expected pressure for the correct functioning.
3. More than 1 minute between entry and start The shelf should not to be inside the autoclave before the initiation of the cycle for a long period of time, because it compromises the quality of the trees since it increases the temperature slowly and the ceramic part of the tree may be damaged with the wax expansion.
4. Pressure is not at 9 bar during the cycle During the interval of the 15 minutes of the cycle of the autoclave, the main chamber should stay constantly at 9 bar to maintain the quality of the trees and remove as most wax as possible that exists inside the tree. In order to have a range, it is considered that something went wrong if the pressure decreases below 8 bar or it increases above 10 bar.

Table 3.3: Problems that could by occur at the Rotary furnace.

1. Tree less than 15 minutes in first zone To guarantee the correct burn of the remaining wax that exists inside the tree, it has to stay at least 15 minutes in the first zone. If this is not fulfilled, the probability that wax remains inside the tree increases.
2. Temperature of zone 1 and zone 2 different of 1000^{o}C The first two zones of the rotary furnace should always be at 1000^{o}C. If the temperature is different, the quality of the trees could be compromised because a good tree sintering will not be acomplished.
3. Temperature of zones 3, 4 and 5 different from expected Each production order has its specific temperature for zone 3, 4 and 5, the same temperature for the three zones. If the temperature is not respected, the sinterization can be compromised.
4. Failure of the oxygen in the first zone The first zone of the rotary furnace has as main objective to burn the remaining wax that exists inside the tree. If the oxygen injection fails, the remaining wax will not be burn and the final pieces will present defects. It is considered a failure when the percentage of oxygen is less than 7%.
5. Less than 20% of the oxygen in the first zone during the entry of trees The first zone of the rotary furnace has as main objective burn the remaining wax that it exists inside the tree. During the entry of the trees in the rotary furnace, the level of oxygen should be 20%, in order to guarantee a correct burning of the remaining wax.

Chapter 4

Software Solution - Simulation and Visualisation

The development of a data visualisation tool is the main objective for this work. The proposed software solution is called *SmartVis4.0*, because it is a visualisation tool that should help the user make informed and intelligent decisions in the context of Industry 4.0, while also having its own ability to identify important or abnormal events, based on its built in knowledge about the manufacturing process at *ZOLLERN & Comandita*.

SmartVis4.0 should work correctly in different devices. In order to develop this web application, the framework *Angular* was used for the frontend component and *Django* framework for the backend.

Since it was not possible to collect data in the shop floor, it was necessary to develop a simulator, that will be used to test *SmartVis4.0*. When data collected from the shop floor exists, *SmartVis4.0* will work correctly by reading and interpreting the real values from database.

In this chapter, the requirements, the architecture and the data structure for the developed work, will be presented. At the end, more details about data flow and relevant implementations, will be explained.

4.1 Requirements

Having identified the base goals for *SmartVis4.0*, the requirements for this system were created. The functional requirements will be presented with Use Cases (UC).

4.1.1 Functional Requirements

The main requirement of *SmartVis4.0* is, as it was mentioned, to show collected data from the foundry department of *ZOLLERN & Comandita* identifying the id of the PO or the tree id that the user wants to see. This data should be presented in graphics and diagrams to help the user to understand it. Alternatively, the user should have the possibility to choose which data they want to see. This requirement is summarised in UC 1.

UC 1. *The user wants to see data that was collected from a specific furnace, selecting a PO or a tree id.*

Another requirement is that the section managers should have the possibility to see data related to what is happening on the shop floor, in real time. That way, they can quickly perceive what is being produced and if everything is working within the expected parameters. This is summed up in UC 2.

UC 2. *The user wants to see data that is being collected in real time to control the shop floor.*

The rotary furnace is loaded one tree at a time and it is important to know the state of each bud, previous shown in Figure 3.2. Thus, it is useful if the data visualisation tool has a simple and intuitive way to see, in real time, the state of the rotary furnace. This requirement is collected in UC 3.

UC 3. *The user wants to see the state of the rotary furnace in real time.*

Another requirement identified for the system is to alert the department manager when something deviates from the expected values, without the manager needing to constantly monitor the data. This requirement is represented by the UC 4.

UC 4. *The user wants to be alerted when something does not work correctly.*

The diagram in Figure 4.1 illustrates the four statements above, which compromise the initial functional requirements established for *SmartVis4.0*.

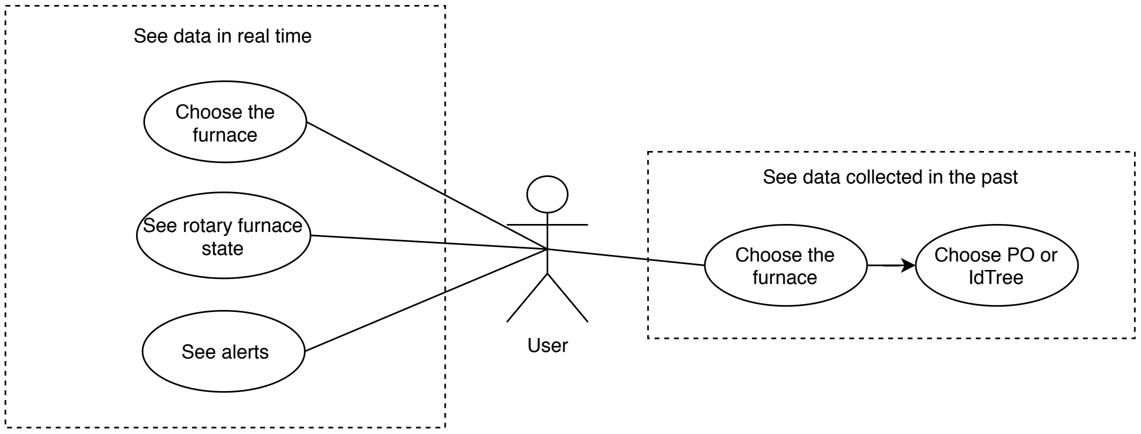


Figure 4.1: UML use cases diagram.

4.1.2 Non-Functional Requirements

In terms of non-functional requirements, *SmartVis4.0* should work in different devices, with different screen sizes, so it should be platform compatible. To have an easy and intuitive interpretation by the user, usability was also a non-functional requirement for the system.

The system will show data in real time and it could change every second. Thus, the performance of the system was considered as another non-functional requirement, because the existence of delays in the presented data would implicate that the real-time feature would not be fulfilled.

Scalability is also a concern, since the tool is meant to be extended to the whole manufacturing process and because, without the manager needing to constantly monitor the data. The managers may want to see the history of the shop floor. The tools selected for the development of this system were chosen so that its evolution is as easy as possible, while also being easy to change the visual appearance.

Security requirements are not a big concern and were not considered. Even though some data could be sensitive, the system is meant to only be available in the internal network of the company. Assuming that the local network is properly secured, there is no need for extra security measures in the visualisation tool. Eventually, as more data becomes available, if the company feels the need to have access control, user authentication and authorisation can be added.

4.2 Architecture

To obtain a good data flow, an architecture for the whole system was created, compromised by its different parts.

This system is composed of three main components, as shown in Figure 4.2. The first component is the simulator, which generates simulated data that will be stored in the second component, the database. Later, the simulator will be substituted for the shop floor, since the data will be collected directly, in real time, from the foundry department. The third component is *SmartVis4.0*, which is divided in two subcomponents, frontend and backend.

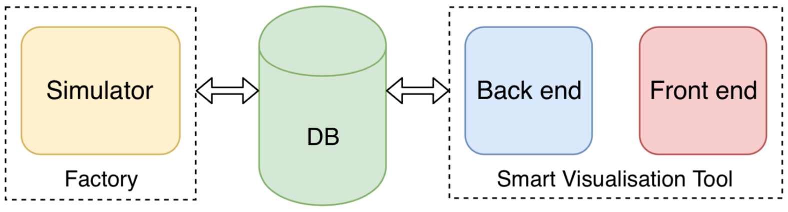


Figure 4.2: System architecture.

The following subsections will describe the architecture adopted for each component.

4.2.1 Simulator

Since the simulator should generate data as close as possible to the real data that exists in the foundry department at *ZOLLERN & Comandita*, it was developed to be as similar as possible to the shop floor. The simulator generates data that corresponds to the correct work of the shop floor, without any problem. The injection of simulated faults was implemented with probabilities and will be explained with more detail in Section 5.1.

The simulator consists of three main parts or subcomponents, as shown in Figure 4.3.

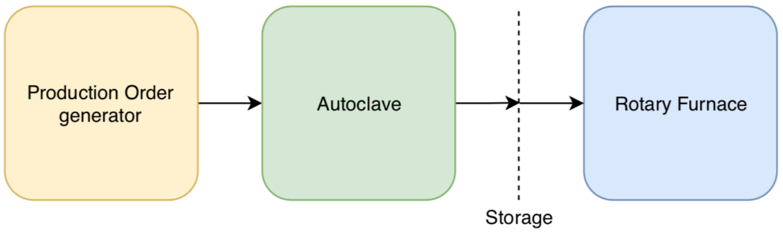


Figure 4.3: Simulator architecture.

The first subcomponent, the PO generator box in Figure 4.3, has the function to generate data regarding the parts that will be produced, with their respective features. The necessary specifications for a PO, are generated in this block as well. This data was generated with a small sample of real data. Based on that, a necessary quantity of POs was created to simulate and generate a good amount of data to be shown.

Each PO is identified by a sequential number that currently has seven digits. When generating a new PO, a random part previously generated and that already exists in the list of possible parts to be produced, is selected. The number of parts to be produced in each PO is generated following a normal distribution, centered in 10 and with a high standard deviation, because this feature could have a large variation. Still in this subcomponent, the POs are organised in shelves to enter the foundry department, like in the shop floor. Each shelf is identified with a sequential id. This part of the simulator is executed just once to generate the necessary data for the rest of the simulation.

The other two subcomponents of the simulator, represented by the green and blue boxes in Figure 4.3, correspond respectively to the autoclave and the rotary furnace. These two parts generate data by simulating the operation of each furnace. This simulation is executed in real time, and the data is generated at the same rate that the real furnaces can monitor. These two blocks work independently, like the two furnaces in the case study.

The autoclave has two important zones for its operation, as described in Chapter 3. The part of the simulator that corresponds to the autoclave operation, was developed with two state machines that correspond to each part of each zone. The two state machines are represented in Figure 4.4 and Figure 4.5.

For each phase of each zone of the autoclave, the simulated data was generated with normal distributions centered in the target value.

When a shelf ends a cycle inside the autoclave, it is added to the storage, where the trees wait to go to the rotary furnace. To help the connection between the simulator of the autoclave and the rotary furnace, a trigger was added to the database that automatically inserts into the storage the shelf that ends the cycle in the autoclave.

Once the storage table has all shelves containing a PO, the trees in that PO are ready to enter the rotary furnace. This part of the simulator works as in the real world, independently of the autoclave.

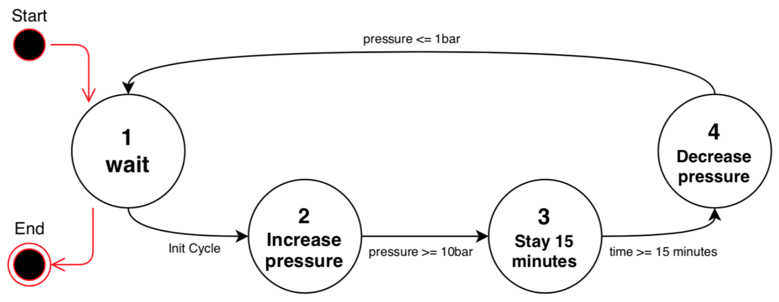


Figure 4.4: State machine for the main chamber of the autoclave.

Simulating the behaviour of the rotary furnace is more complex, because the entrance and exit of trees from that furnace, together with the advances and retreats of the furnace base, are controlled by the operators, making these movements difficult to normalise. Of course, it is also due to that manual operation that the department managers need to have better control of this work flow. Parallel to that, the temperatures need to be collected and they are set for each zone individually, thus they are potentially different for each zone and consequently for different trees that it are inside the rotary furnace. Considering this, two threads were developed. One thread simulates the events of input and output of the trees in the rotary furnace, as well as, the advances and retreats of the base. The algorithm developed in this thread will be explained in Subsection 4.5.1. The other thread simulates the temperatures monitored in each zone. Similar to the autoclave, the simulator of the rotary furnace generates the real temperatures that follow a normal deviation, centered in the target value.

4.2.2 *SmartVis4.0*

SmartVis4.0 was envisioned to be as efficient as possible, because it will process a large quantity of data. This tool needs to be simple, and have an intuitive interface. Because of that, it was divided in two parts, the backend and the frontend. Each part has its architecture to be more efficient in its functions. The deployment diagram for *SmartVis4.0* is presented in Figure 4.6. The system uses the company's corporate PostGreSQL database server.

