INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Mixed Reality for Manufacturing Execution Systems

VÍTOR MANUEL PINTO MOREIRA
Julho de 2018

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Mixed Reality for Manufacturing Execution Systems

Vítor Manuel Pinto Moreira

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Paulo Gandra de Sousa

Supervisor: José Pedro Silva

Co-supervisor: Francisco Miguel Maciel

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Julho 2018

Dedicatória

Aos meus pais, por tudo o que me proporcionaram na vida.

Resumo

A quarta revolução industrial, também conhecida como Indústria 4.0, marca o início de uma nova era onde novos avanços tecnológicos são incluídos e integrados em processos industriais com o intuito de os tornarem mais eficientes. A introdução de sistemas *Cyber-Physical*, *Industrial Internet of Things*, *Cloud Computing*, entre outros, permite a implementação de uma fábrica inteligente, onde todos os sistemas comunicam e interagem de uma forma autónoma e assíncrona, resultando num processo de tomada de decisão descentralizada.

Manufacturing Execution Systems (MES) são sistemas que permitem, entre muitas funcionalidades, a gestão e execução de processos de manufatura e do chão de fábrica, com o objetivo de manter um alto desempenho num ambiente complexo e com mudanças constantes. Com esta nova revolução industrial, surgiu a necessidade de os repensar e reestruturar, de modo a que contemplem e sejam compatíveis com as novas tecnologias, de modo a que possam oferecer uma abordagem de tomada de decisão descentralizada. Atualmente, a Critical Manufacturing desenvolve e oferece um sistema MES que está adaptado às necessidades da Indústria 4.0 através da implementação de tecnologias avançadas. Porém, estes avanços tecnológicos originam uma alta quantidade de dados que nem sempre são transformados em conhecimento.

O uso de sistemas inteligentes e interoperáveis na Indústria 4.0 não implica que os operadores e outros trabalhadores sejam substituídos. Pelo contrário, estes terão ainda mais responsabilidades e uma maior complexidade nas suas tarefas diárias, exigindo que sejam altamente flexíveis e adaptáveis. Contudo, ainda que se conformem com este tipo de ambiente, a complexidade acrescida resulta numa diminuição de produtividade.

A Realidade Mista é um conceito onde o mundo real é combinado com conteúdo virtual que, ao contrário do que acontece na Realidade Aumentada, está ligado e conectado com a realidade, implicando que variações num extremo provoquem alterações no outro. Esta mistura é visualmente exibida através de dispositivos de apresentação, proporcionando ao utilizar uma perceção aumentada do seu meio envolvente e oferecendo funcionalidades através da interação com o conteúdo virtual. O uso deste conceito na Indústria 4.0 abre uma nova porta na interação entre humanos e máquinas e suporte ao operador. Contudo, verifica-se uma falta de investigação e estudo nesta área, onde utilizações e casos de uso básicos ainda não estão explorados.

Esta dissertação propõe uma solução onde a realidade mista é usada e integrada no sistema MES da Critical Manufacturing, tirando partido das suas tecnologias e dos dados gerados por outros sistemas e tecnologias de forma a oferecer uma nova abordagem de interação com os objetos físicos presentes no chão de fábrica. O objetivo principal é o de diminuir a complexidade das tarefas diárias dos operadores e dos engenheiros, contribuindo para a preservação do alto desempenho dos processos de manufatura e aumento da produtividade. A intenção desta

solução é de servir como uma prova de conceito, revelando o potencial da introdução da realidade mista em sistemas MES e no contexto da Indústria 4.0.

Palavras-chave: Indústria 4.0, Manufacturing Execution System, Realidade Mista

Abstract

The fourth industrial revolution, also referred to as Industry 4.0, outlines the beginning of a new era where new technological advancements are put together to make industrial processes more efficient. The introduction of Cyber-Physical Systems (CPS), Industrial Internet of Things (IIoT), cloud computing, and other concepts, drives what is referred to as a "smart factory" in which the different systems are able to autonomously communicate and interact, resulting in a decentralized decision-making process.

With this revolution, Manufacturing Execution Systems (MES) need to be rationalized and restructured, contemplating the introduction of these advancements and being able to offer a decentralized approach to manufacturing processes. Currently, Critical Manufacturing develops and delivers a MES that is adapted to the requirements of the Industry 4.0. However, the amount of data generated by these technologies is not always transformed into knowledge for the customer.

The use of intelligent and interoperable systems in the Industry 4.0 does not mean that human workers are being replaced. On the contrary, shop-floor operators and engineers are beginning to perceive an increased complexity in their daily tasks, requiring them to be highly flexible and adaptable in a hard-working environment. Nevertheless, this increased complexity results in a decrease in the overall productivity.

Mixed Reality (MR) allows to blend the real world with digital content that is connected to it, resulting in a mixed environment that is further presented to the user in the same display device. The usage of this technology in the Industry 4.0 opens a new door to human-machine interaction and operator support. However, this topic is still very under-researched and basic use cases are still unexplored.

This dissertation proposes a solution where mixed reality is used and integrated with Critical Manufacturing's Industry 4.0-ready MES, taking advantage of its technologies and data flowing in the system in order to ease the shop-floor engineer daily tasks. The main intention of this subject is to serve as a proof of concept, revealing the potential of the inclusion of mixed reality in a manufacturing execution system in the context of the Industry 4.0.

Keywords: Industry 4.0, Manufacturing Execution System, Mixed Reality

Agradecimentos

Em primeiro lugar, quero agradecer ao Professor Paulo Gandra de Sousa por todo o apoio e dedicação enquanto meu orientador, tendo sido fulcral ao longo deste caminho.

À Critical Manufacturing, pela oportunidade que me concederam e em especial ao José Pedro Silva e ao Francisco Miguel Maciel por toda a ajuda e acompanhamento dados durante todo o desenvolvimento deste tema. Agradeço também aos meus colegas de trabalho pelo companheirismo e pelos seus conselhos.

Ao Instituto Superior de Engenharia do Porto, nomeadamente aos professores do Departamento de Engenharia Informática, por todo o conhecimento transmitido enquanto estudante deste estabelecimento de ensino, pelo rigor e seriedade que aplicam às suas tarefas e pela preocupação por uma boa formação e educação transmitidos aos seus alunos.

À minha família e amigos, pela amizade e apoio concedidos durante todo este tempo e pela paciência nas alturas mais difíceis.

A todas as outras pessoas que, de uma forma ou de outra, contribuíram para que concluísse este passo importante.

A todos, o meu sincero obrigado.

Table of Contents

List of Figures

List of Tables

Acronyms

AHP	Analytic Hierarchy Process
API	Application Programming Interface
AR	Augmented Reality
BI	Business Intelligence
CMF	Critical Manufacturing
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
CSS	Cascading Style Sheets
ERP	Enterprise Resource Planning
FFE	Fuzzy Front End
FPS	Frames Per Second
GUI	Graphical User Interface
HMD	Head-Mounted Display
HoQ	House of Quality
HTML	HyperText Markup Language
HUD	Heads-up Display
IIoT	Industrial Internet of Things
IoT	Internet of Things
IWCG	Immersive Web Community Group
JS	JavaScript
KPI	Key Performance Indicator
LLVM	Low-Level Virtual Machine
MES	Manufacturing Execution System
MR	Mixed Reality
NCD	New Concept Development
QFD	Quality Function Deployment
SDK	Software Development Kit
VC	Value for the Customer
VR	Virtual Reality

1 Introduction

This first chapter presents a contextualization of the subject of this dissertation, underlining and summarizing some of the key concepts that need to be understood, followed by a description of the problem that this dissertation addresses. In order to present a solution that aims to solve the problem, the objectives are also presented alongside the expected contributions of this work. It is finalized by describing the proposed approach and the structure of the document.

1.1 Context

The industry 4.0 is evermore a near-future reality, marking the beginning of a new industrial revolution where many different systems are put together to reduce production time while experiencing a growth in revenue as well as product quality (Critical Manufacturing, 2017). Big Data, autonomous robots, horizontal and vertical integration, Internet of Things (IoT) and Augmented Reality (AR) are some of the Industry 4.0's scientific pillars (Rüßmann et al., 2015), that together make this industrial revolution possible. To stay up to these changes, Manufacturing Execution Systems need to be rethought and restructured, providing full customization of interfaces that can work with any particular case, making them highly flexible and adaptive (Yew, Ong, & Nee, 2016). As a result, human operators and engineers are required to also be "[…] highly flexible and adaptive in a very dynamic working environment." (Longo, Nicoletti, & Padovano, 2017).

Critical Manufacturing (CMF), a leader company in developing software solutions for advanced manufacturing environments, currently develops and delivers a MES solution for high-tech manufacturers, *e.g.* semiconductors, medical devices, etc., all around the world. Its product is built for advanced manufacturing environments, Industry 4.0-ready, providing rich actionable information and bringing more benefits to the entire value chain (Critical Manufacturing, 2017).

Mixed reality is the concept of blending virtual content on the real world, through display devices, providing a rich and interactable user interface. This human-computer interaction

opens a new door to the Industry 4.0, allowing operators and engineers to more easily interact with computers and machines. The integration of this technology with Industry 4.0-ready MES solutions is yet an unexplored scenario that can bring gains to manufacturers.

1.2 Problem

With the development and implementation of the Industry 4.0, a change in tasks and demands for the operators and engineers that work on factories' shop floors is expected (Gorecky, Schmitt, Loskyll, & Zühlke, 2014). The amount of increased variety and complexity in their daily tasks require high flexibility and adaptability in a dynamic working environment (Longo et al., 2017). The introduction of advanced and state-of-the-art technologies as a support tool leverages that these workers can continue to fulfill their tasks, while not lowering their efficiency and productivity, and having more autonomy and assuming decision making roles and responsibilities (Gorecky et al., 2014).

As MES solutions become more mature and ready for the Industry 4.0, more complex manufacturing scenarios will be handled (Gorecky et al., 2014), followed by an increased amount of computer interfaces that need to be accessed by workers to reach information and perform tasks. This implies more time spent on computer devices and a longer and harder learning process. The introduction of CPS, sensors, actuators, and other components, generates more and more data that flows through software (Rüßmann et al., 2015) that is not always translated into meaningful information for workers. Critical Manufacturing works to convey this large amount of data into resourceful information by improving its MES solution to better handle the Industry 4.0 manufacturing environments.

Another problem is that operators that work in advanced manufacturing environments are often equipped with safety glasses and thick gloves, making it difficult to continuously access computers and terminals.

Lastly, the implementation of mixed reality for advanced industrial environments is still a very under-researched topic, with basic interactions and use cases not yet fully explored.

1.3 Objectives

The goal of this project is to understand and explore how mixed reality can be implemented in MES and applied to industry 4.0 with the intention to help in solving the problems mentioned above. It is expected to contribute with research, usage examples and a platform that showcases the benefits and the drawbacks of the use of the technology for the given context.

Regarding the research and usage examples, it is intended to explore how mixed reality can be applied to advanced manufacturing environments, understanding its place in the industry 4.0. As this is a new concept to be introduced in Critical Manufacturing's MES solution, it is also

needed to explore the currently available tools for building a mixed reality application that can be easily integrated with the system.

To serve as a proof of concept, a functional prototype of a mixed reality application integrated with Critical Manufacturing's MES solution is expected, providing manufacturing production and process related information to engineers that work on the shop floor through exploring the available information present on the system. The solution requirements, non-functional requirements and other functionalities are addressed with more detail in chapter 6. The final platform is intended to, initially, be used as an additional tool for live demonstrations as a way to promote how this technology can help in an advanced and complex working environment.

1.4 Expected Contributions

Following the objectives stated above, the first expected contribution is the research itself, where it is understood what mixed reality is and how can it be applied to the Industry 4.0 through the integration with MES software, alongside the advantages and the disadvantages of its usage in advanced manufacturing environments.

From a software engineering point of view, it is expected a requirements analysis that enumerates and details each functionality and particularity of the prototype application. Secondly, the design of the system supported by informed and reasoned choices. Lastly, the development of the prototype application that follows the design and that implements all the functional and non-functional requirements.

As a conclusion, it is expected to understand the limitations of the built system and the technologies chosen, and how it can be improved in the future.

1.5 Proposed Approach

In this dissertation it is proposed a mixed reality system, integrated in CMF's MES HTML5 GUI, that follows a marker-based approach, relying on the use of visual indicators, also referred as markers and tags, attached to physical entities, i.e. instead of recognizing an object, it is recognized the tag attached to it. In this case, the tags are specific to the used augmented reality library, and not generic such a QR Code. For each recognized tag, it is presented information through holographic content and mixed reality interfaces that allows an interaction with the physical object. The application is integrated into the CMF's MES HTML5 GUI using its dashboards module to interact with the object and subscribed in the message bus implementation to be notified of the physical objects' state changes.

The system provides different interfaces for various contexts in order to serve several use cases. It is also extensible and configurable so that it can be modeled in different ways while still offering the same functionality.

The proposed approach is highly based on the research made prior to its design and development, contributing with knowledge and research topics around the integration of Mixed Reality to the Industry 4.0.

1.6 Document Structure

This dissertation follows a document structure divided into ten chapters.

In the first chapter, Introduction, the context of this dissertation's topic is briefly introduced, followed by the description of the problem and the objectives that are defined to be achieved. The expected contributions of this dissertation are also specified to complete and understand what is to be expected from the objectives. It is then finalized by presenting the proposed approach.

The next chapter, Context, details the different topics that need to be understood in order to better comprehend the problem and the objectives, as well as the presented solution.

Afterward, there is the State of the Art chapter where the literature is revised and presented the most known and used approaches that can help in achieving the desired results. This is highly based on the details found in the Context chapter along with the problem and the defined objectives.

The technologies investigated in the State of the Art are then analyzed in the next chapter, where it is summarized the main functionalities, advantages, and disadvantages of each one, followed by a comparison and a decision of the best tool that can be used to help achieve the objectives and the application requirements.

The fifth chapter, Value Analysis, presents how the idea came to life followed by the analysis of its value the value that it brings to the company, Critical Manufacturing, and its customers.

The following chapter, Requirement Analysis, presents the functional and non-functional requirements analysis, stating, respectively, the requirements for the use case and the non-functional requirements in the form of a FURPS+ supplementary document.

Given those requirements, a design for the system is presented in the next chapter, Design, analyzing different alternatives and its advantages, disadvantages, and limitations. This design also has in mind the general constraints defined for the project.

The implementation of the design is presented in the following chapter, detailing each part of the application, how it behaves and its construction with the chosen technologies.

In the ninth chapter, the presented solution is evaluated through some defined methods that show the achieved and non-achieved results. Some metrics are also evaluated.

The last chapter, Conclusion, presents the achieved objectives and results, and the limitations of the built prototype application. Future work and possible improvements are also addressed,

showing how this concept can be hereafter integrated into Critical Manufacturing's MES solution and how its limitations can be surpassed.

2 Context

The concepts introduced in section 1.1 are vital to the understanding of the problem and how it can be addressed. For that purpose, this chapter presents a deeper investigation of industry 4.0 and a brief introduction to Critical Manufacturing and its products and services.

Mixed reality is often confused and not acknowledged as much as virtual and augmented reality. These concepts have their differences and this chapter also addresses how they differ, followed by a vast explanation of mixed reality.

One of the objectives is to understand how mixed reality can be applied to the industry 4.0. This chapter finalizes by presenting that research, not only real use cases but how it can be integrated with current manufacturing software and other technologies.

2.1 Industry 4.0

In the past, there have been three industrial revolutions driven by technological advancements (Brettel, Friederichsen, Keller, & Rosenberg, 2014; Rüßmann et al., 2015). The introduction of water and steam-powered mechanical manufacturing facilities in the end of the 18th century, the introduction of electrically-powered mass production by having a division of labor in the beginning of the 20th century and, lastly, the introduction of electronics and Information Technology (IT) for manufacturing automation in the start of the 1970s that has continued until the present days. Currently, technological advancements are enabling a fourth industrial revolution, also known as the Industry 4.0, based on the introduction of autonomous CPS, physical objects that are enhanced with embedded software, communication, and computing power (Almada Lobo, 2017). To complement these systems, manufacturing equipment with computing power to leverage data from embedded sensors and actuators, known as Cyber-Physical Production Systems (CPPS), will also be introduced. These systems will change how manufacturing production currently works, evolving into decentralized processes through a decentralized system that connects the entire shop-floor chain, the area of a factory composed

by machines, operators, and where production is carried out. This transformation focuses on achieving manufacturing product and process automation, rapid product development, mass customization and dealing with complex environments (Brettel et al., 2014; Kagermann, Helbig, Hellinger, & Wahlster, 2013) through the integration of nine foundational technology advancements, namely, Autonomous Robots, Simulation, System Integration, IoT, Cybersecurity, Cloud Computing, Additive Manufacturing, Augmented Reality and Big Data. Ultimately, this will result in a unified, decentralized, automated, fully integrated and optimized production flow with increased efficiency, productivity and revenue growth (McCabe, 2016; Rüßmann et al., 2015).

To accommodate this shift, MES systems will have to either change, adapt or evolve. A MES is an information system that focusses on the execution of manufacturing operations such as scheduling and its execution and control, monitoring and execution of production processes, monitoring and control of materials used in the production process, gathering of information about the production process, data analysis, amongst (Critical Manufacturing, 2018g; Zhong, Dai, Qu, Hu, & Huang, 2013). It allows factories to more easily deal with the manufacturing complexity, having well-defined objectives: productivity, by having more production throughput, reduced costs, higher efficiency and lower maintenance and labor costs; quality, through monitoring and control of the manufacturing process, management of exceptions, compliance to norms and traceability; and, lastly, flexibility, by being adaptable to customer needs and mass production. In the Industry 4.0, MES solutions must have these objectives fulfilled and integrate the nine foundational technology advancements if they are to work in this new industry.

2.2 Critical Manufacturing

Critical Manufacturing is a company that focusses on empowering high-tech manufacturers by creating and delivering cutting-edge software products and services (Critical Manufacturing, 2018f) for the business domain of advanced manufacturing environments, such as semiconductor, photovoltaics, electronics and medical devices. Its flagship product is web-based MES solution that leverages several years of MES engineering experience. It has a low total cost of ownership, wide variety of functionalities and capabilities that can deal with the requirements of complex manufacturing processes, empowers end users, is highly extensible and configurable and is Industry 4.0-ready (Critical Manufacturing, 2018b). The total cost of ownership is lowered due to its modular software architecture and scalable pricing based on the customer needs, scalable hardware builds, fast and full server-side installation with no client installation needed, GUI modeling through drag-and-drop technology that empowers end-users, amongst other particularities. The software solution provides functionalities such as Connectivity, through IIoT and device connectivity, mobile friendly, cloud computing, advanced analytics, decentralization, and vertical and horizontal system integration (Critical Manufacturing, 2018d). All the functionalities are divided and grouped among several software modules, namely, Factory Management, Online Visibility & Traceability, Operational Efficiency,

Quality Management, Factory Integration and Operations Intelligence (Critical Manufacturing, 2018c). Lastly, it can also be integrated with other software solutions such as an Enterprise Resource Planning (ERP) and lower level automation software (Critical Manufacturing, 2014).

Alongside the main product, Critical Manufacturing also provides services for individual customer needs. The evolution of the shop-floor automation is increasing the amount of data generated that is being continuously stored. The translation of that data to knowledge opens a door for predictions and decisions that otherwise may come with a high risk. Critical Manufacturing develops Data Analysis and Business Intelligence (BI) solutions that take information from several software solutions, such as MES and ERP, and translate it into knowledge, leveraging standard and ad-hoc queries and reports, automatic generation and distribution of reports, Online Analytical Processing (OLAP) analysis for data slice and dice and, lastly, Data Mining for advanced data analysis (Critical Manufacturing, 2018e). Another branch of delivered services is legacy MES enhancements and migration that tend to help manufacturers to deal with End-of-Life Support, Maintenance Operational Cost and modification, extensibility and integration that together can increase overall costs, risks and missed revenue (Critical Manufacturing, 2018a).

2.3 Mixed Reality and its Difference from Virtual and Augmented Reality

Over the years, the barrier between the real and the virtual world has been diminishing rapidly. The continuous evolution in both hardware and software has allowed the possibility to create such “illusions” that were once only part of science fiction movies. Thus, the emergence of virtual reality and, later, augmented reality and mixed reality, have changed the way of how user experience can be achieved. Though they are different approaches, they are still related and there is the need to differentiate these concepts, understand what mixed reality is and how it can be accomplished.

2.3.1 Virtual Reality

Firstly, virtual reality is an immersive experience in a virtual computer-generated environment (Cruz-Neira, Sandin, & DeFanti, 1993) (Intel, 2018) that may or may not be related to the real world. Although it can be manipulated, it is a complete simulation, both graphics (imagery), sounds, the idea of presence and the ability to manipulate the environment (Steuer, 1992). While immersed, the user is limited to the virtual world that is experiencing, completely losing perception of the real world. This is by far the most advanced kind of immersive experience, already with different tools and applications on the market that make the development easier. Virtual reality is used is many different fields such as entertainment, education, military, shown in Figure 1, science, e-commerce, shown in Figure 2, among others (Virtual Reality Society, 2017). To be able to experience these applications, users use virtual reality headsets, a type of

head-mounted display (HMD) which has a display, sensors and sometimes integrated headphones, and controls, usually for the users' hands, to give control over the application and a more concise way of interacting with it.

Figure 1 - Virtual reality flight simulations in the military[1]

Figure 2 - IKEA virtual kitchen[2]

2.3.2 Augmented Reality

On the other hand, augmented reality, as its name suggests, is the experience of a real world that has been augmented through the addition of some kind of virtual environment, such as computer-generated information that enhances and supplements the perception of the user (Azuma, 1997) (Furht, 2011). Here, the user is not immersed in a virtual environment but rather an environment where the real world is present and can be observed while being overlaid with

[1] Bohemia Interative Simulations training solutions for military organizations. Retrieved May 29th, 2018 from https://bisimulations.com/company/news/press-releases/thu-05122016-1452/bisim-demonstrate-vr-flight-simulator-itec-2016-and-sea-air-space-2016

[2] IKEA virtual kitchen application footage. Retrieved May 29th, 2018 from https://www.ikea.com/ms/en_US/this-is-ikea/ikea-highlights/Virtual-reality/index.html

content to enrich it, making it augmented. Such experience combines virtual and real objects in a real environment, registering them so that they run interactively in real time (Van Krevelen & Poelman, 2010). One important thing to denote is that even though this combines two different realities, real and virtual, they are not connected other than visually. Unlike virtual reality, here the need to be immersed is not the focus, although there are HMDs which can be used for augmented reality, such as the Microsoft HoloLens and Meta 2 (Meta, 2018; Microsoft, 2018a). These experiences can also be used with other devices, such as smartphones, tablets and heads-up displays (HUDs), which are used by fighter pilots and, more recently, incorporated in automobiles (Kipper & Rampolla, 2012). (Bimber & Raskar, 2005) presented a model, see Figure 3, that contains the main concepts that any augmented reality-based application must contain. The lowest and most fundamental layer is made up of three different modules, Tracking and Registration, Display Technology and lastly, Rendering.

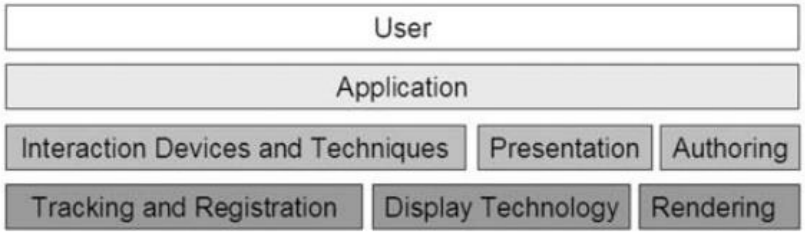


Figure 3 - Building blocks of augmented reality[3]

When combining real and virtual environments, a registration between the synthetic objects and the real world must occur so that both environments are aligned. Most of the times, the virtual information is added once a point of interest is seen or sensed, i.e. detecting the presence of a real object that will trigger the addition of a virtual environment. The continuous identification of the desired object is called tracking. According to (Zhou, Duh, & Billinghurst, 2008), tracking papers from the International Symposium on Mixed and Augmented Reality (ISMAR), of the Institute of Electrical and Electronic Engineers (IEEE), have been the most popular topic for research.

The display technology refers to the display device used to present the combined information to the user. This information, which is made of the real world combined with virtual content, is first rendered before it is presented in the display. The rendering process may execute differently depending on the display device used.

On the layer above, more advanced and complex concepts are introduced. There's the interaction devices and techniques which are responsible for the human-computer interaction (HCI), presentation techniques that use the rendered content and interact with the display device to show the information and, lastly, authoring, tools used to create the association

[3] Building blocks of augmented reality as presented by (Bimber & Raskar, 2005). Retrieved May 30th, 2018 from http://pages.cs.wisc.edu/~dyer/cs534/papers/SAR.pdf

between the physical objects and the generated graphics that will augmented them, e.g. mapping the real object geometry and texture and correctly applying the new virtual content.

