UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

Technologies and concepts for enhancing interaction and enabling collaboration in Augmented and Mixed Reality

Tesis Doctoral

DIEGO VAQUERO MELCHOR
Máster universitario en Sistemas Inteligentes

2020

Departamento de Señales, Sistemas y Radiocomunicaciones

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE MADRID

**Technologies and concepts for
enhancing interaction and
enabling collaboration in
Augmented and Mixed Reality**

Tesis Doctoral

Autor:

Diego Vaquero Melchor

Máster universitario en Sistemas Inteligentes

Directora:

Ana María Bernardos Barbolla

Doctora Ingeniera de Telecomunicación

2020

DEPARTMENT:

Departamento de Señales, Sistemas y Radiocomunicaciones

Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad Politécnica de Madrid

PH.D. THESIS:

Technologies and concepts for enhancing interaction and enabling collaboration in Augmented and Mixed Reality

Author:

Diego Vaquero Melchor

Advisor:

Ana María Bernardos Barbolla

Committee named by the Rector of Universidad Politécnica de Madrid, on the ____ of _____ 2020.

President: _____

Member: _____

Member: _____

Member: _____

Secretary: _____

After the defence of the Ph.D. Thesis on the ____ of _____ 2020, at the Escuela Técnica Superior de Ingenieros de Telecomunicación, the committee agrees to grant the following qualification:

President

Secretary

Members

*“It’s not worth doing something unless you were doing something
that someone, somewhere, would much rather you weren’t doing”*

Sir Terence David John Pratchett

“Spoilers! Goodbye, sweetie.”

Dr. River Song

Acknowledgements

To my parents, for always being there

To Carlos, for the encouragement
during all this time

To all those fantasy authors who have
made the real world more bearable

To my past me, for the tenacity

Abstract

La Realidad Aumentada (AR) y la Realidad Mixta (MR) son tecnologías emergentes que permiten inyectar contenido digital en el mundo real. Aunque ambas han hecho avances impresionantes en los últimos años, todavía quedan algunas aristas que pulir. La primera de ellas está relacionada con la experiencia de usuario en los sistemas AR mientras que la segunda se centra en la propia interacción con los elementos digitales AR. En esta misma línea de pensamiento, esta tesis analiza de forma comparativa la Experiencia de Usuario entre la Realidad Aumentada *wearable* y las plataformas tradicionales. En este sentido, los resultados muestran que los usuarios se sienten generalmente más atraídos hacia la AR, debido en parte a que se trata de una tecnología más novedosa y en parte a su gran potencial para la visualización de contenido 3D. Además, se propone el uso de feedback háptico (en particular mid-air) para facilitar la interacción con elementos AR. Mediante un estudio con usuarios se ha demostrado que este tipo de feedback es útil al mejorar la precisión en tareas que así lo requerían. Por otro lado, teniendo en cuenta el auge de las aplicaciones multi usuario, esta tesis las enfoca desde el punto de vista de la AR y MR. Con ese objetivo en mente se propone un análisis teórico de la colaboración en esos sistemas. Aparte, se propone una taxonomía para clasificar sistemas colaborativos AR que también sirve como referencia de cara a futuros desarrollos. Basado en ese análisis se presenta SARA, una arquitectura cuyo objetivo es facilitar el desarrollo y despliegue de experiencias colaborativas AR. SARA permite compartir elementos AR entre varios dispositivos (e.g. smartphones o tablets). SARA también se encarga de manejar la interacción con el contenido digital, permitiendo que ésta sea adaptada al dispositivo usado en cada caso. Además de una serie de artefactos listos para ser integrados en desarrollos futuros, SARA introduce el concepto de *Modelos de Colaboración*, los cuales orquestan las capacidades de visualización e interacción de los usuarios, pudiendo ser sustituidos en medio de la colaboración. Se propone un conjunto de cinco modelos, cada uno con un nivel de abstracción diferente (basados en turnos, en capas, en jerarquía y en la propiedad). Sin embargo, otros modelos pueden ser fácilmente integrados en la arquitectura. Dichos modelos pueden ser reutilizados entre distintas instancias de la colaboración, lo que ayuda a reducir el tiempo de desarrollo de nuevas experiencias AR.

Abstract

Augmented Reality (AR) and Mixed Reality (MR) are emerging technologies that allow displaying digital content superimposed on the real world. Although both technologies have made impressive strides in recent years, there are still some sharp edges to polish. The first of them is related to the User Experience in AR systems and the second one with interaction with digital AR assets.

Following this line, this thesis focuses comparatively on the User Experience of wearable Augmented Reality with respect to conventional platforms. In this sense, results show that users are generally more attracted towards AR partly due to its novelty and partly due to its high power in terms of 3D visualization. In addition, the use of haptic feedback (in particular mid-air) is proposed to improve interaction with AR digital assets. This type of feedback has been analyzed in a user study and has proved useful in promoting precision in tasks that required it.

Taking into account the rise of collaborative multi-user applications, this thesis approaches them from the point of view of Augmented and Mixed Reality. To do so, a theoretical analysis of collaboration in this type of systems is first presented. Besides that, a taxonomy is presented, whose intention is classifying AR collaborative systems as well as serving as a consultation for future developments.

Based on that analysis, the SARA architecture is presented, whose objective is to facilitate the development and deployment of AR collaborative experiences. SARA allows AR content to be shared between various devices that can be of different types (e.g. smartphones or Head Mounted Displays). SARA also manages the interaction with that digital content and allows it to be tailored to the device in use. In addition to a series of artifacts ready to be incorporated in future developments, SARA introduces the concept of *Collaboration Models*. These models allow to orchestrate the visualization and interaction capabilities of the participants and can be changed on the fly. Five models of different abstraction levels are initially proposed (*Unconstrained*, *Turn-based*, *Layer-based*, *Hierarchy-based*, *Ownership-based*), but others could be easily integrated into the architecture. Furthermore, these models can be reused between different instances of the collaboration, which also saves time in the development of new AR experiences.

Contents

Acknowledgements	V
Abstract	VI
Abstract	VII
List of Figures	XIII
List of Tables	XVIII
1. INTRODUCTION	1
1.1. Reading guide and brief summary of the content of the chapters . . .	3
1.2. Methodological formalism of the present thesis	5
1.3. Expected contributions of this thesis	5
2. STATE OF THE ART	9
2.1. Augmented and Mixed Reality, definition, current status and future projection	10
2.2. Augmented and Mixed Reality Devices	13
2.3. Collaboration in Augmented and Mixed Reality Context	18
2.4. Improving the interaction with Augmented Reality content	23
2.5. Frameworks and architectures for Augmented and Mixed Reality . . .	27
2.6. Comments on the challenges identified in the literature	29
3. ASSESSING AND COMPARING USER EXPERIENCE THROUGH WEARABLE AUGMENTED REALITY	31
3.1. Use Case: Drone Flight Missions. General view and main components	33
3.2. Web tool for the mission creation	36
3.2.1. Main interface	36
3.2.2. Creation and Manipulation of an Inspection using the Web User Interface	38

3.3.	Holo-MIS: A wearable-MR based tool for the drone mission definition	42
3.3.1.	Main interface	42
3.3.2.	Creation and manipulation of the <i>inspection</i> using the Holo-MIS implementation	43
3.3.3.	Integration of Digital Surface Model data for 3D ground content rendering	45
3.4.	Validation with users	46
3.4.1.	User Study Objectives	46
3.4.2.	Participants and Apparatus	46
3.4.3.	Procedure, Tasks and Experimental Design	46
3.5.	Experimental results	49
3.5.1.	Performance and errors	50
3.5.2.	User Experience and Fatigue during the validation	57
3.5.3.	Comments on technology preference and strengths and weaknesses of both systems	64
3.6.	Summary	67
4.	AUGMENTED REALITY BASED COLLABORATIVE EXPERIENCES: ANALYSIS AND TAXONOMY	69
4.1.	Context and Motivation	69
4.2.	Analysis of the Collaboration in AR and MR Context	70
4.2.1.	Collaboration definition	70
4.2.2.	Collaboration characteristics	70
4.2.3.	Collaboration layers	72
4.3.	Presenting the Taxonomy	75
4.3.1.	Limit number of participants	76
4.3.2.	Location of the participants	76
4.3.3.	Collaboration Model	78
4.3.4.	AR content visualization platform	78
4.3.5.	Interaction method	79
4.3.6.	Method for aligning the AR content	80
4.3.7.	Integration of real-world information	82
4.4.	Applying the Taxonomy to the State-Of-The-Art Systems	83
4.5.	Summary	86
5.	SARA: AN ARCHITECTURE FOR SHARED-AUGMENTED REALITY EXPERIENCES AND APPLICATIONS	87
5.1.	Context and motivation	87
5.2.	SARA features	88
5.3.	SARA. Presenting the Components of the Architecture	91
5.3.1.	The Communication Service (CS)	92

5.3.2.	The SARA Client and Content Generators	93
5.3.3.	The Collaboration Manager Services	96
5.4.	Data model	97
5.4.1.	Sessions	97
5.4.2.	Events	100
5.4.3.	Collaboration Models	103
5.4.4.	Orchestrating the communication between system components	109
5.4.5.	Incremental updates and Interaction management	110
5.5.	Practical issues on implementation	114
5.5.1.	Detecting changes in nodes properties	114
5.5.2.	Adding interaction to the AR elements	114
5.6.	Creating and deploying applications over SARA: an use case approach	118
5.6.1.	A first Approach. SmARt city: A data source viewer based on hierarchical roles	118
5.6.2.	Detailing the creation process. A voxel-based cooperative game prototype	119
5.7.	Summary	125
6.	ENHANCING INTERACTION WITH AUGMENTED REALITY THROUGH MID-AIR HAPTIC FEEDBACK	131
6.1.	Context and Motivation	131
6.2.	Harptics: haptic representation of an Augmented Reality element . .	132
6.3.	HARP: An architecture for Haptically-Enhanced AR Experiences . .	136
6.3.1.	Data model, SARA integration and architecture overview . . .	136
6.3.2.	Practical Issues on Haptic Representation	138
6.4.	Developing over HARP, overview of the process and use cases . . .	142
6.4.1.	Designing and implementing the Haptic Inspector prototype .	144
6.4.2.	Notes on the implementation of the Simon prototype	146
6.5.	Validation With Users	148
6.5.1.	User Study Objectives	148
6.5.2.	Participants and Apparatus	148
6.5.3.	Procedure, Tasks and Experimental Design	149
6.6.	Experimental Results	154
6.6.1.	Performance and errors	154
6.6.2.	User experience and Fatigue during the validation	165
6.7.	Summary	168
7.	CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS	171
A.	Representation of Inspections On a Map and its related calcula- tions	175

B. Ground microservice: Height map generation using Digital Surface Model (DSM) information	185
C. Detecting changes in nodes of the SARA architecture	197
References	207

List of Figures

1.1. An overview of the methodological formalism based on use-cases followed in this thesis	6
2.1. Milgram and Kishino's Virtuality Continuum (VC)	11
2.2. Magicverse layers. This Figure has been extracted from the Magic Leap webpage	12
2.3. Example of Augmented Reality devices according to their position: (a) Head-worn; (b) Hand-held; (c) Spatial AR	15
2.4. Studierstube AR project. Two users collaborate while sharing a set of 3D elements	19
2.5. Example of the Nilsson's proposal	20
2.6. Integration of digital cranes in the real world, in such a way that all participants can see the elements controlled by the others	21
2.7. Manipulating the miniaturized avatar of a remote user as proposed by Piumsomboon	22
2.8. Schematic vision of the system proposed by Rhee et. al., in which VR and AR users can collaborate.	22
2.9. OpenXR before (a) and after (b)	28
3.1. A screenshot of the DJI Ground Station Pro iPad application	32
3.2. Components of a drone mission: a set of <i>inspections</i> , a take-off point, a landing point and the estimated trajectory to perform the flight (as the orange line)	34
3.3. Data model of the Mission Definition System (MDS)	35
3.4. Overview of the MDS architecture	36
3.5. Screenshot of the web interface	37
3.6. Example of a panel used for adding measurement processes associated with an <i>inspection</i>	38
3.7. Appearance of the drop-down used to select the Inspection to be added in the web tool	39