As it is demonstrated in the deployment diagram, the user will interact with *SmartVis4.0* by a web browser, which communicates with the frontend server. Each time that is necessary to communicate with the database, the frontend communicates with the backend server, that will fetch the necessary data and will send it to the frontend to be shown to the user.

Frontend

Due to the fact that *Angular* is a Model View Controller (MVC) based framework, but also because this architectural style is well suited for web applications, the architecture of the frontend follows the MVC pattern. As such, there is a representation of the data structure in the Model,

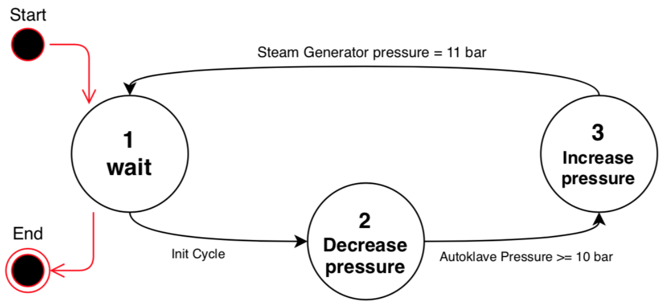

Start
End
1
wait
2
Decrease pressure
3
Increase pressure
Steam Generator pressure = 11 bar
Init Cycle
Autoklave Pressure >= 10 bar

Figure 4.5: State machine for the steam generator.

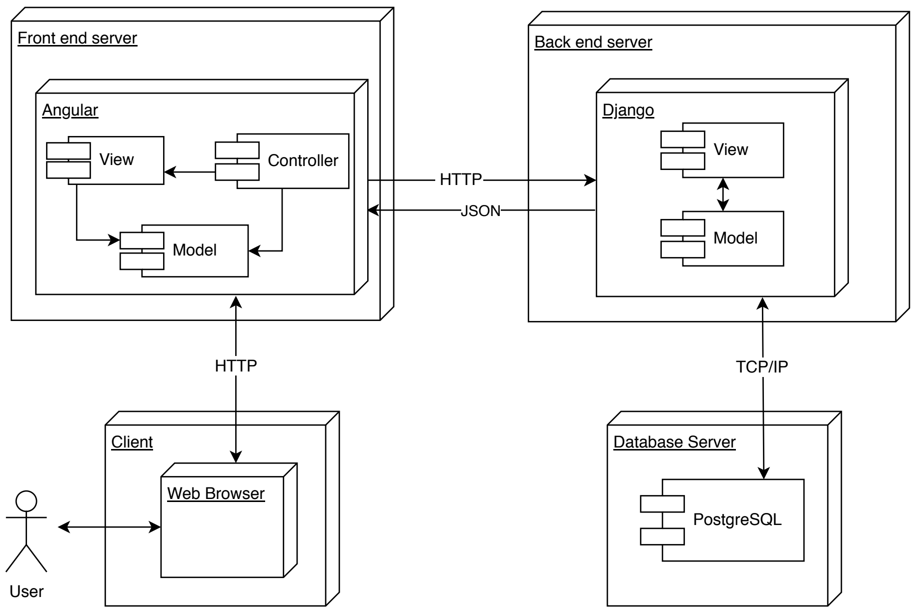


Figure 4.6: Deployment diagram for *SmartVis4.0*.

which is simultaneously used as source of the information displayed in the View and queried by the Controller upon user interaction.

Backend

As for the server, the *DJango* framework was used, which follows the Model Template View (MTV) architectural style. MTV is similar to MVC, as mentioned in Subsection 2.3.2, but depending on the source, there are some differences. In any case, the core ideas of code separation according to its purpose are the same. For the backend, the Model was used to represent the data and communicate with the database, while the View is responsible for replying to the HTTP requests from the frontend using the JSON format. There is also a layer of business logic, that is responsible for the necessary processing of data before sending it to the frontend. Typically, the Template layer is responsible for presenting the information to the user, not the content itself, but the way it is presented. For the backend component, this layer was not developed, because the template is done in the frontend component.

4.3 Data structure

Data is one of the main elements for the development of this project. The work presented is based on preliminary work presented in [24]. As such, an architecture was created for the database, facilitating its access and interpretation. The architecture developed for the database is presented in Figure 4.7 with a relational diagram.

The database was organised in three main parts. The first one contains all information of the parts to be developed and their respective features to be produced. The second one contains all data monitored with the respective time stamp for what is being produced. The last one saves the alerts to be shown in the data visualisation tool.

The next sections will explain with more detail each part of the data structure in the database.

4.3.1 Information data

To study and analyse what is being produced, it is traceability to the tree to keep all the information from POs. This information is inserted by the administration of the company and contains all features concerning the parts that the customers require. The tables shown in Figure 4.8 were created to store only that informative/static data.

As previously shown, a part is associated to a PO. Since each PO has its specific features and it is possible to have more than one PO to the same part, the relation between the table *PartType* and *ProductionOrder* is $1:n$.

A PO has associated several trees. Even though all trees have the same features, they will not be manufactured in the same exact conditions, so it is important to uniquely identify each tree. In that way, the *Tree* table was created, that preserves the features for each tree. The relation between the table *ProductionOrder* and *Tree* is $1:n$.

Once the trees enter the foundry department they are organised in shelves, hence there is another table, *Tree_Shelf*, that associates the trees with their respective shelves.

4.3.2 Monitoring data

At this part of the database, all data collected from the shop floor is saved. It corresponds to the critical attributes that should be controlled for the correct work of each furnace, but also all events from the shop floor in the foundry department.

The monitored data on the database is organised in two subsections, Subsubsection 4.3.2.1 and Subsubsection 4.3.2.2, that corresponds to the autoclave and the rotary furnace respectively.

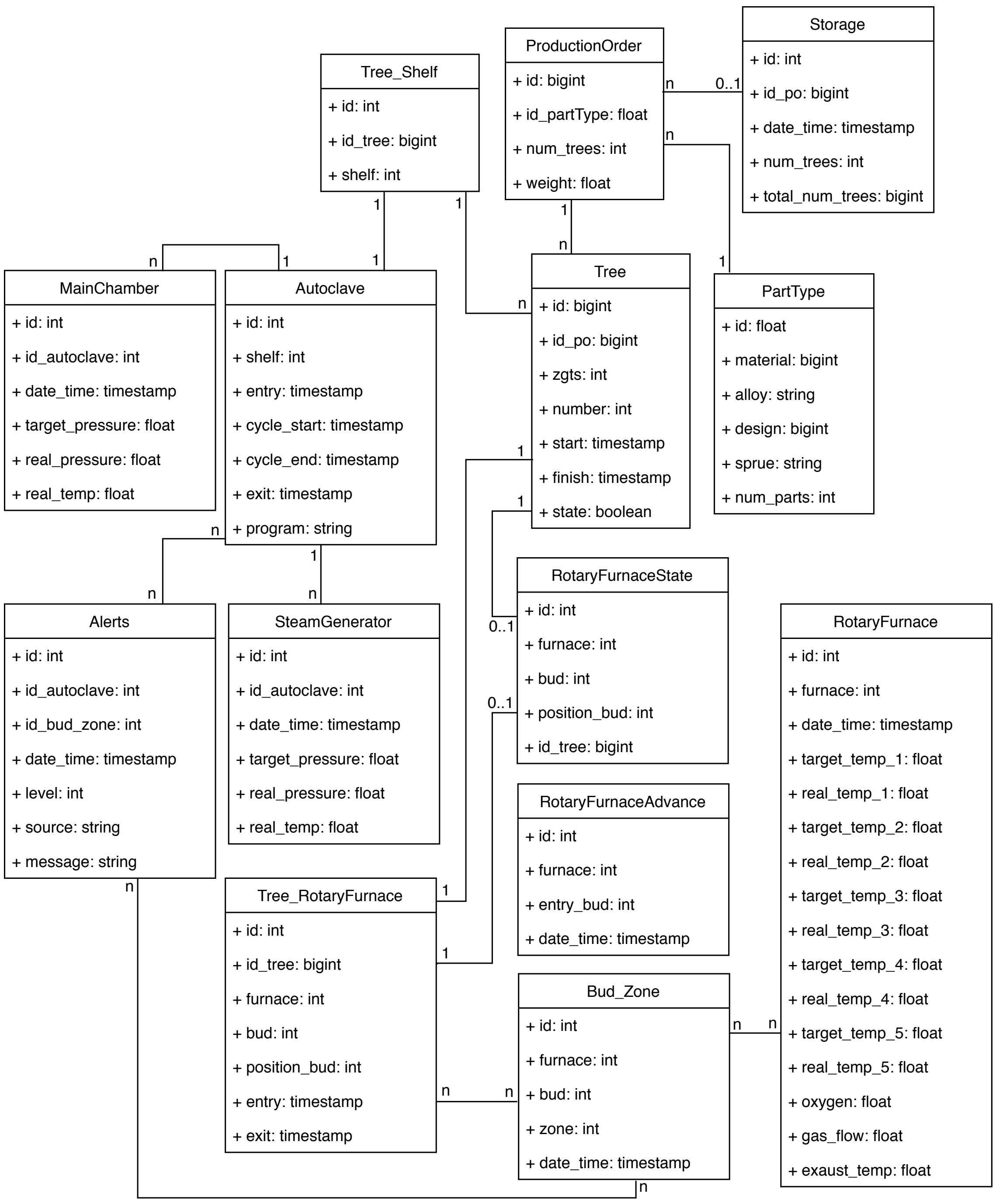


Figure 4.7: Relational diagram.

4.3.2.1 Autoclave Data

The foundry department starts with the processing of shelves of trees through the autoclave. The data of this part of the process is stored in three different tables, presented in Figure 4.9.

The *Autoclave* table stores information about the autoclave cycles. It is important to save the shelf that was processed in that furnace and the respective timestamp for its entry and exit, as

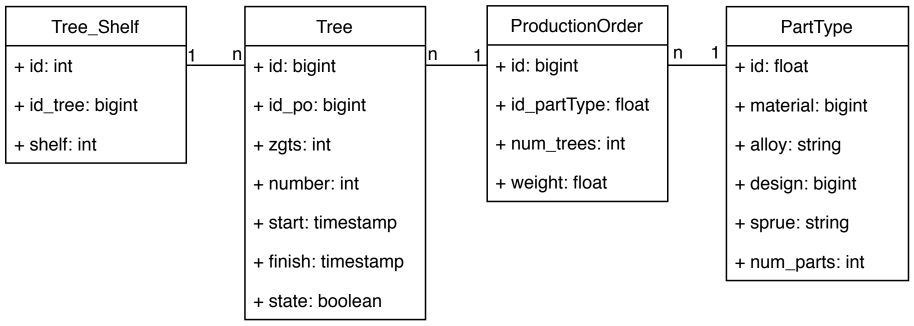


Figure 4.8: Relational diagram for the main production information.

well as the begin and the end of its cycle. It is important to save all this timestamps in order to identify exactly the pressure of the main chamber and the steam generator during the cycle. The time difference between the entry and the start of the cycle, as well as, the time difference between the end of the cycle and the exit, are important to control, because if those intervals are large, they compromise the quality of the trees.

The tables *MainChamber* and *SteamGenerator* store temperature and pressure data, for each second that the cycle lasts, in the main chamber and the steam generator, respectively.

The relation between these three tables is that for each new insert in the table *Autoclave*, there are *n* inserts at the tables *MainChamber* and *SteamGenerator*.

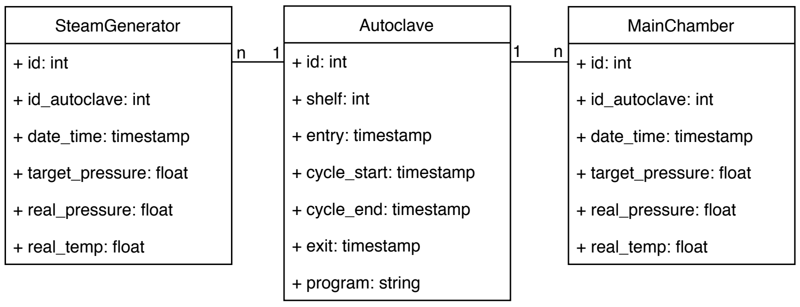


Figure 4.9: Relational diagram for the autoclave section.

4.3.2.2 Rotary Furnace Data

The tables for the rotary furnace are depicted in Figure 4.10. This part of the database is composed by five tables. One of them, the table *RotaryFurnace*, saves the essential attributes that are necessary to guarantee that the furnace is working correctly. The remaining tables save all the events that occur in this furnace.