Currently, augmented reality is starting to be more robust and both hardware and software are able to handle it more properly, making it an emerging technology and already being tested and used for different areas such as military, medical, GPS navigation and gaming (Perdue, 2017), presented in Figure 4.

Figure 4 - Pokémon Go in augmented reality mode[4]

2.3.3 Mixed Reality

The term "mixed reality" first appeared in 1994, introduced by Paul Milgram and Fumio Kishino in their paper "A Taxonomy of Mixed Reality Visual Displays" (Microsoft, 2018b). They presented the differences between virtual reality and augmented reality and further categorized a special subset in which both realities were merged as one, experienced in a single display as if they co-existed (Milgram & Kishino, 1994): the mixed reality. In that same paper, they introduced the concept of the *virtuality continuum*, shown in Figure 5, in which they defined a spectrum with two extremes, the real environment and the virtual environment. Along this continuum, they presented augmented reality as a concept which involves both real environment and virtual environment, with the latter concept only used as a graphics overlay, and augmented virtuality as a type of virtual reality where the scene is augmented with real meaning, e.g. the virtual content is not fictional but a graphical representation of reality. The combination of these concepts, merging both environments as one while leaving the perception that they co-exist, is mixed reality.

[4] Pokémon GO application running in AR mode. Retrieved May 29th, 2018 from https://www.cgsinc.com/blog/pokemon-augmented-reality-corporate-learning

Figure 5 - The *Virtuality Continuum*[5]

It was then defined that the best way to experience mixed reality was in "[…] [an environment] in which real world and virtual world objects are presented together within a single display, that is, anywhere between the extrema of the virtuality continuum." (Milgram & Kishino, 1994). Because there were already different display devices and other approaches starting to appear in the literature, they defined a taxonomy where they categorized different types of mixed reality experiences depending on the type of display device.

Currently, technology has evolved enough to explore all these concepts, and, in fact, the concept of mixed reality now goes beyond than just displays devices and the visual perception of two realities, real and virtual, being merged (Microsoft, 2018b). According to (Foundry, 2018; Intel, 2018), mixed reality involves the merging of synthetic content with the real environment, while they are able to react to each other in real time and are both manipulable, breaking down the barrier between virtual and augmented reality. (Microsoft, 2018b) took it further, and included environmental input, spatial sound, and location as three concepts of today's mixed reality, as illustrated in Figure 6.

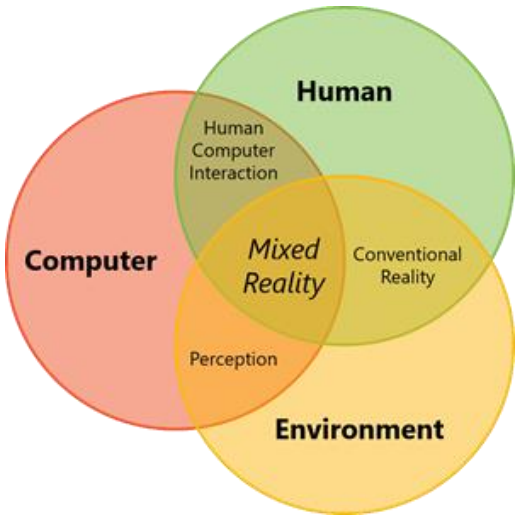


Figure 6 - Environmental input and perception in Mixed Reality[6]

[5] The *virtuality continuum* presented by Paul Milgram and Fumio Kishino in their paper "A Taxonomy of Mixed Reality Visual Displays". Retrieved and adapted May 29th, 2018 from http://www.alice.id.tue.nl/references/milgram-kishino-1994.pdf

[6] Environmental input and perception concepts by Microsoft. Retrieved May 30th, 2018 from https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality

Regarding human-computer interaction, a discipline that studies the interaction of humans with computers through input and output, there's already a wide variety of devices such as touch screens and pads, voice recognition, controllers, skeletal tracking, among others. The *conventional reality*, perceived as the relationship between humans and the environment, plays a big part in mixed reality, as it determines the context of the real environment. Lastly, perception, the concept of environmental understanding by the computer, is made possible through the usage of advanced sensors, such as cameras, gyroscopes, orientation and positional sensors, that are responsible of capturing input from the environment, providing information such as position tracking, spatial mapping, spatial understanding, ambient lighting, spatial sound, among many others. The blending of the virtual and the real environment, through the incorporation of these concepts, define a true immersive mixed reality experience.

Based on the *virtuality continuum* model, presented by (Milgram & Kishino, 1994), Microsoft defined that spectrum as the *mixed reality spectrum*, illustrated at Figure 7, presenting certain examples of mixed reality experiences from two different perspectives: one where the real world has more focus and another where the digital environment is the starting point. For the first case, examples such as observing and interacting with a virtual hologram and tracking the movements and actions of a person while rendering a digital representation in real time are valid and true mixed reality experiences. On the other hand, from a digital world starting point, the user could be immersed in a scene that contains the same physical aspects of the real world, such as floors, walls, and other objects, as captured by the device's environment perception sensors, consequently revealing the user the existing physical boundaries.

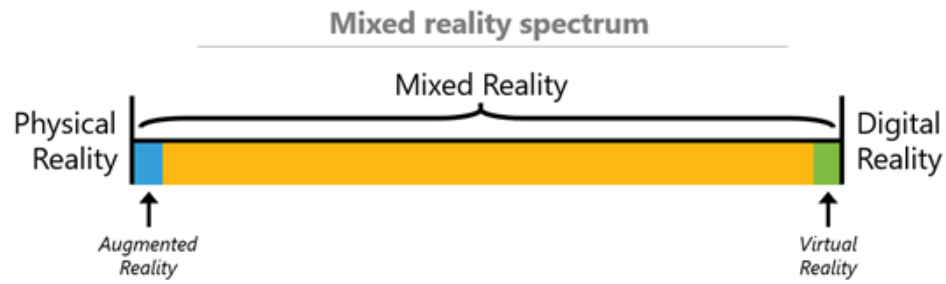


Figure 7 - The Mixed Reality spectrum[7]

Depending on the mixed reality experience perspective, there are different display devices that can be used. Such devices can be defined as Holographic devices, for the perspective where the real world is the central focus, and Immersive devices, for the perspective where the virtual environment is the starting point.

[7] The Mixed Reality spectrum as adapted by Microsoft from the *virtuality continuum* model of (Milgram & Kishino, 1994). Retrieved May 30th, 2018 from https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality

In short, mixed reality can be experienced in many ways, but the key difference to virtual and augmented reality is that is blends both real and virtual world, connecting them in some way that they react to each other in real time. Unlike VR and AR, the use and practice of this concept is still new, and not much explored. More recently, Microsoft advancements in mixed reality revealed experiences such as holographic computing, through the usage of HoloLens, to experience a Skype video call where two users can draw, and those drawings will appear to the other user, see Figure 8. A more immersive example is the HoloTour application for the HoloLens, where users can virtually visit historical places that are blended to the real world while persisting its physical boundaries.

Figure 8 - Skype application on HoloLens[8]

2.4 Mixed Reality Applied to the Industry 4.0

Mixed reality applications can relieve users of work and simplify their interaction with computers and machines. In advanced industrial environments, HCI is a constant task and sometimes the access to computers is not easy, it may force the operator or engineer to remove its hand safety equipment and even if the software is intuitive and fast, the time that it takes to search and get the information needed lowers efficiency and, consequently, may reduce productivity (Michalos, Karagiannis, Makris, Tokçalar, & Chryssolouris, 2016). Machines and other manufacturing equipment are also becoming more and more complex, making it harder for the engineers to deal with servicing and maintenance. From a production process and assembly stage point of view, having quick and meaningful information about the production status, instructions about the assembly process are also good and useful optimizations that mixed reality can bring to the industry (Michalos et al., 2016). The main objective of the incorporation of this technology in the Industry 4.0 is to improve the interaction between the shop floor workers and computers by presenting useful and interactable real-time information to their daily tasks aiding them in decision making and work procedures (Rüßmann et al., 2015; Yew et al., 2016).

[8] Skype application interaction on Microsoft HoloLens – a user draws on the tablet and the other user can see the drawing on the HoloLens display. Retrieved May 30th, 2018 from https://compass-ssl.surface.com/assets/d5/de/d5dee7ae-9a08-407a-86d1-a425c5982f84.jpg?n=Skype_Hero_img_1920.jpg

Research in augmented reality applications and several laboratory prototypes have already developed to understand in what circumstances this technology can be helpful, revealing practical utility in assembly planning, assembly guidance, maintenance, servicing, amongst others (Michalos et al., 2016; Regenbrecht, Baratoff, & Wilke, 2005). Some examples rely on displaying each step of an assembly process to an operator, how to perform the maintenance of a machine to an engineer. However, as discussed in section 2.3, a particularity of mixed reality is that the virtual content is not only superimposed on the real world but also reacts to it and vice-versa. If this detail is considered, it can enhance those use cases by presenting real-time changes as the worker fulfills his tasks or even interact with the machine through the augmented interface. For this to be possible, the software needs to be integrated with the rest of the factory equipment. The implementation and introduction of IIoT in MES solutions allows the integration of the different equipment throughout the factory, which allows each one to communicate and report activity. By listening to these messages and events, the mixed reality application can adapt in real-time. Another approach is to take advantage of messaging infrastructure such as a message bus: the mixed reality application is registered as a subscriber to the equipment messages, where the equipment publishes messages regarding its status, activity, and other changes. Modern MES solutions that are Industry 4.0-ready, have these decentralized systems and infrastructures that allow the introduction of a mixed reality application, as another layer or software module, that can react to and be integrated with the entire shop floor chain and, thus, providing the operators and engineers with valuable, rich and interactable information to simplify their work and better deal with the daily complexity of their tasks.

3 State of the Art

Several concepts of mixed reality rely on using different techniques that together can achieve a certain outcome, as described in section 2.3. Topics such as object tracking, graphics rendering software, and display devices are vital to the development of a mixed reality system and. This chapter presents a state of the art review in those fields, by describing several research topics and approaches for each one.

There are already built frameworks and libraries that offer some tools to ease the development of mixed reality applications. A survey of technologies that follow the objectives and the restrictions of the expected functional prototype is presented in the last section.

3.1 Object Tracking

From subsection 2.3.3, it was noted that object tracking is an important component of a mixed reality application. It is a popular research topic, revealing different base approaches for different contexts. In this section, the most used and state of the art approaches to object tracking, for an augmented and mixed reality context, are presented and further categorized by different types of technology and science involved.

3.1.1 Sensor-Based Tracking

Sensor-based tracking techniques make use of different types of sensors, such as, magnetic, acoustic, inertial, optical and mechanical (Zhou et al., 2008). Although optics are considered sensors, they are used in conjunction with computer vision algorithms. This type of object tracking is vision-based and so it is referred to in subsection 3.1.2. Sensor-based object tracking is already well-known, with its peak being around the time of the 1998 International Workshop of Augmented Reality (IWAR). According to (Baillot, Davis, & Rolland, 2001), approaches to this tracking technique can be categorized as time of flight (TOF) and phase-difference, spatial scan

- through optics and so not referred here, inertial sensing, mechanical linkages and, lastly, direct-field sensing.

Time of Flight (TOF) systems rely on the measurement of the distance between the different features attached to references and targets. The time that the pulsed signals take to propagate between pairs of points, these being the emitter and the receiver, is measured assuming a constant speed of propagation, resulting in the time that it took to reach one point to another. Global Positioning System (GPS) is one of the most important and widely used implementations of a TOF system on a world scale with a high performance in none occluded areas. However, if the direct lines of sight to satellites are occluded, these systems often present poor accuracy and tend to fail. Similar to these systems, phase-difference is also based on the same structure with emitters and receivers, with features attached to references and targets. The difference is that it is not determined the time that the signal takes to propagate between each pair of points, but it is measured the relative phase of an incoming signal and that value is then compared to the signal located on the reference. This technique has several limitations such as occlusion, ambiguity and increased error for each sequential measuring. However, an early approach with this technique presented a head-mounted display for virtual reality (Sutherland, 1968).

Inertial sensing is a tracking technique that is based on preserving an axis of rotation, for a mechanical gyroscope, or position, for an accelerometer, seen as a complementary technique for location-based tracking. A mechanical gyroscope retrieves the orientation of the target for one axis through the principle of conversion of the angular momentum. On the other hand, an accelerometer is responsible for calculating the target's linear acceleration, again for one axis. These sensors have been incorporated in several different devices such as smartphones, tablets, entertainment consoles' controllers, head-mounted displays, amongst many others.

A mechanical linkage system is comprehended of several mechanical parts connected by mechanical links equipped with encoders or potentiometers. These links are able to rotate independently from each other, providing a higher rotation flexibility. The system can also use spring mechanical linkages which are able to measure the distance from the reference point where the system main structure is mounted on. The main disadvantage of this system is that it depends on a physical reference point, tying the system to a physical object and only retrieving values relative to that same reference.

Direct-field sensing relies on the measurement of the signal of a certain field type, such as magnetic and gravitational. Magnetic field sensing is done through the usage of an emitter, composed of three coils with circulating electric current that generates a magnetic field, and a receiver that will have a flux inducted by the emitter's generated magnetic field and will measure both relative position and orientation through three sensors, each one for each of the emitter's coil. An example of such sensor is the magnetometer, a sensor that measures the orientation of the device that it is attached to relative to the Earth's magnetic field, widely adopted by mobile devices and head-mounted displays. Gravitation field sensing, while not widely adopted because it is limited to one degree-of-freedom (DOF), i.e. can only measure orientation for one direction, rely on electrolytic or capacitive sensing of fluids to compute its

orientation. This technique implies several disadvantages such as being inaccurate, due to vibration and acceleration, and having an inconstant reaction time depending on the viscosity of the used fluid.

These sensor-based tracking techniques are mostly used to track a device that it is attached to instead of other surrounding objects, making them very widely adopted for mobile devices, head-mounted displays and smart-glasses. Also, they can be used in conjunction with other tracking techniques to build hybrid systems as it is explained in subsection 3.1.3.

3.1.2 Vision-Based Tracking

Vision-based tracking techniques are, without any doubt, the most used mostly because they are very intuitive and usually require a small system, making them non-intrusive for users. Also, and according to (Zhou et al., 2008), vision tracking research represents 80% of all tracking research of ISMAR. Recently, computer vision suffered an increase in its usage, with the appearance of deep-learning technologies that enabled some computer vision algorithms to be possible to be used and elevating the reliability of image recognition. Still, this approach was already used before, with algorithms relying on different image processing techniques, like feature detection and edge detection, and continue to be widely used due to being lightweight when compared to systems that make use of deep-learning in conjunction with image processing implementations.

Vision-based tracking can be divided into two types, Feature-based and Model-based. Feature-based is a well-known and used technique and have a large variety of algorithms and implementations. This approach is based on image recognition, by analyzing an image and retrieving points of interest, edges, lines, and other features (Szeliski, 2010) (Forsyth & Ponce, 2002), and use them to recognize it. The most successful research papers that ended in widely used feature-based implementations are using planar square or fiducial circular markers as the target to be recognized. These markers, shown in Figure 9, are simple markers, that may or may not contain encoded information, and are built to be fast in the recognition process, representing a point of interest in an image that is, usually, completely different from the surrounding environment, so that recognition errors are diminished.

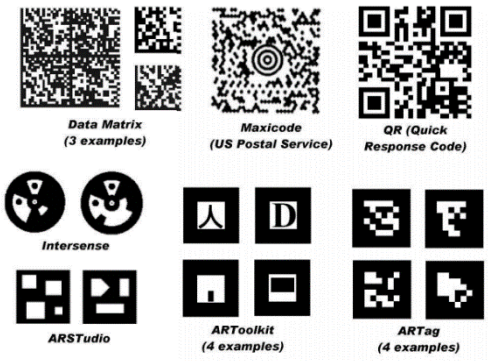


Figure 9 - Different types of planar markers and its usage in different frameworks[9]

These markers also denote patterns that can be easily recognized with specific pattern recognition algorithms (Zhou et al., 2008), and represent one of the most used types of object tracking approaches in AR (Hirzer, 2008) (Fiala, 2005) (Chauhan & Kayasth, 2014). Another approach is by recognizing natural features in an image, apart from a very well-defined object, like a marker. Although requiring less physical setup, *i.e.* there is not the need to place a marker in the environment, this technique does not produce results as well as using markers, because of the less uniqueness in the target. However, its usage in AR and MR is also possible and there are implements of some algorithms in those areas (Neumann & You, 1999). Ultimately, image recognition has been seen to produce the best results with newer technologies that make use of neural networks, *e.g.* Deep Convolutional Neural Networks (DCCN) (Simonyan & Zisserman, 2014) and Residual Learning (He, Zhang, Ren, & Sun, 2016). The problem with these approaches to AR and web environments is that they take a lot of time to set up, there is the need to intensively train models, and they use a lot of computational resources, depending on high tech machines to be fast and accurate. Although JavaScript browser engines are getting better every day, they are still no match when compared with native platforms that make use of high-level hardware that is used to the most.

On the other hand, there is Model-based tracking. Unlike feature-based, this approach tries to analyze an image and reconstruct a pre-known 3D model of the target. They both share some techniques of image feature extraction, *e.g.* line or edge detection and extraction, but they use that information in a different way. Although it is also a popular research topic for AR/MR (Reitmayr & Drummond, 2006) (Gavrila & Davis, 1996), it comes with some drawbacks, such as the need to have the 3D models of the targets known and pre-loaded, and the usage in portable devices is diminished due to their limitation in hardware when comparing to native platform computers.

Unlike sensor-based tracking, using optics to perform object tracking is usually used for tracking an object that is not the system where the optic is attached to. They are particularly useful for marker-based augmented reality or to identify certain objects on the video feed captured by

[9] Different planar markers tags from several systems. Retrieved June 8th, 2018 from http://inside.mines.edu/~whoff/courses/EENG512/lectures/other/ARTag.pdf

the device's camera. Still, optical sensors are used for state of the art self-tracking techniques, such as Spatial Scan. Spatial scan is the principle of analyzing 2D features and/or analyze sweep-beam angles of objects and use this information to compute both spatial position and orientation of the device that contains the optical sensors (Baillot et al., 2001). There are two configurations in spatial scan, outside-in and inside-out. Outside-in relies on using a fixed optical sensor, the reference, and have moving objects, the targets, which produces a small motion of the image. Together with computer vision algorithms, such as pattern recognition and feature detection, changes in sequential images can be found and they can be analyzed to find 2D projections in space to retrieve both special position and orientation. In the case of an inside-out configuration, the optical sensor is moving, e.g. is attached to a head-mounted display or a mobile device, and the objects do not need to be moving. Here, the camera is the target and each object is a reference. By rotating and moving the camera, a large motion is produced and there are even more changes between sequential images. As opposed to outside-in, inside-out configuration is able to have higher resolution and better results because of the larger information that it is able to analyze (Baillot et al., 2001).

Furthermore, vision-based tracking techniques can also be used in hybrid systems.

3.1.3 Hybrid Tracking

Hybrid tracking refers to the combination of different tracking techniques of the same above categories or between them and so reduce their limitations (Baillot et al., 2001). An example using sensor-based tracker only is to combine three gyrometers and three accelerometers. Each gyrometer and accelerometer will, respectively, preserve the orientation and measure the acceleration of an axis. This combination is a system that can retrieve the orientation of a target in space as well as its acceleration that, if well measured, can be used to compute the device's spatial position. Another combination is to use the inside-out configuration of spatial scan along with the above inertial sensing hybrid combination. Vision-based tracking techniques generally are very sensitive to occlusions, have processing lag that varies with the hardware used and tend to degrade with distance. On the other hand, sensors don't have these problems but usually, their results are worse and degrade with movements and drifts. The combination of both these techniques results in a system that provides accurate results, have small processing lag, deal better with distance and are more compact, while only maintaining occlusion problems (Baillot et al., 2001). Similar to this approach, magnetometers can be used instead of gyroscopes and accelerometers, which leads to insensitivity to occlusions while having to maintain processing lag.

Hybrid systems can be further equipped and built with more sensors to retrieve better results, at the expense of a higher cost, complexity, and compactness. An example of a hybrid tracking system for augmented reality is one proposed by (Azuma et al., 1998), where he makes use of both GPS and computer vision sensing technologies to detect and track the position of a target.

3.2 Graphics Rendering Software

As explained before, in MR, the virtual environment plays an important part by extending the reality in a meaningful way. In web applications, the structure of a web page is defined through HyperText Markup Language (HTML) along with Cascade Style Sheets (CSS) that enhance its style appearance. However, with HTML5, it was introduced the Canvas API, which opened a new possibility to bring more power to the visualization and rendering on a web page. With it, new technologies emerged, allow 3D rendering and visualization in web possible, long seen in native platforms. With that being said, it is presented a state of the art revision of technologies built for advanced web user interface and graphics development.

3.2.1 WebGL

Generally, in every platform, either native or web, there is a standard technology which exposes a low-level 3D graphics API. WebGL is a cross-platform low-level 3D graphics API, based on OpenGL ES, and works with the HTML5 Canvas element (Khronos Group, 2011). It is plugin-free, which means that its implementation is done on the browser, with the responsibility of every browser vendor. Most major browsers not only are compatible with WebGL (Mozilla, 2018b), but they are also members of the WebGL Working Group (Khronos Group, 2011), contributing to a widely used standard and making the web easier and more reliable for 3D graphics computation. Currently, WebGL features two specifications, 1.0 and 2.0, that are exposed, respectively, to the OpenGL ES 2.0 and the OpenGL ES 3.0 feature sets. The WebGL 2.0 specification is still in an Editor's Draft status, which means that it is not yet recommended as it lacks support on some browsers (Mozilla, 2018b).

Direct development with this API is possible, however, other solid technologies are already available in order to facilitate computer graphics development, abstracting certain low-level logic.

3.2.2 Three.js

Three.js is an open-source lightweight JavaScript 3D library that makes use of canvas and svg HTML5 elements, offering both CSS3D and WebGL renderers (Mr.doob, 2010/2018).

As opposed to WebGL, this library does not provide a direct low-level access to 3D graphics computation but rather provides an API with easier access which takes care of low-level logic. Its 3D engine is fast and uses many best practice techniques, wrapping WebGL while being feature-rich, offering many built-in features that make the development of web graphics easy (Parisi, 2012). Amongst many other features, is it interactable, which makes possible to use it to create a rich virtual interactable environment. It is also perceived as the leader among 3D libraries for JavaScript and its popularity even reached major companies, like Google, that created projects based on Three.js (Google, 2017) (Warner Bros Entertainment & Google, 2018).

3.2.3 A-Frame

A-Frame is a web framework originally developed by Mozilla and later became an independent open-source project. It is based and built with Three.js that is accessible through JavaScript, exposing access to DOM APIs, WebVR and WebGL, with a core composable and reusable entity-component framework architecture (Mozilla, 2018a).

Unlike other technologies, A-Frame is used and developed through declarative HTML syntax, contributing to an easy integration with a web application. Its key features, besides the powerful architecture and being a declarative HTML language, are having a high performance, being tool and technology agnostic, providing a 3D visual inspector, is extensible through the creation and addition of custom entities and components, amongst many other possibilities. This framework was mostly built with the intention of being an easy and powerful way to create VR experiences and, as an open-source project, it maintained its purpose which led it to grow and be one of the largest 3D projects with a vast and supportive community (Mozilla, 2018a).

3.3 Display Devices

As said before, one of the main concepts in mixed reality is the display technology, which is where the virtual and/or real information is presented. There is a wide range of different choices already on the market focusing on different real-world scenarios and areas, e.g. industry, entertainment, education, and military. Here, only manufacturing industry driven, and/or compatible technologies are taken into account. These types of solutions can be arranged into three groups: smartphones and tablets, smart glasses and optical see-through head-mounted displays (HMDs). Each offers a slightly different approach to how mixed reality is experienced and have their own advantages and disadvantages depending on the context that they are used.

3.3.1 Smartphones/Tablets

Smartphones and tablets are one of the most used technologies in the world, with statistics showing that, in 2017, there were around 1.32 billion users that used a tablet at least once per month (Statista, 2018b) and 2.32 billion users own and used a smartphone at least once per month (Statista, 2018a), with expectations of a rapid growth. By consequence, it makes these devices one of the most important targets for innovation. Most augmented and mixed reality technologies target mobile devices because they can reach a higher level of users, because not every person is able to acquire a most sophisticated higher-priced display device. This type of hardware, and more commonly industrial rugged tablets, tablets with better components for more computing power and specially designed for harsh environments (Thompson, 2015), are being used more and more on the manufacturing industry, replacing legacy manual physical processes, e.g. documentation and maintenance, and, at the same time, streamlining those same processes and others, as the work is now done on a device that can be connected to the software operating on the shop floor (Chaneski, 2015), increasing productivity and mobility.