3.8. Parameters of a Cone Inspection (right) and their representation on the 3D component of the interface as anchors (blue spheres)	40
3.9. Manipulation of the basic features of a Cone Inspection using the three-axis manipulator in the web version	41
3.10. 3D terrain view	43
3.11. Inspections palette as presented to the users.	44
3.12. Manipulating a <i>cone inspection</i> using the Holo-MIS AR version . . .	44
3.13. Representation of the same mission with and without the integration of Digital Surface Model (DSM) data	45
3.14. The map with the annotated positions as shown to the participants. .	47
3.15. Stages of the user test	50
3.16. Parallel coordinates representation of the times required by the users to complete each task. Time is measured in seconds.	51
3.17. Comparison of the specifications given to users (blue marks) with respect to the average data obtained for each implementation: web (red marks) and AR (green marks).	54
3.18. Comparison of the specifications given to users (blue marks) with respect to the average data obtained for each implementation: web (red marks) and AR (green marks) in 3D.	55
3.19. Comparison of the mean value per item in both implementations . . .	61
3.20. Mean values and deviation for each UEQ category in the Web implementation	62
3.21. Mean values and deviation for each UEQ category in the Mixed Reality-based implementation	62
3.22. Average and standard deviation values for each category of the fatigue test. Values range from 1 (if the users experienced strong problems in that category) to 7 (where they had no problem at all)	63
4.1. Collaboration Layers	73
4.2. Visual representation of local, remote and mixed locations	76
5.1. Overview of the microservices in which SARA has been divided . . .	91
5.2. Communication Service details	92
5.3. SARA Client and Content Generators in detail	94
5.4. Collaboration management services in detail	96
5.5. Class diagram of the data model associated to a Session.	98
5.6. Blender node editor where various types of material and their properties can be seen. Material called <i>Principled BSDF</i> is an advanced version of the material representation used in SARA.	99
5.7. Class diagram of the data model relative to Events	100
5.8. Class diagram of the data model relative to the Collaboration Models	104

5.9.	Combination of multiple collaboration models in the SmARt city prototype	108
5.10.	Sequence diagram of the overview of the collaboration process	110
5.11.	Sequence diagram of the process of applying incremental updates without the Collaboration Manager Service	111
5.12.	Introducing the <i>Collaboration Model Manager Service</i> into the process of incremental updates	112
5.13.	Sequence diagram of the process of propagating an Interaction Event by facing it to the Collaboration Model	113
5.14.	Screenshots of the SmARt city prototype	120
5.15.	Overview of the five steps in which the process of developing with SARA is divided	122
5.16.	Screenshot of the Unity implementation	123
5.17.	Capture of the voxel-based prototype taken from the iOS client . . .	124
5.18.	Capture of the voxel-based prototype taken from the HoloLens client	124
5.19.	Overlay of a collaborative session between three clients: a Unity client on PC and two iOS clients	125
5.20.	Sequence diagram that illustrates the voxel-based world prototype . .	128
5.21.	Comparison of the development of collaborative applications AR with and without SARA	129
6.1.	Illustrating the halo problem when representing haptic points with mid-air technology	133
6.2.	High-fidelity and feature-based haptic representations of a model: (a) High-fidelity representation; (b) feature-based representation.	135
6.3.	Minimum combination of a SARA client with haptic support and the <i>Communication Service</i>	136
6.4.	Combination of three Haptic Clients to perform a haptically enhanced AR collaboration. One participant would use the combination of the HoloLens and Haptic Clients while the other participant would use the iOS one, with no haptic feedback	137
6.5.	The Ultrahaptics device with three stickers on three of its corners, which are used for helping the user to set the coordinate system manually	139
6.6.	Overview of the process of generating the haptic representation of a 3D model	140
6.7.	Creating the <i>edge-based</i> haptic representation of a cube, visual representation of the matrix	142
6.8.	Overview of the process of creating a HARP based application	143
6.9.	Overview of creation process of the <i>Haptic Inspector</i> prototype	144

6.10. List of the 3D figures used within the <i>Haptic Inspector</i> prototype as presented to the users.	150
6.11. Two participants trying to guess a figure from haptic feedback only in Task 2	151
6.12. Overview of the tasks of the HARP user validation	154
6.13. Box plot of the interpretability values given for Task 1 to each figure	156
6.14. Times used by the participants to <i>inspect</i> each shape during Task 1 .	156
6.15. Times used by the participants to <i>inspect</i> identify the 3D figures during Task 2	157
6.16. Heat map of the relation between the 3D figures provided to the users (vertical axis) and their guesses (horizontal axis)	158
6.17. Relationship between the perception of users about the difficulty of the figures and their result when guessing. The result of the divination of each figure is represented for each user (Task 2). Each attempt generates a rectangle divided into two triangles. The upper triangle shows if the user was right (green) or failed (red) in their attempt to guess. In the lower triangle appears the user's perception about the difficulty of haptically identifying the figure (Task 1), especially difficult (black color) or easy (blue color)	160
6.18. Results for Task 4 and Task 5 using the <i>Simon</i> prototype by user . .	163
6.19. Mean values and deviation for each UEQ category for the <i>Haptic Inspector</i>	168
6.20. Mean values and deviation for each UEQ category for the <i>Simon</i> prototype	168
A.1. Class diagram of the Inspection datamodel	176
A.2. Locating the corners of the map	182
B.1. Height map image tile, with the heights encoded from black to white	186
B.2. RGB image, with the color information extracted from the LAS files .	186
B.3. Visualization of the content of a LAS file, as a colored point cloud . .	188
B.4. General view of the process of generating the information matrix from LAS files.	189
B.5. Example of the surface covered by the Ground Microservice. Darker tiles represent those portions of lands for which no data has been integrated	190
B.6. An example of how the generated map represents the minimum area that encompasses all the requested coordinates (represented with the yellow dots)	191

B.7. Different scenarios that can occur when generating a height map. This distinction is made based on the number of tiles through which the requested surface extends	191
B.8. Example of a <i>Monotile</i> scenario, in which all the corner coordinates belong to the same tile	192
B.9. Different types of tiles that have to be considered in the <i>multitile</i> case	193
B.10. Relation between the min and max coordinates of the tiles and the corner coordinates for the <i>edge</i> type	194
B.11. An example of a generated 3D terrain that includes color information	195

List of Tables

2.1.	An overview of the main Head-Mounted Displays for Augmented and Mixed Reality. Part 1	16
2.2.	An overview of the main Head-Mounted Displays for Augmented and Mixed Reality. Part 1	17
3.1.	Times measured in seconds for each Task and user grouped by the implementation	50
3.2.	Difference between the sum of Tasks 1 to 3 times (creation of the mission in a guided way, step by step) with respect to Task 4 times (complete creation of a mission in an unguided way), measured in seconds.	53
3.3.	Error measured in meters for the WEB implementation that compares the positions asked to the participants and the introduced one. For each user and parameter, the average position between the individual tasks (T1, T2 and T3) and the global task (T4) has been calculated.	55
3.4.	Error measured in meters for the MR implementation that compares the positions asked to the participants and the introduced one. For each user and parameter, the average position between the individual tasks (T1, T2 and T3) and the global task (T4) has been calculated.	56
3.5.	UEQ scale means per participant and implementation. Values are in range [-3,3]. Attractiveness (Attr), Perspicuity (Pers), Efficiency (Effi), Dependency (Depe), Stimulation (Stim), Novelty (Nove)	58
3.6.	T-test results for the UEQ parameters	58
3.7.	Mean and standard deviation values for each pair of adjectives and their related scale.	60
4.1.	Applying the taxonomy to the state-of-the-art literature. Part 1 . . .	84
4.2.	Applying the taxonomy to the state-of-the-art literature. Part 2 . . .	85
5.1.	Use cases of the collaboration models and it suitability for the voxel-based game and the SmARt city prototypes	107

5.2.	Features of the SmARt city prototype	119
5.3.	Features of the voxel-world game prototype	121
6.1.	Offset in centimeters between the required size and the one entered by the user. The order of the figures is random.	162
6.2.	UEQ Scale means per participant, as collected for the Haptic Inspec- tor prototype. Values are constrained in range [-3, 3]. Attractive- ness (Attr), Perspicuity (Pers), Efficiency (Effi), Dependency (Depe), Stimulation (Stim), Novelty (Nove)	166
6.3.	Mean and standard deviation values for each pair of adjectives and their related scale as observed by the UEQ for each of the two ana- lyzed systems: the <i>Haptic Inspector</i> and the <i>Simon</i> prototype.	167

Chapter 1

INTRODUCTION

Help me, Obi-Wan Kenobi. You're
my only hope

Princess Leia Organa of Alderaan
Star Wars, a new Hope

With the famous phrase leading the top of this page, the film industry brought the world of Augmented Reality (AR) to movie theaters and spectators in 1977. In the *Star Wars* film, a small robot reproduces a “hologram”, a small 3D video fragment in which one of the protagonists (*Princess Leia*) asks for help from an old hermit. Thirty one years later, in the 2008 *Iron Man* movie, thanks to *Tony Stark* we were able to see advanced user interfaces, also based on holograms, with which Tony could interact using his hands in a natural way. That same year, the *Wall-e* movie invited us to dream with a future in which humanity left the planet after putting it on the brink of collapse and engaged in a space retreat (also spoiling the diet, but that is another subject). Thus, the spaceship offered ads to the space travelers in the form of holograms while they could make video calls using an AR interface. In 2009, the premiere of the movie *Avatar* presented Augmented Reality and holograms as sources of information in real time, showing the movement of blue creatures on the planet surface. More recently, the 2017 *Ghost In the Shell* film presented a dystopian society in which the protagonist city was flooded with huge 3D AR ads. Thus, drop by drop, today's society has been assimilating the integration of Augmented Reality into its day to day. Although we still have a little way to reproduce these corporeal holograms without additional hardware, every day we have more capabilities at our fingertips.

In a nutshell, Augmented Reality is a technology that consists of introducing digital content into the physical world. In the mentioned films, this generally involves the use of *Holograms*, which are three-dimensional images formed by the

interference of light beams from a laser or other coherent light source. Nowadays, a physical device is required to see the digital content. Some examples of popular AR devices are Head Mounted Displays (HMDs) and mobile phones. Thus, the digital information is generally over-imposed on a real-world capture. So, for now, these almost corporeal holograms projected in the middle of the air are a bit out of our reach.

The limitations of today's technology not only affect the process of displaying AR content, but they also touches the interaction with that digital content. In the *Iron Man* movie, the Augmented Reality content was shown located over a table. Thus, *Tony Stark* was able to grab some digital element from one position to another while he gave orders to Jarvis (his personal digital assistant) by means of voice commands. Nowadays, there are hardware devices that allow the detection of the position of the hands and their posture, but the results they offer are not mature enough. Devices such as gloves, bracelets and other haptic elements allow the user to receive *physical* information created on demand (e.g. when the user *grabs* an AR element). For example, some haptic gloves allow the user to notice collisions with digital elements in the palm. In return, the user must wear gloves full of sensors that tend to be heavy, connected by fat cables to a control device of a considerable size. Thus, the more accurate those devices are, the bulkier tends to be. In that case, these types of devices are not very portable. Another possibility is to present haptic information in the middle of the air. Thus, mid-air haptic devices present haptic feedback as vibrations on them, which the user can notice with his hand without having to wear any device on.

However, conflicts do not stop there. Society has been using the same person-computer interaction method for many years: a 2D screen with an associated keyboard and mouse. To what extent are users willing to leave that comfort zone for the sake of a new interaction method? Can we facilitate that leap by adapting the traditional 2D interfaces to the open 3D space of Augmented Reality?