Every minute, already defined by the furnace, the *RotaryFurnace* table saves the measured temperature for each zone. This corresponds to the attribute *real_temp_n*, where *n* is the number of the corresponding zone. The element *target_temp_n* indicates the nominal value for each zone, in each insert, where *n* is the number of the corresponding zone. As the first zone is different from

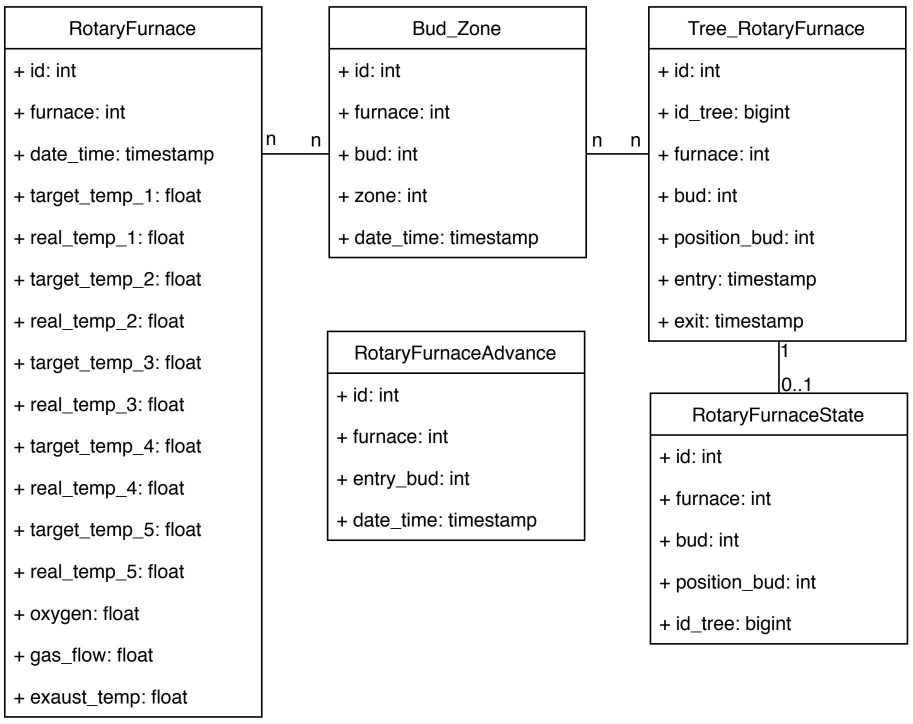


Figure 4.10: Relational diagram for the rotary furnace section.

the other ones, because it has oxygen injection to burn the trees, for each insert in this table, the oxygen level, the gas flow and the exhaust temperature are also stored, that correspond respectively to the attributes *oxygen*, *gas_flow* and *exaust_temp*, in the *RotaryFurnace* table.

The main events of the rotary furnace are the entry and exit of trees, and the bud advances inside the furnace. Since the company wants to control the position of each tree during its passage in the rotary furnace, for each tree, the id of the bud and its respective position, are saved in the *Tree_RotaryFurnace* table, in the attributes *bud* and *position_bud*, respectively. The timestamps of the respective entry and exit, are also stored in this table.

To simplify the reading of the current state of the rotary furnace, each time a new tree is inserted into the rotary furnace, the *RotaryFurnaceState* table is updated by a trigger. This trigger will be explained in chapter Subsubsection 4.4.2.2. The *RotaryFurnaceState* table always has the current state of each bud, saving the *id* of the trees at that moment, without maintaining a log.

It is also necessary to record the location of each bud over time. To do that, for each new advance of the rotary furnace, the id of the new bud in the first position on the first zone, that corresponds to the entry of the furnace, is stored on the *RotaryFurnaceAdvance* table. This is stored because knowing the bud that is in the furnace entrance at each moment, it is possible to know the ids of the remaining first buds of each zone. The *Bud_Zone* table, stores the bud at the start of each zone over time. This information is generated with a trigger on each rotary furnace advance. This trigger will be explained in Subsubsection 4.4.2.2. Thus, it is easier to know how much time each tree spent in each zone, given that it is also possible to know the bud in which each tree is loaded.

4.3.3 Alerts

The alerts section is composed of a single table, as shown in Figure 4.11. This table stores the alerts to be shown to the user and it keeps the history of previous alerts.

The alerts have their specific features, so it is important to save them in order to make it possible for the data visualisation tool to show them easily. These features are the level and the source of the alert, that is stored in the *Alerts* table, in the attributes *level* and *source*, respectively. The *level* of the alert indicates the importance of such alert, in a range, whether it is something informative for the manager department or if something catastrophic is happening. The *source* identifies where the problem was encountered, to help the department manager to identify and solve that problem.

Once the *level* and the *source* of the alert is identified, the last parameter, *message*, specifies in an objective form, what generated the alert.

The *Alerts* table is filled by triggers in other tables in the database, which will be explained in Subsubsection 4.4.2.3.

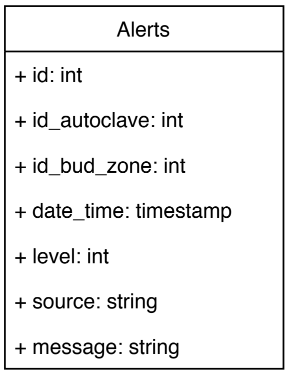


Figure 4.11: Relational diagram for the alerts.

4.4 Smart data dynamics

Data flow is one of the most important aspects of this work. To guarantee that the data arrives correctly to the user, each block communicates with others blocks, with different types of structures, facilitating the implementation of different functionalities. However, the knowledge that the system has also flows inside the database. The Figure 4.12 shows how each block communicates with other blocks.

Next sections will explain the data flow between blocks and the smart data used in each block.

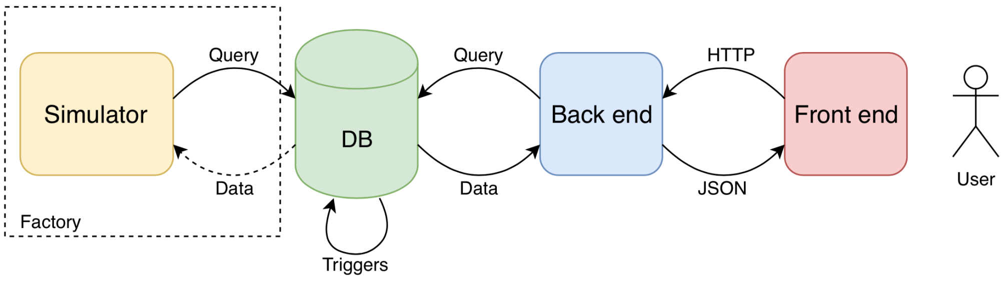


Figure 4.12: Data dynamic in all system.

4.4.1 Inter Blocks

As discussed, the simulator generates data and stores it in the database. This communication is performed by queries, which are represented in Figure 4.12. Sometimes, it is necessary to send some of that data from the database to the simulator, since the latter needs data to work properly. As the simulator will be disabled when the real data from the shop floor is available, the communication from database to the simulator is represented with a dotted line because it will not be necessary.

The other component that communicates with the database is the backend of *SmartVis4.0*. This component receives requests from the frontend by the HyperText Transfer Protocol (HTTP). The backend receives the request, interprets it and sends the necessary queries to the database, in order to receive from the database the required data. The frontend receives the response from the backend in JSON format, since the frontend is a web interface and it uses TypeScript language, this is the most appropriate format to exchange data.

4.4.2 Intra Blocks - DB

The system developed has some knowledge about the process of investment casting, more specifically the production method in *ZOLLERN & Comandita*. Thus, some relevant data is interpreted and reorganised in the database with the use of triggers. Although, this involve a new technology, the alternative of the use of triggers is multiple periodic queries. However, this approach is more exhaustive to the communications with database.

In other cases a summary resulting of the data interpretation is created, that helps the user to get some information without the need to be constantly analysing and interpreting the data collected in the shop floor. The knowledge that the system has from the foundry department at *ZOLLERN & Comandita*, gave it the skills such as those that would be found in an expert system.

The information that results from the interpretation of the data collected from the shop floor is organised into three big parts. These are the storage, the rotary furnace and the alerts. The first two use the data collected to save, in an easy way, the data that will be interesting to the user. This reorganisation of the information will spare some processing needed each time that the user requires that information. The last part uses its knowledge about the factory process and generates summaries when it is necessary to inform the factory manager when something needs attention.

In what follows more detail will be given about how each part was implemented.

4.4.2.1 Storage

Between the autoclave and the rotary furnace in the foundry department at *ZOLLERN & Comandita*, the POs wait in storage. To know exactly which trees finished their cycle on the autoclave and are waiting to go to the rotary furnace, a table was created on the database called *Storage*, presented on Figure 4.13.

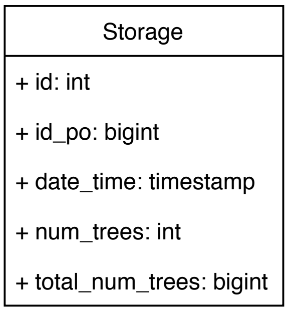


Figure 4.13: Table with the data from the storage.

The attribute *po_id* identifies the id of the PO that is in the storage and the *total_num_trees* indicates the total number of trees that the PO has. Since all trees from a PO may not go through the autoclave at the same time, the attribute *num_trees* indicates the number of trees that are in the storage zone. The timestamp *date_time* saves the instant that the first tree from that PO arrives at the storage.

The information saved on the *Storage* table is inserted by a trigger on the *Autoclave* table. The trigger is represented in Figure 4.14.

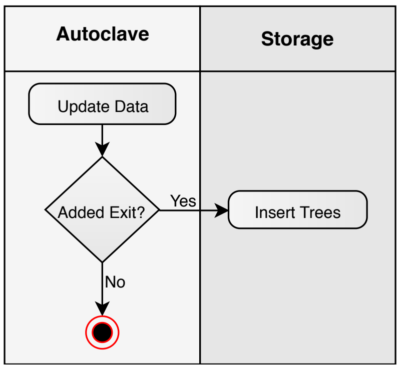


Figure 4.14: Swim lane diagram about trigger for storage table.

Each time that the timestamp of the attribute *exit* on *Autoclave* table is updated, the trigger is activated. It selects all trees that exist on the shelf that is exiting the autoclave and identifies the ids of the POs that they correspond. If there is an entry on *Storage* table from that PO, the number of trees is updated, if not, the id of the new PO is inserted with the respective number of trees.

4.4.2.2 Rotary furnace

The rotary furnace is very human dependent and its movements are somewhat unpredictable. The manager of the foundry department needs to track the movement of each tree in each instance, but it is hard to control that just by storing each advance of the buds in the rotary furnace. As mentioned in Subsubsection 4.3.2.2, two tables were created, *Bud_Zone* and *RotaryFurnaceState*, that save information that helps the access to relevant data. In this way, each table is updated with triggers, that are presented in Figure 4.15.

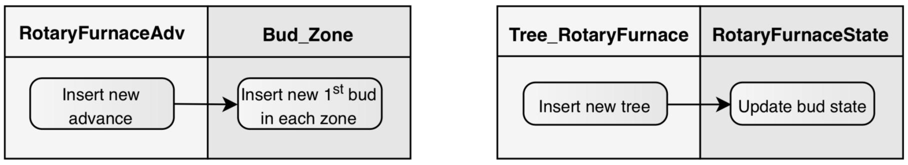


Figure 4.15: Triggers on the rotary furnace database.

The *Bud_Zone* table, stores the bud at the start of each zone over time. This information is generated with a trigger on each rotary furnace advance, which is represented by the scheme of left side of Figure 4.15. Just by knowing the bud that is in the furnace entrance at each moment, it is possible to know the ids of the remaining first buds of each zone, using Equation 4.1, where e is the id of the entrance bud and n_k is the number of buds from the entrance to the start of zone k and b_k is the id of the bud at the start of zone k.

$$\begin{aligned} b_k &= ((((e+n_k-1) \bmod 24)+24) \bmod 24)+1, \\ k &= \{2,3,4,5\}, \\ n_k &= \{5,12,17,21\} \end{aligned} \tag{4.1}$$

To simplify obtaining the current state of the rotary furnace each time that a new tree enters or exits the rotary furnace, the *RotaryFurnaceState* table updates the attribute *id_tree* for the respective bud and position, where the entered or exited, with a trigger.

4.4.2.3 Alerts

The entries of the *Alerts* table are entirely generated by triggers. Thus, the database will always be responsible for controlling the data and inserting a new alert in the database when something does not work correctly.

The database is able to identify all problems mentioned in Section 3.4. Each problem is identified by each trigger and has a predefined alert level. The Table 4.1 indicates, for each problem presented in Section 3.4, its respective level. The level could be one, two or three, with one corresponding to an informative message, two for a median alert and three for an important or dangerous alert. Each level was attributed for each identified problem.

Table 4.1: Level of the problems listed.

Furnace	**Problem**	**Level**
Autoclave	Delay less or more than 8 seconds to arrive 9 bar	2
	Pressure in steam generator is not at 12 bar	3
	More than 10 seconds between entry and start	2
	Pressure is not at 9 bar during the cycle	3
	More or less 15 minutes at 9 bar	3
Rotary	Tree less than 15 minutes in first zone	3
	Temperature of zone 1 and zone 2 different of 1000°C	2
	Temperature of zones 3, 4 and 5 different of the expected	2
	Failure of the oxygen in the first zone	3
	More than 2 hours inside the rotary furnace	1

4.5 Relevant implementation details

The principal components of the developed work and how these are implemented, are described within this section. First, the events regarding the rotary furnace implemented in the simulator are covered. After that, the visualisation tool developed to represent the state of this furnace is described. Finally, the most relevant alerts, implemented with triggers, will be explained.