The usage of these devices with augmented and mixed reality is a simple approach, regarding that running an application and displaying the final content is done on the same hardware. In this case, a video see-through is the displaying method used, i.e. the real-world information is captured by the device's camera and the virtual overlay is added on top, with the final content being displayed all-together on the device's screen. Because each model from each brand is different, and there are better and worse devices, regarding performance and hardware components, the same application will end up providing better results on two different devices. Another disadvantage is that not every device is AR-capable, as it does not have the required hardware components, although this is diminishing every day as newer devices are starting to be included with enough capabilities for these technologies.

3.3.2 Smart Glasses

Smart glasses are a type of wearable computer glasses that allow the user to see additional information, apart from what he sees from the real world, rendered on a special type of displays either imbued on the glasses' lenses or on a mono or binocular display, shown in **Error! Reference source not found.**, attached to the glasses (Schweizer, 2014). By extension, this is an augmented reality-based hardware that can be extended for a mixed reality approach through the use of its computer capabilities. One of the pioneers of this technology is Google, that back in April 2012, announced Project Glass and later presented the Google Glasses (Glass Almanac, 2018).

Figure 10 - Smart glasses[10]

As this type of technology evolved and became better over the years, other companies started to develop their own solutions and nowadays there are several different products developed for different contexts and environments, from day-to-day social usage and fitness to industrial and manufacturing purposes (Schweizer, 2014). In most cases, these smart glasses run an operating system on its imbued computer, e.g. Android, with native custom applications that are launched and controlled either through voice recognition and/or integrated touchpads and can take advantage of the hardware components as it was any other device. Because there is

[10] On the left, Google Glass smart glasses equipped with a monocular attached display. Retrieved June 8th, 2018 from https://www.x.company/glass/
On the right, Vuzix blade smart glasses with a built-in display on its lenses. Retrieved _June 8th, 2018 from https://www.wareable.com/ar/vuzix-blade-smartglasses-review

no standard for these technologies, each company has its own particularities about what they offer in its products and how they can be used.

The approach for augmented and mixed reality applications is slightly different from smartphones and tablets. In this case, it's an optical see-through, i.e. the real world is seen by the user's eyes, through the glasses' lenses, and only the virtual content is rendered on the display. A great advantage is that because they are used like any regular glasses, the device's camera is already facing and looking in the same direction as the user is. The same goes for tracking and positional sensors like GPS, gyroscope, accelerometer, and others because they are built-in on the glasses, they will reflect exactly the user's position, orientation and so on. Another advantage is that some companies build their solutions in a way that if the user needs prescribed lenses, they can be modified to have those included instead of regular lenses (Eyecare Business, 2018; LaForge, 2018). The downside of this type of display device is the price. Although it varies, they are still recent, and companies usually integrate high-grade components so that they can deliver the best experience possible. The price of manufacturing and industry solutions are even higher because they need to be reliable and rugged due to the environment that they are used in.

3.3.3 Optical See-Through Head-Mounted Displays

A head-mounted display, as its name suggests, is a type of display device that is mounted on the user's head. Usually, these are more intrusive devices than smart glasses. A great example of head-mounted displays are the virtual reality headsets, such as the HTC Vive and Oculus Rift. On the augmented and mixed reality spectrum, these devices are also head-mounted but offer an optical see-through approach, similar to smart glasses. The main difference here is that these tend to be the top-level display devices, offering better position and motion tracking, better cameras, hardware, are more robust and rugged and offer more extensibility by software and integration with other systems. The usage of these devices for mixed reality is often referred to and known as "holographic computing" (Microsoft, 2018a). They can be seen as an upgraded version of smart glasses, as they work on the same basis but have more computing power and can take advantage of better hardware components.

Unlike smart glasses, head-mounted displays don't have a lens for each eye but do have a display that covers all the ear-to-ear area. However, they are built in a way that the user can wear his own prescribed or safety glasses underneath if he needs to. Another flexibility of some of these products is the integration with hard hats for industrial and manufacturing health and safety (Trimble, 2018), shown in **Error! Reference source not found.**.

Figure 11 - Microsoft HoloLens integrated with a hard hat[11]

One thing to denote here is that some of the high-end smart glasses also offer similar hardware components as these head mounted displays (DAQRI, 2018; ODG, 2018). In this case, choosing between one or other solution falls to software and hardware integration, convenience and preference.

3.4 Augmented and Mixed Reality Frameworks and Libraries

AR and MR are still cutting-edge technologies and its development is a bit limited, especially for web platforms. In this case, there are no standards nor established reference tools for this kind of software development. However, there are already some libraries and frameworks that provide tools to build this type of applications that can be easily integrated with web applications. In this section, the most promising, solid and used solutions are presented along with their particularities, benefits, and drawbacks.

3.4.1 ARCore

ARCore is Google's augmented reality SDK for Android smartphones, which makes use of several different APIs to provide core functionalities such as device's motion tracking, environmental understanding and light estimation (Google, 2018c). Version 1.0 was released in late February of 2018, with support for 13 different Android mobile devices (Gosalia, 2018a), and is currently in version 1.2 with more features, improvements, and support for a wider variety of Android devices (Gosalia, 2018b). Although being a native platform framework, there are SDKs for other platforms like Unity, iOS, Unreal and the web, through prototype browsers based on Google Chrome for both Android, WebAROnARCore, and iOS, WebARonARKit (Google, 2018a).

[11] Trimble hard hat solution for Micro HoloLens integration. Retrieved June 8th, 2018 from https://bimchapters.blogspot.com/2018/04/first-look-trimble-hard-hat-for.html

ARCore's motion tracking is based on a technique called odometry and mapping (COM), where the device's pose, position, and orientation, estimation is obtained by a combination of tracking visual features, named feature points, and the usage of the device's sensors (Google, 2018d). The environmental understanding relies on the constant tracking of feature points that appear to be laying on the same surface, making it an available plane where objects can be anchored to. Light estimation is achieved by analyzing the lighting of the environment being observed followed by a conversion to an average intensity and color correction. Lastly, ARCore also provides support for user interaction, through hit-testing, a technique that finds the projection of a tap on the screen on the real world, oriented points, which lets the developer access angled surfaces, anchors and trackables, the first is a defined plane, tracked over time, where the user wants to place objects and the second is the objects that are being tracked (in the case of the plane, it's each of its feature points), augmented images, 2D images that trigger a reaction, e.g. an animation, and could anchors API, which lets the developers build augmented reality applications that can interact with others devices and share the same augmented reality experience.

The main advantage of this technology is that it is targeting all Android mobile devices that feature the minimum requirements, hardware, and sensors. On the other hand, for the web platform, it only works on prototype chrome-based browsers which are unstable and are not kept up-to-date as the main chrome distribution, making them unreliable.

3.4.2 WebXR

WebXR is the new API that is being proposed and developed to be the standard for the entire range of the immersive web, from virtual reality to mixed reality and everything in between (Medley, 2018). It is replacing the former WebVR API, which was only meant to support virtual reality. Regarding Google Chrome, some virtual reality functionalities are already available in Chrome beta (version 67), where augmented reality support is available in Chrome Canary (version 68), as an early experimentation platform (Google, 2018b). This new API is being written by the Immersive Web Community Group (IWCG) with contributors from major companies such as Google, Mozilla, Microsoft and others and it is intended to be highly flexible and extensible, in order to provide an easy way to switch between virtual reality and augmented reality, because not every device supports both functionalities, and to accommodate future technologies, such as display devices (Immersive Web Community Group, 2018).

This web API works as a bridge to native APIs, taking advantage of them. Currently, for Android, it makes use of ARCore to get access to functionalities such as motion tracking, environmental understanding, and user input, amongst others, and ARKit for iOS devices. What this means is that although this is a web API and runs on the browser, the device where it is being used must be compatible with ARCore and have it installed. While still limited to devices with compatibility for this framework, this establishes a way to add mixed reality to the web with expectations that it will become a standard, opening a door for cross-browser compatibility.

3.4.3 ARToolKit

ARToolKit is an open-source software library for augmented reality applications (ARToolKit, 1999), providing SDKs for several different native platforms, such as Windows, Linux, and macOS (ARToolKit, 2015/2018). Originally, it was developed by Hirokazu Kato and released by the Human Interface Technology Laboratory (HIT Lab), University of Washington (Kato & Billinghurst, 1999) and, in 2001, the projected continued under the ownership of ARToolworks, which lasted until 2015, when the company was acquired by DAQRI (Lardinois, 2015). At that time, the project was at its fifth version and a version 6 was announced by DAQRI, although it was never released. Around late 2017, the projected started to lose support and maintainability and, in late March 2018, the ARToolKit website was no longer available, which led to the creation of ARToolKitX by Ben Vaughan and Phil Lamb, the former CEO and CTO of ARToolWorks, to "[…] ensure that the software is developed and maintained […]" (Vaughan & Lamb, 2018). To be able to be used in web applications, the project was ported from C/C++ to JavaScript, using Emscripten, an LLVL to JavaScript compiler (MDN, 2015), known as JSARToolKit.

The library is mainly focused on the use of computer vision algorithms to solve problems such as object tracking, from 2D barcodes and pattern markers to NFT (ARToolKit, 2016) and pose estimation of the device relative to the object being tracked on the real world. Regarding the rendering of virtual overlay, it offers a simple graphics library, based on GLUT, and it is also easily integrated with Three.js. The main advantage of this library is that it works on any browser that supports WebRTC, for media capture, and WebGL, for 3D rendering. It features a very complete API and fundamental concepts documentation. On the other hand, its inconstant project development and maintainability reveal a possible risk of product termination.

3.4.4 AR.js

AR.js is an open-source library for augmented reality web applications based on JSARToolKit, created by Jerome Etienne (Etienne, 2017/2018a). The purpose of this project is to make augmented reality more efficient on the web. It makes a different approach to how it is implemented and released. As opposed to JSARToolKit, it offers a different API based on the rendering technology that it is going to be used with. It is integrated with Three.js and A-Frame, and for each of these technologies, AR.js is used in a different way. Regarding the first one, the project has a main architecture made of 3 main components: the ARToolKitSource, the source media to be analyzed, captured by the camera; the ARToolKitContext, the main engine responsible for finding markers on the source media; and, lastly, the ARMarkerControls, which is based on the Three.js controls API, responsible to have the marker connected to the 3D overlay (Etienne, 2017/2018c). For the A-Frame integration, it follows the same declarative HTML syntax, providing components for a camera that follows a marker and a camera that is static and reports the marker position (Etienne, 2017/2018b).

The abstraction of the JSARToolKit API can be an advantage or a disadvantage, depending if there is the need to have control over the API or not. A more serious disadvantage is having

different implementations depending on the 3D rendering library that it is used with, which not only makes it impossible to change it at any time but also makes it impossible to be used without the rendering library. Another downside is its lack of documentation about the API.

3.4.5 ArUco

Aruco is an open-source helper library for augmented reality applications (AVA, 2018). It's a fiducial marker system that was proposed in a paper with three main contributions: an algorithm for generating marker dictionaries, an algorithm for detecting these markers with error correction and a solution to the occlusion problem in augmented reality (Garrido-Jurado, Muñoz-Salinas, Madrid-Cuevas, & Marín-Jiménez, 2014). Similar to ARToolKit, this technology focuses on the usage of computer vision algorithms to solve problems of augmented reality, such as object tracking, pose estimation and occlusion. Its implementation relies on native cross-platform OpenCV and Eigen3 (OpenCV, 2018), although it has been ported to JavaScript so that it can be used in the web (Mellado, 2015/2018).

Regarding its functionalities, object tracking is only done through the usage of fiducial markers that can either be generated or directly used through the predefined implementations. The generation algorithm accepts dictionary size and marker size configurations, making it a flexible system that can be adapted to different use cases. Marker detection works similar to ARToolKit and pose estimation is done through the usage of OpenCV posit algorithms (OpenCV, 2018). The occlusion problem is addressed by the usage of a square chessboard-like board filled with several markers in conjunction with an occlusion mask calculated by color segmentation. An advantage of this system is its high controllability, as its usage is not abstracted by a high-level API. On the downside, pose estimation works better if the multi-markers approach is used instead of a single marker per object and it directly depends on OpenCV and Eigen3, forcing these technologies to also be used and included in a project.

3.4.6 Argon.js

Argon.js is a web framework for adding augmented reality content to web applications, by taking advantage of the Argon4 augmented reality enabled web browser developed for iOS and Android devices (Georgia Tech Research Corporation, 2018a). Currently, it also works on other browsers, being limited to their base capabilities. The framework was created from a research project at the Augmented Environments Lab at Georgia Tech, with support from several companies. Currently, it is an independent open-source project supported by the Augmented Environments Lab and by Mozilla.

Its core functionalities are geospatial augmented reality and object tracking, using computer vision algorithms from the Vuforia AR SDK (Georgia Tech Research Corporation, 2018b; Vuforia, 2018). With the support from Mozilla, an integration with A-Frame was also developed. Although having a very complete documentation, from its concepts to the API and several

tutorials, Argon.js based web applications will not feature all augmented reality functionalities on a common browser such as Google Chrome, because it lacks the custom implementations done on the Argon4 web browser.

4 Augmented and Mixed Reality Frameworks and Libraries Comparison

Each software and technology referred and analyzed in section 3.4 has its own functionalities, advantages, and disadvantages. Some are more complete than others, offering a wide variety of functionalities. However, choosing a technology to help achieve the objectives of this dissertation must be based on solid reasons that are highly related to the requirements of the prototype application. This chapter presents an analysis summary, based on the details discussed in section 3.4, and a comparison of those technologies, in order to understand the best option or options to use for the development of the application.

Firstly, ARCore is a native SDK for Android-based devices, but, currently, it is only supported by a small number of mid and top-end smartphones. Its development for the web is possible through the usage of a Chrome prototype browser for Android. Regarding tracking, it allows tracking of complex images and feature points. For content rendering, it is not bound to any software. Table 1 contains the main important facts about this SDK.

Table 1 - ARCore analysis summary

Main Functionalities	**Advantages**	**Disavantages and Limitations**
Image tracking Feature points tracking Light estimation Anchors Cloud-ready	SDK developed and supported by Google Robust hybrid tracking Documentation	Native SDK Only runs in a small number of Android devices Web development only possible with prototype Chrome browsers

WebXR is a still-in-development and proposal API that is intended to become a standard for the web. At the time of writing, the initially available API experimentations for augmented reality only work in Google Chrome Canary. It makes use of ARCore for Android-based devices and for iOS systems it uses ARKit. Regarding its functionalities, it is intended to follow both ARCore and ARKit and allow their usage in web applications. Currently, it is only possible to use feature points tracking in order to perform hit-testing and understand horizontal surfaces in the world. Table 2 summarizes the most important features of this web API.

Table 2 - WebXR analysis summary

Main Functionalities	**Advantages**	**Disadvantages and Limitations**
Feature points tracking Hit-testing	Web API Intended and proposed to become a standard Support from several companies such as Google, Mozilla, and Microsoft	Still in proposal development Only a few features available as an experimentation Dependant on ARCore and ARKit (consequently, only on works on devices that support those SDKs)

For the analysis of ARToolKit, it is only taken into account the ported JavaScript version, which runs on plain web applications. It's an already established technology that does not follow any standards. Its version 5 revealed a ported version to JS, allowing its use in web applications. However, not every functionality from the main SDK is available in this version. It currently

offers tracking of 2D barcodes and square fiducial patterns. There is also an optional direct integration with ThreeJS in order to ease the development of 3D-based augmented and mixed reality content. It offers an event-driven API or total control over the process if that is needed. It also retrieves a wide variety of data when it tracks barcodes or patterns, such as the center point on the web page, vertices, lines, transformation matrix, among other information. A summary of its particularities, advantages, and disadvantages is presented in Table 3.

Table 3 - JSARToolKit5 analysis summary

Main Functionalities	**Advantages**	**Disadvantages and Limitations**
2D Barcode tracking 2D Fiducial pattern marker tracking Optional direct integration with ThreeJS	Event-driven API or total control over it High amount of retrieved data from tracking Runs on plain web applications Usage examples and documentation on the project's GitHub page	Discontinued No support

The fourth discussed technology is ARjs, a library that is based on JSARToolKit5 and essentially offers a straight-up optimized integration with either ThreeJS or A-Frame, abstracting their APIs. The author claims the library to have good performance even lower-end smartphones and it got popular on the project's GitHub page. Lack of API documentation is also noticeable. This technology is more desired and developed for augmented reality applications that only use 3D virtual content.

Table 4 - ARjs analysis summary

Main Functionalities	**Advantages**	**Disadvantages and Limitations**
2D Barcode tracking 2D Fiducial pattern marker tracking Integrated with 3D libraries	Based on JSARToolKit5 Good performance on lower-end old smartphones	Lacking API documentation Direct and non-optional integration with 3D libraries ThreeJS or A-Frame JSARToolKit5 API abstraction offering almost no control over it

ArUco is also similar to JSARToolKit5 where its tracking is also based on 2D barcodes and patterns. However, it's more optimized for using multiple barcodes or pattern markers as one, in order to compensate for the less robust and more performant tracking. Trackings don't retrieve much information as additional data must be calculated. It depends on OpenCV for image processing, marker tracking, pose estimation, and occlusion problem solving, and Eigen3 for algebra and mathematical operations. The main specifics of this library is presented in Table 5.

Table 5 - ArUco analysis summary

Main Functionalities	**Advantages**	**Disadvantages and Limitations**
2D Barcode tracking 2D Fiducial pattern marker tracking	Faster tracking Occlusion problem solution Control over the API	Dependant on OpenCV and Eigen3 Less robust tracking Little data about trackings

Lastly, Argon.js is an open-standards web framework for building augmented reality applications and is highly dependent on the Argon4 browser. However, some functionalities also work in browsers that implement WebRTC and WebGL APIs. It uses Vuforia SDK for image processing and object tracking. It has good documentation, but its development has not been updated for a year, at the time of writing. Unlike the other technologies, it supports spatial augmented reality. Table 6 summarizes the details of this framework.

Table 6 - Argon.js analysis summary

Main Functionalities	**Advantages**	**Disadvantages and Limitations**
Spatial augmented reality Object tracking	Good documentation Control of the API Supported by Mozilla	Highly dependent on Argon4 web browser and the Vuforia SDK

Each of these technologies has its own particularities, but some of them share a few concepts. The main functionalities presented for each one are not widely different and reveal the existence of two patterns of augmented and mixed reality, marker-based and markerless. The

first case is based on using barcodes, fiducial pattern markers or another type of visual indicators that are tracked by the technology and are can be used either as a point of interest or a reference point. They are also used as an identifier of an object because of its uniqueness. In this case, JSARToolKit5, ARjs, and ArUco are all based on using markers for their tracking capabilities. These reveal a more generic approach although impose the user to use printed visual indicators that are placed in the real world. Despite this limitation, they are useful in contexts where there is the need to have unique tracking and reliability. In conjunction with rendering libraries, it is possible to build applications that despite being less immersive, from an environmental understanding point-of-view, allow the recognition of unique features, i.e. the visual indicators, and provide augmented and mixed reality features on top of them.

The latter case, markerless, is based on the concept of not using any markers and have the technology being able to understand the real environment and learn about its surroundings. Often, concepts such as light estimation and surface detection are the core functionalities that these technologies offer. ARCore, WebXR, and Argonjs are all markerless augmented reality tools and are used in contexts where there is the need to detect floors, walls, measure the light intensity, among others. They offer a more modern approach that takes advantage of hardware and software advancements in order to understand the surrounding environmental features. However, regardless of having these great features, they fall behind when there is the need to have a more reliable object tracking.

Considering the practical objectives presented in section 1.3 and the functional and non-functional requirements described in chapter 6, there is the need to use a technology that allows implementing object recognition and tracking and it must run in a plain web application on Google Chrome. This excludes the usage of ARCore and Argonjs as they are highly dependent on other specific browsers. At the time of writing, WebXR is still in proposal and development and was also only presented in Google I/O 2018, halfway through the internship, which means that it wasn't initially considered as a choice for the development. However, it still does not offer the needed functionalities for this project. JSARToolKit5, ARjs, and ArUco are all marker-based technologies and are the more indicated alternatives. Still, they have their own differences and limitations which must be taken into account. ARjs is based on JSARToolKit5 but it is integrated with either ThreeJS or A-Frame and there isn't the option to exclude these rendering libraries, which enforces that the visual content must be 3D graphics. This limits the integration and reusability of Critical Manufacturing's MES visual components. The other two technologies are very similar given that both operate on different types of markers and offer more control over the API. However, ArUco depends on OpenCV and Eigen3 libraries which implies that they also must be included in the application as for JSARToolKit5 there is not the need to add additional libraries for it to fully work. Although being discontinued, ARToolKit went through several years of development and became more mature, revealing that its last version can still be used, having in mind that there is no future support and development. Given this comparison and analysis, JSARToolKit5 is a strong choice among the six different technologies and provide the needed functionalities to help in the development of the prototype application.

5 Value Analysis

Value is often associated with only its monetary portion, the cost. This is a precarious statement, as one must not measure the value of a product or service by how much it costs. In his book, (Miles, 2015) states that "[…] best value is determined by two considerations: performance and cost." where performance is defined by capability of a product to "[…] serve the customer's needs and wishes to the degree that he expects; […]". To better analyze this dissertation's value, a technique called Value Analysis is applied. It can be defined as (Miles, 2015):

> *"[…] a problem-solving system implemented by the use of a specific set of techniques, a body of knowledge, and a group of learned skills. It is an organized creative approach that has for its purpose the efficient identification of unnecessary cost, i.e., cost that provides neither quality nor use nor life nor appearing nor customer features."*

As such, this chapter provides information about the fuzzy front end of innovation, the value for the customer and the value proposition itself. It is also presented a House of Quality (HoQ) for the Quality Function Deployment (QFD) method.

5.1 New Concept Development

The Fuzzy Front End (FFE) of innovation is a process that takes place at the beginning of an innovation process where, generally, it produces valuable ideas, opportunities are found, and decisions are made. These activities will later influence development and production processes such as New Product and Process Development (NPPD) (Koen et al., 2001). However, this stage lacked a common well-structured process as the attempt to compare the FFE of one company to another failed (Koen et al., 2001), revealing the need to develop a model that could be established as a standard for the FFE phase of companies' innovation processes. The New Concept Development (NCD) model, shown in **Error! Reference source not found.**, developed by members of a total of eight companies (Koen et al., 2001), brings a common language to the

FFE stage, making this process more defined, structured and reliable by losing its fuzzy concept and its lack of organization.

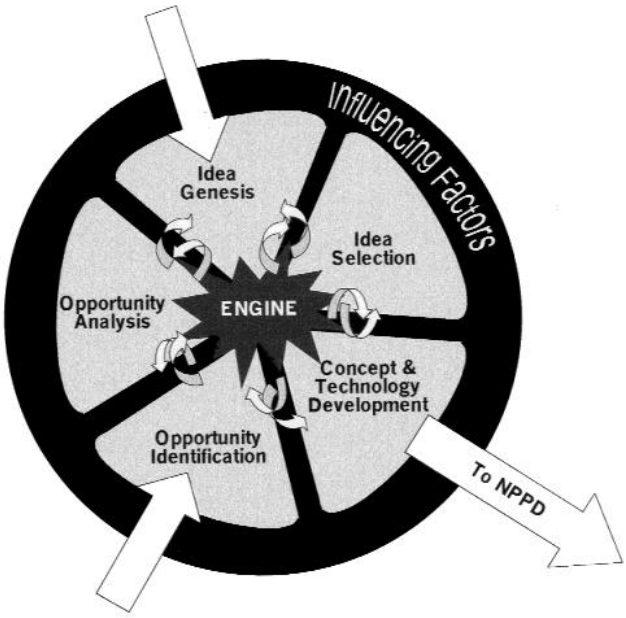


Figure 12 - The NCD model[12]

As described by (Koen et al., 2001), this model has three different parts: "[…] the inner area consists of five key elements, [Opportunity Identification, Opportunity Analysis, Idea Genesis, Idea Selection and Concept & Technology Development], where the ideas flow through these elements in a not sequential way, *i.e.* there is not a particular order that it must be used; the engine, which is what makes the five key elements work in conjunction with the leadership and culture of the organization and, lastly, the influencing factors that affect the entire innovation process while not being directly involved in it."[1].

In this section, the focus is on the inner area of the NCD model, by defining each key element relating each one with this dissertation's subject.

5.1.1 Opportunity Identification and Opportunity Analysis

These elements represent, respectively, the phases where companies identify opportunities to pursue and further analyze them by adding more information to make them more business-focused. The Opportunity Identification is a common start element for the innovation process. In this case, these two elements have already been thought and worked on by Critical Manufacturing. This originated opportunities to innovate by incorporating mixed reality in their product which, in the future, may provide new ways to how human-machine interaction is achieved by their clients.