This whole situation becomes even more complicated when developing AR and MR experiences, as there are more and more devices with the necessary power for AR processing, coming from different manufacturers and with different characteristics and interaction capabilities. In the single-user applications case, this issue does not involve much trauma. An application is implemented for a set of devices and all users who have one of those compatible devices can enjoy the experience. However, the current trend implies that more and more applications support collaborative work among several participants. Applications such as shopping lists, video games, development support tools and many others, allow several users to work together in order to achieve a certain goal.

In the case of AR-based collaborative applications, one of the possibilities is to have an army of programmers making the same application for different platforms or using some “meta-programming” framework that generates multiple applications

from the same source code. In addition, it will be necessary to establish viable communication protocols for each of these platforms, establish a flexible data model and many others. But what if this process is sped up? What if those programmers already had the minimum set of tools for the creation of collaborative AR applications and they would only need to adapt them to the logic of the requirements? In fact, that could make both the development and deployment stages easier and faster, as well as simplifying the process for non-specialist designers.

1.1. Reading guide and brief summary of the content of the chapters

This doctoral thesis aims to contextualize, among others, the questions present in the opening of this Chapter and offering an answer. Thus, the reader interested in knowing the current state of AR systems and their associated technologies, can consult Chapter 2. That reader interested in the transition from traditional platforms to AR platforms can consult Chapter 3. Chapter 4 offers a theoretical perspective of the concept of collaboration in Augmented Reality based experiences. The reader interested in the topic of streamlining the development of collaborative multi-platform AR applications can consult chapter 5. Finally, that reader interested in the topic of how to improve the interaction with the AR content, making use of haptic feedback in this case, could consult the Chapter 6. Below, we present an overview of each Chapter.

Chapter 2 summarizes the current state of the art of the topics present in this thesis. In that line of thought, a definition for Augmented Reality will be presented first, as well as its current state and its projection of the future. It will also analyze how AR is related to Mixed Reality (MR) and Virtual Reality (VR). After this, a study of the main devices on which the visualization of AR content is based is presented. At that point there will also be a review of the main applications and systems in which the AR is can be found. Once this is done, we will analyze multi-user collaboration in the context of Augmented Reality. Then, we will present the state of the art related to the improvement of the interaction with AR content, as well as its possibilities. Finally, we will review the main frameworks and development tools that allow developing collaborative AR applications and systems.

Chapter 3 addresses the study of the effectiveness of interaction in wearable-AR content with respect to conventional platforms (e.g. the web). This study has been based on a specific use case: a tool for the definition of flight missions for drones.

Thus, the components of a mission will be detailed first and the data model on which they are based will be shown. The Mission Definition System architecture will also be presented, of which the definition tool is part. After this, the details of the Web mission editor will be presented, and how users can manipulate the components of a mission to adapt it as required. Next, the implementation of the same editor based on wearable Mixed Reality technology will be presented. The Holo-MIS editor is an adaptation of the web tool to the Mixed Reality context and has its own advantages and interaction peculiarities. Once both implementations have been presented, a validation with users will be presented, in which the main objective is to analyze the degree of acceptance of AR applications adapted from conventional applications. Finally, the experimental results of the validation will be presented.

Chapter 4 is focused on proposing a taxonomy that serves to classify services and systems based on collaborative Augmented and Mixed Reality. Not only that, but it also serves as a consultation aid: selecting the characteristics desired for a future development, it is easy to consult references in the literature that use them. To do so, first an analysis of the collaboration is presented in the context of Augmented and Mixed Reality. In particular, a definition will be proposed to it, and its main characteristics will be analyzed. After this, a theoretical framework based on layers will be proposed that incorporates the *collaboration models* associated with AR systems. Finally, the components of the taxonomy itself will be presented, referencing the associated literature systems.

Chapter 5 is dedicated to SARA: An Architecture for Shared-Augmented Reality experiences and Applications. This architecture has been designed to facilitate the development and deployment of cross-platform collaborative applications based on Augmented Reality. First, the main features of SARA will be presented. Second, the main components of the architecture and the relationship between them will be presented. Third, the data model on which SARA is based will be presented, which has the advantage of being easily extensible for adding new functionalities to the architecture. After this, a brief review of some practical issues in the implementation of SARA will be made. Finally, two prototypes of SARA-based applications will be presented: a tool for displaying AR injected content into the smart city context and a collaborative game prototype of voxels-based world editing.

Chapter 6 is centered on exploring the improvement of interaction with AR content through mid-air haptics feedback. The concept of haptics will first be presented, which is strongly related with the haptic representation of an AR element. After this, the HARP architecture will be presented, designed to facilitate the integration of haptic platform-agnostic feedback. After this, the HARP development process will be explained by proposing two systems: a Haptic Inspector that allows

the user to haptically explore 3D AR models and an implementation of the Simon game, which can be played alone by combining haptic feedback and wearable AR or in collaborative manner(introducing a hand-held device). Once both prototypes have been presented, a validation will be presented with users whose main objective is to understand the contribution of haptics to AR-based tasks. Finally, the results of the validation with users will be presented.

1.2. Methodological formalism of the present thesis

In this section we summarize the usability testing methodology shown in chapters 3 and 6. As stated before, an use-case based approach has been used for the analysis of the different contributions of this thesis. In particular, for Chapters 3 and 6. Figure 1.1 presents an overview of the validation process as well as the standards used to analyze the User Experience and fatigue.

Regarding Chapter 5, the validation of SARA has been through the development of functional prototypes. In addition, the architecture proposed in Chapter 6 and focused on the improvement of interaction with AR content through haptics, takes SARA as a starting point, extending its capabilities.

1.3. Expected contributions of this thesis

In this Section we present the general contributions of this doctoral thesis in a summarized way.

- A study on the assessment and comparison of user interaction in the context of wearable Augmented Reality. This study has been carried out focused on one use case: the definition of drone flight missions. We make a comparison between the User Experience of a traditional, web implementation of the tool versus a wearable-AR based implementation.
- An analysis of collaboration in multi-user systems based on AR. This analysis has allowed us to define the main features of the collaboration in such systems, as well as establishing a theoretical framework in which to place each component of it: from the point of view of high-level visualization, passing through the interaction components including the collaboration management itself.
- A taxonomy to classify collaborative AR systems according to their main features. This taxonomy is the result of the analysis of the current state of the art

Figure 1.1: An overview of the methodological formalism based on use-cases followed in this thesis

in collaborative Augmented Reality systems and serves two main objectives. First of all, it abstracts the main characteristics of collaborative AR-based systems, which have been taken into account for the development of our collaborative AR architecture. Secondly, this taxonomy serves as a consultation aid for future developments. Thus, given a number of desirable characteristics in a future AR experience, taxonomy can be used to search for related references in the literature.

- An architecture to facilitate and expedite the development and deployment of collaborative multi-platform AR experiences together with a set of artifacts ready to be used for future developments. This architecture has the code name SARA, and it applies the theoretical knowledge acquired both from the analysis of the state of the art in related fields and the study of collaboration in AR systems, and from the aforementioned taxonomy.

- Finally, we present a study on the improvement of interaction with AR content by means of haptic feedback. This study has been based on two main use cases: a haptic inspector (which combines AR elements with mid-air haptics) and a haptic based game. From the combination of AR elements and haptic feedback arises the term *harptics*.

Chapter 2

STATE OF THE ART

History will be kind to me for I
intend to write it

*Sir Winston Leonard
Spencer-Churchill*

When designing this state-of-the-art chapter, we have done it in such a way that the interested reader can focus on the desired topic. As a result of this, this chapter has been divided into five pillars closely related to the structure of this thesis. Below we summarize the contributions of each field.

First, a state-of-the-art regarding Augmented and Mixed Reality is presented, as a gateway to the fundamental body of this thesis. Thereby, we take an overview of the most popular definitions of AR and MR as well as their relations.

Second, we present a state-of-the-art related to the devices used to display AR and MR content. This analysis has mainly influenced the development of the Taxonomy presented in Chapter 4.

In third place, we present a compilation of the state-of-the-art related to collaboration in the context of AR and MR. As before, this analysis has served as a starting point for the definition of the Taxonomy presented in Chapter 4 and as inspiration for the design of SARA in Chapter 5.

Fourth, we present a state-of-the-art on interaction with AR and MR digital content, as well as the process of improving it. We have placed special emphasis on the analysis of haptic devices since the main ideas involved in Chapter 6 were extracted from there.

Fifth and finally, we have dedicated a Section on its own to these frameworks and architectures that has been proposed for creating AR experiences, whether they are collaborative and multiplatform or not. The design of the SARA architecture

in Chapter 5 has been heavily influenced by this analysis given that it allowed us to detect the limitations and advantages of the proposed solutions.

The last part of this Chapter will summarize the main challenges found in the literature, challenges that this thesis address. Next, we proceed to detail the Sections of the state-of-the-art one by one.

2.1. Augmented and Mixed Reality, definition, current status and future projection

At the time of writing this document, Augmented Reality (AR) technology is becoming more and more popular. With the emergence of massive smartphone games like Pokémon Go (pok, 2020), many users have been able to test their possibilities in an accessible way. Moreover, thanks to the success of this application, other similar re-implementations have appeared, but including dinosaurs (jur, 2020) and the wizarding world (har, 2020). Likewise, it is increasingly popular for news TV programs to use Augmented Reality for emphasizing their content, such as weather prediction (Feldman, 2019) or election results (Antena3, 2019).

Thanks to the last advances in image processing and mobile computation power, nowadays it is enough to have access to a smartphone (more or less new) to open a digital window in the real world. But nevertheless, this does not stop here. More and more smartphones have multiple cameras, which allows a better analysis of the physical world, which in turn allows digital content to interact more reliably with it. Due also to the emergence of development frameworks and libraries that facilitate the development of such applications, it is increasingly easier to integrate AR functionalities in different contexts.

Thereby, the future of Augmented Reality seems brighter than ever. So much so, that some sources in 2019 indicated that by 2020 (year of this document), 100 million consumers will use Augmented Reality when shopping, both from home and in physical stores (gar, 2019). At this point, the question arises of what exactly Augmented Reality is. In its most basic definition, Augmented Reality is a technology based on the superimposition of digital content over the real world. His journey began in the 60s, when Ivan Sutherland published in 1965 a brief paper (Sutherland, 1965) which three years later, in 1968, would conclude with a new type of device: a Head-Mounted Display. This new device would allow to present digital content between the user and the physical world. According to the author himself: “*With appropriate programming such a display could literally be the Wonderland into which Alice walked*”. It was not until 90s when the term Augmented Reality appeared, coined by Thomas P. Caudell and David Mizell (Thomas and David, 1992). In 1994, Paul Milgram and Fumio Kishino introduced the *Virtuality Continuum* concept (Milgram

and Kishino, 1994), on which they established a classification based on the user location and its relation with the digital content. Figure 2.1 shows an adaptation of the widespread and well-known representation of the virtuality continuum.

Figure 2.1: Milgram and Kishino’s Virtuality Continuum (VC)

Thus, in Figure 2.1 the real / physical world occupies the left side of the classification while the virtual world is located on right side. If the user stays in the real world, and the digital information is added to it, we have the case of Augmented Reality (AR). Otherwise, if the user is introduced into the virtual world while keeping some contact with physical elements (i.e. isolated from the real world), we would be in the case of Augmented Virtuality (AV). Nowadays, this last situation is commonly known as *Virtual Reality*. Although it is not directly contemplated within the scope of this thesis, many of the proposals applicable to Mixed and Augmented Reality can be applied to it. For Milgram and Kishino, the *Mixed Reality* (MR) concept appears here as a mix between the virtual content and information from the physical world, in such a way that digital assets may interact with the real world. One of the most widespread interpretations today is the one from Azuma in 1997 (Azuma, 1997), who specifies that AR has to meet three basic requirements: i) to combine the real world with virtual information, ii) to be interactive in real time and iii) to be presented in 3D. This definition leaves aside those systems that display superimposed digital information about the real world in a non-interactive way (e.g. 2D overlays).