4.5.1 Rotary furnace - Simulator

As previously mentioned, the simulation of the rotary furnace events are not linear and it is difficult to normalise its work. In this way, an algorithm to simulate this part of the rotary furnace was developed in order to be the closest as possible to the work in the shop floor. This will simulate the work of the furnace, without any failure.

The rotary furnace follows Algorithm 1 and executes it in an infinite loop. The variable *newPO* saves the id of a new PO that is available in the storage and *treesPO* is an array with the ids of all trees that belongs to the *newPO*. The next trees that will be placed in the rotary furnace are the trees which were saved in *queue* and were removed from the array *treesPO*. With the number of trees that will be placed next in the rotary furnace, it is easy to know how many buds are necessary to be available, so the number of buds that the trees will need is saved in *numBuds*. The variable *free_buds_exit* and *available_buds_advance* are flags that indicate, respectively, if there are *numBuds* empty in the rotary furnace's exit and *numBuds* in the first zone ready to advance.

The part of the simulator explained in Algorithm 1 begins verifying the next PO that is available to enter the rotary furnace. Sometimes, the PO needs to be divided. This happens in two

Algorithm 1: Function responsible for simulating the rotary furnace data

```
if empty queue then
    newPo = PO from Storage table;
    treesPO = Array with trees from newPO;
    if newPO.weigth > 200 then
        queue = array trees which total weight < 200;
        treesPO = array treesPO without trees that are in queue;
    else
        queue = treesPO;
        treesPO = null;
else
    numBuds = round($queue.length/3$);
    free_buds_exit = verify the availability of buds in exit;
    available_buds_advance = verify the availability of advances in first zone;
    if available_buds_advance and free_buds_exit then
        Add trees from queue to rotary furnace;
        if treesPO then
            queue = treesPO;
            treesPO = null;
        else
            queue = null;
    else if available_buds_advance then
        Remove trees from rotary furnace;
```

possible cases. The first one, if the PO is too large, that means that the metal is too heavy, more than 200Kg, to be produced in one time. The second one, if the number of trees is bigger than the number of spaces available in the first zone, because the trees need to stay fifteen minutes in this part of the rotary furnace. After that, if necessary, one part is with the maximum number of trees that can enter at the same time, and the second part with the remaining.

When the simulator of the rotary furnace has the set of trees that will begin their rotary furnace cycle, the simulator verifies if there are trees at the end of the rotary furnace that can be used for the new trees that need to enter. After this verification, and removal if there are any trees at the exit, the simulator will verify if the rotary furnace can advance the trees of the first zone, and if there are buds available to receive the new trees. If the conditions are verified, new trees are added to the rotary furnace, if not, the cycle of the Algorithm 1 continues to check the availability to insert new trees in the rotary furnace cycle.

4.5.2 Rotary furnace diagram - *SmartVis4.0*

The diagram of the rotary furnace in *SmartVis4.0* was developed in order to be as similar as possible with the architecture of the real furnace, as mentioned previously. That way, it will be better for the foundry manager get the necessary information about the state of the rotary furnace.

The diagram of the rotary furnace in *SmartVis4.0* has its base structure that is presented in Figure 4.16.

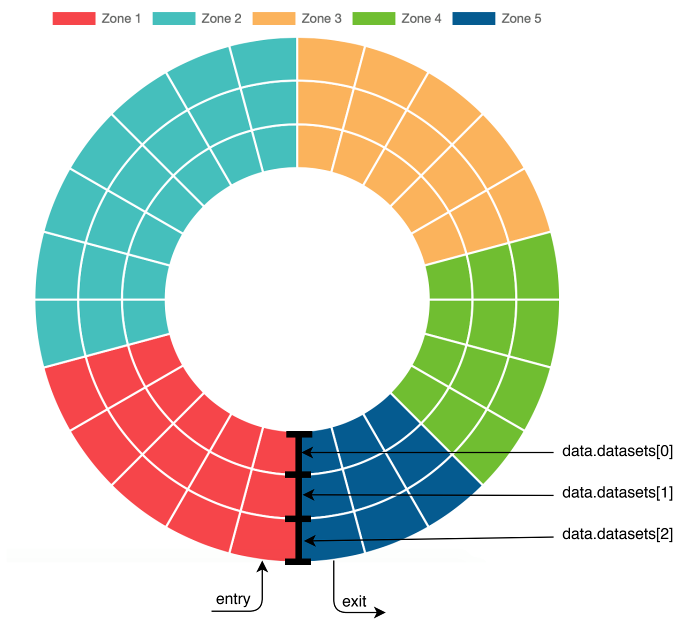


Figure 4.16: Base diagram of the rotary furnace in *SmartVis4.0*.

To develop this visualisation diagram, the framework *Chart.js* was used. As mentioned in Subsection 2.3.3, this framework uses the TypeScript language, which allows the developing of an appealing and dynamic visualisation module. This framework uses the *Chart* class to represent different types of graphs just by initialising the attributes of the class with specific values. These principal attributes are *type*, *data*, *options* and *plugins*. With these it is possible to define multiple features for each graph.

The rotary furnace is round, as demonstrated in Figure 3.2. The chart in the framework *Chart.js* that has the shape more similar with the rotary furnace is the *Doughnut Chart*, so the attribute *type* is initialised as *'doughnut'*. The attribute *data* is what defines how the graph will be presented. This can be composed by more than one dataset, organised in a vector. For each dataset it is possible to define different features. The attribute *data*, of doughnut chart that was created to represent the rotary furnace, was initialised as presented in Code 4.1.

Code 4.1: Initialise attribute *data* in the class *Chart*.

```
data: {
        datasets: [{
          data: sizesVector,
          backgroundColor: coloursVector,
          label: labels3,
        }, {
          data: sizesVector,
          backgroundColor: coloursVector,
        label: labels2,
      }, {
        data: sizesVector,
        backgroundColor: coloursVector,
        label: labels1,
      }],
    },
```

The doughnut chart is round, like the rotary furnace, and it has a free space in the centre, which makes each part of the graph look like a bud. The number and the size of each part of the graph is defined with the values that the attribute *data*, inside the *datasets*, is initialised. Since this graph is used to compare the dimension of different values, the attribute *data* needs to be initialised with an array of numbers. The dimension of each bud of the doughnut chart follows Equation 4.2, where i is the index of the *data* array, $data[i]$ is the value in *data* array with index i and the $size_{bud}[i]$ is the dimension of the bud in a range $[0:1]$, where 1 corresponds to the whole graph and 0 does not appear in it.

$$size_{bud}[i] = \frac{data[i]}{\sum_{j=0}^{n} data[j]} \tag{4.2}$$

Since this part of the graph should represent each bud of the furnace that is composed by twenty four buds, all with the same size, the array *sizesVector*, presented in Code 4.1, should be initialised with twenty four equal values. The doughnut chart was initialised with three layers, corresponding to the three possible positions of each bud. For this, the *datasets* is initialised by a set of three datasets representing three aligned layers. Thus, doughnut chart gets the shape of the rotary furnace.

The rotary furnace is composed of five zones. Each zone has its number of buds and some other specific features, as it was mentioned in Section 3.3. For an easier interpretation of the diagram, it is necessary to highlight each zone. To do that, a vector of twenty four colours was created, that correspond to the colour of each zone, and the buds of the each zone have the same colour. The Code 4.2 demonstrates how the colour vector is created. The background in all datasets was initialised with same vector of colours.

Code 4.2: Vector of colours.

```
colours = [
  // Zone 1 -> 5 buds
  red, red, red, red, red,
  // Zone 2 -> 7 buds
  turquoise, turquoise, turquoise, turquoise, turquoise, turquoise, turquoise,
  // Zone 3 -> 5 buds
  yellow, yellow, yellow, yellow, yellow,
  // Zone 4 -> 4 buds
  green, green, green, green,
  // Zone 5 -> 3 buds
```

```
    blue, blue, blue
  ];
```

The next phase of the creation of the rotary furnace representation in *SmartVis4.0*, is to get the state of the furnace from the database. With that, the label of each bud is initialised based on the existence of tree in the specific position of the rotary furnace. For that, the function *initLabels()*, that is partially represented in Code 4.3, was created.

Code 4.3: Function to initialise labels with the existing trees in the rotary furnace

```
initLabels() {
    // res -> Result of the getStateRotationOven()
    this.getStateRotationOven().then(res => {
      this.resData = res;
      // bud -> first bud of the first zone
      this.getFirstBud().then(bud => {
        let i = 0;
        let tempPos = 0;
        const entryBud = this.getAttribute(this.getAttribute(bud[0], 'fields'), '
            entry_bud');
        this.resData.forEach(y => {
          const x = this.getAttribute(y, 'fields');
          tempPos = this.getAttribute(x, 'position_bud');
          i = (this.getAttribute(x, 'bud') - entryBud  + 24) % 24;
          if (this.getAttribute(x, 'id_tree') !== null) {
            this.label[tempPos - 1][i] = this.getAttribute(x, 'id_tree') - (Math.
                floor(this.getAttribute(x, 'id_tree') / 1000) * 1000);
          } else {
            this.label[tempPos - 1][i] = null;
          }
        });
        this.initChartDonut();
      });
    });
  }
```

Firstly, this function gets the state of the rotary furnace with the *ids* of the trees that are in each position in each bud, and *resData* vector. If there are no trees, the corresponding value is *null*. It is necessary to know the *id* of the bud that is in entry of the furnace, in order to represent each tree in the respective place of the diagram. This corresponds to the label with the index equal to zero. After that, it is possible to initialise the matrix with all labels. The relation of the matrix and each part of the diagram is represented in Figure 4.17.

To initialise the matrix with all values of the trees in their correct place, each index of the *resData* vector is filled. For each element the respective *bud id* and position are accessed. With the value of the position, it is possible to get the index of the line in the matrix of labels. To know

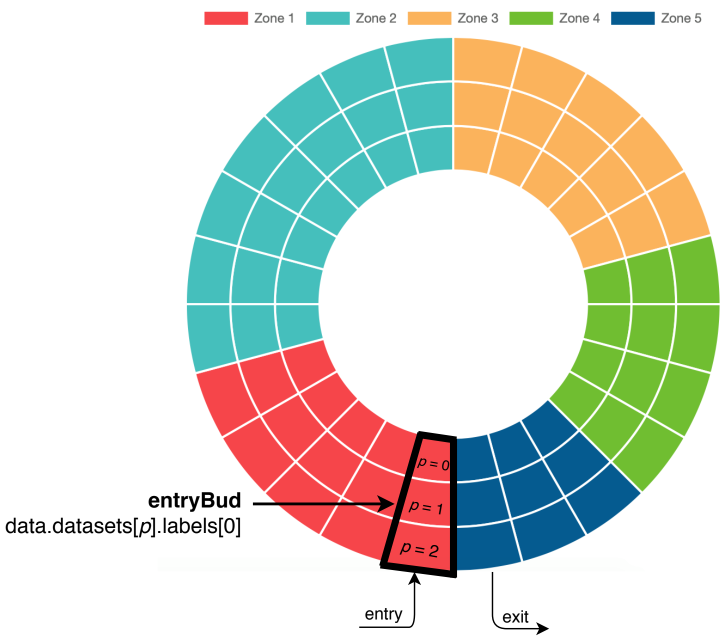


Figure 4.17: Relation between the attributes and the real diagram in *SmartVis4.0*.

the index of the bud in that moment, Equation 4.3 is used.

$$index = (idBudTree - entryBud + 24) \bmod 24 \tag{4.3}$$

After the *resData* vector is filled with the state of the rotary furnace, all the labels are ready to be presented. For a better visualisation, the opacity of each bud is changed when it does not have a tree, in other words, when the label is equal to *null*. The opacity of the respective bud is changed to 50%.

The other two parameters of the *Chart* class, *options* and *plugins*, where initialised with features that give details in the presentation of the diagram and user interaction. One example of that is when there is no tree in a specific position and the user places the mouse over that position, the label does not appear. Other feature defined within these parameters, is the text that appears on top of each position, in this case, the *id* of the corresponding tree, as it is shown in Figure 4.18 and this is responsive for different screen sizes. The information that will be shown in the label when the user places the mouse over a position with a tree, is also defined in these parameters.

4.5.3 Example usage

To guarantee that all data arrives correctly to the user, there are a lot of communications and functions that exists behind the browser. These functions and communications could be triggered by the user, or triggered by a specific function that runs periodically in the frontend server. Figure 4.19 represents as an overview of all communications that could occur in parallel.