[12] The New Concept Development model as defined by (Koen et al., 2001). Retrieved June 13th, 2018 from https://web.stevens.edu/cce/NEW/PDFs/Clarity_FEE.pdf

5.1.2 Idea Genesis

This third key element of the NCD model is also another common start point in the innovation process. Companies may have had more opportunities identified and/or analyzed that were not initially considered to be upgraded to an idea and select them to take part in another innovation process. Or, this stage could have transitioned directly from the last one. In either case, the core here is the "[…] birth, development and maturation of the opportunity into a concrete idea." (Koen et al., 2001). In the most part, this has already been made by Critical Manufacturing itself, although, throughout the first meetings between the supervisory team, the advisor, and the student, the idea suffered some changes that came from thoughts and brainstorming concepts. The more defined concept that was created was the addition of a new module/application to Critical Manufacturing's product that uses mixed reality to present information to the middle managing team of an Industry 4.0 environment, easing their work while innovating that same process. Some other concerns about this new platform also came across during this time and were considered to later stages. By the end of these meetings, the idea was already ready to go into the fifth key element, the Concept and Technology Development, defined in subsection 5.1.4.

5.1.3 Idea Selection

This element represents the part of this model where the company must choose the idea(s) that they want to develop and work on. It may be a simple step depending on how it is done but it is a serious activity because wrong choices have a great impact on the expected value. This is a case where the Front-End Innovation (FEI) process did not follow the five elements sequentially. The idea that had originally been selected by Critical Manufacturing, went back to suffer upgrades and became more mature.

5.1.4 Concept and Technology Development

This final element of the NCD's inner area is very important in the way that behaves like the bridge to a development process such as NPPD. The focus here is to take the idea even further and "[…] build a business case based on estimates of market potential, customer needs, investment requirements, competitor assessments, technology unknowns, and overall project risk." (Koen et al., 2001). In the NCD's final stage, a document was produced containing an overall view of the project, specifically the use case to be addressed, non-functional requirements and restrictions of the prototype application that is going to be developed, that resulted from work and interaction between the supervisory team, the advisor, and the student. This document is presented in Appendix A.

5.2 Value for the Customer

As said in chapter 5, the term "Value" is often defined wrongly or not measured in a correct way. Depending on the context, it can be more concisely defined as "[…] need, desire, interest, standard/criteria, beliefs, attitudes, and preferences." (Nicola, Ferreira, & Ferreira, 2012). In any business, value is the key element that can make a company successful and competitive in the market. The problem is that value can be perceived differently by customers and the perception of the producer may also differ from customer itself (Ulaga & Eggert, 2006) (Lindgreen & Wynstra, 2005). On the other hand, Value for the Customer (VC) can be described as (Woodall, 2003):

> *"[…] any demand-side, personal perception of advantage arising out of a customer's association with an organization's offering, and can occur as reduction in sacrifice; presence of benefit (perceived as either attributes or outcomes); the resultant of any weighed combination of sacrifice and benefit (determined and expressed either rationally or intuitively); or an aggregation, over time, of any or all of these."*

That same study (Woodall, 2003) denoted that most studies suggested sub-forms of VC that, generally, tended to be part of two categories: the Nature of Derived VC, which contains properties to evaluate VC in its derived form, and the Contingent VC, a sub-form of VC that categorizes it as by decomposing in a longitudinal (temporal) perspective, as shown in **Error! Reference source not found.**.

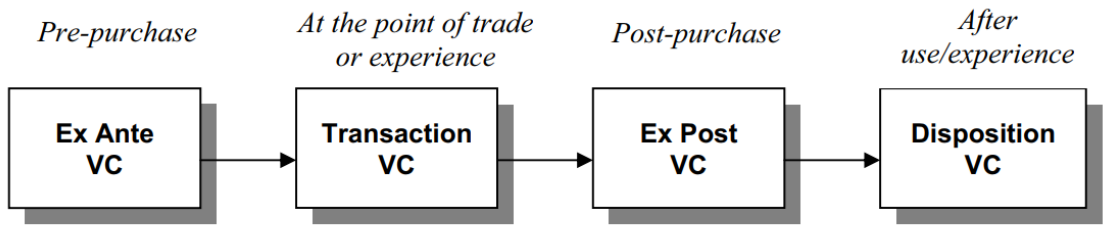


Figure 13 - Longitudinal Perspective on VC[13]

Each of these "temporal" phases, in **Error! Reference source not found.** above, relate to a different kind of value of the Contingent VC (Woodall, 2003) and can be further categorized as shown in Table 7 below:

[13] The longitudinal perspective on value for the customer, presented by (Woodall, 2003). Retrieved June 13th, 2018 from https://www.researchgate.net/profile/Tony_Woodall/publication/228576532_Conceptualising_'Value_for_the_Customer'_An_Attributional_Structural_and_Dispositional_Analysis/links/556c7c1708aeab7772231484/Conceptualising-Value-for-the-Customer-An-Attributional-Structural-and-Dispositional-Analysis.pdf

Table 7 - Contingent VC values categorized in a Longitudinal Perspective on VC[14]

Longitudinal perspective temporal position	Contingent VC value(s)
Ex Ante VC	Desired Value Expected Value
Transaction VC	Transaction Value Acquisition Value Exchange Value
Ex Post VC	Delivered Value Received Value Use Value Post Purchase/Performance Value
Disposition VC	Redemption Value

With this clarification of the longitudinal perspective on VC, it is presented below, in Table 8, the proposed benefits and sacrifices for the customer in each temporal position:

Table 8 - Proposed benefits and sacrifices for the customer on a Longitudinal Perspective on VC

Longitudinal perspective temporal position	Benefits	Sacrifices
Ex Ante VC	Service quality Utility Alternative Solutions Reliability Performance quality	Price
Transaction VC	Trust Support	Acquisition costs
Ex Post VC	Quality Utility Service support Results for the customer	Effort Costs of use Delivery and installation costs Training costs
Disposition VC	Features	Costs of repair Maintenance costs

5.3 Value Proposition

A value proposition is one of the most important building blocks of a company's business model, not only because it's how it proposes its value, but also because it possesses valuable information and provides an overall view of the company's products and services, the target customers, the value itself and its uniqueness (Osterwalder, 2004) (Weill & Vitale, 2001). Given

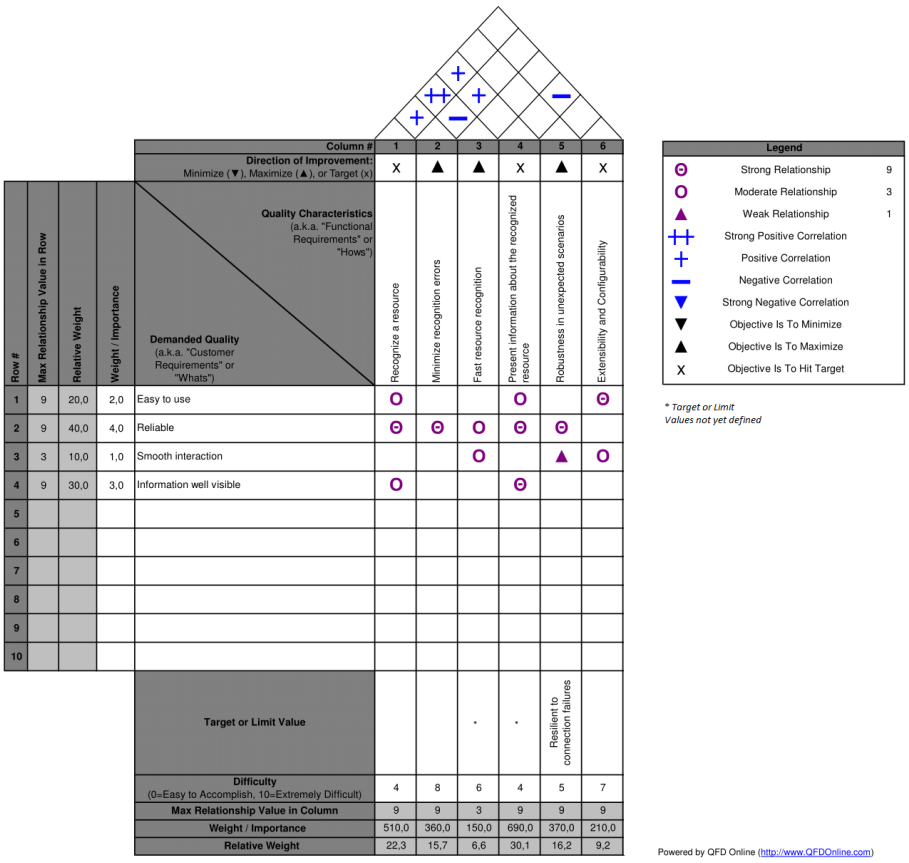
[14] Adapted from (Woodall, 2003, pp. 9–10)

this importance, the value proposition for the service that is developed throughout this dissertation is as follows:

> *Immerse yourself in the industry's environment and simplify your work. The Industry 4.0 does not have to cause fear.*
>
> - Straight up easy to use application
> - Customizable and configurable user interfaces and application data
> - Independent informative and interactable information for each recognized entity
> - Easier and faster access to the desired actions
> - Less time spent on searching for and switching between user interfaces
> - Overall increased productivity and less complexity in the operators and engineers' daily tasks

5.4 Quality Function Deployment

QFD is a technique used to design a product or a service based on customer needs and demands that involves communication and involvement from all members from the producer and supplier organizations, including technical and non-technical personnel (Warwick Manufacturing Group, 2007). The first step of this system is the HoQ where the attributes and needs of the customers are translated into technical and engineering features. In this case, these demands and technical characteristics are taken out and adapted from the functional and non-functional requirements, described in chapter 6. This model is used to describe the business idea with a focus on the relationship between customer needs and technical requirements. Figure 14 presents the HoQ model developed for the prototype application that is to be built.



Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Recognize a resource	Minimize recognition errors	Fast resource recognition	Present information about the recognized resource	Robustness in unexpected scenarios	Extensibility and Configurability
				Column #	1	2	3	4	5	6
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	x	▲	▲	x	▲	x
1	9	20,0	2,0	Easy to use	O			O		Θ
2	9	40,0	4,0	Reliable	Θ	Θ	O	Θ	Θ	
3	3	10,0	1,0	Smooth interaction			O		▲	O
4	9	30,0	3,0	Information well visible	O			Θ		
5										
6										
7										
8										
9										
10										
				Target or Limit Value			*	*	Resilient to connection failures	
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	4	8	6	4	5	7
				Max Relationship Value in Column	9	9	3	9	9	9
				Weight / Importance	510,0	360,0	150,0	690,0	370,0	210,0
				Relative Weight	22,3	15,7	6,6	30,1	16,2	9,2

Legend		
Θ	Strong Relationship	9
O	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

** Target or Limit*
Values not yet defined

Powered by QFD Online (http://www.QFDOnline.com)

Figure 14 - House of Quality

6 Requirement Analysis

The solution proposed in this dissertation follows a set of requirements in order to present the necessary functionality to the user. The document found in Appendix A holds a description of a use case that the system must be able to do perform, and a summary of non-functional requirements that it must comply to.

This chapter presents a requirements analysis, based on that information. Functional requirements are extracted from the use case definition, underlining each functionality that the application must have. Non-functional requirements are more detailed and arranged in the form of a FURPS+ model.

6.1 Use Case Definition

One of the main objectives is to build a prototype application that is intended to serve as a proof of concept which, initially, will be used in live demonstrations. These showcases will follow an existing demo use case that is being extended to be performed by the mixed reality application. Below, the use case is defined and structured having in mind is description and details in the Appendix A document.

Main Actor:

Factory shop floor engineer

Description:

The actor points the device to the coffee machine, which recognizes the devices and shows the current relevant Key Performance Indicators (KPIs), such as the state of the capsule loading port, the current water level, and capsule count, and the current coffee processing state (standby). This information can be represented through a mix of text, colors, and graphics. He can check-in into the machine (using the existing web interface) and see that he is the current employee

responsible for the machine. The actor can then open and close the capsule port, seeing in the interface the change of state with a minimum delay. He should then put a capsule in the machine and close the port. Then, he may proceed to order a small, medium, or large coffee through the mixed reality interface, or alternatively, he can press one of the physical buttons to do so. As the coffee is served, the interface is again updated, and the water level can be seen decreasing, and approaching minimum level. As the coffee is served, the actor can see that the resource needs maintenance, or request a new maintenance action, using the interface. He can then see that the interface signals the new need for maintenance, along with relevant information for the maintenance required. He should be able to complete the maintenance, filling the water level, and signal the completion either through the standard web interface or through the application

Pre-conditions:

The actor has the device configured and connected to the MES.

The actor has the device configured with the mixed reality application.

Post-conditions:

If the user interacts with the device through the mixed reality interface, the chosen action must occur on the device and further changes on the device must also be visible in the interface.

6.2 Functional Requirements

The functional requirements presented below were extracted from the use case definition and describe the core functionalities of the prototype application.

6.2.1 FR01 – Recognize an entity

The first functionality that the application must have implemented is the recognition of an entity through image recognition techniques. Initially, and regarding the use case definition, the system would only recognize a resource. However, this requirement was extended to also include other entities.

If needed, the entity can be tagged with a visual indicator, i.e. a tag, to help in the image recognition process. The recognition is not limited to one entity at a time, i.e. one or more entities can be simultaneously recognized. Another particularity is that an entity may be tagged with more than one visual indicator to display other information.

6.2.2 FR02 – Present information about the recognized entity

The second functionality underlines what happens after an entity is recognized. When a new recognition happens, information regarding the entity must be presented until it is no longer visible. The displayed information is, in case of a resource, related with the entity's KPIs, current checked-in employee, and maintenance management. For other entities, there are no mandatory requirements regarding what should be presented.

Also, there should be interactable information that works as the bridge between the interface and the physical object, allowing the user to perform actions through mixed reality interfaces.

6.3 Non-Functional Requirements

Additionally, there are some non-functional requirements that the prototype application must have implemented. These are presented in the form of a FURPS+ model, a form of structuring functional and non-functional requirements in a supplementary specification artifact in the IBM's Rational Unified Process (RUP) (Eeles, 2004). The functional requirements will not be considered for this model as they have been detailed above. The non-functional quality requirements are presented first, in Table 9.

Table 9 - Non-functional requirements

Category	**Requirement**
Functionality	-
Usability	NFR01 – Smooth UI experience
Reliability	NFR02 – Robust and reliable in unexpected scenarios NFR03 – Strive to avoid false positive identifications NFR04 – Minimize interface flickering with recognition errors
Performance	NFR05 – Recognize the entity as fast as possible
Supportability	NFR06 – Moderate degree of extensibility NFR07 – Moderate degree of configurability

The FURPS+ model is also constituted by other constraints regarding the software design, implementation requirements, interface requirements and, lastly, physical requirements. These are presented in Table 10.

Table 10 - Non-functional requirements additional constraints

Category	Requirement
Design constraint	NFR08 – Application integrated with CMF's MES solution
Implementation requirements	NFR09 – Application should be web-based NFR10 – Application should be built in Typescript over an Angular 4 framework NFR11 – Application should run on Google Chrome NFR12 – Application follows the company coding guidelines and is reasonable documented
Interface requirements	NFR13 – Follow the design guidelines of CMF's MES HTML5 interface
Physical requirements	NFR14 – Application must run in a Tablet-like device

More details regarding these requirements are presented below in order to provide more information and to better understand its implications.

6.3.1 Usability

Usability relates to details involving the user interface and its usages, such as accessibility, visual design, and user experience.

NFR01 – Smooth UI experience

Considering the use case definition and other real case uses, the person holding the device won't perfectly hold it still without causing any movement. Normal human movements must be considered along with possible distractions that may occur. These movements will, consequently, cause the device to also me moved, producing unwanted jittering and flickering to the interface. This lowers the user experience. This non-functional requirement involves the implementation to consider this possible flickering, minimizing it so that the user experience is smooth and pleasant.

6.3.2 Reliability

This category includes aspects such as the availability of the application, how it behaves in the case of failure, recoverability, and accuracy.

NFR02 – Robust and reliable in unexpected scenarios

Live demonstrations may present unstable connections between systems, e.g. the client and server applications. The system must be prepared to deal with these situations and not fail when the connection fails.

NFR03 – Strive to avoid false positive identifications

The object recognition requirement implies the identification and possible further tracking of entities. False positive identifications may occur when trying to recognize an entity, resulting in a different outcome than what it is expected. The prototype application must be robust and avoid false positive identifications which induct the user in mistake.

NFR04 – Minimize interface flickering with recognition errors

This non-functional requirement is slightly related to the NFR03. If recognition errors from false positive identifications happen constantly, e.g. the system is having problems on a recognition and is constantly changing the recognized entities, the interfaces will also suffer the same outcome. This flickering promotes a bad experience and lack of reliability. The same thing happens for missing recognitions, as the user is expecting to see an interface pop-up and it does not appear. It is intended that the system is prepared and robust for these types of scenarios.

6.3.3 Performance

The performance category of the FURPS+ model involves aspects such as system response time, algorithms time to execute, and recovery time.

NFR05 – Recognize the entity as fast as possible

FR01 dictates that the system must recognize an entity when a device is pointed at it. This non-functional requirement complements it, by indicating that performance must be taken into account for the recognition process. It should be as fast as possible while still implementing the other requirements and non-functional requirements.

6.3.4 Supportability

The last category is supportability, which includes such as testability, configurability, extensibility, and scalability.

NFR06 – Moderate degree of extensibility

The application should be able to be extended in order to use it with additional functionality. This relates to adding additional mixed reality interfaces, having access to recognitions, among other possibilities. It should also be extensible through the use of the MES HTML5 GUI dashboards module.

NFR07 – Moderate degree of configurability

The user interface should be configurable to display different information, having options to switch between different views. Application data such as the associations between tags and entities and view styles, e.g. zoom and rotation, should also be configurable in order to offer a more flexible application.

6.3.5 Additional constraints

For design constraints, the application must be integrated with CMF's MES solution, as a new module available in its HTML5 interface.

Regarding implementation requirements, the application should be web-based and built-in Typescript over an Angular 4 framework, following the same structure as CMF's MES HTML5 GUI. This web application should fully run on Google Chrome and its development must follow the company guidelines and all the code should be reasonably documented.

The user interface needs to follow the design guidelines of CMF's MES HTML5 GUI, maintain the same aspect, look and feel, reusing components when possible and maintain its design patterns.

Finally, physical requirements involve that the application should run on a tablet-like device, making use of its camera.

7 Design

This chapter addresses the design of the system, based on the requirement analysis of chapter 6. The different views of the system are organized in the 4+1 architectural view model (Kruchten, 1995). Each view is presented with diagrams that follow the Unified Modelling Language (UML) notation to better understand how the application was designed to fulfill all the functional and non-functional and the all the constraints. The scenarios view is not presented, as the only existing scenario is defined in section 6.1. Henceforward, visual markers are referred to as tags.

7.1 Architecture Overview

Being an extension to CMF's MES product, the application must adopt its the higher-level architecture. CMF's MES HTML5 GUI is built in Typescript over an Angular 4 framework. Angular is both a platform and a framework, which inducts applications to follow its own architecture in order to get the best results. Currently, Angular architecture is component-based, see Figure 15, which mostly focus on high reusability and the single responsibility pattern. The most important subjects of its architecture are modules, components, and services. Henceforward, when referring to components, it is related to Angular components and not UML components, unless said otherwise.

Modules can incorporate a set of components that are related to the same domain, workflow or related capabilities. They can export functionalities and import functionality from other modules as well. Some best practices are dividing the application into different modules, keeping distinct responsibilities separated from each other, which improves high cohesion, testability, and maintainability.

Components are the main part of Angular's architecture. A component has a class, which should contain some application data and logic, and a template, which defines the view of that component. These two entities are bound to each other, defining that the component has all the logic for its specific view and nothing more. The template is a combination of HTML and CSS

to visually present data to users. It can be enriched with directives that provide additional program logic. Data binding is responsible for binding information between the component class and the template. It is a two-way binding, i.e. user input and interaction will update the application data and, at the same time, if the bound application data is changed in the component, e.g. through services, the view will suffer changes to comply with the new data. Pipes can be used to transform information to be displayed differently to users, e.g. present currency according to the user's location. Finally, a component allows inputs and outputs, which can be used to pass data to the component and use its outputs to get access to the data that it exports, respectively.

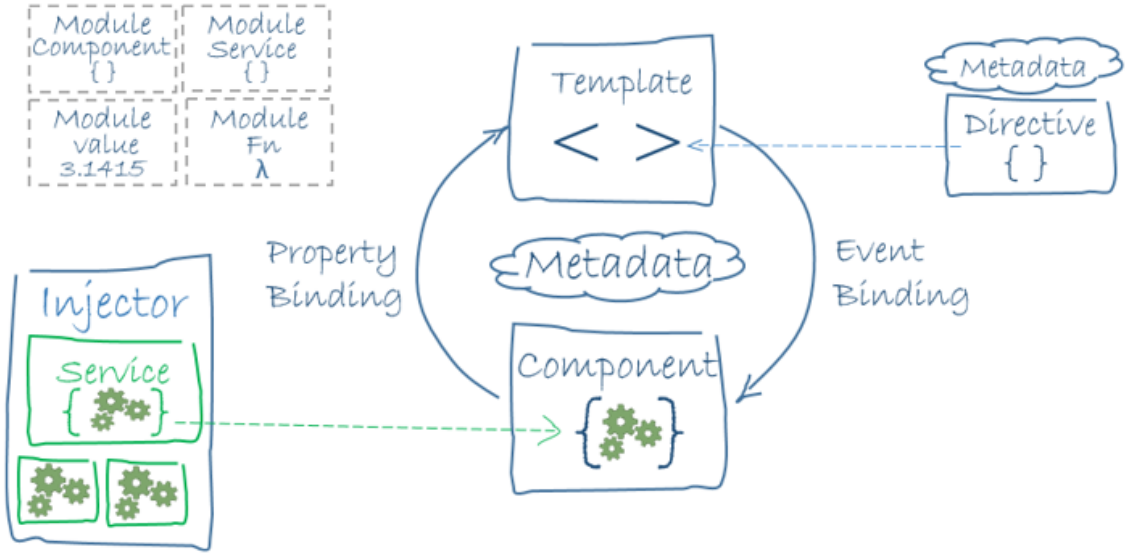


Figure 15 - Angular architecture[15]

Services are used to separate data and logic that aren't directly associated with a view. They can be used through Dependency Injection (DI) that dynamically provides services to components, reducing coupling and separating concerns.

Finally, most of these entities are just plain classes. To be distinguished from each other, they use decorators, annotations used right before the class definition, which provide the necessary metadata to Angular.

The prototype application follows this architecture, by having different components that are related to different views and different user interaction concerns. Each component is only related to one module and have its own necessary services to interact with the backend. These can be reused by other components if that is needed. This addresses the non-functional requirements NFR10 and NFR11, implying a web-approach to the application, using Typescript and Angular 4.

[15] Angular architecture pieces and how they relate to each other. Retrieved June 19th, 2018 from https://angular.io/guide/architecture#architecture-overview

For the main part of the application, the mixed reality functionality, it, internally, follows an event-driven architectural style for the communication and interaction between different layers. These layers are based on the (Bimber & Raskar, 2005) building blocks model for augmented reality, referred in section 2.3.2, with additional logic that makes the virtual information, i.e. the views, integrated with the physical objects.

Details, decisions, and alternatives are presented below in each view, providing more information about the design and how it was accomplished.

7.2 Logical View

The high-level overview of the system presents how the main UML design components are connected and complain to non-functional requirements. They are all integrated with CMF's MES HTML5 GUI, as required by NFR08. Firstly, the system must have some configurability, as defined in NFR07, allowing users to change the information that they want to see and allowing them to make changes to the application. Also, and according to NFR06, the application must have a moderate degree of extensibility, allowing the application to be extended to alter or add additional behavior. The first is achieved by including configuration wizards that allow the user to change some parameters of the application, such as managing what entities are recognized by the application and bootstrapping an entire facility that already exists in the system. Other configurations such as choosing different views, changing appearance styles and zooming are also included. These particularities are included in a higher-level component, named "PageMixedReality", that is responsible for having tools to perform configurations and to use them in conjunction with the mixed reality component. Figure 16 presents the high-level logical view of the application with the different developed and used components.