Nowadays, the *Augmented Reality* concept is sometimes used in substitution of Mixed Reality by developers, users and manufacturers. In the context of this thesis that same assimilation will be done. In this line of thought, the terms of AR and MR will be used with similar meaning. Thereby, when we refer to the Augmented Reality, we also assume that virtual content will be related to the physical world, thus being the proposals and concepts presented in this document directly applicable to Mixed Reality context.

More recently, in 2019, the company Magic Leap presented the concept of *Magicverse* (mag, 2019). In their words, Magicverse is *an Emergent System of Systems that brings the physical with the digital in a large scale, persistent manner within a community of people*. Figure 2.2 shows the layers in which they divided the Magicverse.

MAGICVERSE SPATIAL APPLICATION LAYERS

COPYRIGHT © 2019

Figure 2.2: Magicverse layers. This Figure has been extracted from the Magic Leap webpage

In the bottom side of Figure 2.2 two different base layers appear: the *physical* world and the *digital* world. The *physical* world represents the real-world infrastructure disposed across the terrain. On its behalf, the *digital* world represents the virtual information or the virtual representation generated from the physical elements. By taking the combination of those two base layers, they define a set of *spatial application layers*, which combines several data sources that share a certain scope. Thus, for example, the *health and wellness* layer could incorporate information from the ambulances, the hospitals and other sources like air pollution. Within the interpretation of the Magicverse, there is a concept of vital importance: "*Data, information and experiences within these environments are unlocked from screens and servers to persist at scale in contextually relevant physical environments*". In

other words, the Magicverse they propose makes digital content independent of the physical platform used to consume it, also having the context of the physical environment integrated in some way into digital content. For example, if a real-world traffic light has an associated digital representation on the Magicverse (which for example shows the remaining time to change its color), its position in the real world would be relevant in order to show it to different users who, for example, were by car.

We believe that the current trend will follow a channel similar to that proposed in the Magicverse. Instead of having independent services for which adapted AR experiences are created, in our opinion some kind of mediating layer will be established between them, in such a way that the creation of AR and MR experiences is not slowed down by the technology itself.

2.2. Augmented and Mixed Reality Devices

Regarding the physical devices used for displaying AR and MR content, there are several possibilities to display Augmented Reality and Mixed content. Mainly two possible categories are usually established. On the first one, devices are divided according on how the digital content is presented to the user. Briefly, the following three categories isolated:

- Video see-through: The virtual environment in which the user is immersed is replaced by a video feed of the real world and the AR content is overlaid upon the digitized images.
- Optical see-through: In this case there is no video feed of the real world. The user can directly see it while the AR digital overlay is shown upon it by means of transparent mirrors or lenses. In other words, the digital elements are shown between the user and the real world.
- Projective AR: This technique consists of projecting AR digital information on real, physical elements. By doing that, the physical objects themselves become projective displays.

The second category for classifying AR devices is established by taking into account *where* the AR device is positioned. Thus, there are mainly three categories:

- Head-worn: In this case, the user must wear a device similar to a helmet. This type of devices allows the user to keep their hands free, which can be used for the interactive component. However, with current technology they tend to be heavy and bulky, which can lead to neck pain. The devices appearing on the market are becoming lighter and lighter, which can lead to a boom in this type

of devices in the not so distant future. Besides that, current trends indicate that these devices will increasingly resemble the common glasses that are used on day-to-day life. Figure 2.3a shows the HoloLens 2, the most recent Mixed Reality device presented by Microsoft (Microsoft, 2020).

- Hand-held: In this case, the user has to grab with her hand the AR device. A well-known example of this type of devices are smartphones. Those devices draw the AR content over imposed to a video feed obtained with the camera, and the user can touch the screen in order to interact with the AR assets. Figure 2.3b shows a hand-held device, in particular an iPad, that a user is holding with his hand. Thus, by adjusting the position and rotation of the device, he can inspect a 3D model with architecture details ((Apple, 2020)).
- Spatial AR: In this situation it is not required for the user to wearing nor grabbing any device. Usually, the AR component is located on a fixed position. Figure 2.3 shows an example of Spatial Augmented Reality (SAR). In that example, an AR overlay is projected over a sandbox. The digital content is updated according to the heights of hills and valleys of the sand and then different colors are associated to different heights. (University of California, 2020). In this type of systems, the user can interact with the AR content by using her hands.

Figure 2.3: Example of Augmented Reality devices according to their position: **(a)** Head-worn; **(b)** Hand-held; **(c)** Spatial AR

More details about both categorizations can be found in the research of Rick Van Krevelen and Ronald Poelman (Van Krevelen and Poelman, 2010).

Due to the importance of Head-Mounted Displays (HMD) for the scope of this thesis, it has been decided to dedicate a review of its own. We have focused on those that present Augmented and Mixed Reality functions, leaving aside those that are exclusively focused on Virtual reality. Table 2.1 collects the basic information of each device.

	Launching Year	Launching Cost	Technology	Comments
Vuzix star 1200	2011	\$4999	-	The capabilities of this device were quite limited and did not offer spatial understanding. Its scope has focused on military applications and more recently (2017) with the enterprise
Google Glasses	2013-2014	\$1500	-	Despite being one of the pioneers of AR, HMD devices, its low market acceptance and skepticism towards user privacy caused the product to be discontinued.
Meta 2	2016	\$949	Meta SDK. Unity. It requires an external PC	Due to a series of controversies with the investors and staff readjustments, in 2019 the company was declared insolvent. Thus, the access to these devices may be affected. They are still considered in development.
HoloLens V1	2016	\$3000	Universal Windows Platform Unity, Direct X, Vuforia	They represented a great step in the HMD AR field. They do not require an external PC to work. In addition, they included gesture detection and voice recognition as well as physical world understanding
Magic Leap	2018	\$2000	Unity, Unreal Engine	The product has been surrounded by controversy regarding the fact that the promising results did not match the ones presented.

Table 2.1: An overview of the main Head-Mounted Displays for Augmented and Mixed Reality. Part 1

	Launching Year	Launching Cost	Technology	Comments
Holoboard	2018	\$249?	Unity plugin	Android and iOS compatible. Cross-platform and Cross-device
HoloLens V2	2019	\$3500	Universal Windows Platform Unity, Direct X, Vuforia	The represent an improvement on the Hololens V1 in terms of power and weight reduction. Additionally, they allow direct interaction between the user's hand and MR content
Nreal	2019	\$1199	Unity. Unreal engine and native Android to be released	They require a connection with a smartphone. Focused on their small size
Third eye Gen X2	2019	\$1950	Android only	An Android smartphone is required

Table 2.2: An overview of the main Head-Mounted Displays for Augmented and Mixed Reality. Part 1

In the next section we will analyze the current state of the art regarding collaborative systems based on AR and MR.

2.3. Collaboration in Augmented and Mixed Reality Context

Collaboration in the context of Augmented and Mixed Reality appeared shortly after the first AR systems emerged (Caudell and Mizell, 1992). One of the first approaches was the Transvision System from Rekimoto (Rekimoto, 1996), which allowed multiple users to share virtual content graphics positioned over a physical table. This system also enabled the users to interact with the content by choosing an action among a set of possible ones (e.g. selection or manipulation) and to perform it by means of a physical device.

Let us highlight two important concepts of this system. First, the *ownership* of the physical control element had to be transferred among users. Hence, two or more users could not work on the same digital element at the same time. This concept of *blocked ownership* has been adopted in many of later systems, with its associated advantages (easy to implement and little prone to conflict situations) and its limitations (a user must wait for another user to end-up the interaction in order to do hers). Second, each user's application had access to its own 3D model database which was kept synchronized with the remaining ones by propagating modifications between the users. This concept of synchronization between multiple devices to maintain a coherent state is inherent to multi-device or multi-user applications, which is commonly known as *master-slave* configuration.

The Studierstube system and its implementations (Schmalstieg et al., 1996; Szalavári et al., 1998; Schmalstieg et al., 2002) substituted that process of sharing the same digital content between all the users by the concept of *information layers*. Thus, control could be established on which layers each user was able to see and which ones were not. This in turn implied that the information presented to each user could be individually controlled. In order to interact with the AR content, participants may use a physical device, but unlike what happened with the Rekimoto system, in this system each user had their own manipulation device. In other words, several users could be interacting with AR content at the same time without causing a problem. The same concept of a *master device* and multiple *slaves* used in the Rekimoto system is adopted here. This is due that a master device stored the current, central state of the virtual content and when required, slave devices may download it. Additionally, any changes performed to the state of the virtual content on the slaves' side was propagated to the central state maintained by the master. This system stands out also because it represents more a service provider

than a single-purpose application. Thus, it was used for different fields like education (Kaufmann, 2003) and gaming (Fuhrmann and Purgathofer, 2001). Figure 2.4 shows an original example presented in Schmalstieg et al. (2002). In that example, two users share a view composed of 3D geometric entities and the objective of the collaboration is to teach the other participant to use the tool.

Figure 2.4: Studierstube AR project. Two users collaborate while sharing a set of 3D elements

The idea of combining different platforms into the collaboration was explored by Hollerer in 1999, under the MARS project (Höllerer et al., 1999). In this case, mobile devices and stationary computers was involved into the collaboration. In this system, authors made a distinction between two different contexts where users might be: *outdoors* and *indoors*. A user located indoors could manipulate virtual annotations associated with real world positions and objects while the outdoor users would see them properly placed. Later, both users may swap their roles. In this system each role is associated with a different AR device and hardware set, so that when exchanging the roles, so did the physical elements. Despite the limitations imposed by the hardware required when this research was published, this system stands out as one of the first platforms that integrated different types of devices into the context of Augmented Reality collaboration.

Benko et. al. offered with the VITA project (Benko et al., 2004) a system for off-site visualization of an archaeological excavation. The authors combined Head Mounted Displays, a tracked hand-held device, a high-resolution display and a multi-touch projected table surface to enable the collaboration. From a high-level point of view, the collaboration was thought as the navigation, manipulation and browsing of digital data. Thus, each user had their own representation of the digital content and

they did not have a shared visualization component. In particular, this system did not enable multiple users to manipulate the same digital element. In other words, each of the users worked independently but with a common goal.

Five years later, Nilsson et. al. (Nilsson et al., 2009) proposed the use of Augmented Reality to provide collaboration capabilities in crisis management scenarios between different organizations. In particular, the authors faced the process of sharing common organization-specific information on a map. In this system, AR content was shared among the participants, in such a way that they may see the same element positioned in the same location. However, the digital assets to be presented could be adapted to each participant, according to her affiliation. The proposed solution consisted of decoupling the data (e.g. the position of a car on the map) from its visual AR representation. Hence, the associated AR component for a piece of information was generated based on the affiliation of the participant. Besides that, each client device was notified when any change in its own virtual representation was made by another user (e.g. if he or she moved the vehicle). Then, that change was propagated to the other client devices and each one updated its own AR representation. Regarding the interaction with the digital assets, it was performed by means of a physical device, which in this case was a joystick. Thus, only the user with possession of that joystick was able to interact with and to manipulate the AR content. Figure 2.5a shows how the participants are positioned around the shared position while one of them uses the manipulation control. Figure 2.5b shows an example of what one of the participants would see.

Figure 2.5: Example of the Nilsson's proposal

That same year (Hammad et al., 2009) presented the DARCC system (Distributed AR for Collaborative Construction simulation). This system allowed the user to locate and to operate digital construction elements, in particular cranes. That could be done in a collaborative manner by means of Augmented Reality. The

collaboration was performed within the construction site itself, with all the participants visualizing the same AR content (each user may see the digital crane that he was controlling and those controlled by the other users). In order to show the digital content aligned between all the participants, they combined GPS position and device sensors to convey the idea that the digital assets were located in the physical world. Although all the AR content was shared among all users, the interaction of each one was limited to control a single associated element: the crane. Thus, the concept of ownership appears again. Given that the content alignment was based on GPS, the system had to be used outdoors, because GPS does not work well inside buildings. Figure 2.6 shows how the digital AR assets would appear integrated into the real world.