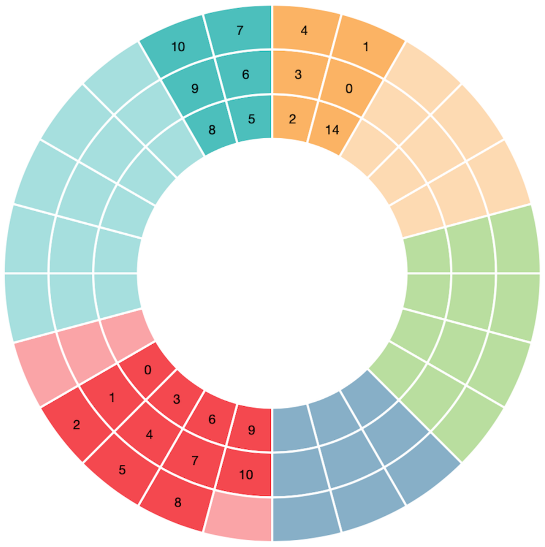


Figure 4.18: Final appearance of the rotary furnace diagram in *SmartVis4.0*.

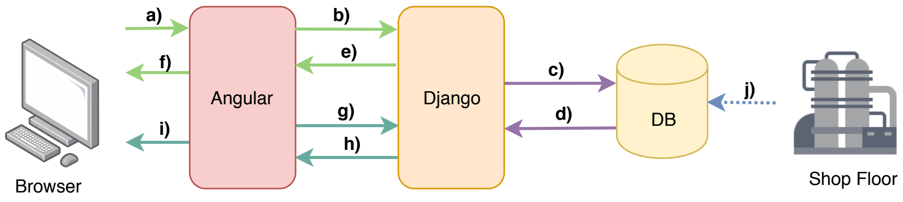


Figure 4.19: Scheme with all communications when the *SmartVis4.0* works.

As it was mentioned above, the communications could be trigger by the user. Considering this case, its respective communications are represented by the green arrows in Figure 4.19. The user begins to select what wants to see and the browser sends the information to the frontend server, Angular, corresponding to the **a)** arrow. The interaction of the user with the browser, will triggered a specific function in the frontend server. The respective function sends a HTTP request to the backend server, Django, corresponding to the **b)** arrow.

The backend server is always waiting for a HTTP request. When one arrives, the backend server activates the corresponding function of the request. This function will be sent a query to database, **c)** arrow, and the database returns the respective required data, **d)** arrow. At this step, Django serialises the data in JSON to sends it to the frontend server, **e)** arrow.

When an answer of a specific request arrives in Angular, the frontend server interprets if it is an error or not. If the answer was an error, the respective message will be presented to the user, but, if it was an "OK", the Angular initialises the variables of the graphs with the respective parameters from the answer. The information that the frontend server sends to the browser, is represented by the **f)** arrow.

Some events could be triggered by functions that exist in the frontend server, as an example, the function to check if exists new alerts in the database. This part of the frontend server makes periodically to the backend server, **g)** arrow. The backend server receives the join and verifies in the database if new alerts exists. After that verification, sends the respective answer to the frontend server, **h)** arrow. The Angular receives the answer and interprets it. If no alerts exists, the Angular does not do anything, but, if answer brings new alerts, according to their levels, they are sent to the browser and presented to the user in the *SmartVis4.0*, **i)**.

In the case of a user makes a request to see data in real time, this will be triggered a function that will make periodical to the backend, like the function of the alerts. However, the function of alerts runs all the time, while if it was a request of the user and if he wants to stop to see data in real time, the function could be suspended.

In parallel of all communications and functions that guarantee the correct operation of the *SmartVis4.0*, the data collected in the shop floor is stored in the database, **j)** arrow. With the existence of the triggers in the database that were explained above, once the data arrives in the database, it is immediately reorganised and interpreted, in order to guarantee that the information arrives at the user, as soon as possible.

Chapter 5

Experiments and Results

Testing the system is fundamental to guarantee that all requirements were fulfilled. To help this task, some features were added in the simulator, in order to create specific situations. For validating all principal objectives of the system, all use cases where tested and validated.

Finally, the client review is crucial for the software validation. In this work, *ZOLLERN & Comandita* was the client. Some software validations were made during the software development, in order to discuss some details and to make some small validations to guarantee that the work is complying with the client's needs.

5.1 Simulation with faults

The simulator was developed to generate data as closely as possible with what's happening in the shop floor but without failures. However, the process does not work correctly all the time and sometimes, some failures can occur. The possible failures were identified and compiled into a list, presented in Section 3.4. *SmartVis4.0* has the capacity to show data, but it was developed to identify the problems listed and to present alerts when some occur.

In order to test in *SmartVis4.0*, the alerts when problems occur in the shop floor, while real data does not exist, it was added the possibility of generating failures in the simulator. The failures are injected during the work of the simulator with the changing of the correct work of the process with failures before saving the data in the database.

Each problem identified was tested isolated to guarantee that *SmartVis4.0* can easily identify all the cases. To help this implementation, each problem was developed in their respective part of the simulator and it is always integrated into the simulator, but it is isolated to the remaining parts by a condition.

For each problem, a probability is defined that failure could occur, p_x that is a number between zero and one. The Figure 5.1 demonstrates the possible range of the occurrence of the failure according to the value of the p_x. To define if the failure occurs, a random value is generated between zero and one, that follows a normal distribution. If that number is lower or equal to p_x, the failure will occur, if not, the simulator will work normally.

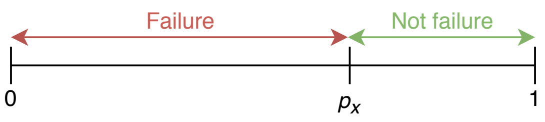


Figure 5.1: Probability to a failure occur

To test some specific failure, its probability of occurrence is defined with a high value to guarantee that the failure happens in a short time, however, the probability is not defined equal to one, which makes the failure occur all the time, because it is more interesting that the simulator generates failures sometimes, and other times it works correctly. Thus it is possible in the same simulation the ability of *SmartVis4.0* to identify the problems when they happen. When testing a failure, the probabilities of the other failures that are not under study are equal to zero, to guarantee that they never occur, so they don't interfere with the failure under observation.

After all the possible problems were tested, it was defined, for each one, the probability of the failure different to the zero, but under 0.5 to make some failures occur and generate some alerts for any problem. Thus, the simulation is closer to the real world and what really could happen when *SmartVis4.0* will use the real data monitored in real time.

5.2 Requirements Validation

In order to validate the system developed with all parts integrated, the requirements presented in Section 4.1, functional and non functional will be validated with images of the final appearance of *SmartVis4.0*.

5.2.1 Functional

The functional requirements define functions of the system. For *SmartVis4.0*, they were determined in Section 4.1. To test the system and if it achieves the main objectives, each requirement will be analysed with examples of *SmartVis4.0* usage. All the use cases were tested with simulated data.

UC 1. *The user wants to see data that was collected from a specific furnace, selecting a PO or a tree id.*

With *SmartVis4.0*, it is possible to visualise data that represents the passage of a certain PO by the existing furnaces. As an example of this case, the autoclave was selected. For that autoclave, the user wants to see what was the pressure in the main space during the cycle of the PO with *id* equal to 8127326. Figure 5.2 contains a graph of the pressure variation in the main chamber during a full autoclave cycle, corresponding to a given PO. This information is preprocessed in the backend, which determines in which shelves that order was stored, before entering the autoclave, and then obtains the cycle data for those shelves. In this case, only one graph is shown because the whole order was stored in a single shelf.

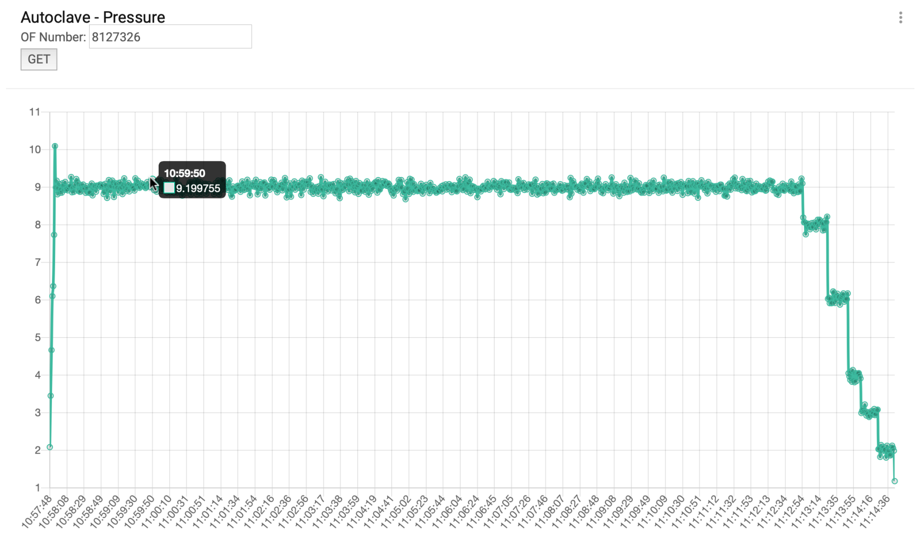


Figure 5.2: Variation of the pressure in the main chamber of the autoclave during the cycle of the shelf containing PO 8127326.

UC 2. *The user wants to see data monitored that is being collected in real time to control the shop floor.*

Control the data in real time is easier with *SmartVis4.0* in real time mode. With that, the platform receives new data regarding the furnaces, with a refresh rate of a second.

In the case of the pressures and temperatures from each furnace, since these parameters are constantly monitored, a temporal window of data is maintained in *SmartVis4.0*. For the autoclave, the window has 30 minutes because the period of its cycle is more or less 20 minutes, therefore, it is possible to see a complete cycle. In the case of the rotary furnace, since the variations of temperatures are less than the autoclave and a complete cycle of a tree inside of it is higher, there is an one hour time window.

Any time that a new value is collected in database about one of the furnaces, it is added to *SmartVis4.0* and the last value presented is removed from the data visualiser, keeping the size of each window.

UC 3. *The user wants to see the state of the rotary furnace in real time.*

One of the features requested by the managers of the foundry department was the ability to quickly check the state of the rotary furnace, which was implemented through the scheme in Figure 5.3. This diagram shows how the furnace is loaded in real time, but it can also be used to check the progress over time of a given PO within the furnace. For each position of each bud, one can see if it is loaded and the number of the tree in that position. The chart is also interactive, allowing the user to scroll the mouse over to obtain a quick overview of the contents, or to click in a bud and obtain more detailed information.

UC 4. *The user wants to be alerted when something does not work correctly.*

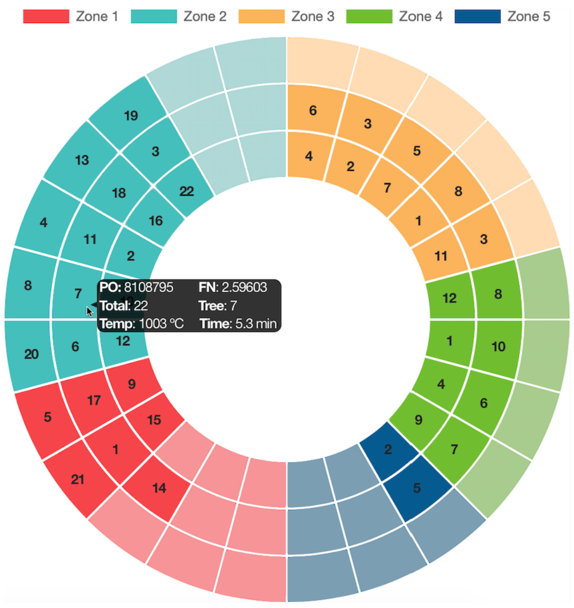
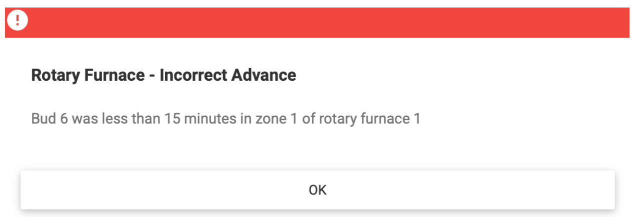


Figure 5.3: Representation of the rotary furnace in the data visualisation system.

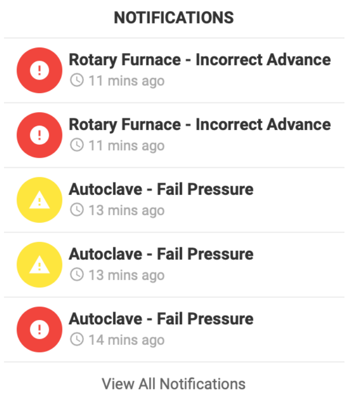
To verify this use case, failures were introduced in the simulator following the ideas presented in Section 5.1, thus activating the triggers that generate alerts in the database. These alerts are received in *SmartVis4.0* and are shown both in the form of a pop-up and in a drop-down notifications menu, as depicted in Figure 5.4.