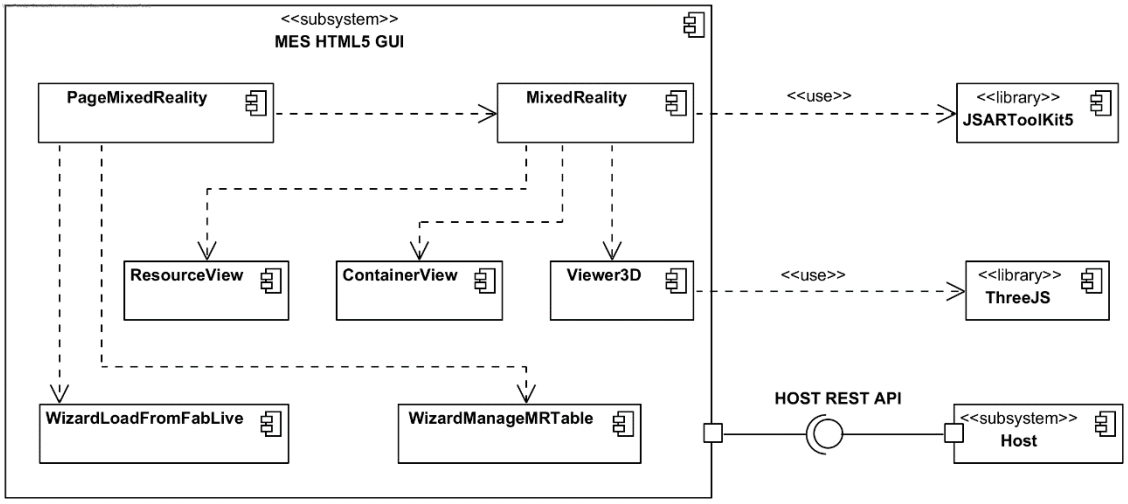


Figure 16 - Application High-Level Logical View

The MixedReality component is responsible for the logic involving the recognition and tracking of objects and presenting the mixed reality views, i.e. has the logic for the functional

requirements FR01 and FR02, and several non-functional requirements, namely, NFR01, to present a smooth UI experience, NFR02, NFR03, and NFR04, to deal with reliability, and, lastly, NFR05, which implies a fast recognition of entities. It also makes use of the JSARToolKit5 library API, to deal with the recognition and tracking of objects. The “ResourceView” and “ContainerView” components are special views that show different information for resources and containers business entities, respectively. The Viewer3D component is responsible for rendering and showing a 3D object representation of the recognized entity.

Finally, all the business information and data shown in the different views is retrieved from the CMF’s MES Host through REST APIs.

7.3 Development View

This view presents more details to each UML component and how the application is designed in a more fine-grained level. Firstly, the mixed reality core functionalities are described and explained how they interact with each other. After that, the other components of the application are addressed.

As said in section 7.1, the internal layers of the mixed reality functionalities are based on the (Bimber & Raskar, 2005) building blocks model for augmented reality applications. As these share some core concepts, such as tracking, registering and rendering, this model was adopted as the base starting point. Along with these layers, there is the media capture element, responsible to retrieve the video feed of the real world and the base concepts of Angular, described in section 7.1.

Angular’s base concepts rely on the definition of a component and a template view that is bound to it. The component acts like a middleware and performs a slight orchestration as it gets new information, e.g. new object recognition. Figure 17 presents a fine-grained UML component diagram that illustrates the core functionalities. The template is bound to the component via Event Binding and Property Binding, allowing them to change their shared data in real-time.

The “MediaCapture” component is responsible for retrieving the video feed from the device’s camera by using the WebRTC Media Capture API. This API allows the access to the device’s camera and microphone and retrieve, respectively, video and audio that can be pre-defined to be retrieved with different resolutions and can also be configured to access either a device’s front or back camera (Mozilla, 2018c). The stream that is retrieving from API is sent to the component which will then add it to the view and provide it to the tracker.

Although not directly required, it is necessary to track the position of objects in real-time, and not only perform a recognition. That is because if tracking is not performed, the position of an object won’t be known and the virtual content that holds the information for that object could only be statically positioned in the view, just like a plain overlay on a web page. Here, the tracker

is responsible to indicate the component whether an object is recognized and when it is no longer in the device's line of sight. However, it must also retrieve its position in real-time while it is visible. Two approaches for how this is done were designed and can be seen in Figure 17 and Figure 18.

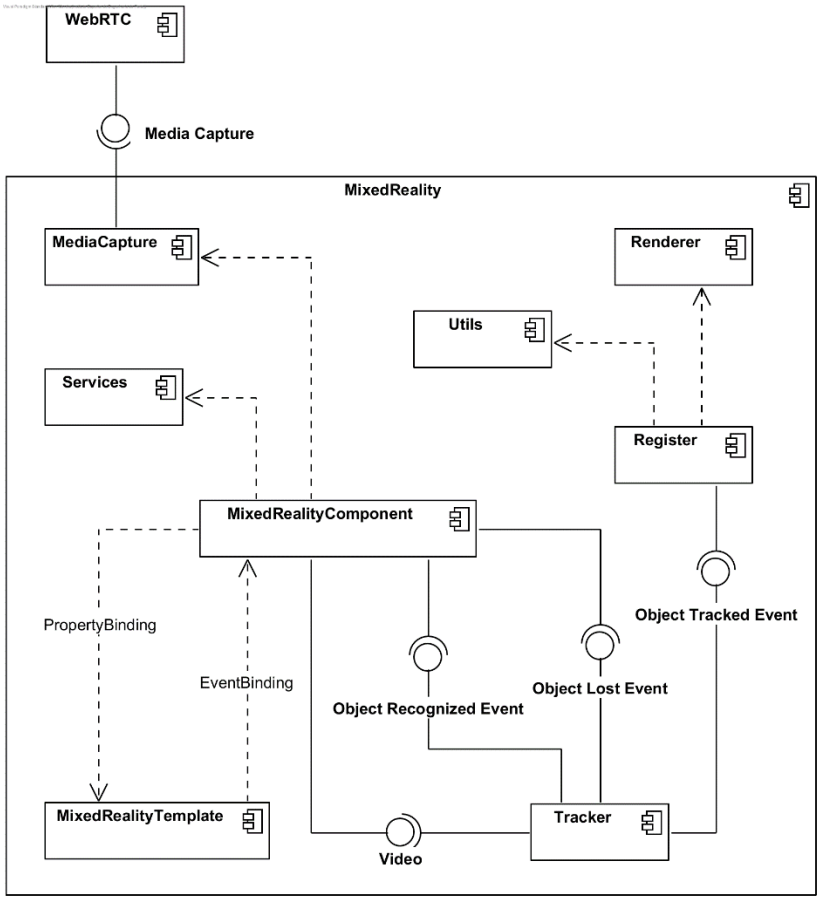


Figure 17 - Mixed reality core functionalities component diagram

In the first case, the tracker is designed under an event-driven architectural style which takes advantage of the event-driven API that JSARToolKit5 offers. That functionality is extended to the whole tracker, which fires events when a tag is recognized, lost and tracked. This is an asynchronous approach that improves performance by not constantly blocking the JavaScript's event loop, which could lead to bad user experience and bad results if the there is a large amount of processing to be done.

The second approach, illustrated in Figure 18, presents an alternative that does not take advantage of JSARToolKit5 event API. Here, the processing is less agile and needs to be manually done. This implies using a loop where, for each iteration, a frame from the video is analyzed to see if there are any tags present. The best advantage of this alternative is that there is more control over the API, as each function needs to be requested to JSARToolKit5. However, on the other hand, the event-driven API is already optimized and retrieves tracking events as they are

found, i.e. it does not wait for the entire frame to be analyzed and just fires an event each time a tag is found. Choosing this alternative would cause more blocking to the event-loop thread.

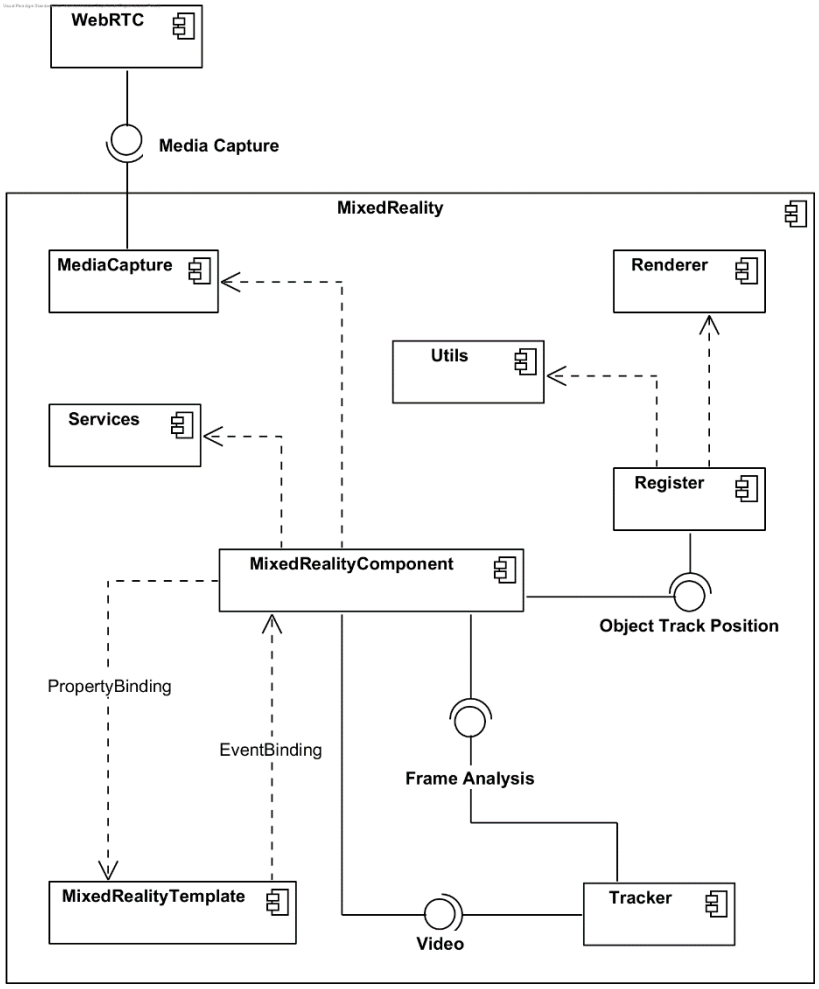


Figure 18 - Mixed reality core functionalities alternative component diagram

Because of the above arguments, the event-driven design was chosen, which also ends up complying with the programming language that the application is built on, Typescript. However, in order to be able to know if a tag is a new recognition instead of just a tracking event and if a tag is lost from the line of sight, the tracker had to be extended. JSARToolKit5 only notifies that a tag is found each time it is found in a frame from the video. This extension is referred further ahead in more detail.

The “MixedRealityComponent” is subscribed to listen for the recognition and lost events fired by the tracker. These events inform the component about a new tag and a lost tag, which will cause it to either use services to retrieve the object that is identified by that tag along with information to add to the view or remove a certain view that was displaying information about an object that is no longer visible. Each time a new recognition is performed, the new view is sent to the Register component and cached by it. This component is listening for tag tracking events from the Tracker which are fired more often, and by having the register listening to those

events, it will automatically align the view with the real object and show the view that is related to a certain tag. An alternative to this process is to not cache the views in the register and have tracking events being listened by the component. This would force it to act as a bridge and pass this information to the Register. Although saving memory by not caching the views, it would create more dependencies and make the process less agile and performant.

Finally, the Register component makes use of a Renderer to apply graphical transformations to the views, such as scaling and rotation, and then makes the view visible.

A UML class diagram that provides more information about the development view of the mixed reality core concepts is presented in Appendix B. In that diagram it is possible to see that the Tracker is composed by a "TagTracker" and a "TrackingCorrectionMechanism" classes. The latter class is responsible to analyze the tracking state and process data to understand if there are new recognitions and losses and fire the necessary events, where tracking events are directly fired by the "TagTracker" which only surpasses JSARToolKit5 tracking errors and fires tracking events with the data retrieved by that library. Section 7.4 details and illustrates how the overall activity flow is designed.

The gap between augmented and mixed reality is closed by having interactable views that perform actions in real objects. There are four different defined views: a plain informational view without interaction; an interactable view that uses CMF's dashboards module to interact with real-world objects; a dynamic view that displays different information depending on the recognized object type, displaying real-time information about the object state which changes if the object state also changes, using CMF's message bus; and, finally, a custom view that may be customized. The first one only acts like an augmented reality interface while the second and the third are interactable and somewhat bound to the physical objects through actions and changes. The last is completely custom which can end up being anything.

To finalize the details about the "MixedReality" component, Angular allows components to have inputs and outputs which contributes to extensibility and reusability. In this case, this component allows several inputs to alter its behavior such as turning on and off some rendering styles, e.g. zoom, rotation and scale, pass a custom view to be used, allow views to be dragged through touch-and-drag events and change the currently displayed view. As outputs, the component emits events each time a new object recognition is done and also when an object is lost. These contribute to both extensibility and configurability non-functional requirements.

In Section 7.2, the high-level component diagram also presents a component named "PageMixedReality". This is the parent component that uses the mixed reality functionalities, extends and configures its behavior, and can be accessed through the MES HTML5 GUI. It also reuses several MES HTML5 GUI components in order to main its graphical design, as defined by NFR13. Figure 19 illustrates how this component interacts with the rest of the application.

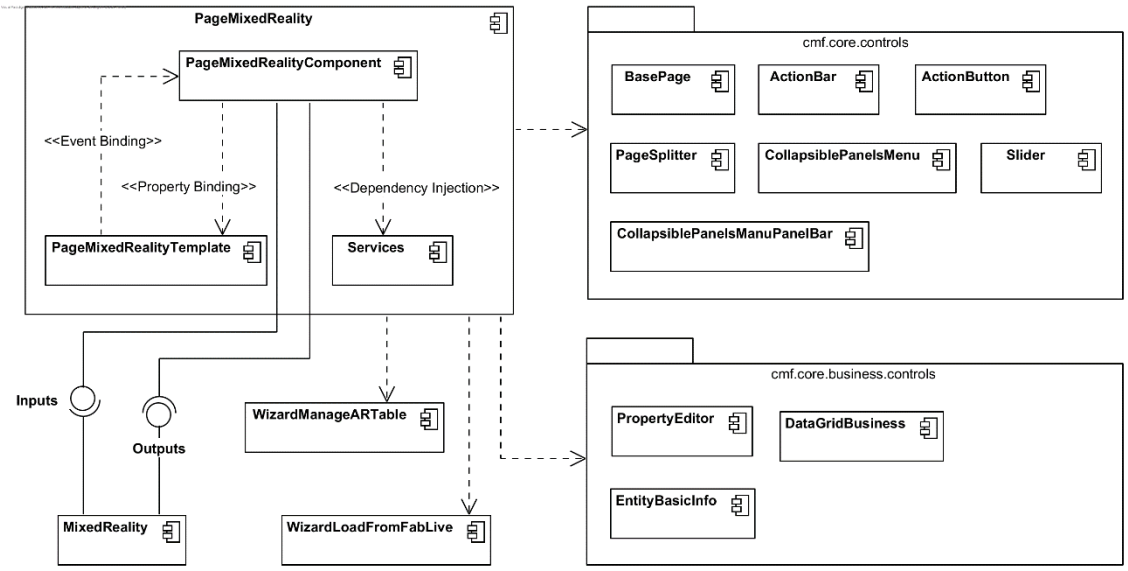


Figure 19 - PageMixedReality component diagram

It uses the “MixedReality” component via inputs that it sends to that child component and also reads its outputs to have access to more information.

Apart from using the mixed reality functionalities and reusing some UI components from the MES HTML5 web application, it also uses two wizards that were developed in order to allow more configurability. The first, “WizardManageARTable”, allows the user to change which objects are associated with tags and allows the user to download the tag image. The second, “WizardLoadFromFabLive”, allows the user to bootstrap the application by choosing a FabLive3D instance and mapping each entity to a tag. FabLive3D is a module of MES HTML5 GUI that allows the user to see a real-time facility in 3D with access to visual information about each entity. All the data changed by these wizards is persisted to tables that exist in the MES, so that if the user configures the application in a certain way, that configuration is saved.

7.4 Process View

To better understand the main flow of the application, an activity diagram is presented in Figure 20. This only concerns the process of recognition, tracking, registering, and rendering.

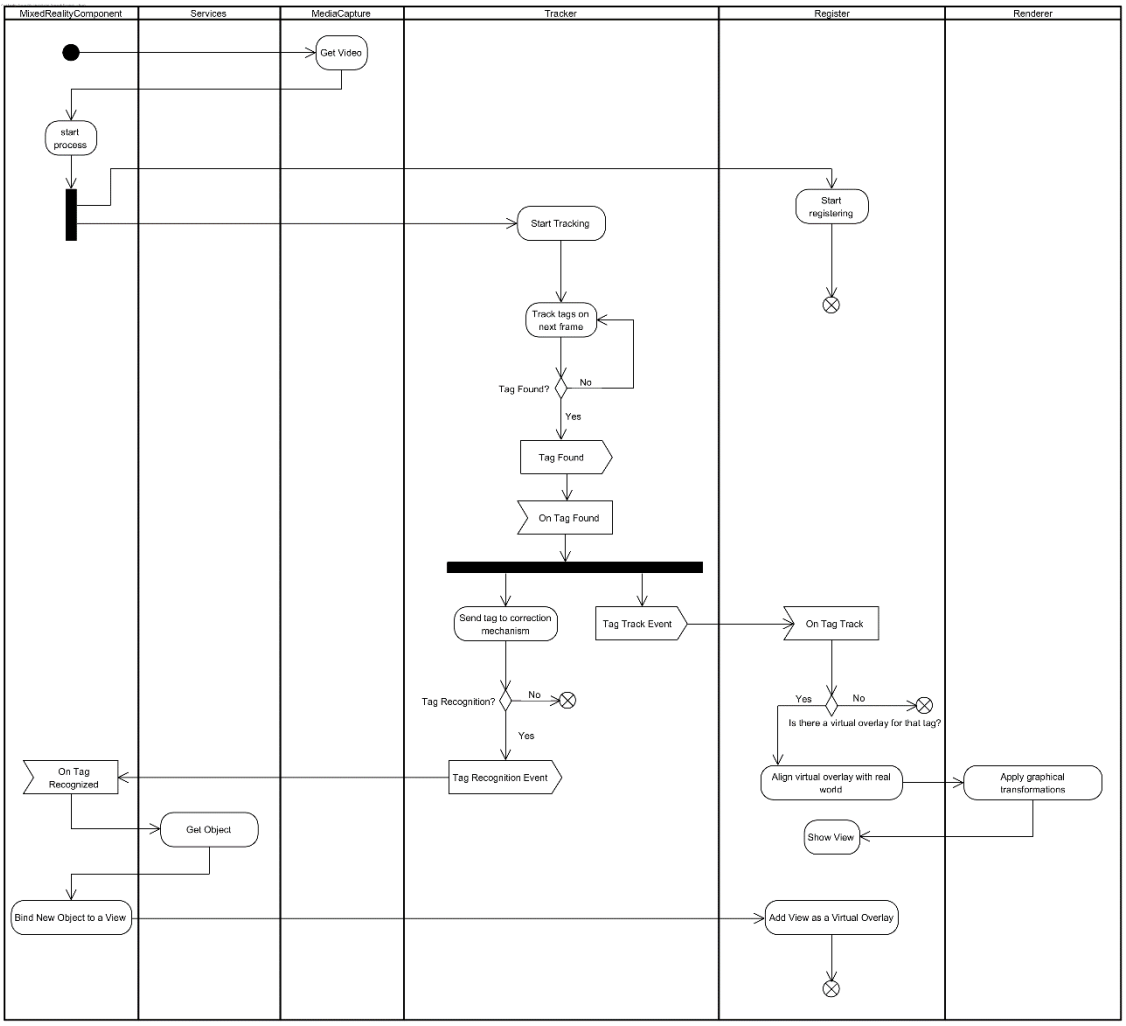


Figure 20 - Application main flow activity diagram

When the application is initialized and ready, it starts by getting the required video stream. After that, the process starts by turning on the tracker and the register. The register will be listening for track events while the tracker will start analyzing the video. As explained in section 7.3, this process is abstracted by the JSARToolKit5 API, and it only fires events when a tag is found. However, when it fires a tag found event, there are two different steps executed next. Firstly, there is error checking and if it there are no problems, a tag track event will be fired and the register will handle it by checking if it has a view for that tag, properly aligning it and displaying it in the device. The other step is sending the tag to a correction mechanism which processes it to know if it's a new recognition or not. Periodically, this mechanism also checks if a tag has been lost. If there is a new tag recognition, a tag recognition event is fired, and the mixed reality component will handle it, by using services to find the object that is related to that tag. That object will then be bound to the view and that view is then sent to the register to be cached and used. When a tag is lost, a tag lost event is fired, and the mixed reality component will remove the view from the template and tell the register to delete it from its cache.

7.5 Physical View

The use case definition presents a scenario where the user uses a tablet device to access the mixed reality interface. It implies a physical constraint, described in NFR14, where the application is accessed through a tablet-like device. NFR11 also specifies that the application must run on Google Chrome. Together, these constraints force a physical system distribution where a device with Google Chrome is used to access the web application. Figure 21 presents the system deployment diagram and illustrates how each node is connected.

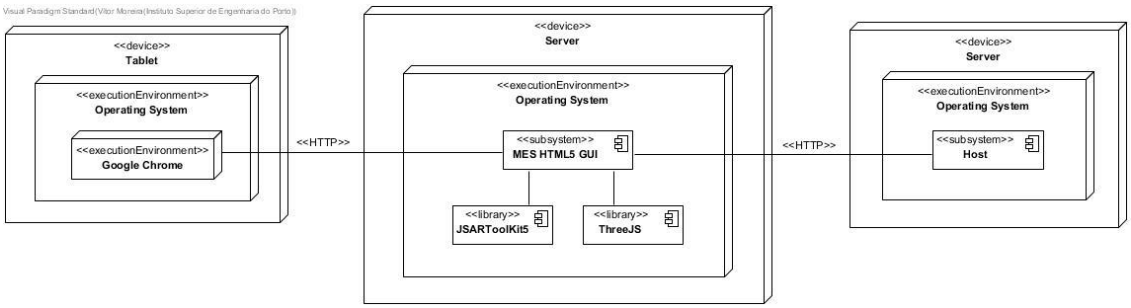


Figure 21 - System deployment diagram

The tablet device must have Google Chrome running and accessing the CMF's MES HTML5 GUI through the HTTP protocol. This application is running on a machine and has JARToolKit5 and ThreeJS as dependencies. Lastly, the application communicates with the host backend server.

8 Implementation

This chapter exposes some of the most important implementation details of the system designed in chapter 7. The different developed components are addressed individually, explaining how they work in practice and presenting, when necessary, code extractions and image examples.

8.1 Mixed Reality Core Functionalities

The mixed reality core functionalities are the main part of this dissertation and, as such, a solid explanation of its implementation is, hereby, more deeply discussed. For a better understanding of the built application, this section is divided into four subsections, one for each core functionality.

8.1.1 Media Capture

Video capture is achieved through the usage of the Web RTC Media Capture API. It allows capturing of both audio and video and to set some constraints. This was helpful as it was set in the constraints that audio is not required, and the ideal width and height were also given. An async function was used to wrap this behavior to simplify code reading and to better underline that this code is asynchronous.

```
public async getUserMediaStream(): Promise<MediaStream> {
        const videoConstraints = {
               audio: false,
               video: {
                      width: {
                             ideal: this._width
                      },
                      height: {
                             ideal: this._height
                      },
                      facingMode: {
                             ideal: this._facingMode
                      }
               }
        }
        this._mediaStream = await
        navigator.mediaDevices.getUserMedia(videoConstraints);
        return this._mediaStream;
}
```

Code 1 - Retrieve video from Web RTC

8.1.2 Tracker

This component uses JSARToolKit5 to perform object tracking. JSARToolKit5 allows different tag types to be used, with different limits of total available tags and with or without error correction. Its tags are based on 2D matrix barcodes, from 3x3 matrices to 6x6 and for each type, there are variations with error correction, as shown in Table 11.

The larger the matrix size, the higher the number of tags supported that can be used. However, larger matrices result in more complex processing that degrades in reliability with more distance (ARToolKit, 2015). Also, versions with larger error correction and detection support fewer tags but result in less likelihood that they will be misrecognized. Currently, the system uses the AR_MATRIX_CODE_5x5_22_12_5 matrix code type because it is a good balance between the number of available tags and reliability of recognitions.

Table 11 - JSARToolKit5 tag types and maximum number[16]

Matrix code type	Maximum number of tags	Number of bit errors that can be detected	Number of bit errors that can be corrected
AR_MATRIX_CODE_5x5_BCH_22_7_7	127	6	3
AR_MATRIX_CODE_4x4_BCH_13_5_5	32	4	2
AR_MATRIX_CODE_5x5_22_12_5	4095	4	2
AR_MATRIX_CODE_3x3_HAMMING63	8	2	1
AR_MATRIX_CODE_4x4_BCH_13_9_3	512	2	1
AR_MATRIX_CODE_3x3_PARITY65	32	1	0
AR_MATRIX_CODE_3x3	64	0	0
AR_MATRIX_CODE_4x4	8192	0	0
AR_MATRIX_CODE_5x5	4194304	0	0
AR_MATRIX_CODE_6x6	8589934592	0	0

The first block to analyze is the configuration and the start method of the tracker, see Code 2. The JSARToolKit5 tracker is controlled and accessed through the ARController class instance which needs to be initialized. It takes the media element that contains the stream to the video as the first parameter and a camera configurations file URL as the second parameter. This file contains data about the camera calibration. For this project, the default file was used. However, ARToolKit also provides a native application that can be used to calibrate a camera and generate the calibration data file.