Figure 2.6: Integration of digital cranes in the real world, in such a way that all participants can see the elements controlled by the others

Apart from these works, there are many other examples of collaborative systems that can be divided according to their area of application: *Industrial* (Datcu et al., 2014b; Aschenbrenner et al., 2018), *Architecture and Construction* (Hammad et al., 2009; Coppens and Mens, 2018), *Education and training* (Blanco-Fernández et al., 2014; Sanabria and Arámburo-Lizárraga, 2017) and *Entertaining* (Datcu et al., 2014c; Huo et al., 2018). An exhaustive review of collaborative systems based on AR and MR and classified by application field to date has been proposed by de Belen et al. (de Belen et al., 2019).

More recently, some authors like Piumsomboon (Piumsomboon et al., 2019) have explored different visualization metaphors to achieve the collaboration between several participants. In their proposal, a 3D scan of one of the participants is generated and shared with the remaining users. Instead of being full-size, that 3D model is scaled in such a way that other participant can *summon* and physically manipulate it. Figure 2.7 shows a screenshot taken from (Piumsomboon) that combines the point of view of the scanned user and the remote one.

Figure 2.7: Manipulating the miniaturized avatar of a remote user as proposed by Piumsomboon

A similar technique has been proposed by Rhee et. al. (Rhee et al., 2020). However, instead of using a 3D scan of one of the participants, a generic avatar was integrated into the visualization. In particular this system combines Virtual Reality and Augmented Reality and with the objective of integrating the real-world environment into the VR visualization, they used 360° panoramic video. Figure 2.8 shows an schema on how their system works.

Figure 2.8: Schematic vision of the system proposed by Rhee et. al., in which VR and AR users can collaborate.

These are just some examples of how some of the most recent contributions in the field try to somehow introduce the remote participants into the collaboration in an immersive way. Finally, we propose the reading of the classification recently proposed by Pidel and Ackermann Pidel and Ackermann (2020), in which they perform an overview focused on collaborative systems not only based on Augmented

Reality, but also based on Virtual Reality.

2.4. Improving the interaction with Augmented Reality content

When interacting with computers or machines, *natural interaction* usually refers to the final goal of the user experience that aims at being smooth, easy, and evident, i.e., natural to the user. However, *natural interaction* is a subjective concept (Chu and Begole, 2010) that can be applied to a wide range of fields. This is mainly due to different users having different preferences, expectations and background.

Several studies have been conducted regarding the interaction with virtual content without hands, that is, without direct contact from the user. Some approaches (Nilsson, 2007; McNamara and Kabeerdoss, 2016) have made use of users' gaze to show or hide information when using AR displays. It has also been studied whether the use of head gazing or eye pointing (Wang et al., 2019) is more effective when interacting. Results indicate that users tend to prefer the head approach because of its better stability. Given that this type of interaction is quite widespread, some frameworks such as EyeMRTK (Mardanbegi and Pfeiffer, 2019) have been appearing over time to facilitate the development tasks. The survey by Mohamed Khamis et al. (Khamis et al., 2018) collects the history and current situation of gaze-enabled interaction in the context of mobile handheld devices. Finally, a survey about interaction methods for head-mounted displays that emphasizes touch and touchless-based interaction was presented by Lee (Lee and Hui, 2018).

Regarding direct manipulation, users may apply actions over elements like *touching* or *moving*. In the end, the better the interaction adapts to what the user expects, the more *natural* it will feel. Hence, different systems and devices may offer different *natural* ways to interact with them. In wearable AR context, the use of hand gestures has been widely explored in the literature (Billingshurst et al., 2009; Lee et al., 2010). Thanks to the apparition of more powerful devices, nowadays, it is possible not only to detect gestures but also to perform image analysis techniques in real time for tracking hand movements (e.g., Kinect and Leap Motion (Shotton et al., 2011; Marin et al., 2014)). The preference of users for certain gestures has already been explored by some researchers (Piumsomboon et al., 2013). Nevertheless, the interaction with the virtual content is not completely *natural* yet. This is mainly due to one key component that has not been systematically integrated into the interaction: *tactile feedback*. Thereby, the concept of *haptic feedback* promises to tear down the boundaries between the real, physical world and the virtual one. The combination of hand gestures and haptic feedback is positioned as a middle ground between precision and comfort, and its advantages have already been explored in

the literature (Morris et al., 2007).

On its most basics, haptic feedback consists of presenting synthetic tactile stimuli to the users. Since this type of feedback can be generated on demand, it can be used to enhance the perception of Augmented Reality content. Whereas before the user could only see a digital AR asset, now she can *touch* it and perceive it with her hands. There are many different types of haptic devices available today. Two main groups can be distinguished depending on whether the feedback is based on the perception of forces (*kinesthetic*) or produced in the skin of the user (*cutaneous*, tactile if the perception is over the fingers).

Kinesthetic devices form the first group. Some examples of these devices are the one used in the PHANTOM project (Massie et al., 1994)—which consists of a grounded pen attached to a device that generates haptic feedback—or the one proposed by Aleotti et. al. (Aleotti et al., 2014)—a grounded, graspable haptic device that kept track of the user’s hand and generates the feedback, e.g., to simulate contact with digital elements. This kind of device tends to present low portability since they required a grounded hardware component.

The second group is composed by *cutaneous devices*, and it includes devices such as wristbands (myo, 2019; Cœugnet et al., 2017; Pezent et al., 2019), haptic rings (Nunez et al., 2015; Hussain et al., 2016), and gloves (Blake and Gurocak, 2009; hap, 2019; vrg, 2019) as well as mid-air tools. More in detail, cutaneous devices can be divided into the following groups:

- *Basic cutaneous devices.* This kind of device is one of the most popular, since they are simple and provide great results. The vibration of the device is perceived by the user through cutaneous perception that the user is accustomed to, such as the reception of an alert in the smartphone. However, this kind of device presents some limitations; for example, users may find it difficult to understand complicated vibration patterns. Some examples of this kind of device are vibration wristbands ((myo, 2019), discontinued from 2018) (Carcedo et al., 2016). Also, there are possibilities like fingertip devices (Gabardi et al., 2016) as has been said, the multipurpose vibration capabilities of smartphones.
- *Active surfaces.* This kind of devices uses physical components to provide the feedback. One possibility may be to use vertical that can be touched by the user. This kind of device behaves well for rendering surfaces, having nice resolution and accuracy. However, they have limited functionality due to the actuators required to perform the feedback. The biggest limitation of this kind of device is the lack of portability, since they are usually required to be installed in a certain manner. Some examples of this kind of device are Relief project by Leithinger et. al. (Leithinger and Ishii, 2010) and the Materiable system, by Nakagaki et. al (Nakagaki et al., 2016).

- *Mid-air* devices provide the haptic feedback without requiring the user to wear any device (in contrast with devices such as gloves, bracelets, or rings). The main difference with the previous group is that, in the previous one, the user came into physical contact with the haptic device, while in this group, that interaction occurs in the air (hence, the name). There are different kinds of mid-air haptic devices, but they can be divided into three groups according to the technology they use: air vortices (Sodhi et al., 2013), air-jets (Tsalamlal et al., 2014), and ultrasounds (Carter et al., 2013; Iwamoto et al., 2008).

Due to the practical use of mid-air haptic devices in this thesis, these will be emphasized. As their name suggests, Ultrasound-based devices are based on ultrasonic speakers to focus sound waves at a point in the space above them. At the point where the sound waves converge, a pressure point is generated. That pressure point is strong enough to be noticed in the user's hand and, thus, to induce a tactile sensation. By changing the frequency of the ultrasounds, this kind of devices are also able to control the strength of the sensation. A representative of this type of device is Ultrahaptics (Carter et al., 2013; ult, 2019). Some works have already explored the idea of using those haptic points to haptically render 3D objects (Long et al., 2014). In this case, the authors understood by *rendering*, to dispose a set of haptic points in a certain manner and to let the user perceive them with her hands.

In opposition to the ultrasound-based devices, air-vortex ring devices use air flows to provide the haptic feedback. According to some authors, this second group of devices do not present the limitations of the first group, i.e., weak perceived sensation, limited range and the limitation of representing small volumes above the device space. Shtarbanov and Bove for example, (Shtarbanov and Bove Jr., 2018) presented an approach to augment 2D and 3D displays with haptic capabilities.

The combination of AR assets and haptics is present in literature for Head-Mounted Displays (San, 2015), physical traditional screens (Monnai et al., 2014), and custom hardware solutions (Kamuro et al., 2009; Eck and Sandor, 2013). In comparison, the idea of combining Augmented and Mixed Reality with mid-air haptic feedback has not been so thoroughly explored. Moreover, literature related with this topic tends to be focused on the application stage. Some works are only focused on the haptic side of the equation, such as the work of Leithinger (Leithinger, 2018). In his work, he replaced visual aids with haptic feedback using an Ultrahaptics UHDK5 device. This system provided haptic sensations surrounding the hand of the user like a haptic aura. Thus, that feedback was used to guide the user towards a physical, real-world control element in order to provide information about its functionality and type of control.

In the same line of not incorporating AR information, Goncu and Kroll (Goncu and Kroll, 2018) presented the GraVVITAS system, a multi-modal system for presenting 2D graphics. The system they exposed had the limitation of not being able

to display 3D graphics, and to solve this problem, they extended the system to integrate the Ultrahaptics UHEV1 device. In that system, a 3D model was generated using an external tool. In the next step, that 3D model was divided into slices according to its volume. The result of that slicing process was then haptically presented to the user by selecting the slice associated to the user's palm position. From this work, two main limitations can be obtained. First, their system had the limitation of requiring a specific program for the generation of the 3D model, OpenSCAD. Second, their system was limited to iOS and MacOS devices, without offering possibilities for the integration of arbitrary devices.

Other authors have created interaction metaphors with the objective of generating tools to manipulate digital elements using mid-air devices (Song et al., 2012). In Geerts's work (Geerts, 2018), the author goes one step further and removes the physical component of the interaction. Instead of having a real button, the element was presented as a haptic, Augmented-Reality button to the user. His work explored the idea about how to design visual and haptic interaction patterns. This system demonstrated that a physical component could be simulated through haptic interaction and that users were really able to interact with them. Dzidek et al. (Dzidek et al., 2018) combined Augmented Reality and mid-air haptics and conducted an evaluation with users. Their focus was on how gestures and actions are performed over the haptic device. Results shown that the improved graphic components with haptic rendering happened to be more memorable. For the authors, this suggested that active exploration of haptic and AR content may be more convenient rather than in a purely receptive manner.

A little ahead of events, this same train of thought has been followed in our investigation. Going forward with what will be indicated later, one of the things we seek is to know how the user explores AR models enhanced with haptic. Frutos-Pascual et al. (Frutos-Pascual et al., 2019) conducted an experiment to analyze the influence of mid-air haptic feedback as supplementary feedback in Virtual Reality (VR) context. They found that visual feedback was preferred among the users as supplementary cue, but haptics did increase the accuracy when dealing with small targets.

Mid-air haptic technologies can also be combined with gesture recognition techniques to generate attractive and powerful results. Kervegant et al. (Kervegant et al., 2017) briefly explored the idea of incorporating mid-air haptic feedback to the AR experience by combining HoloLens and Ultrahaptics. Their work consisted of displaying a virtual sphere over an Ultrahaptics device with which the user can interact. Although that interaction was limited to exploring the collisions between the user's hand and the sphere, this concept opens the door for more advanced experiences.

2.5. Frameworks and architectures for Augmented and Mixed Reality

The ideation of frameworks and architectures to facilitate the development of new cross-platform applications has already been raised previously in the literature. Some of these works have focused on wearable devices (Chi and Li, 2015) or smartwatches (Houben and Marquardt, 2015), infrastructure devices like the Kinect (Nebeling et al., 2014b) or the generation of adaptive web applications (Yang and Wigdor, 2014) and user interfaces (Nebeling et al., 2014a). Following the same approach of facilitating the development of cross-platform application but focusing on the Augmented Reality Context, Speicher et al. (Speicher et al., 2018) developed the XD-AR framework, created to address the issues found when developing AR applications in a multiplatform context. More specifically, they focused on the more recent devices and frameworks used in that moment (e.g. Tango, ARCore and ARKit).