Rotary Furnace - Incorrect Advance

Bud 6 was less than 15 minutes in zone 1 of rotary furnace 1

(a) Pop-up notification



NOTIFICATIONS

Rotary Furnace - Incorrect Advance
11 mins ago

Rotary Furnace - Incorrect Advance
11 mins ago

Autoclave - Fail Pressure
13 mins ago

Autoclave - Fail Pressure
13 mins ago

Autoclave - Fail Pressure
14 mins ago

(b) Drop-down notifications

Figure 5.4: Notifications on Visualisation Tool.

All problems presented in Table 4.1 was implemented. Each one was tested singly, with the probabilities of occurrence, explained in Section 5.1, in order to verify that all respective alerts work correctly in *SmartVis4.0*.

Table 5.1: Combinations to test the platforms compatibility requirement.

	Firefox	**Chrome**	**Safari**
Computer	test1	test2	test3
Mobile phone	test4	test5	test6

5.2.2 Non-Functional

Some of the non functional requirements that were mentioned in Subsection 4.1.2 are essential to fulfil some of the principal objectives of *SmartVis4.0*. They are **platform compatibility** and **performance**.

In order to test the **platform compatibility** of *SmartVis4.0*, three different browsers were used. They are *Firefox*, *Google Chrome* and *Safari*. Each browser was tested in computers and in mobile phones to verify the responsiveness that makes web pages render well on a variety of devices and window or screen sizes. The Table 5.1 identifies the test for each combination.

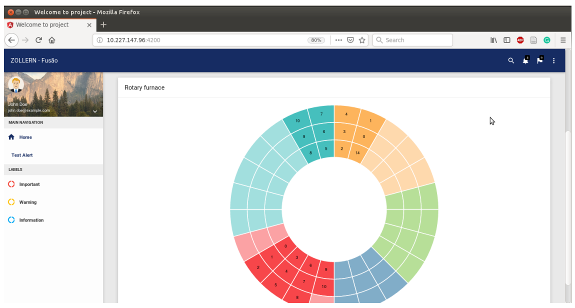
The results of each test are presented in Figure 5.5 and Figure 5.6, that shows how the *SmartVis4.0* is presented in each browser in computers and in mobile phones, respectively. It is possible to verify that the web page is exactly equal in the tested browsers. When the page is accessed in devices with different sizes of the screens, *SmartVis4.0* is responsive, because it adapts its menus and the size of the page elements depending on the screen, it is possible to see comparison results in Figure 5.5 and in Figure 5.6.

The **performance** requirement is fundamental in *SmartVis4.0*, as it shows data in real time and the response time of the system needs to follow the data arrive rate. Since data arrives every second, it was verified that *SmartVis4.0* is able to present data at that rate, so the performance requirement was fulfilled.

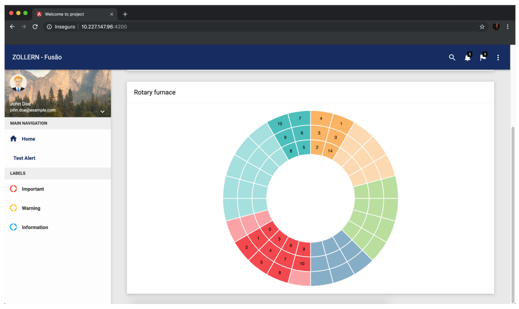
5.3 Validation by the client

At the time of writting, *SmartVis4.0* is not installed in the *ZOLLERN & Comandita*, because there is no collected data. Since the only data that exists is the data generated by the simulator, *SmartVis4.0* wil be usefull in future for the company.

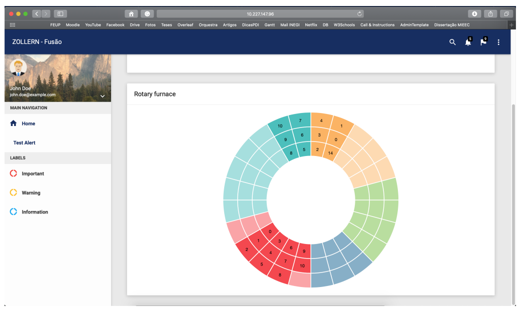
However, the future users of *SmartVis4.0*, the foundry department managers, have been following the evolution of the tool in order to guarantee that it is developed with the necessary features to help their work. At this moment, they consider that *SmartVis4.0* is a step in the right direction. The funcionalities of *SmartVis4.0* will facilitate their work, when the real data is available.



(a) Test 1 - Firefox

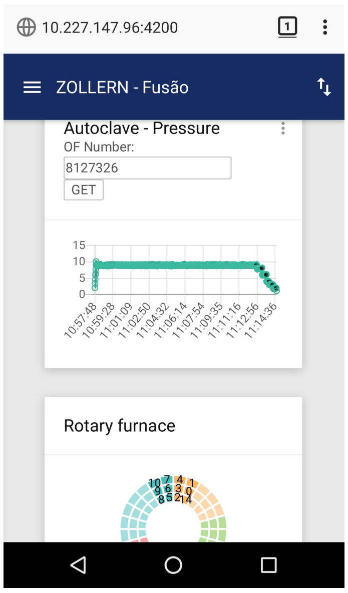


(b) Test 2 - Google Chrome

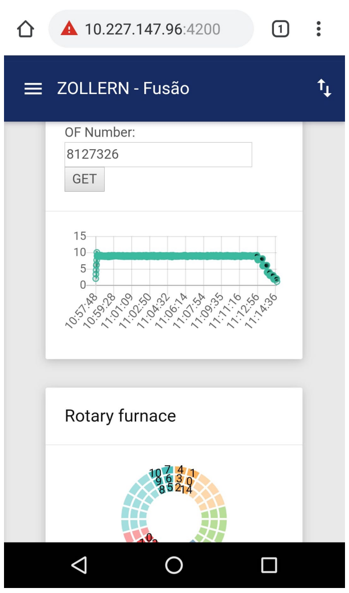


(c) Test 3 - Safari

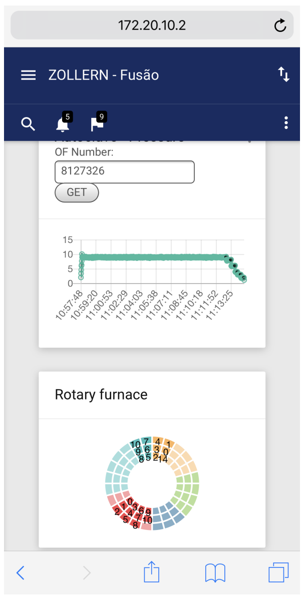
Figure 5.5: Tests in computer using different browsers.



(a) Test 4 - Firefox



(b) Test 5 - Google Chrome



(c) Test 6 - Safari

Figure 5.6: Tests in mobile phone using different browsers.

Chapter 6

Conclusions

As demonstrated, it is essential that data visualisation systems are developed to facilitate access and interpretation of the large amounts of acquired data. This work proposes a data visualisation system, *SmartVis4.0*, that allows the company *ZOLLERN & Comandita* the possibility to understand what happens on the shop floor in real time in a simpler and more intuitive way. Wherever the users are, they can know what is being produced at that moment and verify if the whole process is following its normal operation.

A study of the recent trends in web applications development made *SmartVis4.0* a state of the art web platform. Ideas like responsiveness and single page application were taken into account during the development of this tool. This allows the manager of the foundry department an easy and quick access to the platform, in any device that has a browser. Independently of the size of the screen, *SmartVis4.0* is responsive, so the page could adapt to be more user friendly. The software can easily be extended mainly due to the MVC concept used in its development. Scalability is expected to be good but no tests were conducted. Performance of the battery held device was taken into consideration as no polling operations are used.

The approaches taken have been validated by the client. *SmartVis4.0* allows the users to follow a given production order through the foundry department and to better understand the probable origins of high defect rates. It is possible to know exactly when an order passed through each furnace and the parameters registered during its processing. It will, therefore, be easier to see whether all the rules of the manufacturing process have been met or not.

Existing tools have some similar features to the visualisation tool developed in this work, like seeing in real time data saved in a database. However, *SmartVis4.0* has some schemes developed specifically to be applied at *ZOLLERN & Comandita*, to help the interpretation of what is happening in the shop floor. In addition, the system developed is an expert system, it compares process data with expected parameters and generates alerts when something is wrong, so that the managers can take immediate action.

SmartVis4.0 is expected to run without modification once real data is available. With this tool, the company may also conduct a cause-and-effect study in the foundry department. This will improve its manufacturing process, making it more efficient and decreasing the percentage of defective parts.

The developed software is expected to be adequate to other companies of the same process. The strategies for analysis and software development seem adequate for other cases where development of (smart) industrial visualisation tools are of interest.

6.1 Future Work

Since this work was developed in an initial phase of Industry 4.0 revolution at *ZOLLERN & Comandita*, there is still a long way to go. To improve *SmartVis4.0*, the following is proposed:

- Adding push notifications so that *SmartVis4.0* can alert the users when they are not running the application;
- Adding user authentication to protect the information that exists in *SmartVis4.0* from everyone that is inside the company network and does not have permission to access;
- Add the possibility for each user to define the alerts that they want to see;
- Adding the week plan of the foundry department in *SmartVis4.0*;
- Comparing the week plan with what is really happening in the shop floor, in order to identify deviations.

Appendix A

Provisional version of paper accepted for ROBOT2019

Smart Data Visualisation as a Stepping Stone for Industry 4.0 - a case study in investment casting industry

Ana Beatriz Cruz[1], Armando Sousa[2], Ângela Cardoso[3], Bernardo Valente[4], and Ana Reis[5]

[1] FEUP
[2] INESC-TEC and FEUP
[3] INEGI
[4] Zollern & Comandita
[5] INEGI and FEUP

Abstract. With present day industries pressing for retrofitting of current machinery into Industry 4.0 ideas, a large effort is put into data production, storage and analysis. To be able to use such data, it is fundamental to create intelligent software for analysis and visualisation of a growing but frequently faulty amount of data, without the quality and quantity adequate for full blown data mining techniques. This article case studies a foundry company that uses the lost wax method to produce metal parts. As retrofitting is underway, modelling, simulation and smart data visualisation are proposed as methods to overcome data shortage in quantity and quality. The developed data visualisation system is demonstrated to be adapted to the requirements and needs of this company in order to approach full automation ideas. Such data visualisation system allow workers and supervisors to know in real time what is happening in the factory, or study the passage of manufacturing orders for a specific area. Data Analysts can also predict machinery problems, correct issues with slow changing deviations and gather additional knowledge on the implementation of the process itself.

1 Introduction

Nowadays, we are going through the fourth industrial revolution, also known as Industry 4.0. This is a digital revolution, whose main objectives are increasing the efficiency of operation and productivity, as well as increasing the level of automation, thus making the companies more competitive, as concluded in [1].

This digital revolution is driving the implementation of tools and intelligent platforms that produce a greater amount of data and information for analysis, as explained in [2]. The storage of large amounts of data allows an analysis of the conditions in which a product was created, as well as the use of machine learning techniques, to make an early prediction of the occurrence of product defects or machine failures.

However, not all companies are prepared for this type of revolution, because many have at least some very primitive processes, where human work and control predominate. An example are investment casting companies that use the lost wax casting method, where at least the last part of the process is mostly manual. In this type of manufacturing, the occurrence of failures in some sections, more specific in the casting department, is quite frequent and difficult to control. Generally, data acquisition and monitoring are also very scarce, due to the existing manufacturing processes.

For this project, the investment casting company *Zollern & Comandita*, was used as a case study, with the objective of developing a data visualisation system. The project began with a study of the manufacturing process. Then, because it is the section where the managers have less access to the information necessary to make their decisions, the focus was placed in the foundry department. Since data collection is at its early stages, it was necessary to develop a simulator, which produces relevant data, very close to what would be real.

Subsequently, an expert data visualisation system was developed, to allow an intuitive comprehension of the information. This analysis can be done in real time, in fact the system is capable of recognising deviations from what is expected and emits alerts when something goes wrong. The tool also allows to view data from a finished production order, which makes it possible to understand the conditions of its production, as well as to make the cause-effect study of its defects or qualities.

2 Industry Context, Objectives and Requirements

Lost wax casting is a method of producing metal parts with high precision. This process has eight stages of development until the final product is obtained, as shown in Figure 1 and demonstrated on the web page [3].

First, wax forms of the product to be manufactured are injected. Then one or more of these pieces are welded to a common trunk of wax, which is called the tree. Subsequently, several layers of ceramic are made around this tree. After the ceramic layers are thoroughly dried, the inner wax is removed so that the metal alloy can be poured in a liquid state into the tree. After the metal solidification, the ceramic is broken and the final product is obtained by separating the parts of the common trunk.

A study was done at *Zollern & Comandita* to understand which section has the most issues and the foundry was the one that stood out in the high number of incidences. This section consists of the phases of wax removal, tree sinterization, metal alloy preparation and pouring.