[16] The total different types of tags, maximum number per type and error correction information of JSARToolKit5 library. Retrieved and adapted June 21st, 2018 from https://github.com/artoolkitx/artoolkitx/wiki/Creating-and-using-square-barcode-markers

```
public startTracking(): void {
        if (this._isTrackerInit || this._isAnimationLooping) { return; }
        this._arController = new ARController(this._mediaElement,
        this._cameraParams);

        this._arController.onload = () => {
              this._arController.setPatternDetectionMode(artoolkit.AR_MATRIX_CO
              DE_DETECTION);
              this._arController.setMatrixCodeType(AR_MATRIX_CODE_5x5_BCH_22_12
              _5);
              this._arController.setThresholdMode(artoolkit.AR_LABELING_THRESH_
              MODE_AUTO_OTSU);
              this._arController.addEventListener("getMarker", event =>
              this.processTag(event.data));
              this._trackerCorrectionMechanism.start();
              this.animationTick();
              this._isAnimationLooping = true;
              this._isTrackerInit = true;
        }
}
```

Code 2 - Configuration and start of tracker

After creating the ARController instance, it is set an onload function which will be ran when the controller is fully initialized. This function will set the controller to only track matrix-based tags, set the matrix code type and set an automatic threshold applied to the image so that it handles better different exposures and contrasts. Next, an event listener is added to that controller, which will fire events when it finds a tag that corresponds to the above constraints, along with a callback function to handle the event. Finally, as it is now ready to run, the tracker correction mechanism is started along with the animation loop, which will tell the controller to process each frame, and the boolean control flags are set to true.

The callback to handle the event, shown in Code 3, is responsible to check for recognition errors and emit the tag track event, followed by sending that recognition to the tracker correction mechanism. When JSARToolKit5 tracker recognizes a possible tag but can't compute its value, it will fire an event with the tag matrix id value of -1. This particularity is useful as it allows to easily surpass recognition errors. If the value is correct, an event "tagTrack" is fired, containing an object with all the relevant data from the JSARToolKit5, namely, the area of the tag on the image frame, the center point, the tag id, the lines and vertices of the tag image, and, lastly, the transformation matrix. To finalize this process, the tag id is sent to the tracker correction mechanism.

```
private processTag(eventData): void {
        if (eventData.marker.idMatrix !== -1) {
                this.framework.sandbox.mediator.emit("tagFound", {
                        result: {
                                area: eventData.marker.area,
                                center: {
                                        x: eventData.marker.pos[0],
                                        y: eventData.marker.pos[1],
                                },
                                id: eventData.marker.idMatrix,
                                lines: eventData.marker.line,
                                vertices: eventData.marker.vertex,
                                transformationMatrix: eventData.matrix
                        } as TagFoundDetails
                });
                this._trackerCorrectionMechanism.pushValue(eventData.marker.idM
                atrix);
        }
}
```

Code 3 - Callback function to handle the get marker event from JSARToolKit5

To conclude the main tracking process, the animation loop was implemented using the Request Animation Frame API. There were two main options to implement, either using this API or just using a loop. Using the Request Animation Frame, each iteration runs in the browsers animation loop whether using a loop would not run in the browsers animation loop and would create new iterations, blocking the main thread and highly decreasing performance and usability. This function, presented in Code 4, is very simple as it only defines a function expression named "tick" which will be run in the animation loop. It checks for errors and not enough data from the video first and then it tells the ARController instance to process the next frame. An important note here is that this function is run outside the Angular's zone, boosting performance.

```
private animationTick(): void {
        if (!this._isAnimationLooping) {
            const tick = () => {
                this._requestAnimationFrameId = requestAnimationFrame(tick);
                if (this._arController && this._mediaElement.width > 0 &&
                this._mediaElement.height > 0) {
                        if (this._mediaElement.readyState ===
                        this._mediaElement.HAVE_ENOUGH_DATA) {
                                this._arController.process(this._mediaElement);
                        }
                } else {
                        this.shutdownTrackingProcess();
                }
            };
            this._ngZone.runOutsideAngular(tick);
        }
}
```

Code 4 - Tracker animation loop function

As described in 7.3, the tracker is extended by a correction mechanism which is responsible for acknowledging tag recognitions and losses. JSARToolKit5 does not support this functionality and

it had to be implemented. This correction mechanism uses a buffer that holds a finite number of tag values and when it gets full, it is analyzed and determined whether there are new recognitions by analyzing if a value appeared more than a certain value threshold. If that value is being constantly recognized it tends to be a true value and not a recognition error. This process is done each time the buffer gets full, so its size it had to be determined by analyzing some cases. Firstly, it must not be a high amount, e.g. over 30, because if the application runs at 30 fps (frames per second) this would mean that would take a whole second to fill a buffer. Secondly, if the buffer length is very low, e.g. under 10, and the camera was capturing several tags at the same time, such as 4, the buffer would get full of a lot of different values but few entries for each value. This would be bad in the case that the tracker, for some reason, misrecognized a certain tag two or three times: the available data would not be enough to be analyzed and concluded. To ensure performance and reliability in this process, it was introduced a limitation to the maximum current tags that can be recognized of 3. The final buffer length is that value multiplied by 7, which is the number of entries that a tag should have to be approved as a recognition, resulting in a total of 21 entries. This limits the number of different tags that can be recognized at a certain time but ensures reliability and performance. Also, if more than 3 tags would be used at the same time, the interface would appear overflowed with much information, contributing to a bad user experience.

To ensure the maximum number of current tags, the correction mechanism also has an array with the current values that are being tracked. If that array is full, the value is discarded. Also, when a tag appears more than certain times in the buffer, implying a recognition, it is not fired a tag recognition event because it was already recognized and is still being tracked.

When the tracker correction mechanism is started, it initializes the arrays and other properties and then declares a setInterval function which will run each two seconds. This function will make sure to iterate the array containing the current tags and check if there have been updates to its recognitions. If not, it runs a callback function that will fire a tag lost event. This process is described in Code 5.

```
public start() {
        this.init();
        this._checkTagStateInterval = this._ngZone.runOutsideAngular(() => {
           return setInterval(() => {
               for (const tag of this._currentTags) {
                      if (tag.state === 1 &&
                      tag.tagDetail.lastIntervalRecognitions ===
                      tag.tagDetail.recognitions) {
                             tag.state = 0;
                             this._currentBufferIndex = 0;
                             this._onTagLostCallback(tag.tagDetail.tag);
                      } else {
                             tag.tagDetail.lastIntervalRecognitions =
                             tag.tagDetail.recognitions;
                      }
               }
            }, 2000);
        );
 }
```

Code 5 - Tracker correction mechanism update recognitions state

The rest of its functionality happens when tag id values are sent from the tracker. Each time a tag is tracked, the id is sent to the buffer. When it gets full, the buffer is analyzed, and it is calculated the most occurrent values, shown in Code 6.

```
private calcMostOccurrents(): number[] {
        const mostOccurent: number[] = [];
        const valueOccurrences = {};
        let currentValue: number;
        for (let index = 0; index < this._bufferLength; index++) {
                currentValue = this._buffer[index];
                if (valueOccurrences[currentValue]) {
                        valueOccurrences[currentValue]++;
                } else {
                        valueOccurrences[currentValue] = 1;
                }
        }

       for (const value in valueOccurrences) {
                if (valueOccurrences[value] > ((1 / (this._maxConcurrentTags +
                1)) * this._bufferLength)) {
                        mostOccurent.push(parseInt(value));
                }
       }

       return mostOccurent;
}
```

Code 6 - Tracker correction mechanism full buffer most occurrent values method

Each unique value is determined as one of the most occurrences if its total number of entries is at least ¼ of the buffer length. For each most occurrent value, it is processed to understand if it's a new recognition or if it's already being tracked, and, if so, its total number of recognitions needs to be incremented in order to not be deleted from the current tags array.Code 7 presents this logic.

```
private processNewTag(value: number) {
        const tagPosition = this._currentTags.find(position =>
        position.tagDetail.tag === value && position.state === 1);
        if (tagPosition) {
                tagPosition.tagDetail.recognitions++;
        } else {
                const freePosition = this._currentTags.find(position =>
                position.state === 0);
                if (freePosition) {
                        freePosition.state = 1;
                        freePosition.tagDetail = {
                                tag: value,
                                recognitions: 1,
                                lastIntervalRecognitions: 0,
                        } as TagDetails;
                        this._onTagRecognizedCallback(value);
                }
        }
}
```

Code 7 - Tracker correction mechanism process new tag method

If the value does not exist in the current array and there is at least one free position, out of the total of 3 max positions, the tag will be added to that position and a tag recognized callback is called to fire a tag recognition event.

8.1.3 Register

The process of registering, or aligning, the real and the virtual environment is simpler when using tag-based real-time tracking because the position of the tag is already known. However, there is still the need to address some concepts such as depth, rotation, and jittering. JSARToolKit5 provides some data about the position of the tag, its center point, lines and vertices, and a transformation matrix when a tag is tracked. The callback function to handle the tag track event, shown in Code 8, extracts these details and further sends the information to be processed in order to have the overlay aligned with the real world.

```
private onTagTrack(event: EventArgs): void {
        const details: TagFoundDetails = event.result;
        const htmlOverlay = this._virtualOverlays.find(overlay => overlay.tagId
        === details.id);
        if (htmlOverlay) {
                htmlOverlay.details.decomposedTransformationMatrix =
                MatrixUtils.decomposeTransformationMatrix(details.transformatio
                nMatrix);
                htmlOverlay.details.center = details.center;
                htmlOverlay.details.area = details.area;
                this.alignAndDraw(htmlOverlay);
        }
}
```

Code 8 - Tag track event callback function

The center point is used to correctly position the virtual content. A transformation matrix holds data related to linear transformations, in this case, translation, rotation, and scale. For translation, the values are all 0 because the library already gives the values of the center point. By experimenting with the API, it was noted that applying the transformation matrix as a CSS 3D matrix would only cause the content to be rotated and not scaled, as the scale values weren't correct. This forced an implementation of an approach which uses the computed area of the tag in the image frame as a reference point. When an object is closer to the camera it appears larger and if the object is placed further away, it looks smaller. This concept was applied to depth calculation, where further away tags will have virtual content smaller than closer ones. Also, for each tag-virtual content pair, if one is closer to the camera than the other, the virtual content will appear over the other's, creating a more immersive and correct experience. The transformation matrix was still used by extracting the rotation values and applying them as a rotation matrix. Regarding jittering, there were three algorithms developed that detect jittering for translation, depth, and rotation. For translation, it was implemented an algorithm that simply checks if the center point from one frame to another changed enough to be considered as a translation or is too small and is considered as camera instability. This process is presented in Code 9.

```
private detectTranslationJitter(htmlElement, currentX, lastX, currentY, lastY):
boolean {
        return Math.abs(currentX - lastX) < X_AXIS_DISPLACAMENT_THRESHOLD *
        htmlElement.getBoundingClientRect().width &&
        Math.abs(currentY - lastY) < Y_AXIS_DISPLACAMENT_THRESHOLD *
        htmlElement.getBoundingClientRect().height;
}
```

Code 9 - Translation jittering detection method

Depth jittering detection was a similar process, where the area was compared to understand if there was enough difference, see Code 10.

```
private detectDepthJitter(currentArea: number, lastArea: number): boolean {
        return Math.abs(Math.sqrt(currentArea) - Math.sqrt(lastArea)) <
        DEPTH_DISPLACAMENT_THRESHOLD;
}
```

Code 10 - Depth jittering detection method

Lastly, rotation jittering was applied to each axis by also comparing if there was too much angle difference in each axis from on frame to another, see Code 11.

```
private detectRotationJitter(currentEulerAngles, lastEulerAngles) {
        return Math.abs(currentEulerAngles.x - lastEulerAngles.x) <
        ROTATION_DISPLACAMENT_THRESHOLD &&
        Math.abs(currentEulerAngles.y - lastEulerAngles.y) <
        ROTATION_DISPLACAMENT_THRESHOLD &&
        Math.abs(currentEulerAngles.z - lastEulerAngles.z) <
        ROTATION_DISPLACAMENT_THRESHOLD;
}
```

Code 11 - Rotation jittering detection method

After all calculations are done, the different stylings are applied using CSS transformations. To ease this process, it was created a CSSRenderer class which abstracts the creation of these transformation and applies them one by one.

Regarding the mathematical operations such as decomposing a transformation matrix to extract rotation angles, matrix multiplication, and other operations, it was created a matrix utils file which contains each function.

Finally, it was also implemented an option to allow the user to not have the virtual content placed on top of the tag and rather be fixed and draggable. This logic is also present in the register component and is used through mouse and touch events on the browser's window. All the above options are configurable, as a user might want to switch between having each option on or off.

8.1.4 Mixed Reality Template and Component

As explained before, the template and the component are bound together through property and event bindings in order to keep to the state up to date. In this application, the view is composed by both the real world and the mixed reality interfaces overlaid. Code 12 presents the outer structure of the template, i.e. how the real-world video feed is structured along with the other elements.

```
<div class="mixed-reality-placeholder" cmf-core-controls-element-query cmf-
core-controls-progressIndicator #mrPlaceholder>
        <video class="source-video" autoplay #sourceVideo></video>
        <div class="mr-overlay" #mrOverlay *ngFor="let dummy of '
        '.repeat(_maxOverlays).split(''), let i = index">
                <ng-template [ngIf]="_mixedRealityInterfaceSlots &&
                _mixedRealityInterfaceSlots[i].state === 1 &&
                _mixedRealityInterfaceSlots[i].renderReady === 1">
                        <ng-container [ngSwitch]="mixedRealityView">
                                // THE DIFFERENT VIEWS GO HERE //
                        </ng-container>
                </ng-template>
        </div>
</div>
```

Code 12 - Mixed reality template outer structure

Both the video and each mixed reality interface are placed inside an HTML div element, with additional CSS stylings which makes the video element to occupy all the available space and be behind the rest of the elements. Each overlay is added subsequently after the video element. This is done through using the ngFor Angular directive to repeat certain elements. This directive iterates over a collection and, in this case, it was needed to add as many views as concurrent tags that can be analyzed. The Tracker component limited the amount of current tags to 3, which means that there must be a total of 3 views in the template, one for each tag. Because the ngFor only iterates over a collection, it was implemented a work-around which emulates a collection: an empty string is repeated 3 times and further split by an empty separator, making the string be split between character, originating an array of strings with 3 empty strings. By not using a defined array in the component the memory used for caching of that array is saved and is one less property bound between the component and the template. Each iterator creates an ng-template structural directive which will hold some content enriched with an ngIf directive that controls whether it should be presented or not. Each view is composed by an ngSwitch directive in order to have different a different interface depending on the chosen view. Each different case is detailed below.

The first possible view is an entity tile, that holds information about a certain object instance. This is a component of the MES HTML5 GUI that is here reused to present straightforward basic information, as shown in Code 13. This is an augmented reality interface and is also the default view in cases that there isn't a predefined choice.

```
<ng-template [ngSwitchCase]="MixedRealityView.EntityTile"
#defaultMixedRealityView>
        <div class="container-entity-tile">
                <cmf-core-business-controls-entityTile
                [instance]="_mixedRealityViewSlots[i].entity">
                </cmf-core-business-controls-entityTile>
        </div>
</ng-template>
```

Code 13 - Entity Tile view template

The follow-up view is more complex as it offers full access to a real object as long as it is connected and configured with the MES. It makes use of the dashboards module, shown in Code 14. In this case, a UI Page was created in the MES HTML5 GUI and is used to interact with an object. That page is loaded in the application startup and used for each view.

```
<div class="container-ui-page" *ngSwitchCase="MixedRealityView.UIPage">
        <cmf-core-dashboards-pageCore
                [page]="_mixedRealityViewSlots[i].uiPage"
                [pageModel]="_mixedRealityViewSlots[i].uiPageModel"
                [currentLayout]="_mixedRealityViewSlots[i].uiPageCurrentLayout"
                >
        </cmf-core-dashboards-pageCore>
</div>
```

Code 14 - UI Page view template

The third view, detailed in Code 15, is a dynamic approach where it renders a different template depending on the type of the object being tracked. Once more, a ngSwitch directive is used to control this behavior. Three types of objects were considered: resource, material and container. When a resource is being tracked, it is presented information such as its real-time state, maintenance management information and current checked-in employee. For a material, t is shown its 3D object representation, and, in the case of a container, it is shown its materials. The entity tile is also the default view.

This view also presents mixed reality as, in the case of a resource, changes to its state are shown in real time, by connecting this view to the message bus. More details on how this is implemented are presented in section 8.2.

```
<div class="container-dynamic" *ngSwitchCase=" MixedRealityView.Dynamic">
        <ng-container [ngSwitch]="_mixedRealityViewSlots[i].entityType">
                <uxar-core-resourceView *ngSwitchCase="'Resource'"
                [instance]="_mixedRealityViewSlots[i].entity"
                [showState]="true">
                </uxar-core-resourceView>
                <uxar-core-viewer3D *ngSwitchCase="'Material'"
                [instance]="_mixedRealityViewSlots[i].entity">
                </uxar-core-viewer3D>
                <uxar-core-containerView *ngSwitchCase="'Container'"
                [instance]="_mixedRealityViewSlots[i].entity">
                </uxar-core-containerView>
                <ng-container *ngSwitchDefault
                [ngTemplateOutlet]="defaultMixedRealityView">
                </ng-container>
        </ng-container>
</div>
```

Code 15 - Dynamic view template

Lastly, the custom view is an approach where the component that makes use of the mixed reality component can configure it to display something else. This was implemented to offer more configurability and extensibility. In this case, Angular's ngTemplateOutlet is used in conjunction with an input to achieve this objective. An HTML template can be sent as an input from a component to another via template variables. The parent component defines a template and annotates it with a template variable and then sends it as in input. In order for the parent component to have access to some information, a context as to be used. Here, in Code 16, the child component sends the current object instance being tracked in the context.

In the case that the custom template is not sent as in input, the default template also references the default view, i.e. the default custom view is also the entity tile.

```
<div class="container-custom" *ngSwitchCase="MixedRealityView.Custom">
        <ng-container
        [ngTemplateOutlet]="customMixedRealityViewTemplate ?
        customMixedRealityViewTemplate : defaultCustomMixedRealityViewTemplate"
        [ngTemplateOutletContext]="{instance:_mixedRealityViewSlots[i].entity}"
        >
        </ng-container>
        <ng-template #defaultCustomMixedRealityViewTemplate>
                <ng-container
                [ngTemplateOutlet]="defaultAugmentedView">
                </ng-container>
        </ng-template>
</div>
```

Code 16 - Custom view template

The component is responsible to start the mixed reality process and have the information ready for the view to display it. It is configured through the use of its inputs, shown in Table 12, and outputs, shown in Table 13.

Table 12 - Mixed reality component inputs

Input	**Logic**
tagEntityList	Holds the entity-tag association pairs
fixedPosition	Set the renderer to not apply translation
allowRotation	Set the renderer to apply rotations
allowDepth	Set the renderer to apply depth
allowJitterCorrection	Set the register to correct interface jittering
zoom	Change zoom value
mixedRealityView	Change the current displayed mixed reality view
customMixedRealityViewTemplate	The custom template value for the custom mixed reality view

None of the inputs above is mandatory for the component to work without errors. However, if no tag entity list is sent as an input, the tags are recognized but aren't associated with entities, which will end up not displaying anything. All the other inputs have default values if none are passed to override them.

Table 13 - Mixed reality component outputs

Output	**Logic**
tagRecognizedId	Sends an event holding the id of the recognized tag
tagLostId	Sends an event holding the id of the lost tag

Regarding the information used in the view that is relative to each one of the three maximum views, it is defined implementing an interface that gives more context and type information.

```
interface MixedRealityViewSlot {
        tag: number,
        readonly htmlElement: HTMLElement,
        state: 0 | 1,
        renderReady: 0 | 1,
        entity: Cmf.Foundation.BusinessObjects.Entity,
        entityType: string,
        uiPage: Cmf.Foundation.BusinessObjects.UIPage,
        uiPageModel: Page,
        uiPageCurrentLayout: PageLayout
}
```

Code 17 - Mixed reality view slot interface

When the component is initialized, there are created three objects that implement this interface, later used to map the information from the component to the view. The tag property is used to hold the id of the tag recognized, the htmlElement references the entire HTML block, presented in Code 12, for a slot. The state and renderReady are used to control, respectively, if the slot is occupied by a tag and if its ready to be rendered. The latter case is not to be mistaken with one where the view is only shown when it is rendered, implicating more time for the user to see something on the screen, as it is only as a safety measure, used to wait until the service is able to retrieve the associated entity instance from the MES Host and there is enough information for the view to be rendered without any errors. The properties entity and entityType hold the value of the entity's name and type, respectively, and, the last three properties hold the value to render a UI Page when that view is engaged.

To start the mixed process, as shown in section 7.4, the component needs to start each core functionality individually through orchestration. The logic is done by accessing the camera and retrieving video, followed by starting the register and the tracker. The method that holds this logic is shown in Code 18.

```
private async startAR() {
        this._mediaCapturer.changeVideoSizeConstraints(this._mediaWidth,
        this._mediaHeight);
        let mediaStream: MediaStream;
        mediaStream = await this._mediaCapturer.getUserMediaStream();
        this._mediaElement.srcObject = mediaStream;
        this.framework.sandbox.feedback.stopProgressIndicator(this.mixedReality
        ElementRef);
        this._tracker.startTracking();
        this._virtualRegister.start();
}
```

Code 18 - Mixed reality process start method

Lastly, this component listens for evens of tag recognition and tag lost. For each of them, a callback is used to hold the logic of, respectively, add information to the mixed reality view slots so the view template is updated and remove the data of a certain tag from the corresponding slot. The first case makes use of services that fetch data regarding the object that tag is associated with from the MES Host. This data is the entity instance and is fetched through its name and type.

```
private async onTagRecognized(value: number) {
        this.foundTagId.emit(value);
        const freeMixedViewSlot = this._mixedRealityViewSlots.find(overlay =>
        overlay.state === 0);
        if (freeMixedViewSlot) {
                const tagEntity = this.tagEntityList.find(entry => entry.tagId
                === `${value}`);
                if (tagEntity) {
                        freeMixedViewSlot.state = 1;
                        const entity = await
                        this._mrTableService.getObjectByNameAndType(tagEntity.en
                        tityName, tagEntity.entityType);
                        freeMixedViewSlot.tag = value;
                        freeMixedViewSlot.entity = entity;
                        freeMixedViewSlot.entityType = tagEntity.entityType;
                        const entityProperty =
                        freeMixedViewSlot.uiPageModel.properties.find(property
                        => property.name === "instance")
                        entityProperty.value = entity;
                        freeMixedViewSlot.renderReady = 1;
                        this._virtualRegister.addOverlay(value,
                        freeMixedViewSlot.htmlElement);
                }
        }
}
```

Code 19 - Tag recognition event callback function

All the information, when ready, is added to the slot and the view is then rendered. Lastly, it is sent to the register so that when the tag is tracked, the register is able to align in correctly.

Regarding the tag lost event, the callback function, presented in Code 20, is simpler, as the only logic is to clear the view slot and remove the overlay from the register.

```
private onLostTagId(value: number) {
        this.lostTagId.emit(value);
        const mrOverlayWithTag = this._mixedRealityViewSlots.find(overlay =>
        overlay.state === 1 && overlay.tag === value);
        if (mrOverlayWithTag) {
                mrOverlayWithTag.renderReady = 0;
                this._virtualRegister.removeOverlay(value);
                mrOverlayWithTag.entity = null;
                mrOverlayWithTag.tag = -1;
                mrOverlayWithTag.state = 0;
        }
}
```

Code 20 - Tag lost event callback function

8.2 Resource View

This view is specific for resource entities and englobes showing information about the current materials in a resource, the checked-in employees, maintenance management information and its state in real-time. Because of the amount of data to be displayed would make result in a large interface, and, consequently, conflicting with other possible interfaces that are also being presented, the different context is switchable. An example is shown in Figure 22.