In industry context, Microsoft has presented their *Spatial Anchors* system (azu, 2020), which consists of a cross-platform developer service to deliver multi-user Mixed Reality experiences. It enables to store and afterwards retrieve the position of the users' devices and their relationship with the real world. Microsoft has also launched recently their *Spectator View* system (spe, 2019), which enables to stream the MR digital content from a HoloLens device (in the form of *holograms*) to be shared with other users who are carrying 3D hand held devices. Thanks to this feature, the digital content can be easily shared among several users. Spatial synchronization is performed among the devices to present the digital content aligned. Although it is a very powerful and easy to integrate tool, its main drawback is that the cross-platform support is offered by means of the Unity environment, which is itself cross-platform. When integrating it into an iOS or Android device, for example, it must be done through a Unity application, i.e. without using the native frameworks.

OpenXR is a royalty-free, open standard that provides high-performance access to AR and VR platform and devices (Usually, the acronym XR refers the spectrum of hardware, applications and techniques related with Augmented and Virtual Reality). In other words, it enables applications to reach multiple hardware platforms without requiring its re-implementation. the main challenge this standard address is the fragmentation of the XR market. Thus, each AR application and engine must use the proprietary APIs associated to them. This situation is illustrated in Figure 2.9a.

Figure 2.9: OpenXR before (a) and after (b)

As can be seen in Figure 2.9a, multiple applications can be built with different frameworks (e.g. Unity or Unreal). Then, these applications have to be adapted for different hardware. In opposition to that, OpenXR seeks to provide a common API with the aim of homogenizing the development process for different platforms. Figure 2.9b shows a diagram of this situation. Then, a certain application (e.g. AR APP 2) could run in different devices (e.g. the windows MR ones or the Daydream). With respect to multi-user context and collaborative applications, this standard does not address it directly. This is left to the developer, who, making use in the code of the methods that OpenXR offers, should establish the communication protocol and state exchange between the involved clients.

WebXR on its behalf is a group of standards which according to the Mozilla reference (Mozilla, 2020), “*they are used together to support rendering 3D scenes to hardware designed for presenting virtual worlds (Virtual Reality), or for adding graphical imagery to the real world (Augmented Reality)*”. Thereby, the WebXR Device API implements the core of the WebXR feature set: managing the selection of output devices, rendering the 3D scene on the chosen device and managing motion vectors created by means of the input controllers. From a technical point of view, it is based on WebGL, which is used for rendering the 3D world into the WebXR session. In the provided reference, the life cycle of a WebXR application is as follows. First, it is required to check whether the device has XR capabilities or not. Second, a WebXR session is created. Third, elements can be added to that session. Thus, WebXR offers a series of calls in the code, i.e. an API, which developers can use to create AR sessions using web technology while abstracting the target devices. However, this standard does not offer easy multi-user management.

On top of the WebXR standard, Amazon has presented a set of tools called Sumerian (sum, 2020). Sumerian has a web approach: it allows the creation of 3D scenes that can then be viewed in any browser. In this case, Amazon opts for an approach in which the creation of 3D elements is facilitated to the user through an IDE and its own scripting system. Thus, from the same implementation, this tool allows focusing on three different visualization platforms: Web 3D as well as Augmented and Virtual Reality devices. Despite this versatility, facing AR devices presents several problems. First, it is only available for iOS and Android devices, leaving the HMD suite of devices out. Second, being based on a web implementation, the performance of AR experiences can be affected. Third, being an Amazon proprietary tool, it is not open to improvement or adaptation by the community. However, the fact that high-volume companies like Microsoft and Amazon seek to facilitate the creation of multi-user AR experiences, gives an idea of the potential they have. In the next Section, we present an overview of the challenges that have been identified in the literature and in commercial proposals. A similar approach is presented by PlugXR (plu, 2020), a cloud-based platform that allows the creation of AR experiences targeting both, Web AR as well as iOS and Android devices. The philosophy of this platform is to make the development process as easy as possible to the point of not requiring coding skills. Other than those two, Viscopic Pins (vis, 2020) works in the same line of facilitating development, focused on devices such as the Microsoft HoloLens, smartphones and tablets. In this case, the focus of the platform is oriented to the industry-

As a last note on the frameworks for Augmented and Mixed Reality, we would like to highlight the extensive review of standards related with both technologies presented by the Augmented Reality Framework ETSI Industry Specification Group (, ARF). That document compiles standards related to the Augmented Reality itself, data presentation and communication protocols as well as hardware APIs and User Interaction.

2.6. Comments on the challenges identified in the literature

While performing the analysis of the related literature and the commercial proposals, a set of challenges has been found. In this Section our intention is to draw attention to these challenges and to illustrate how these have been approached in this thesis.

First, to the author's knowledge, there has not been any user study focused on the comparative between interaction in AR and traditional platforms. From our point of view and considering the current evolution of the market and technology,

these types of studies are vital to know the strengths and weaknesses of both types of interaction. With this objective in mind, we offer in Chapter 3 a comparison between both types of systems.

Second, there are some studies like the one of (de Belen et al., 2019) that make a classification of Augmented and Mixed Reality based systems and experiences. However, these classifications are typically focused on one or two parameters, such as the scope or the AR content display technology. The lack of this type of study prevents the comparison and classification of collaborative experiences based on Augmented Reality. To solve this problem, Chapter 4 of this thesis proposes a high-level taxonomy of AR collaborative experiences. This taxonomy in turn will serve as a reference for future developments, thus facilitating the search for references and similar works in the literature.

Third, despite the number of AR systems proposed in the literature, we have not found a study on the characteristics of collaboration in this context. Many of these systems in the literature have been created ad-hoc or their creation has been focused on the development of the prototype in question. However, it should be noted that several systems that use AR and that involve multiple users, do not necessarily have the same characteristics. Thus, the individual and common objectives of the users, the way in which the AR elements are viewed and the way of interacting with them are some of the parameters that differentiate some systems from others. Thus, the analysis of collaboration in AR and MR context proposed in Chapter 4 aims to fill this gap.

Fourth, the current trend regarding the development of collaborative AR experiences can be summarized in three alternatives. First, the use of own libraries, which tend to be restricted to a certain family of devices, offering instead a homogeneous way of development (e.g. ARKit or ARCore). Second, the use of techniques such as WebXR or OpenXR, which while seeking to offer a cross-platform context, is not necessarily focused on multi-user applications and generally forces work at a low API level. Finally, there is a third option, like the one represented by Unity, in which the system itself has a multi-platform implementation.

Fifth, a standard way to generalize the development of experiences that combine mid-air haptics with AR content has not been proposed yet. Some proposals in the literature approach this objective, but they tend to focus on a single device and are not generalizable. This is in part due to each AR and haptic device making use of its own development platform, which makes it difficult to establish a common framework.

Throughout the following chapters of this thesis, we will face, among others, the challenges presented above.

Chapter 3

ASSESSING AND COMPARING USER EXPERIENCE THROUGH WEARABLE AUGMENTED REALITY

It does not do to leave a live
dragon out of your calculations if
you live near him

J. R. R. Tolkien, The hobbit

Conventional 2D screens have been the companions of society for almost 70 years. Usually, these are accompanied by two well-known peripherals: the keyboard and the mouse. Thus, the keyboard serves as a method for data entry while the mouse allows interaction with digital content. However, how do we conceal the transition from these traditional methods to AR-based systems? Could these traditional systems be replaced by AR-based emerging ones? Among those AR-based systems, we focus those on which interaction with digital content and data entry occurs without any peripherals: wearable AR systems. Thus, in this type of systems the user wears an HMD (Head Mounted Display) device that allows him to see the AR elements, while he can interact with them through gestures or voice commands.

One possible way to address these issues is to compare the effectiveness of wearable AR-based systems with respect to a “conventional” system. We will define that effectiveness in terms of precision and usability. With that objective in mind, we will rely on a use case: a drone flight creation system. This type of tools helps the user to define flights that will be carried out by drones. In general, all of them

shares two main characteristics. First, they allow the user to establish a series of points forming a path, that will later be followed by the drone. Second, they rely in standard technologies such as Google Maps (or any other map provider service) and they offer a 2D point of view. Figure 3.1 shows a screenshot of the DJI Ground Station Pro iPad application (DJI, 2020). In the Figure, a set of geographical positions are defined wrapping an area. The final objective of this kind of tool is to generate a flight plan that follows the restrictions generated by the defined positions and annotations.

Figure 3.1: A screenshot of the DJI Ground Station Pro iPad application

Thus, we observe a set of tools with a large geometric load facing the user along with a strong visualization component. Apart from that, great precision is required when defining flights. These three characteristics make drone flight definition systems a perfect use case for the study presented in this thesis. Thereby, we can analyze the user interaction in Augmented and Mixed Reality context by means of the following components:

- A tool designed for conventional platforms, which will be used as the baseline for the comparisons.

- A tool for wearable, Mixed Reality platforms, with functionality equivalent to the previous one
- A comparative analysis of the User Experience and a fatigue study between comparing both implementations
- The characterization of the strengths and weaknesses of each implementation, which will provide us an indication of what factors most influence the acceptance or rejection of the wearable AR systems.

The conventional platform selected to be used as the baseline, consists of a web implementation of the flight editor. On its behalf, the target platform for the wearable-AR implementation has been the HoloLens. Both implementations will be integrated into the MDS (Mission Definition System) infrastructure, which packages the necessary functionalities for the generation and distribution of drone flights based on the definition from the users.

Despite the technical limitations of the MR version (mainly related to the device visualization constraints and weight) and the reduction in terms of efficiency (with tasks requiring more time to be completed), during the validation stage several participants have expressed their preference for this version. However, the precision problem in MR HMD context must be always taken into account, which opens the door to new interaction techniques to solve the gap between conventional interaction methods and newer ones. The results

The rest of this Chapter is structured as follows. First, we will expose an overview of the Mission Definition System (MDS) and its main objectives, as well as the data model in which it is based on. Then, the details of the web-based implementation will be presented. Next, we present the details of the Holo-MIS tool, the AR implementation of the mission editor. Once both implementations have been presented, the validation with users is addressed. Then, the objectives of the tests, the participants and apparatus and the procedure, tasks and experimental design are presented. After that, the experimental results will be shown, making a division into three categories: i) performance and errors, ii) User Experience and fatigue and iii) comments on technology preference and strengths and weaknesses of both systems.

3.1. Use Case: Drone Flight Missions. General view and main components

The main objective of the Mission Definition System (MDS) is to provide drone flight generation functionalities. Thus, a user can define *Flight Mission* that can later be carried out manually by a pilot or automatically by a drone. They also

have an operational objective associated such as surveillance or infrastructure maintenance tasks. From this point, we will refer to them simply as *missions*. Figure 3.2 shows an overview of the main components of a *mission*: a takeoff and a landing point and multiple inspection with different geometries.

Figure 3.2: Components of a drone mission: a set of *inspections*, a take-off point, a landing point and the estimated trajectory to perform the flight (as the orange line)

As shown in Figure 3.2, each mission is defined by one or several *inspections*, each one of them with an associated geometrical shape. Users may use those *inspections* that best suit the purpose of the flight. Additionally, multiple *inspections* can be combined to generate advanced geometries. Let us illustrate this concept with two examples. As a first example, in order to inspect a telecommunications antenna in all its height, a possible way is to use a cylindrical shaped *inspection*. However, if according to the specifications of the mission, we are only interested on inspecting its top, we may use a *circle inspection*. As a second example, let us imagine a scenario in which it is required to fly a drone along a road, with the hypothetical objective of looking for damages in the asphalt. In order to do so, we may combine several *line inspections* to follow the path of the road. Moving on, in addition to that set of *inspections*, missions require two more parameters: the *take-off* point from which the drone will start the flight and the *landing* point.