At the foundry, the trees are placed inside a furnace (autoclave), and are subjected to a pressure of 9 bar for 15 minutes. This high pressure removes about 95% of the wax inside the tree. After that, the trees are stored until there is room to move to a second furnace. This is a rotary furnace, where the trees are placed in small sections, called buds, with at most three trees per bud, and

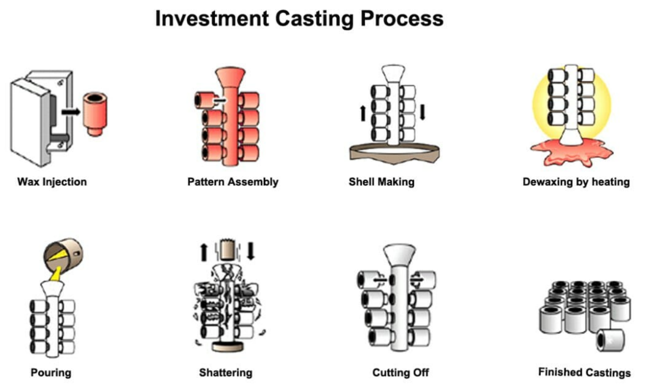


Fig. 1. Investment casting process, as shown in [4].

run through the five different zones within it until they reach the exit. Figure 2 represents this furnace and its structure.

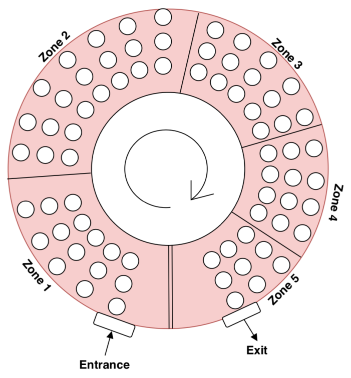


Fig. 2. Scheme of the rotary furnace.

The first zone of the furnace aims to burn the remaining wax inside of the tree. Trees must stay in this area for 15 minutes or more, depending on the type of metal part being produced. Zones 2, 3, 4 and 5 aim to increase the temperature and to sinter the trees to be cast. This phase is important to avoid thermal shocks and prevent the metal alloy from starting to solidify before reaching all areas of the interior of the tree.

Whilst the trees are in the rotary furnace, the metal alloy to be poured is being prepared in an induction oven. At the moment, there is very little data being collected from the induction ovens, only the temperatures of the alloy are measured and even that is done very sparsely. Once the trees have gone through all zones of the rotary furnace, which takes approximately 2 hours, and the metal alloy is prepared, pouring begins, but no data is collected.

Although there are ideas to eventually collect more data from alloy preparation and pouring, those phases are not included in this project, because the specific way in which the data will be collected is yet to be decided.

Figure 3 explains the scheme sequence of the foundry section. At *Zollern & Comandita* there are two autoclaves, two rotary furnaces and three induction furnaces.

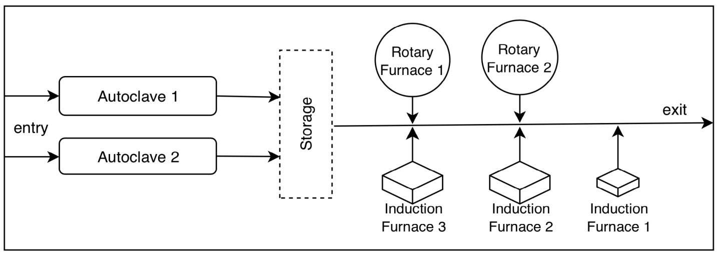


Fig. 3. Scheme of the main components of the foundry department.

This project's main objective is to develop a system where it is possible to visualise the collected data of the production line of the company. It must be able to filter large amounts of data from the past, to something more specific and easy to interpret by the user. This will help in understanding and better tracking what is happening on the shop floor. This system must also have the characteristic of being accessible by any device.

Based on the above objectives, three system requirements were devised:

R1. View data from a finished production order: Viewing data from a production order that has already been completed allows the managers to establish cause-and-effect relationships. These relationships will identify the characteristics in the manufacturing process that led to defects in the final product.

R2. Get real-time insight into what is happening on the shop floor: Understanding what is happening on the shop floor in real time is important for section managers. That way, they can quickly perceive what is being produced and if everything is working within the expected parameters.

R3. Emit alerts if certain variables are outside the expected values: Real-time warnings when important manufacturing conditions are not verified may allow the managers to quickly correct the problem, or at least to interrupt production.

3 State of the Art

With the high amount of data coming from Industry 4.0, some systems were developed to allow its visualisation. These systems have been created with in general purpose, so that they can be applied in any context, giving the user the possibility to choose the way of presenting the data. Following are two well known examples of this kind of data viewers.

The ***Q-DAS*** [5] software is specialised in the computerisation of statistical procedures with a focus on quality management applications including Statistical Processes. This software has several tools, such as *QS-STAT*, which allows the user to make statistical analyses of the collected data by producing analysis reports, which refer to the values collected by sensors at certain time intervals.

Another platform that allows visualisation of data is ***Grafana*** [6], which is a dashboard that allows the user to query, visualise, alert and understand data metrics, regardless of where they are stored.

Despite their wide usage, these two platforms are for local use, that is they can only be used in a computer that has the software installed. As such, these tools are not always the best for companies that are in the initial stages of implementing Industry 4.0 ideas.

4 Data Structure

Data is one of the main elements for the development of this work. As such, an architecture was created for the database, facilitating its access and interpretation.

4.1 Production Data

To study and analyse what is being produced, it is necessary to keep all the information from production orders. The tables shown in Figure 4 were created to store only informative/static data. A production order has associated a type of part to be produced and several trees. Once the trees enter the foundry department they are organised in shelves, hence there is another table that associates the trees with their respective shelves.

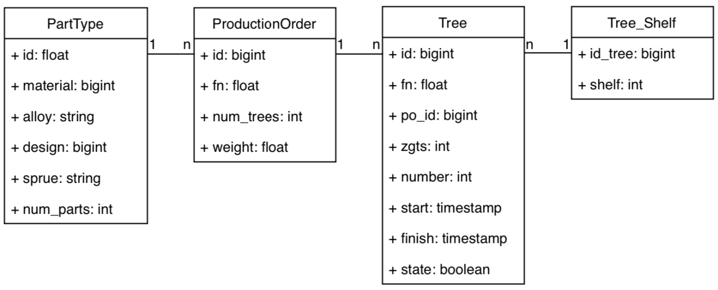
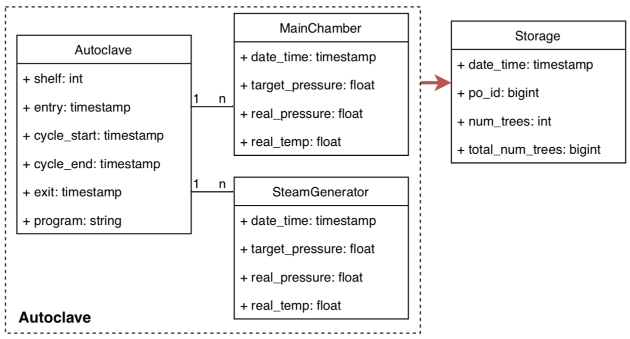


Fig. 4. Database scheme for the main production information.

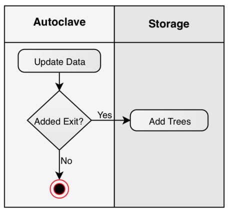
4.2 Autoclave Data

The foundry department starts with the processing of shelves of trees through the autoclave. The data of this part of the process is stored in three tables. The first table stores general information about the autoclave cycles, while the second and third tables store temperature and pressure data, for each second that the cycle lasts, in the main chamber and the steam generator. There is also a table which keeps information about the production orders that have already exited the autoclave and are ready for the rotary furnace. The data in this table is inserted through a trigger that runs when a shelf exits the autoclave.

Figure 5 demonstrates the architecture of autoclave part of the database and the trigger (red arrow) between the *Autoclave* table and the *Storage* table.



(a) Autoclave tables



(b) Trigger

Fig. 5. Database scheme for the autoclave section with the trigger from the *Autoclave* table to the *Storage* table.

4.3 Rotary Furnace Data

In the case of the rotary furnace, it is necessary to record the location of each bud over time. Just knowing the bud that is in the furnace entrance at each moment, it is possible to know the ids of the remaining first buds of each zone, using Equation 1, where e is the id of the entrance bud and n_k is the number of buds from the entrance to the start of zone k and b_k is the id of the bud at the start of zone k.

$$\begin{aligned} b_k &= ((((e + n_k - 1) \bmod 24) + 24) \bmod 24) + 1, \\ k &= \{2, 3, 4, 5\}, \\ n_k &= \{5, 12, 17, 21\} \end{aligned} \tag{1}$$

The tables for the rotary furnace are depicted in Figure 6. In particular, the table *Bud_Zone*, stores the bud at the start of each zone over time. This information is generated with a trigger on each rotary furnace advance, which is represented by the green arrow in Figure 6 and the scheme of left side of Figure 7. Thus, it is easier to know how much time each tree spent in each zone, given that we also know the bud in which each tree is loaded.

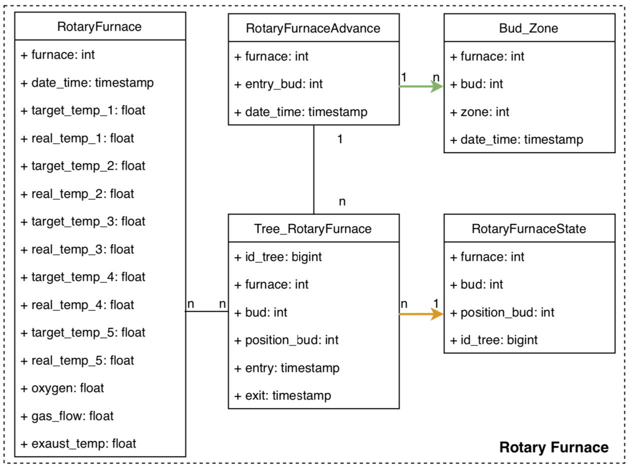


Fig. 6. Database scheme for the rotary furnace section.

To make reading the current state of the rotary furnace simpler, each time a new tree is inserted into the rotary furnace, the *RotaryFurnaceState* table is updated by a trigger, which is represented by the yellow arrow in Figure 6 and the scheme on the right side of Figure 7. The *RotaryFurnaceState* table always has the current state of each bud, saving the *id* of the trees it has at that moment, without maintaining history.

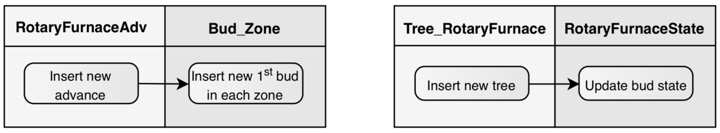
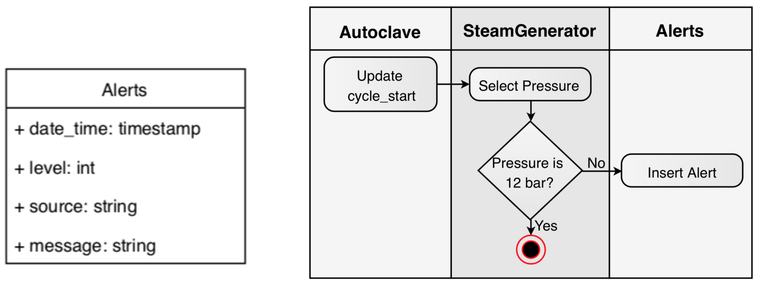


Fig. 7. Triggers on the rotary furnace database.

4.4 Alerts Data

In order for the managers to have instant alerts when an issue arises, the database also includes an alerts table, whose diagram is shown on the left of Figure 8. The entries of this table are entirely generated by triggers. For example, when a new autoclave cycle starts, if the steam generator pressure is not approximately 12 bar, a new alert is added to the table as shown by the scheme on the right of Figure 8.



(a) Table of alerts (b) Example of a trigger of an alert

Fig. 8. Alerts data.

5 Simulation Tool

At the start of this project, *Zollern & Comandita* did not have all the data necessary for the visualisation tool to be built. As such, it was necessary to develop a simulator to generate the required data. This tool is designed to automatically populate all tables in the database, with values very close to the expected ones.

For the tables with the information of the pieces to be produced, some real samples were used while others where simulated to look like the real samples.

After information of the manufacturing orders is inserted in the database, the system begins to simulate the entire manufacturing process of the foundry department in real time. The autoclave and the rotary furnace are simulated by different components, which operate independently and are only connected by the data in the database. The simulator generates the sporadic event of the arrival of a new shelf in the autoclave. As soon as a shelf exits the autoclave, it is added to the *Storage* table by a trigger. With this table, the simulator of the rotary furnace always knows the shelves it has available to use. Every time the rotary furnace has room for a new production order and there are completed orders in the storage, the simulator automatically loads the rotary furnace.

To generate furnace temperature data, normal distributions centred on the target values of each zone were used. The respective standard deviations were adjusted, depending on each situation so that the simulated data would be as similar as possible to real samples of the rotary furnace.