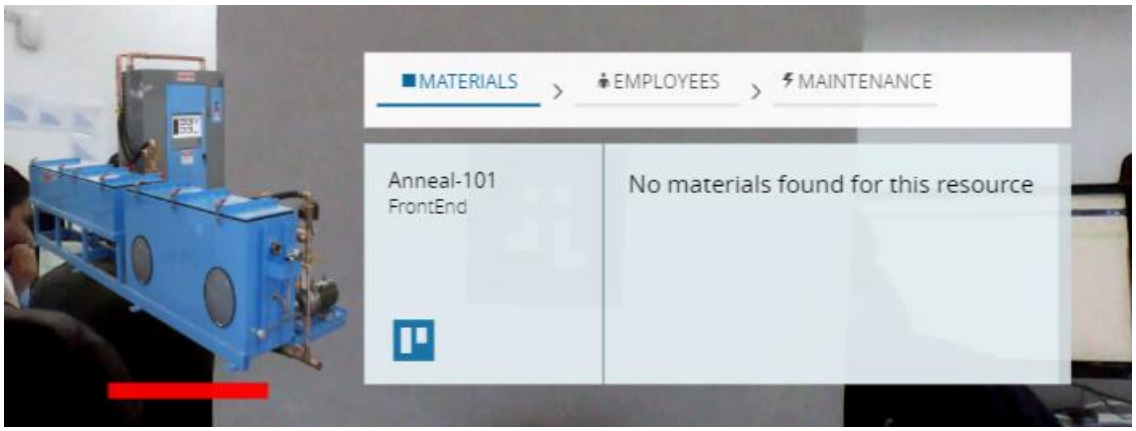


Figure 22 - Resource view example

Switching between the different information is done by clicking or touching the select list present in the top of the interface. Changing to a different perspective will change the information displayed below. The left side is composed by an image, 2D or 3D, representing the resource and a rectangle that illustrates the state of the resource. This state is updated in real-time, through integrating with the MES MessageBus and subscribing to messages regarding the resource's state change. This was developed as a proof of how the integration with a system such as a message bus can, easily, enable mixed reality and, asynchronously, integrate different systems. Subscribing to the message bus is shown in

8.3 Container View

Similar to the resource view, this is specific for container entities, which have materials. This interface is not interactable, only visually informative. It presents, for each container, all its materials and their type. An example can be seen in Figure 23.

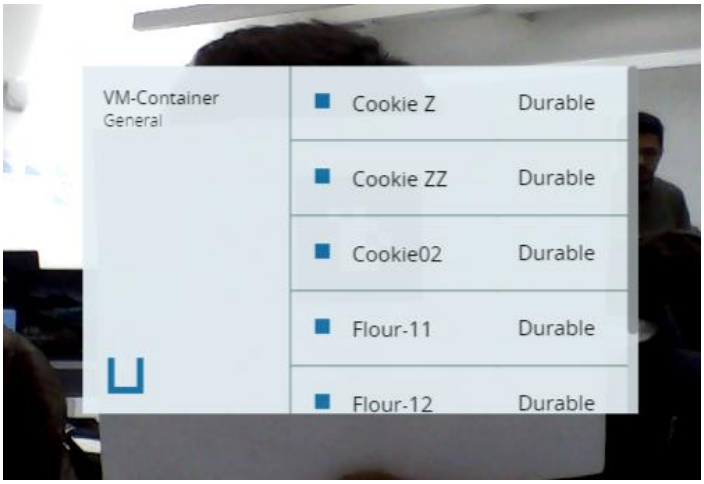


Figure 23 - Container view example

As opposed to the resource view, this is not integrated with the message bus or another system that allows asynchronous updates to the data being used. Consequently, it falls to an augmented reality interface, merely serving as a faster way to present this information to the user.

8.4 Page Mixed Reality

This component was developed to extend and use the mixed reality component. It's through this component that the user can access the mixed reality module and use its functionality. Also, it defines the outer interface, shown in Figure 24 and Figure 25, where the user can configure customize the application. Firstly, the interface is divided into 3 main parts: the action bar, marked as black, the center page, marked as green, and the sidebar, marked as red. The action bar holds the action buttons to access configurations, namely, the wizard to manage the table that holds the tag-entity associations, the wizard that bootstraps the application from a FabLive3D instance, and, lastly, the refresh and full-screen actions. The refresh button restarts the mixed reality process and forces the list of the tag-entity associations that is passed to the mixed reality component to be updated. Regarding full-screen functionality, which hides everything but the center area. The center area presents the interface of the mixed reality component, as analyzed in subsection 8.1.4. It's where the user visualizes both the real world and the mixed reality interfaces. Finally, the sidebar is composed of the customizations regarding styles, views and other information.

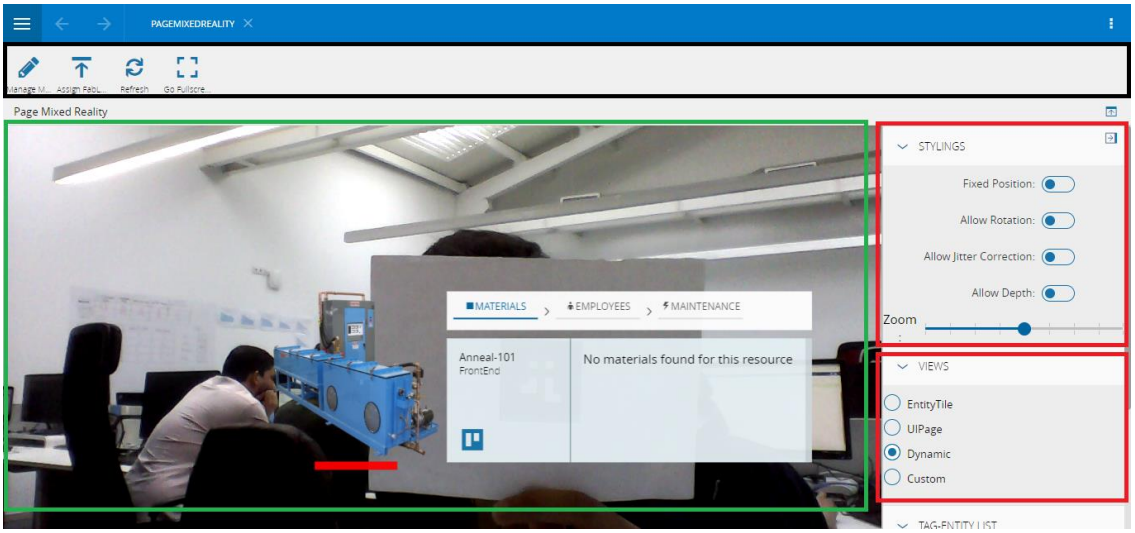


Figure 24 - PageMixedReality user interface details

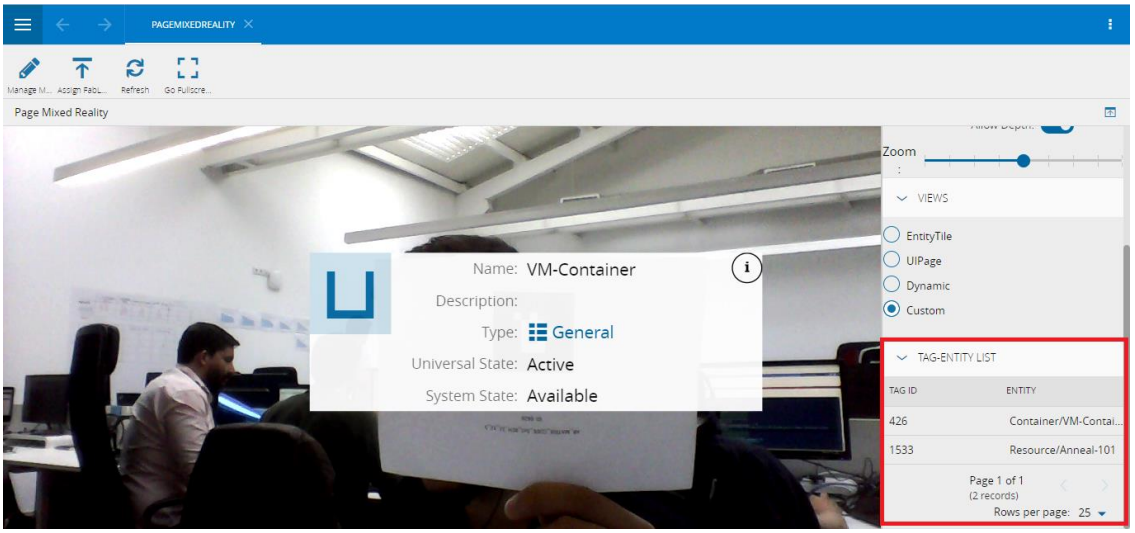


Figure 25 - PageMixedReality user interface details 2

Because the mixed reality component is more generic, the parent component that uses it can extend it anyway, as long as it holds the logic for managing its inputs and outputs. In this case, this component presents an interface that allows a simple and straightforward interaction, having options to change the stylings, the current view and even configure the application. Besides this, when it is initiated, it makes use of services that fetches all the tag-entity associations and sends that information to the child component

Regarding the view template, several components from the MES HTML5 GUI were reused to maintain the same design and appealing, as requested by NFR13. These are the ones illustrated in Figure 19, section 7.3. To make use of the mixed reality component, it's needed to import and use it in the template, as shown in Code 21.

```
<div class="container">
        <ng-template #customView let-instance="instance">
                <cmf-core-business-controls-entityBasicInfo
                [instance]="instance">
                </cmf-core-business-controls-entityBasicInfo>
        </ng-template>
        <uxar-core-mixedReality
                [tagEntityList]="_tagEntityList"
                [fixedPosition]="_fixedPosition"
                [allowRotation]="_allowRotation"
                [allowDepth]="_allowDepth"
                [allowJitterCorrection]="_allowJitterCorrection"
                [zoom]="_zoom"
                [mixedRealityView]="_selectedViewItem.value"
                [custommixedRealityViewTemplate]="customView"
                (foundTagId)="onFoundTagIdChange($event)"
                (lostTagId)="onLostTagIdChange($event)">
        </uxar-core-mixedReality>
</div>
```

Code 21 - Including the mixed reality component in the parent component

The inputs are all overridden with values because the interface's options allow to change the customization and configurability in real-time. This component also defines the custom view that is sent as in input to the mixed reality component. Currently, the outputs' callback functions are declared but have no logic, only because there weren't any functionalities needed to be developed on top of this data. However, an example of its usage is to have another category in the sidebar that displays, in real-time, the current tags and entities being tracked.

8.5 Wizard Manage Mixed Reality Table

The first wizard accessible through the action bar allows editing the table that has the information relative to the tag-entity associations, contributing to the configurability of the application. To make the process easier, this wizard allows the user to add new associations, update the current ones and even remove them. In the process of adding and updating, the user is able to set the desired value of the tag, which must correspond to its ID, and then he can choose the type of entity followed by an entity instance that corresponds to that type, as shown in Figure 26. All these options are mandatory. When setting the ID of the tag, its image is presented so that the user can download it and print it to add to the entity.

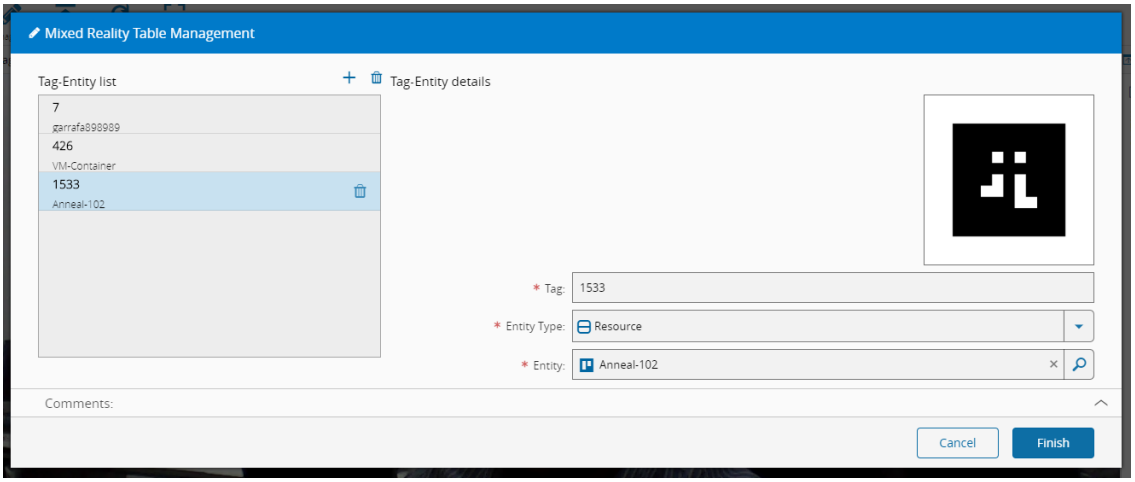


Figure 26 - Wizard Manage MR Table example

All the changes done to the list are persisted after the finish button is clicked. However, for the changes to take effect on the mixed reality component, the page must be refreshed by clicking on the refresh button of the action bar, which will force to fetch the information that just got updated and send it as the correct information to be tracked.

8.6 Wizard Assign FabLive3D Instance to MR Table

The second and last wizard accessible through the action bar allows the application data, namely, the table that has the tag-entity associations, to be bootstrapped from an existing FabLive3D instance that exists on the system. It's a more straightforward and simpler way to configure the application, reusing work that was already done to the system. This wizard takes the chosen instance and maps each entity instance to the table. Similar to the wizard manage mixed reality table, is also requires the page to be refreshed in order for the changes to take effect. The interface, however, is much simpler, as illustrated in Figure 27, as it only asks the user to choose an instance and hit the finish button for the bootstrapping process to start.

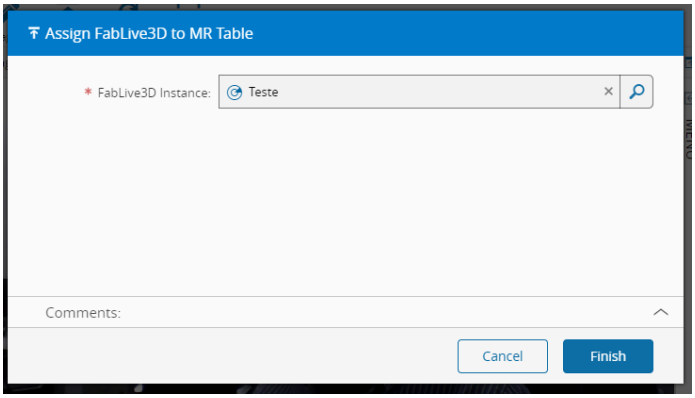


Figure 27 - Wizard Assign FabLive3D to MR Table

One thing to denote is that if the table already had some data, it is overridden.

9 Evaluation of the Solution

The proposed and developed solution is evaluated through the use of a methodology that defines the different dimensions of the system and the requirements of each dimension. The model built using that methodology underlines an ideal system, and it was further applied to two versions of the systems. In each one, the system was evaluated and understood how it compares to an ideal system and what quality and technical requirements were already implemented. In this chapter, it is presented the used methodology alongside the developed model. Lastly, it is evaluated each version of the system using that model, showing the results alongside an analysis.

9.1 Evaluation Methodology

The methodology applied to the evaluation of this project is based on analyzing and measuring the overall system state by applying a Quality Evaluation Framework (QEF), which is used to describe, at a certain point in time, its state and how it compares to an ideal system (Heidari & Loucopoulos, 2014). The model developed to describe the proposed system, presented in Appendix C, is divided into three different dimensions, named Functionality, Adaptability, and Reliability. Each dimension is further divided into factors and each aggregate one or more requirements which are adapted from the requirement analysis presented in section 6. Henceforward, when referring to requirements, it is the requirements defined by the quality evaluation model and not the requirements described in that chapter.

The Functionality dimension relates to functional requirements and user interaction factors. In the first case, they, FFR01 and FFR02, are measured either by having access to the functionality or not. In the latter case, FU01 relates to the usability non-functional requirement and is measured by analyzing if there is a smooth interaction between the user and the user interface and if, visually, there is instability correction to the motion of mixed reality interfaces.

Similarly, the Adaptability dimension is also divided into two factors, supportability and compatibility. The supportability factor regards the supportability non-functional requirements. The first, AS01, is evaluated by analyzing if the application is able to be extended or not, and if so, what those options are. The second requirement, AS02, denotes the degree of configurability of the application, whether it is possible to configure the displaying information, i.e. the mixed reality interfaces, and the application data configurations. The other factor, Compatibility, holds the requirements related to the restrictions of the application of having to run in a tablet-like device and in Google Chrome, AC01 and AC02 respectively. These are also measured by only understanding if they either are accomplished or not.

The last dimension presented in this model is Reliability and is divided into three factors, resilience, performance and reliability. The resilience factor ensures that the application does not suffer errors in unexpected internet connection failure issues, defined by RRS01. Performance holds the requirement RP01 which underlines that the application should recognize the entities as fast as possible., although a reference time value was not determined. Lastly, the resilience factor is composed of two requirements, RRL01 and RRL02 that are equal to the non-functional requirements NFR03 and NFR04, respectively. All the requirements of this dimension are evaluated as either having been addressed and implemented or not.

One last thing to denote is that the described ideal system was designed having in mind the current objectives and requirements, and not a future implementation of this system in the CMF's MES product.

9.2 Evaluation Results

The model discussed in the last section was applied to two versions of the system, a first prototype finished in April 25th, 2018 and a second finished on May 28th, 2018. The first version was underlined by its usage in Hannover Messe 2018 by Critical Manufacturing as a demonstration, where the second version was underlined by the end of the system's implementation.

Several requirements were already implemented in the first prototype, enabling the possibility for its usage. The system had an overall fulfillment of 70%, with the Functionality and Adaptability dimensions with a significant percentage of completion, as presented in Figure 28. The requirement FU01 is the only not fully completed requirement, as the system did not have an instability correction to the user interface. However, the Reliability dimension counted with less implemented requirements, resulting in a 25% of fulfillment.

q	D	Qi	Dimension	Q_j	W_{ij} (Factor Weight *j* in Dim *i*) [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*) [0,100]
70%	0.78	83.33	Functionality	100	0.67	Functional Requirements	10	FFR01 - Recognize an entity	100
							10	FFR02 - Present information about a recognized entity	100
				50	0.33	Usability	8	FU01 - Smooth user interaction	50
		87.5	Adaptability	75	0.50	Supportability	6	AV01 - Moderate degree of extensibility	100
							6	AV02 - Moderate degree of configurability	50
				100	0.50	Compatibility	6	AC01 - Application runs in a tablet-like device	100
							10	AC02 - Application must run on Google Chrome	100
		25	Reliability	100	0.25	Resilience	8	RRS01 - Reliable in unexpected scenarios	100
				0	0.25	Performance	8	RP01 - Recognize the entity as fast as possible	0
				0	0.50	Reliability	8	RRL01 - Minimize interface flickering with recognition errors	0
							10	RRL02 - Minimize false positive identifications	0

Figure 28 - QEF results for the first prototype

Although all the dimensions are balanced in terms of the number of requirements, the Functionality and Reliability dimensions have requirements with higher weight, making them complex to accomplish. In this case, the Reliability dimension has the least completed requirements when compared to the other ones. This reveals that, although the system is evaluated at 70%, this measurement is not linear and, as a consequence, is it needs to be understood that it presents a reliability flaw.

For the second prototype, an effort was put to, mainly, address the reliability related requirements along with a smoother UI experience. The implementation of a tracker correction mechanism to help in the recognition process and instability correction of the mixed reality interfaces presented a more solid system, which was concluded with an estimated evaluation of 94% and individual percentages of 100%, 100% and 75% for the Functionality, Adaptability and Reliability dimensions, respectively, as shown in Figure 29.

q	D	Qi	Dimension	Q_j	W_{ij} (Factor Weight *j* in Dim *i*) [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*)
94%	0.25	100	Functionality	100	0.67	Functional Requirements	10	FFR01 - Recognize an entity	100
							10	FFR02 - Present information about a recognized entity	100
				100	0.33	Usability	8	FU01 - Smooth user interaction	100
		100	Adaptability	100	0.50	Supportability	6	AV01 - Moderate degree of extensibility	100
							6	AV02 - Moderate degree of configurability	100
				100	0.50	Compatibility	6	AC01 - Application runs in a tablet-like device	100
							10	AC02 - Application must run on Google Chrome	100
		75	Reliability	100	0.25	Resilience	8	RRS01 - Reliable in unexpected scenarios	100
				0	0.25	Performance	8	RP01 - Recognize the entity as fast as possible	0
				100	0.50	Reliability	8	RRL01 - Minimize interface flickering with recognition errors	100
							10	RRL02 - Minimize false positive identifications	100

Figure 29 - QEF results for the second prototype

RP01 is the only requirement that was not entirely completed. Although the system was designed and implemented with it in mind, a reference value for comparison was not determined and the actual values and measurements of the recognition process' performance

were not calculated. All the other requirements were implemented, resulting in a stable system for demonstration and proof of concept purposes. It also revealed a growth and progress in the development of the system, with an initial focus on functional requirements followed by enhancements and continuous improvements.

10 Conclusion

In this final chapter, it is made and presented an overall critical balance about the whole work, what objectives were fulfilled, the results, future work, and other additional aspects. It is divided into two sections, Achievements and Results, and Future Work. In the first case, it is discussed what was achieved in the project, i.e. the fulfilled objectives, the implemented requirements, the overall results of the all the work, and other additional and important information. Lastly, in the second section, it is presented an overview of possible future work and recommendations, how this system can grow and become, in the near future, a new module in the Critical Manufacturing's product.

10.1 Achievements and Results

The main purpose of this dissertation was to understand how mixed reality could be integrated with a MES software in order to bring benefits to the Industry 4.0 working environment, reducing the complexity of the worker's daily tasks and thus increasing productivity, and build a functional prototype that shows its benefits and drawbacks. The theoretical problem was first approached by understanding what mixed reality is and how it can be applied to the Industry 4.0. The research revealed that mixed reality differs from other areas by being connected with the real world, and so, it was analyzed how it could be achieved through an integration with MES software. More study revealed that an Industry 4.0-ready MES comprehends IIoT and other communication-based technologies that allow software to be connected with physical machines and with it, being able to communicate and interact. Alongside with this information, it was also investigated real use cases and mixed reality applications in hard environments.

From a more practical perspective, it was developed a functional prototype that implements several functional and non-functional requirements and follows some of the researched real applications of mixed reality for the Industry 4.0. It provides the main functionalities which consist of recognizing a facility entity defined in the system and presenting information related to it. Additionally, it also implements all the non-functional requirements, although there are a

few things to point out. Firstly, the system was developed having in mind that the recognition process should be as fast as possible, as required by NFR05, although the measurement values and statistics were not determined. Secondly, non-functional requirements related to minimizing recognition errors were addressed by enhancing the tracking system with custom additional implementations that work to minimize these problems. Regarding the configurability and extensibility of the system, these were some open requirements that, initially, lacked concrete objectives and details. As the system was being built, different approaches for extensibility and configurability were being added, as long as they were in agreement with the supervisory team. All the other non-functional and additional constraints were fully implemented. Another achievement was the use of the first version of the prototype in Hannover Messe 2018, one of the largest trade fairs, where Critical Manufacturing attended. A small adaptation from the use case defined in section 6.1 was used, where the mixed reality application was connected and configured with the MES, and a coffee machine was also configured in the MES. When using the coffee machine to order a coffee, the water level and capsule count could be seen decreasing in the mixed reality interface. It revealed as an early accomplishment and an early proof of concept.

From a quality perspective, the system was evaluated using a QEF model, applied in both version 1 and 2, exposing and revealing its progress and development. Its results proved that the objectives and technical requirements for the prototype application were accomplished almost in their entirety.

Finally, the system also has its limitations. The maximum number of concurrent recognitions and further tracking of tags is capped at 3, in order to maintain performance. Also, the system is limited to recognize a total of 4095 different tags. Lastly, as any vision-based system, it does not have a 100% accuracy. However, the developed mechanism to help in tracking is able to identify occasional recognition errors. Still, the system is not prepared to deal with constantly misrecognizing a tag.

10.2 Future Work

There are a few possibilities for future work in order to improve this system. Mixed reality is still an area of development with scarce support, lacking standards and technologies. With time, recent technologies that aim to establish standards of development will mature and be ready to be included in production and be integrated with commercialized products. The evolution of hardware, both computer power and mixed reality display devices, will also open new possibilities of enhancing the system, making it more immersive, with smoother interaction and usability.

The system's interfaces are a target for improvement, both visually and in terms of content and interaction. Graphically, they can be improved and redesigned to enhance immersion and usability. From a technical point-of-view, there can be developed more interfaces for different contexts, actions, and use cases, making each one simpler and more user-friendly.

Another improvement is to enhance the tracking system with additional tracking techniques, such as non-optical sensors, driving towards a hybrid-tracking system to provide additional reliability.

Finally, the integration of this system with an IIoT implementation may also bring benefits with more direct interaction and reaction, and communication flows.

Bibliography

Almada Lobo, F. (2017, February 10). Why MES Needs To Change For The Future. Retrieved June 8, 2018, from https://www.mbtmag.com/article/2017/02/why-mes-needs-change-future

ARToolKit. (1999). ARToolKit Home Page. Retrieved June 8, 2018, from https://www.hitl.washington.edu/artoolkit/

ARToolKit. (2015). *artoolkit-docs: Documentation files (.md) for ARToolKit*. ApacheConf, ARToolKit. Retrieved from https://github.com/artoolkit/artoolkit-docs

ARToolKit. (2016, February 15). About the Traditional Template Square Marker. Retrieved February 8, 2018, from https://www.artoolkit.org/documentation/doku.php?id=3_Marker_Training:marker_about

ARToolKit. (2018). *artoolkit5: ARToolKit v5.x*. C++, ARToolKit. Retrieved from https://github.com/artoolkit/artoolkit5 (Original work published 2015)

AVA. (2018). ArUco: a minimal library for Augmented Reality applications based on OpenCV | Aplicaciones de la Visión Artificial. Retrieved June 8, 2018, from https://www.uco.es/investiga/grupos/ava/node/26

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, *6*(4), 355–385.