Once the required parameters have been disposed by the user, the system generates two additional components: a *Flight Plan* and an estimated *Trajectory*. The

Flight Plan is generated first and it contains a series of positions associated to each inspection through which the drone should pass. Those points are calculated by aggregating two different elements. First, each *inspection* has one or several *Measurement processes* associated, each one representing the actions that the drone must perform for that *inspection*. At the same time, each *measurement process* has one or many *sensors*, whose associated information will be considered when calculating the *Flight Plan*. From the calculated points of the *Flight Plan*, the system estimates a trajectory that the drone would perform to follow them. That is done by using a dynamic behavior model of the drone. Figure 3.3 gathers the main entities handled by the MDS system as a class diagram. Aside from that, a more detailed class diagram of the data model can be found on Appendix A.

Figure 3.3: Data model of the Mission Definition System (MDS)

In the upper part of the Figure, the *inspections* are located. Every *inspection* is defined by three elements: i) a *geometry* that sets the shape, ii) a *minimum distance* to the inspection that the drone must respect and iii) a *maximum distance* to which the drone can move away. Each *inspection* also has one or many associated *Measurement Processes*. At the same time, each *Measurement Process* has one or many *Sensors*. Finally, the *Geometry* enumeration illustrates the different shapes the system handles.

Figure 3.4 presents an overview diagram of the MDS system. In the left most part, the user is represented. She will interact with the system by means of a

Human-Machine Interface (HMI). Two different implementations of the HMI are available: a web-based application and a Mixed Reality one. The implementation and details of both tools are presented in Sections 3.2 and 3.3. The calculations of the *Flight Plan* and the *Trajectory* are performed on the *MDS Calculation Engine*, once the user has defined the mission. The results of these calculations as well as the original mission are stored on a *Mission Repository*. From there, a *Drone Interface* application may access to the details of a specific mission in order to do one of two possible actions: either generating the flight instructions necessary for a drone to flight automatically, or offering the details of the mission to a human pilot, who will be manually controlling the drone during the flight. More details of the whole architecture and the calculations performed in the *Calculation Engine* can be found in the article by (Besada et al., 2018).

Figure 3.4: Overview of the MDS architecture

Once the overview of the MDS architecture has been presented, the two possible implementations of the HMI will be presented in detail over the next two Sections.

3.2. Web tool for the mission creation

In this Section we will present the details of the web implementation of the HMI component integrated into the MDS system. First, we will take an overview on the main interface of the tool and its main components. Then, details about how *inspections* are manipulated in the web editor will be presented.

3.2.1. Main interface

Figure 3.5 shows a screenshot of the interface as presented during the definition process. As can be seen in the Figure, the main space is divided into three different zones, labeled as A, B and C:

- A) In this panel the general parameters of the mission are grouped. From here, the user can write a brief textual description of the mission (maybe with details

Figure 3.5: Screenshot of the web interface

for the human pilot) as well as beginning the process of establishing the take-off and landing points. From this panel it is also possible to save the current state of the mission in the repository and to and giving the order to start the *Flight Plan* and *Trajectory* calculations.

- B) This section holds the visual representation of the mission and it is divided in two different components:
- The left side of zone B corresponds to the 3D representation of the mission. In this 3D component, the lower plane represents a portion of the real terrain shown by the Google map of the right side, being both centered on the same geographical position. *Inspections* appear on this plane, being represented according to their basic components. The user can interact with these *inspections* by selecting them and the details of how they can be manipulated will be presented in Section 3.2.2.
 - The right side of zone B is occupied by a 3D standard Google map, with which the user can interact in the usual way. Whenever the user moves the center of this map or changes its zoom, the 3D component gets updated. Thus, both components are linked together and they keep a consistent state.
- C) The rightmost panel of the main interface collects the details of the *inspection* that the user is currently manipulating. Therefore, it will only be visible as long as that editing process goes on. The content of this panel will be adapted to the inspection in question, showing the necessary controls to manipulate

each one of them. For example, in Figure 3.5 the user has selected the *cylinder inspection*. The aforementioned panel shows a field for setting the radius of the cylinder as well as two sections for setting the details of the two circles that shape it. In addition to these basic parameters, the tool also offers an advanced manipulation panel that allows establishing the minimum and maximum distance that the drone must respect, as well as the establishment of the measurement processes belonging to the inspection. Figure 3.6 shows the appearance of the panel when adding a measurement process to the side of a *cylinder inspection*. By using this panel, the user may add new processes or remove them, as well as to manipulate their details. Similar panels are provided for each list of measurement processes and all of them behave in the same way.

Figure 3.6: Example of a panel used for adding measurement processes associated with an *inspection*

3.2.2. Creation and Manipulation of an Inspection using the Web User Interface

This Subsection details the process by which the user creates *inspections* and updates their parameters using the web-tool. The first step to create an *inspection*, is to select the desired one from the drop-down located on the upper side of the interface. Figure 3.7 shows the appearance of the drop-down once opened.

The *inspections* on Figure 3.7 can be divided in two groups. The first group is

Figure 3.7: Appearance of the drop-down used to select the Inspection to be added in the web tool

formed by the *cylinder*, *cone*, *truncated cone*, *circle* and *point* and for adding them to the map, the user has to click anywhere of it. Those *inspections* will then be created centered on the selected point. To define the *inspections* on the second group (*line*, *catenary* and *polygon*), the user has to consecutively click on the 2D map to define the points that form the *inspection*. Once the points have the desired appearance, the user has to confirm the creation. Later, it is possible to modify the parameters using the side details panel.

From the 3D view of the main interface, the user can select any *inspection* by clicking them. That will open the details panel for that *inspection*, allowing the user to modify its details. To illustrate the manipulation process, let us take as an example a *cone inspection*. According to the model, a *cone inspection* is defined by two circles, one at the top with no radius and one at the bottom with a non-zero radius. From the point of view of the user, the top element can just be seen as a position, which represents the tip of that cone. Figure 3.8 shows the parameters of the Cone (right) and how they are graphically represented in the 3D view (left).

As can be seen in Figure 3.8, the positions of the top and bottom circles are represented by blue spheres. Each one of these spheres is identified as an *anchor*. If the user clicks one of them, that anchor will pass to edit mode and display a *three-axis manipulator*. This component consists of three arrows, each of them pointing

Figure 3.8: Parameters of a Cone Inspection (right) and their representation on the 3D component of the interface as anchors (blue spheres)

in the direction of the three coordinate axes. The user can click and drag any of those three arrows to move the anchor in the associated direction. Figure 3.9 shows an example in which the controller has been used to move the tip of the cone and how this has altered the appearance of the inspection in both the 3D view and the 2D map.

Figure 3.9: Manipulation of the basic features of a Cone Inspection using the three-axis manipulator in the web version

The use of the 3-axis control for manipulating the *inspections* is what we call the *graphic* way of doing it. In opposition to that would be the *textual* way. In this second case, it is possible to specify the parameters by directly entering the appropriate value in the associated text fields or sliders of the details panel. In any case, the associated values are properly updated. For example, when the user moves up one of the anchors, the *altitude* parameter of the side panel gets updated.

3.3. Holo-MIS: A wearable-MR based tool for the drone mission definition

In this Section we present Holo-MIS, the wearable AR/MR implementation of the mission editor. Like its web counterpart, we will first present the details of its interface and its main components. Later, the process of manipulating *inspections* in the within the AR implementation will be explained. Additionally, the integration of terrain height information in the mission editing process will be exposed. A working version of the Holo-MIS tool was presented as a demo during the Mobile HCI'18 congress (Vaquero-Melchor et al., 2018). The annotations made by the users of the demo served to refine the final version of the tool, which is the one used during the validations with users.

3.3.1. Main interface

The implementation of the mission editor is based on the same data model set out in Section 3.1. In this case, all the tool has been redesigned to be adapted to the 3D visualization capabilities and interaction peculiarities of the AR context. In Figure 3.10 a capture of the Holo-MIS system interface and its main components can be seen. It is important to highlight the fact that when dealing with MR, the elements on the Figure are arranged in the physical space.

The basic information of the Mission is grouped within the gray panel at the top of Figure 3.10 (A). This one is divided into three sections from left to right: i) basic information of the mission and actions to establish take-off and landing points; ii) a list that shows all the already created *inspections* and allows the user to show/hide them independently; iii) a configuration section, from which the user can begin the *Flight Plan* and *Trajectory* calculations and re-position all the AR elements, as well as toggling the visibility of the mission elements.

The main map appears in the section labeled as (B) in the example of Figure 3.10 and placed above the table. Over this map the *inspections* will be displayed. By using the map controls in section (C), the user can move the center of the map towards the desired cardinal direction and change the zoom level. The combination of both elements gives similar functionality to the one described by the Google map of the web interface. Whenever the center of the map or the zoom level are changed, the added *inspections* are updated and properly adapted to the new context. More details about this process will be presented in Appendix A. The map controls are always positioned between the user and the map itself, making them always in sight and not blocked by the *inspections*.

Figure 3.10: 3D terrain view

3.3.2. Creation and manipulation of the *inspection* using the Holo-MIS implementation

To create an *inspection*, the user first must use the *Add Inspection* button located in the left-most side of the main panel. Once this is done, in front of the user appears the *Inspections Palette*, as depicted in Figure 3.11. This palette can be seen as an adaptation of the drop-down shown in Figure 3.7 to the AR context. Thanks to this palette, the user can see at a glance the 3D representation of each of the inspections. Once selected the desired *inspection*, it is added to the map and the user may now start manipulating it. At this stage, the following should be highlighted. To unify the creation of *inspections* it was decided that all of them would be created in the same way. By default, all the *inspections* have a set of default parameters, and they are created in the center of the map. In the web implementation, both the *line* and the *catenary* inspections, do require the user to specify a start and ending point. This is not the case in the Mixed Reality implementation, as they will also appear with default positions in the center of the map .

The first step towards manipulating an *inspection* is selecting it from the map. To do so, the user must *gaze* the inspection and make a *tap* gesture. To gaze means to point with the head towards the desired element. Once done that, the anchors associated to the *inspection* will appear in a similar manner as in the web-UI. A small panel is also shown above the *inspection* gathering its main parameters and

showing them in a textual way. Figure 3.12 shows an example of the panel.

Figure 3.11: Inspections palette as presented to the users.

Once one of the anchors has been selected, a 3-axis manipulator is shown associated with that parameter. This component behaves the same as in the web-UI. The user can gaze one of the three arrows of the component and perform a *drag* gesture to move the anchor in the associated axis. Figure 3.12 shows an example of moving the position of the tip of a Cone Inspection.

Figure 3.12: Manipulating a *cone inspection* using the Holo-MIS AR version

Regarding the *take-off* and *landing* points, the process is as follows. First, the user must tap the proper button in the main panel (located at the left-most side) which starts the setting process. At this point, the user must gaze the desired position on the map and perform a *tap* gesture. By doing that, the point is moved to a new position imitating the way used in the point-and-click web-UI.

3.3.3. Integration of Digital Surface Model data for 3D ground content rendering

To take full advantage of the 3D visualization capabilities in the Mixed Reality implementation, it was decided to integrate Digital Surface Model (DSM) terrain information, so that the user can see an approximation of how the defined inspections are related to the real world. Figure 3.13 shows an example in which the user has created a mission and how it appears with no DSM support (a) and with DSM support (b). Appendix B shows the details of the *Ground micro-service* that provides the DSM information and how that information can be integrated into the editor.

(a) With no DSM support

(b) With DSM support.

Figure 3.13: Representation of the same mission with and without the integration of Digital Surface Model (DSM) data

3.4. Validation with users

3.4.1. User Study Objectives

The user validation is two-folded. In first place and as a main objective we seek to analyze the user experience of applications based on AR and AR that have been adapted from traditional platforms (web in this case). Secondly, we seek to better understand what the advantages and disadvantages are (from the point of view of the users) of each type of system and the possible inclination of the users towards each of them.