The simulation tool generates data and stores it in the database. This data can be used in developing any tool for the company, as well as studying what is more relevant to monitor. When real data is available, the company can store it in the database, replacing the simulator, and the remaining components of this project will continue to work.

6 Visualisation Tool

To be able to visualise the data, a graphical interface has been developed. The idea is to add features to this interface in due time, as the managers decide what is more relevant for them to see, but also as data becomes available and is integrated in the database, throughout the whole factory. A web interface was chosen so that it is easily accessed from any computer or mobile device that has a network connection. It was taken into account that the interface should be responsive to adapt to the size of the screen of the device being used.

The visualisation system was designed as a typical web application, with a front end component, responsible for presenting the information to the users, and a back end component, which manages the access to the database, as shown in Figure 9. Communication between these two components is achieved through HTTP (Hypertext Transfer Protocol) requests, which comply with the REST (Representational State Transfer) architectural style.

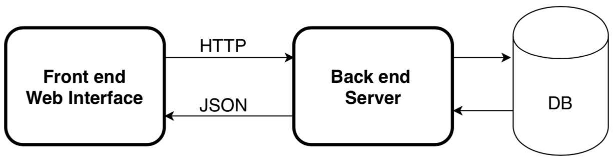


Fig. 9. Architecture of the Visualisation Tool.

For the development of front end component, the web framework *Angular* was used, because it offers a wide variety of tools that help with the development of such interfaces. In particular, the plugin *Chart.js* was used for the graphs, and the framework *Bootstrap* was used in order to easily obtain a responsive user interface.

Due to the fact that *Angular* is an MVC (Model View Controller) based framework, but also because this architectural style is well suited for web applications, the architecture of the front end follows the MVC pattern. As such, there is a representation of the data structure in the Model, which is simultaneously used as source of the information displayed in the View and queried by the Controller upon user interaction.

As for the server, the *DJango* framework was used, which follows the Model Template View (MTV) architectural style. MTV is similar to MVC, but depending on the source, there are some differences. In any case, the core ideas of code separation according to its purpose are the same. For the back end, the Model was used to represent the data and communicate with the database, while the View is responsible for replying to the HTTP requests from the front end using the JSON format. There is also a layer of business logic, that is responsible for the necessary processing of data before sending it to the front end. Typically, the Template layer is responsible for presenting the information to the user, not the content itself, but the way it is presented. For the back end component, this layer was not developed, because that responsibility belongs to the front end component.

7 Results

In this section, the results of the database, simulator and visualisation tool developed up to this point for the foundry department of *Zollern & Comandita* are presented, while validating the requirements elicited in Section 2.

R1. View data from a finished production order

With the developed tool it is possible to visualise data generated by the simulator that represents the passage of a certain production order by the existing furnaces. For example, one can see the data simulated for the autoclave and the resulting charts. Figure 10 contains a graph of the pressure variation in the main chamber during a full autoclave cycle, corresponding to a given production order. This information is preprocessed in the back end, which determines in which shelves that order was stored, before entering the autoclave, and then obtains the cycle data for those shelves. In this case, only one graph is shown because the whole order was stored in a single shelf.

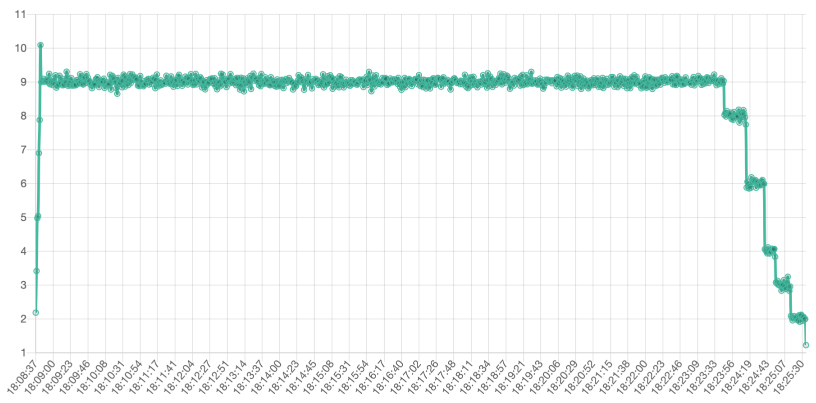


Fig. 10. Variation of the pressure in the main chamber of the autoclave during the cycle of the shelf containing production order 8127337.

R2. Get real-time insight into what is happening on the shop floor

One of the features requested by the managers of the foundry department was the ability to quickly check the state of the rotary furnace, which was implemented through the scheme in Figure 11. This diagram shows how the furnace is loaded in real time, but it can also be used to check the progress of a given production order within the furnace over time. For each position of each bud one can see, if it is loaded and the number of the tree in that position. The chart is also interactive, allowing the user to scroll the mouse over to obtain a quick overview of the contents, or to click in a bud and obtain more detailed information.

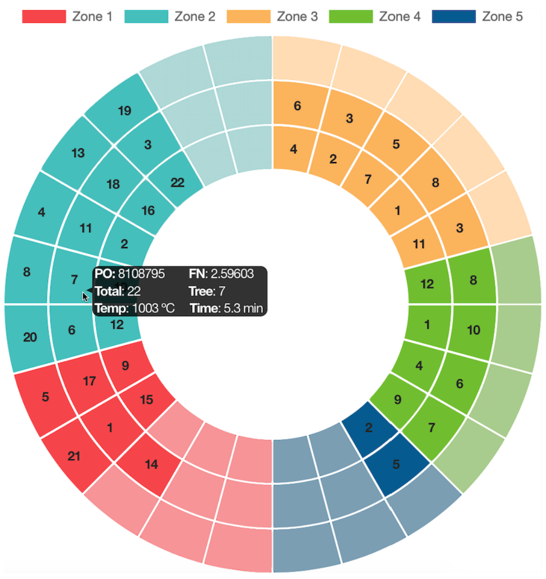
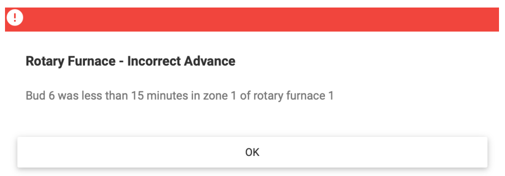


Fig. 11. Representation of the rotary furnace in the data visualisation system.

R3. Emit alerts if certain variables are outside the expected values

To verify this requirement, specific failures were introduced in the simulator, activating the triggers that generate alerts in the database. These alerts are received in the visualisation tool and shown both in the form of a pop-up and in a drop-down notifications menu, as depicted in Figure 12.

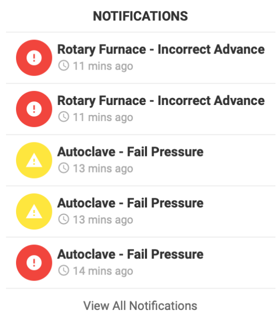


Rotary Furnace - Incorrect Advance

Bud 6 was less than 15 minutes in zone 1 of rotary furnace 1

OK

(a) Pop-up notification



NOTIFICATIONS

Rotary Furnace - Incorrect Advance
11 mins ago

Rotary Furnace - Incorrect Advance
11 mins ago

Autoclave - Fail Pressure
13 mins ago

Autoclave - Fail Pressure
13 mins ago

Autoclave - Fail Pressure
14 mins ago

View All Notifications

(b) Drop-down notifications

Fig. 12. Notifications on Visualisation Tool

8 Conclusions

As has been shown, it is essential that data visualisation systems are developed to facilitate access and interpretation of the large amounts of data acquired.

This work proposes a data visualisation system that allows the company *Zollern & Comandita* the possibility to understand what happens on the shop floor in real time in a simpler and more intuitive way. Wherever the users are, they can know what is being produced at that moment and verify if the whole process is following its normal operation.

The developed tool also allows the users to follow a given production order through the foundry department and to better understand the probable origins of high defect rates. It is possible to know exactly when an order passed through each furnace and the parameters registered during their processing. It will, therefore, be easier to see whether all the rules of the manufacturing process have been met or not.

Existing tools have some similar features to the visualisation tool developed in this work, like seeing in real time data saved in database. However, this visualisation tool has some schemes developed specifically to be applied at *Zollern & Comandita*, to help the interpretation of whats happening in the shop floor. In addition, the system developed is an expert system, it compares process data with expected parameters and generates alerts when something is wrong, so that the managers can take immediate action.

With this tool, the company may also conduct a cause-and-effect study in the foundry department. This will improve its manufacturing process, making it more efficient and able to decrease the percentage of defective parts.

References

1. Lu, Yang. "Industry 4.0: A survey on technologies, applications and open research issues." Journal of Industrial Information Integration 6 (2017): 1-10.
2. John Zysman and Martin Kenney. 2018. The Next Phase in the Digital Revolution: Intelligent Tools, Platforms, Growth, Employ- ment. Commun. ACM 61, 2 (2018), 54–63.
3. Pattnaik, Sarojrani, D. Benny Karunakar, and P. K. Jha. "Developments in investment casting process—a review." Journal of Materials Processing Technology 212.11 (2012): 2332-2348.
4. "Investment Casting Process — Sand Casting, Investment Casting & CNC Machining in China." Sand Casting, Investment Casting and Die Casting in China, 29 Dec. 2009, www.castingquality.com/casting-technology/investment-casting-tech/investment-casting-process.html.
5. Hexagon Manufacturing Intelligence. Q-DAS. Available at https://www.q-das.com/en, 2019. Accessed 2019-07-05.
6. Grafana Labs. Grafana. Available at https://grafana.com, 2019. Accessed 2019-07-05.
7. Morgan, R., Grossmann, G., Schrefl, M., Stumptner, M. A Model-Driven Approach for Visualisation Processes (2019) ACM International Conference Proceeding Series, art. no. a55, .

References

[1] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[2] J. Zysman and M. Kenney. The next phase in the digital revolution: Intelligent tools, platforms, growth, employment. *Communications of the ACM*, 61(2):54–63, 2018. cited By 21. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041610915&doi=10.1145%2f3173550&partnerID=40&md5=2bd88cce61625e452cfd7769adf63525, doi:10.1145/3173550.

[3] Inegi. Institute of science and innovation in mechanical and industrial engineering. Accessed: 2019-07-23. URL: http://www.inegi.pt/instituicao.asp?idm=1&idsubm=5&LN=EN.

[4] Massimo Bertolini, Davide Mezzogori, and Francesco Zammori. Comparison of new metaheuristics, for the solution of an integrated jobs-maintenance scheduling problem. *Expert Systems with Applications*, 122:118–136, 2019. doi:10.1016/j.eswa.2018.12.034.

[5] Greg Cline. Industry 4.0 and industrial iot in manufacturing: A sneak peek - aberdeen. https://www.aberdeen.com/opspro-essentials/industry-4-0-industrial-iot-manufacturing-sneak-peek/, March 2017. (Accessed on 02/13/2019).

[6] IIC. Industrial Internet Consortium - Fact Sheet. (March 2014):1, 2015.

[7] Jay Lee, Hung-An Kao, and Shanhu Yang. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, 16:3–8, 2014.

[8] Hexagon manufacturing intelligence. https://www.q-das.com/en. Accessed: 2019-07-05.

[9] Grafana labs. https://grafana.com. Accessed: 2019-07-05.

[10] Tim A Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Progressive web apps: the definite approach to cross-platform development? 2018.

[11] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[12] Vanessa Donnelly. *Designing easy-to-use websites*. Addison-Wesley, 2000.

[13] Brett S Gardner. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*, 11(1):13–19, 2011.

[14] Andreas Biørn-Hansen, Tim A Majchrzak, and Tor-Morten Grønli. Progressive web apps: The possible web-native unifier for mobile development. In *WEBIST*, pages 344–351, 2017.

[15] Progressive web app checklist | web | google developers. URL: https://developers.google.com/web/progressive-web-apps/checklist.

[16] Madhuri A Jadhav, Balkrishna R Sawant, and Anushree Deshmukh. Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 6(3):2876–2879, 2015.

[17] Sivakumaresan Thangeswaran. Software application architecture, March 28 2019. US Patent App. 16/202,373.

[18] Julia Plekhanova. Evaluating web development frameworks: Django, ruby on rails and cakephp. *Institute for Business and Information Technology*, 2009.

[19] Django. URL: https://www.djangoproject.com/.

[20] Chart.js. URL: https://www.chartjs.org/.

[21] Gurayyarar. Adminbsbmaterialdesign, Oct 2018. URL: https://github.com/gurayyarar/AdminBSBMaterialDesign.

[22] Serope Kalpakjian and Steven R Schmid. Manufacturing processes for engineering materials. *New Jersey, the United States of America: Prentice Hall*, 2003.

[23] InduSoft Web Studio. Mpi, inc. URL: http://www.indusoft.com/Marketing/Article/ArtMID/684/ArticleID/129/MPI-Inc.

[24] Isabel Maria Lousada Soares Figueiredo. Towards "industrie 4.0" in the context of investment casting industry. *MSc. FEUP*, 2019.