Azuma, R. T., Hoff, B. R., Neely III, H. E., Sarfaty, R., Daily, M. J., Bishop, G., … You, S. (1998). Making augmented reality work outdoors requires hybrid tracking. In *Proceedings of the First International Workshop on Augmented Reality* (Vol. 1).

Baillot, Y., Davis, L., & Rolland, J. (2001). A survey of tracking technology for virtual environments. *Fundamentals of Wearable Computers and Augumented Reality*, 67.

Bimber, O., & Raskar, R. (2005). *Spatial augmented reality: merging real and virtual worlds*. CRC press.

Brettel, M., Friederichsen, N., Keller, M., & Rosenberg, M. (2014). How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 Perspective. *International Journal of Mechanical, Industrial Science and Engineering*, *8*(1), 37–44.

Chaneski, W. (2015, December 21). Expanding the Use of Tablets in Manufacturing. Retrieved June 8, 2018, from https://www.mmsonline.com/columns/expanding-the-use-of-tablets-in-manufacturing

Chauhan, V. P., & Kayasth, M. M. (2014). Object Detection and Pose Tracking In Augmented Reality. *International Journal of Science and Research (IJSR)*, *3*(12), 2623–2629.

Critical Manufacturing. (2014, January 16). Integrating ERP, MES and Automation levels. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/newsroom/blog/posts/blog/erp-integration-67

Critical Manufacturing. (2017, November 21). Industry 4.0. From Concept to Reality: Bringing it all together. Retrieved February 25, 2018, from http://www.criticalmanufacturing.com/en/newsroom/blog/posts/blog/industry-4-0-from-concept-to-reality-bringing-it-all-together

Critical Manufacturing. (2018a). Critical Manufacturing - Legacy MES Enhancements and MES Migration Services. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/services/legacy-mes

Critical Manufacturing. (2018b). Critical Manufacturing MES. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/critical-manufacturing-mes/overview

Critical Manufacturing. (2018c). Critical Manufacturing MES | Modular Functionality for Integrated Management of Operations. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/critical-manufacturing-mes/modules-features

Critical Manufacturing. (2018d). Critical Manufacturing V6. Retrieved from http://www.criticalmanufacturing.com/uploads/resources/Critical%20Manufatcuring%20MES%20Brochure_20180502115142.pdf?v67

Critical Manufacturing. (2018e). Manufacturing Business Intelligence | Software Services for Manufacturing Environments. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/services/business-intelligence

Critical Manufacturing. (2018f). Manufacturing Technology Solution Provider | Company Overview. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/en/company/overview

Critical Manufacturing. (2018g). What is MES. Retrieved June 8, 2018, from http://www.criticalmanufacturing.com/de/critical-manufacturing-mes/what-is-manufacturing-execution-system

Cruz-Neira, C., Sandin, D. J., & DeFanti, T. A. (1993). Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (pp. 135–142). ACM.

DAQRI. (2018). DAQRI - Smart Glasses™. Retrieved June 8, 2018, from https://daqri.com/products/smart-glasses/

Eeles, P. (2004, July 1). What, no supplementary specification? Retrieved June 13, 2018, from http://www.ibm.com/developerworks/rational/library/3975.html

Etienne, J. (2018a). *AR.js: Efficient Augmented Reality for the Web*. JavaScript. Retrieved from https://github.com/jeromeetienne/AR.js (Original work published 2017)

Etienne, J. (2018b). *AR.js: with A-Frame*. JavaScript. Retrieved from https://github.com/jeromeetienne/AR.js (Original work published 2017)

Etienne, J. (2018c). *AR.js with Threejs*. JavaScript. Retrieved from https://github.com/jeromeetienne/AR.js (Original work published 2017)

Eyecare Business. (2018, May 1). 3D-Printed Lenses Now for AR Smart Glasses. Retrieved June 8, 2018, from https://www.eyecarebusiness.com/news/2018/3d-printed-lenses-now-for-ar-smart-glasses

Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 2, pp. 590–596). IEEE.

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Foundry. (2018). VR/AR/MR, what's the difference? Retrieved June 8, 2018, from /industries/virtual-reality/vr-mr-ar-confused

Furht, B. (2011). *Handbook of augmented reality*. Springer Science & Business Media.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292.

Gavrila, D. M., & Davis, L. S. (1996). 3-D model-based tracking of humans in action: a multi-view approach. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on* (pp. 73–80). IEEE.

Georgia Tech Research Corporation, A. E. L. (2018a). argon.js. Retrieved June 8, 2018, from https://www.argonjs.io/

Georgia Tech Research Corporation, A. E. L. (2018b). argon.js documentation. Retrieved June 8, 2018, from https://docs.argonjs.io/

Glass Almanac. (2018). The History of Google Glass. Retrieved June 8, 2018, from http://glassalmanac.com/history-google-glass/

Google. (2017). Poly. Retrieved February 25, 2018, from https://poly.google.com

Google. (2018a, February 23). Quickstart for AR on the Web | ARCore. Retrieved June 8, 2018, from https://developers.google.com/ar/develop/web/quickstart

Google. (2018b, March 5). WebXR Device API - Chrome Platform Status. Retrieved June 8, 2018, from https://www.chromestatus.com/feature/5680169905815552

Google. (2018c, August 5). ARCore Overview | ARCore. Retrieved June 8, 2018, from https://developers.google.com/ar/discover/

Google. (2018d, August 6). Fundamental Concepts | ARCore. Retrieved June 8, 2018, from https://developers.google.com/ar/discover/concepts

Gorecky, D., Schmitt, M., Loskyll, M., & Zühlke, D. (2014). Human-machine-interaction in the industry 4.0 era. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on* (pp. 289–294). IEEE.

Gosalia, A. (2018a, February 23). Announcing ARCore 1.0 and new updates to Google Lens. Retrieved June 8, 2018, from https://www.blog.google/products/google-vr/announcing-arcore-10-and-new-updates-google-lens/

Gosalia, A. (2018b, May 8). Experience augmented reality together with new updates to ARCore. Retrieved June 8, 2018, from https://www.blog.google/products/google-vr/experience-augmented-reality-together-new-updates-arcore/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Heidari, F., & Loucopoulos, P. (2014). Quality evaluation framework (QEF): Modeling and evaluating quality of business processes. *International Journal of Accounting Information Systems*, *15*(3), 193–223.

Hirzer, M. (2008). Marker detection for augmented reality applications. In *Seminar/Project Image Analysis Graz* (p. 25).

Immersive Web Community Group. (2018, July 6). WebXR Device API. Retrieved June 8, 2018, from https://immersive-web.github.io/webxr/

Intel. (2018). Virtual Reality Vs. Augmented Reality Vs. Mixed Reality. Retrieved February 16, 2018, from https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html

Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion. Retrieved from http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

Kato, H., & Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on* (pp. 85–94). IEEE.

Khronos Group. (2011, July 19). WebGL - OpenGL ES for the Web. Retrieved February 25, 2018, from https://www.khronos.org/webgl/

Kipper, G., & Rampolla, J. (2012). *Augmented Reality: an emerging technologies guide to AR*. Elsevier.

Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'Amore, R., … Johnson, A. (2001). Providing clarity and a common language to the "fuzzy front end." *Research-Technology Management*, *44*(2), 46–55.

Kruchten, P. (1995). Architectural blueprints—The "4+ 1" view model of software architecture. *Tutorial Proceedings of Tri-Ada*, *95*, 540–555.

LaForge. (2018). Shima. Retrieved June 8, 2018, from https://laforgeoptical.com/pages/meet-shima

Lardinois, F. (2015, May 13). DAQRI Acquires AR Pioneer ARToolworks. Retrieved June 8, 2018, from http://social.techcrunch.com/2015/05/13/daqri-acquires-ar-pioneer-artoolworks/

Lindgreen, A., & Wynstra, F. (2005). Value in business markets: What do we know? Where are we going? *Industrial Marketing Management*, *34*(7), 732–748.

Longo, F., Nicoletti, L., & Padovano, A. (2017). Smart operators in industry 4.0: A human-centered approach to enhance operators' capabilities and competencies within the new smart factory context. *Computers & Industrial Engineering*, *113*, 144–159.

McCabe, B. (2016, November 9). Industry 4.0 and Manufacturing Processes. Retrieved June 8, 2018, from https://www.semiwiki.com/forum/content/6341-industry-4-0-manufacturing-processes.html

MDN. (2015, February 12). Emscripten. Retrieved February 7, 2018, from https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Emscripten

Medley, J. (2018, May 6). Welcome to the immersive web | Web. Retrieved June 8, 2018, from https://developers.google.com/web/updates/2018/05/welcome-to-immersive

Mellado, J. (2018). *js-aruco: JavaScript library for Augmented Reality applications*. JavaScript. Retrieved from https://github.com/jcmellado/js-aruco (Original work published 2015)

Meta. (2018). Meta 2. Retrieved June 8, 2018, from https://meta-eu.myshopify.com/

Michalos, G., Karagiannis, P., Makris, S., Tokçalar, Ö., & Chryssolouris, G. (2016). Augmented reality (AR) applications for supporting human-robot interactive cooperation. *Procedia CIRP*, *41*, 370–375.

Microsoft. (2018a). Microsoft HoloLens. Retrieved June 8, 2018, from https://www.microsoft.com/en-ca/hololens

Microsoft. (2018b, March 21). What is mixed reality? Retrieved June 8, 2018, from https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality

Miles, L. D. (2015). *Techniques of value analysis and engineering*. Miles Value Foundation. Retrieved from https://books.google.pt/books?id=yCf0CQAAQBAJ&printsec=frontcover&hl=pt-PT#v=onepage&q&f=false

Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, *77*(12), 1321–1329.

Mozilla. (2018a). A-Frame - Introduction. Retrieved February 25, 2018, from https://aframe.io

Mozilla. (2018b, January 27). The WebGL API: 2D and 3D graphics for the web. Retrieved February 25, 2018, from https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

Mozilla. (2018c, March 4). WebRTC API. Retrieved June 20, 2018, from https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

Mr.doob. (2018). *three.js: JavaScript 3D library*. JavaScript. Retrieved from https://github.com/mrdoob/three.js (Original work published 2010)

Neumann, U., & You, S. (1999). Natural feature tracking for augmented reality. *IEEE Transactions on Multimedia*, *1*(1), 53–64.

Nicola, S., Ferreira, E. P., & Ferreira, J. P. (2012). A novel framework for modeling value for the customer, an essay on negotiation. *International Journal of Information Technology & Decision Making*, *11*(03), 661–703.

ODG. (2018). ODG - R-9 Smartglasses. Retrieved June 8, 2018, from https://www.osterhoutgroup.com/r-9-smartglasses

OpenCV. (2018). Detection of ArUco Markers. Retrieved from https://docs.opencv.org/3.4.0/d5/dae/tutorial_aruco_detection.html

Osterwalder, A. (2004). The business model ontology: A proposition in a design science approach.

Parisi, T. (2012). *WebGL: up and running*. O'Reilly Media, Inc.

Perdue, T. (2017, July 7). Applications of Augmented Reality. Retrieved June 8, 2018, from https://www.lifewire.com/applications-of-augmented-reality-2495561

Regenbrecht, H., Baratoff, G., & Wilke, W. (2005). Augmented reality projects in the automotive and aerospace industries. *IEEE Computer Graphics and Applications*, *25*(6), 48–56.

Reitmayr, G., & Drummond, T. (2006). Going out: robust model-based tracking for outdoor augmented reality. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 109–118). IEEE Computer Society.

Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., & Harnisch, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consulting Group*, *9*.

Schweizer, H. (2014). Smart glasses: technology and applications. *Student Report*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*.

Statista. (2018a). Smartphone users worldwide 2014-2020 | Statistic. Retrieved June 8, 2018, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Statista. (2018b). Tablet users worldwide 2013-2020 | Statistic. Retrieved June 8, 2018, from https://www.statista.com/statistics/377977/tablet-users-worldwide-forecast/

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, *42*(4), 73–93.

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I* (pp. 757–764). ACM.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Thompson, D. (2015, October 8). Smartphones and tablets in manufacturing | Control Engineering. Retrieved June 8, 2018, from https://www.controleng.com/single-article/smartphones-and-tablets-in-manufacturing/a3a8216401f94def2f4bf7db3d159551.html

Trimble. (2018). Trimble Mixed Reality. Retrieved June 8, 2018, from http://www.tnzmarcom.com/mixedreality/

Ulaga, W., & Eggert, A. (2006). Relationship value and relationship quality: Broadening the nomological network of business-to-business relationships. *European Journal of Marketing*, *40*(3/4), 311–327.

Van Krevelen, D. W. F., & Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, *9*(2), 1.

Vaughan, B., & Lamb, P. (2018). artoolkitX. Retrieved June 8, 2018, from http://www.artoolkitx.org/

Virtual Reality Society. (2017, May 5). Applications Of Virtual Reality. Retrieved June 8, 2018, from https://www.vrs.org.uk/virtual-reality-applications/

Vuforia. (2018). Vuforia | Augmented Reality. Retrieved February 9, 2018, from https://www.vuforia.com/

Warner Bros Entertainment, & Google. (2018). O Hobbit: A Batalha dos Cinco Exércitos. Retrieved February 25, 2018, from http://middle-earth.thehobbit.com/

Warwick Manufacturing Group. (2007). Quality Function Deployment. In *Product Excellence Using Six Sigma*. School of Engineering, University of Warwick, Conventry, UK: Warwick Manufacturing Group. Retrieved from https://warwick.ac.uk/fac/sci/wmg/ftmsc/modules/modulelist/peuss/slides/section_6a_qfd_notes.pdf

Weill, P., & Vitale, M. (2001). *Place to space: Migrating to eBusiness Models*. Harvard Business Press.

Woodall, T. (2003). Conceptualising'value for the customer': An attributional, structural and dispositional analysis. *Academy of Marketing Science Review*, *2003*, 1.

Yew, A. W. W., Ong, S. K., & Nee, A. Y. C. (2016). Towards a griddable distributed manufacturing system with augmented reality interfaces. *Robotics and Computer-Integrated Manufacturing*, *39*, 43–55.

Zhong, R. Y., Dai, Q., Qu, T., Hu, G., & Huang, G. Q. (2013). RFID-enabled real-time manufacturing execution system for mass-customization production. *Robotics and Computer-Integrated Manufacturing*, *29*(2), 283–292.

Zhou, F., Duh, H. B.-L., & Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (pp. 193–202). IEEE Computer Society.

Appendix A - Use Case Description

Mixed Reality for Manufacturing Execution Systems – Use Case

Mixed Reality for Manufacturing Execution Systems Internship

05 January 2018

1 Mixed Reality for MES general information

1.1 Introduction

This document provides a general overview over the use case for which the *Mixed Reality for Manufacturing Execution Systems* internship should aim to address. It also details some general constraints and non-functional requirements that the internship solution should take into account.

1.2 Use case

The main purpose of the internship application is to serve as a proof of concept of the introduction of mixed reality concept in the context of a *Manufacturing Execution System*, as well as serve as an additional tool in demonstrations and live showcases of Critical Manufacturing MES product. As such, the following use case describes the usage of the internship solution, extending the currently existing demo scenario where a coffee machine is connected with the MES, and the user can both control its behavior and visualize relevant information. However, the use case is not restrictive to the coffee machine scenario, as it can be applied to a scenario based on the interaction with one of more *resources*. The scenario also assumes the usage of a tablet-like device to use the application although the hardware device usage can be broadened to an equivalent device that allows image recognition and display of information. When referring to the *application*, we are referring to the mixed reality solution developed in the internship.

The main actor that will interact using the application is not meant to be a typical shop floor *operator* but rather an engineer who desires to view more specific information of a resource which might not be immediately available to the *operator* on his/her daily usage. For this kind of users, the information that is relevant and should be displayed by the application should be related the to the *Maintenance Management* of the *resource*, relevant *Key Performance Indicators (KPIs)* and information regarding the current *Employee* that is *checked-in* (working) on the *resource*.

In the demonstration using a coffee machine, a standard use case can be described as follows:

The actor has the device configured with the application, with the mixed reality application running. Near him, he can find a coffee machine that is configured and connected to the MES. This machine can be visually tagged with an indicator to ease the image recognition.

The actor points the device to the coffee machine, which recognizes the devices and shows the current relevant KPIs, such as the state of the capsule loading port, the current water level and capsule count, and the current coffee processing state (standby). This information can be represented through a mix of text, colors, and graphics. He can check-in into the machine (using the existing web interface) and see that he is the current employee responsible for the machine. The actor can then open and close the capsule port, seeing in the interface the change of state with a minimum delay. He should then put a capsule in the machine and close the port. Although not critically relevant for the internship, the actor can proceed to order a small, medium, or large coffee through the mixed reality interface, or alternatively, he can press one of the physical buttons to do so. As the coffee is served, the interface is again updated, and the water level can be see decreasing, and approaching minimum level. As the coffee is served, the actor can see that the resource needs maintenance, or request a new maintenance action, using the interface. He can then see that the interface signals the new need for maintenance, along with relevant information for the maintenance required. He should be able to complete the maintenance, filling the water level, and signal the completion either thought the standard web interface or thought the application.

1.1 Non-Functional Requirements

The mixed reality solution should take into account some non-functional requirements to reach the standards for usage in demos and to serve as a proof of concept for a future MES integration.

In terms of usage, it should try to follow the design guidelines of the MES to match the currently existing HTML5 interface. It should also be robust and reliable to unexpected scenarios, considering the possibility of unstable connections in a live demonstration environment. It should try to provide the user with a smooth interaction experience, considering the instability of the person holding the device and the possibility that the actor using it might be engaged in active conversation and pointing the camera elsewhere. It should recognize the device as fast as possible and minimize interface flickering with recognition errors. It should also strive to avoid false positive identifications, possibility considering location.

It is important to focus that there should be a moderate degree of extensibility of the recognized *resource* (i.e. carrying other tags), and the user interface, which could be configurable to display different information according to the context. This extensibility and configurability can be achieved by integrating with the existing MES system, and the existing user interface and dashboard configuration module.

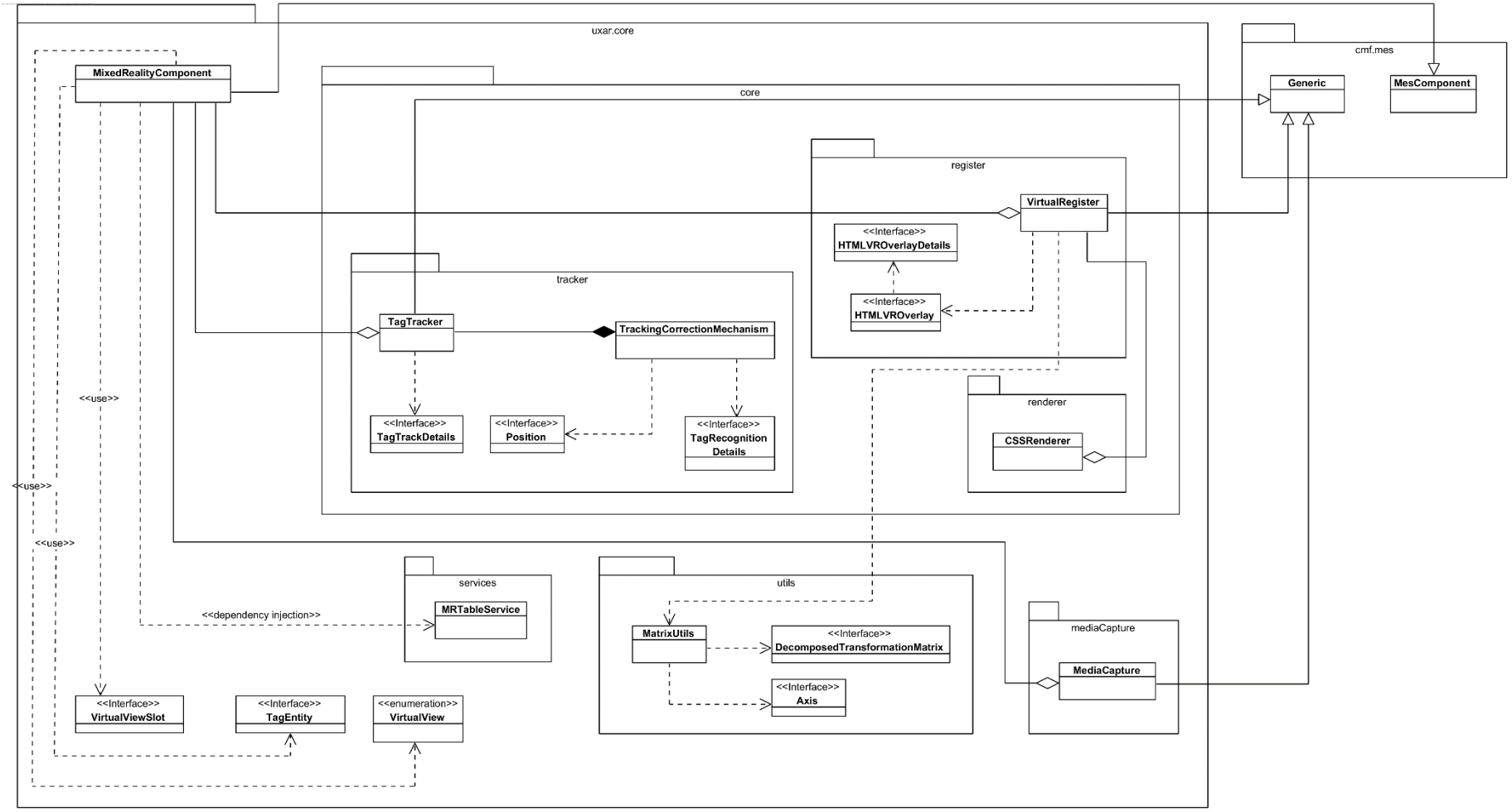
1.2 General Constraints

The application developed ought to prioritize a web approach for the image recognition module, if possible, in order to ease a future integration in the MES product. This would also bring advantages in integrating with the current interface design, modules, and visual components. The current interface is developed in *TypeScript* over an *Angular 4* framework, which should be compatible with any web technologies that work on a plain web environment.

The application should run on *Google Chrome*.

The developed application should follow the company code guidelines and be reasonably documented to provide information for future improvements.

Appendix B – Mixed Reality Core Functionalities Class Diagram



Appendix C – QEF Model

q	D	Q_i	Dimension	Q_j	W_{ij} (Factor Weight *j* in Dim *i)* [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*) [0,100]
0%	1.73	0	Functionality	0	0.67	Functional Requirements	10	FFR01 - Recognize an entity	0
							10	FFR02 - Present information about a recognized entity	0
				0	0.33	Usability	8	FU01 - Smooth user interaction	0
		0	Adaptability	0	0.50	Supportability	6	AS01 - Moderate degree of extensibility	0
							6	AS02 - Moderate degree of configurability	0
				0	0.50	Compatibility	6	AC01 - Application runs in a tablet-like device	0
							10	AC02 - Application must run on Google Chrome	0
		0	Reliability	0	0.25	Resilience	8	RRS01 - Reliable in unexpected scenarios	0
				0	0.25	Performance	8	RP01 - Recognize the entity as fast as possible	0
				0	0.50	Reliability	8	RRL01 - Minimize interface flickering with recognition errors	0
							10	RRL02 - Minimize false positive identifications	0

Dimension	Functionality			
Factor	Functional Requirements, User Interaction			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FR01 - Recognize an entity	The application recognizes an entity when it is on sight by the device's camera	No access to functionality	-	Full access to the functionality
FR02 - Present information about a recognized entity	The application presents information about na entity after it is recognized	No access to functionality	-	Full access to the functionality
FUI01 - Smooth user interaction	The application has a smooth UI experience (interaction and motion stability)	No instability correction and weak interaction	Smooth interaction	Smooth interaction and instability correction

Dimension	Adaptability			
Factor	Versatility, Compatibility			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
AS01 - Moderate degree of extensibility	Application can be extended to provide additional functionality	No	-	Yes
AS02 - Moderate degree of configurability	Application may be configurable to display different information and to edit application data	No	Only configurable to display other information or to edit application data	Configurable display information and application data
AC01 - Application runs in a tablet-like device	Application runs in a tablet or a similar device	No	-	Yes
AC02 - Application must run on Google Chrome	Application fully runs in Google Chrome	No	-	Yes

Dimension	Reliability			
Factor	Resilience, Performance, Reliability			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
RRS01 - Reliable in unexpected scenarios	Application is reliable under internet connection problems	No	-	Yes
RP01 - Recognize the resource as fast as possible	Recognition time as low as possible	No	-	Yes
RRL01 - Minimize interface flickering with recognition errors	Interface flickering is addressed under recognition errors	No	-	Yes
RRL02 - Minimize false positive identifications	Application has a mechanism to reduce false positive identifications	No	-	Yes