3.4.2. Participants and Apparatus

The study has been carried out during July 2019, with 8 independent volunteers recruited in the University (students, professors and PhD students). Apart from these eight users, another one participated in the validation but could not conclude it for reasons external to the test. Therefore, his data has not been included in this document. The ages of the participants range between 23 years and 57 (μ : 28.87, σ : 7.99), all of them males. The following data was obtained from a brief initial survey in which they were asked about their relationship with the AR context. 44.4% of users do use AR applications sporadically and 11.1% once a month, while the remaining 44.4% never use AR. The complete test had an average duration of 65 minutes (σ : 17.2) per user. Finally, the apparatus used for the test is formed by the two implementation of the Mission Definition System explained in Section 3.2 (as a conventional User Interface) and Section 3.3 (as a representative of wearable-MR devices).

3.4.3. Procedure, Tasks and Experimental Design

The procedure has been the same for all the participants and it was as follows. First, all of them were asked to fill the user consent and a brief survey regarding their personal data and their previous experience with Augmented and Mixed Reality. To grant that all users receive the same information, the facilitator has made use of a written script that was read to all of them.

From a high-level point of view, the study has been divided into two big blocks. In the first block users are faced to the process of designing a mission using the web implementation (which is taken as a baseline). In the second block, participants had to first adapt themselves to the typical interaction methods of wearable MR. Once that was done, each one of them has then proceeded to work with the MR implementation of the System. Both blocks have been subdivided into tasks in such a way that they can be directly compared between both implementations (which

from now on we label as web and MR). Briefly, Task 1 to 3 followed a guided creation process of a mission and they serve to make the user familiar with the interfaces. Task 4 on its behalf, asks the user to create a mission following a series of specifications in an unguided way. Below the purpose of each task and its individual details are presented.

- **Task 1, Creating a *Line inspection* and manipulating it to carry out a sweep with the drone along a building facade:** The purpose of this task is to offer a first contact with both implementations of the tool. To do so, the creation of a *line inspection* is demanded. This *inspection* has been selected due to it is one of the most basics (it is defined by only two positions on the map). Then, the user must move the starting and ending points of the line to the positions indicated by the facilitator on a physical map. It is also asked to set the right height of both ends at 10 meters above the ground. Figure 3.14 shows the map with the annotated positions as shown to the participants. The created line must go from point A to point B on that Figure. The time required for the completion of this task is stored for both implementations as well as the position introduced by the user.

Figure 3.14: The map with the annotated positions as shown to the participants.

- **Task 2, Creating a *Cylinder inspection* and manipulating it to take a panoramic view with the drone:** The objective of this task is to face the user to the creation and manipulation of a more advanced inspection. In this case, it is asked to create a cylinder one. This *inspection* is composed by two circles (one at the base of the cylinder and one at the top) that has to be moved independently. The cylinder must be positioned in point C of the map shown in Figure 3.14. Aside from that, it is required for the base circle to have an altitude of 5 meters above the ground the for the top circle to have an altitude of 15 meters. Finally, the radius of the cylinder (i.e. of both circles) must be set at 10 meters. Again, the completion time for this task in each version of the tool is stored. Before continuing, we would like to make a brief note regarding the process of adding this inspection. By default, when both tools are started, the map view appears centered on the same terrain that is shown to the user in the physical map. In fact, point C of the map in Figure 3.14 exactly matches the center of the map. This was done to facilitate the editing process without having to complicate with searches on the map. However, when adding the cylinder inspection, by default it is added in the center of the map, which, as already mentioned, coincides with point C. Therefore, users only have to vertically move both circles according to the details of the mission.
- **Task 3, Setting takeoff and landing points for the mission:** Once the *inspections* have been defined, the objective of this task is to show the user how to establish the takeoff and landing points. In the two previous tasks, the participant has been asked to manipulate the inspections mainly by using the 3-axis manipulator.

In the web-based tool, the process for setting both points is as follows. First, the user has to click the proper button, located on the mission details panel. Then, she has to click anywhere on the Google map, which will set the position. When implementing the Mixed Reality version of the tool, it was sought to translate this kind of *point-and-click* interaction, as faithfully as possible. Thus, in the AR version, once the user selects the right action located on the main panel, the process starts. Then, the participant only has to gaze with her head towards the desired position on the map and then to confirm that position by doing a *tap gesture*. With this Task, we can compare the performance differences in both implementations of the editor. Our hypothesis is that being a simple task, whose interaction have been adapted to be equal on both platforms, the times for each of them should be similar. Takeoff point must be positioned in point D of Figure 3.14, and the landing point should be located on point E. For this task on both platforms, the time required by each user is stored. After this Task has been completed, the mission will ready to be processed. Off-time, the user is asked to launch the calculations that

generate the *Flight Plan* and the *Trajectory*. Once these calculations are done, the results will be shown to the user. It is important to remind that the time measured for this task stops at the moment in which the order for starting the calculations is given. Hence, the execution and response times of the system are not taken into account.

- **Task 4, Creation of a mission autonomously according to a set of specifications:** Once the user already knows how to set the different components of a Mission in each platform, the objective of this task is to face them towards the creation of a whole mission without having the facilitator's guide. In order to do so, at the beginning of the tasks the facilitator provides a set of details of the mission: which inspections to use, their details and where should they be positioned as well as the takeoff and landing point positions. During this task, the participant must: i) define the inspections indicated with the appropriate parameters, ii) establish the takeoff and landing points and iii) launch the calculations of the Trajectory and the Flight Plan. During the task, the user may ask for help, and she is reminded of the details of each inspection if requested. For this task, the total time that the user needs from the beginning until the start of the trajectory calculations is stored.

Figure 3.15 shows a block diagram of the different stages of the test. The whole experiment has been designed as a within-subject one. A total of 64 trials (8 users x two different implementations (web-UI/MR) x 4 tasks) are presented. Dependent variables are the completion times, errors during test as well as an UEQ survey for each of the two implementations and a fatigue test associated with the MR implementation. For the scope of this thesis, the errors analyzed have been limited to the completeness of the task and precision errors.

3.5. Experimental results

The results of the user validations are summarized here. First, we will present the observations about the performance of the users, as well as the errors occurred during the test and in the tasks. Then, the results about the User Experience and fatigue tests will be addressed. Finally, we will present some annotations regarding the technology preference of the users and the strengths and weaknesses of both implementations. An initial analysis of the results was presented as a poster publication (Vaquero-Melchor and Bernardos, 2019a).

Figure 3.15: Stages of the user test

3.5.1. Performance and errors

First, we will address the information gathered about the times required by each user to complete all the task with both implementations: web and MR. These times are gathered in Table 3.1, and they are measured in seconds.

	Web				Mixed Reality			
	Task 1	Task 2	Task 3	Task 4	Task 1	Task 2	Task 3	Task 4
U1	42.72	43.39	12.53	85.46	95.34	48.93	14.83	146.85
U2	16	30	10	35.58	107	70	19	184
U3	70	48	19	65	240	65	26	175
U4	22.78	28	7.36	72	55.54	34.9	10.35	147.75
U6	31.34	20.44	9.6	71.86	175	248	16.97	285
U7	50.34	25.74	16.44	56.59	93.95	113.5	36.34	257.18
U8	16	76	7.92	56.48	179.61	71.36	14.21	188.75
U9	24.18	48.85	7.3	73	214	129	17.37	191.81
	μ : 33.79 σ : 17.97	μ : 40.05 σ : 16.88	μ : 11.26 σ : 4.10	μ : 64.49 σ : 14.07	μ : 145.05 σ : 61.64	μ : 97.58 σ : 63.90	μ : 19.38 σ : 7.66	μ : 197.04 σ : 46.17

Table 3.1: Times measured in seconds for each Task and user grouped by the implementation

Under each column of Table 3.1 the average and standard deviation for that task

and platform are shown. As a visual aid for interpretation, Figure 3.16 shows the same times that appear on the table as a parallel coordinates diagram.

Figure 3.16: Parallel coordinates representation of the times required by the users to complete each task. Time is measured in seconds.

With both sources of information, the following conclusions can be drawn. First, times required to perform a certain task in the Mixed Reality version, are in general greater than its Web-based counterpart. In the parallel coordinates diagram of Figure 3.16 this can be seen given that the slope between the two lines associated with the same task is always increasing. Thus, in a generalized way we can affirm that the tasks carried out with the AR version have required more time than those of the web version. There is only one exception for that affirmation. For task 2 in the Mixed Reality implementation, participant U8 took less time than for the web implementation. User 4 on the other hand, in general has been the fastest to complete each task.

A T-test of paired samples has been carried out to determine whether there is a statistically significant difference among the times required by the users to complete each Task in both implementations. Thereby, our null hypothesis is that there is no difference among the averages ($\alpha = 0.01$).

Regarding Task 1, the null hypothesis can be rejected ($t = 4.959; p - value = 0.0016; p < <$), and hence we can conclude that there is a statistically significant difference for those times.

For Task 2, we cannot reject the null hypothesis ($t = 2.115; p - value = 0.072$). Thus, we can state that there is not a significant difference in the mean times when defining the *cylinder inspection*. That makes us think that the more the user interacts with the system, the less difference in times will appear.

With respect to Task 3, the null hypothesis can be rejected ($t = 4.203; p - value = 0.004; p < <$). Thereby, there is a difference between the times and we can state that the *gaze-and-tap* strategy and the *point-and-click*, although they were designed to be similar, they have a different performance in term of time.

Finally, and regarding Task 4 (creating a mission in an unguided manner), we can reject the null hypothesis ($t = 6.9508; p - value = 0.000221; p < <$). Hence, we can conclude that there is a statistically significant difference between the mean times associated with each implementation.

In summary, there is a statistically significant difference in times required to complete Task 1, Task 3 and Task 4. However, for the creation of the cylinder inspection in Task 2, that difference disappears.

Taking into account that in Task 4 users have to create a complete mission and Tasks 1 to 3 consists on that same task but divided into pieces, it would be expected for Task 4 to have a total length in seconds similar to the sum of the times for Tasks 1 to 3. However, in general lines that does not happen with the Mixed Reality implementation: some times for Task 4, in particular for users U4 and U7, are smaller than the sum of the three others. This seems to imply that after having overcome the learning curve of the AR implementation with Tasks 1 to 3, upon reaching Task 4, the performance of some users is improved and that with use, it could improve even more. Other participants (e.g. U3, U6 and U9) have a greater discrepancy in defining the mission in an unguided way in the AR implementation. The sum of the task times can be seen in Table 3.2. Negative values in Table 3.2 indicate that the user took more time to create the mission in the unguided stage of the task than when doing it step by step.

At this point, a T-test of paired samples has been carried out in order to analyze the difference in terms of times among the aggregation of Task 1, Task 2 and Task 3 with respect to Task 4 for each implementation. Our null hypothesis is that there is not a statistically significant difference in the mean times of each block, given a certain implementation ($\alpha = 0.05$).

For the web implementation, the null hypothesis cannot be rejected ($t = -2.066; p - value = 0.07$) and hence there is not a difference in the mean times. In other words, on average, users take as long to complete individual tasks ($T1 + T2 + T3$) as they do to complete the entire mission autonomously ($T4$).

Results are similar with respect to the MR implementation ($t = -2.14; p -$

	WEB			Mixed Reality		
	T1+T2+T3	T4	Difference	T1+T2+T3	T4	Difference
U1	98.64	85.46	13.18	159	146.85	12
U2	56	35.58	20.42	196	184	12
U3	137	65	72	331	175	156
U4	58.14	72	-13.86	101	147.75	-47
U6	61.38	71.86	-10.48	439.97	285	155
U7	92.52	56.59	35.93	243.79	257.18	-13
U8	99.92	56.48	43.44	265.18	188.75	76
U9	80.33	73	7.33	360.37	191.81	169

Table 3.2: Difference between the sum of Tasks 1 to 3 times (creation of the mission in a guided way, step by step) with respect to Task 4 times (complete creation of a mission in an unguided way), measured in seconds.

$value = 0.6936$). Hence, the null hypothesis cannot be rejected.

Since the test gives similar results for both implementations we can argue that both have a similar behavior. Thus, in both there is not a statistically significant difference in terms of time between the creation of the piece-wise mission ($T1 + T2 + T3$) and the autonomous, unguided way ($T4$).

Below, the details of the errors made by the users during the tests will be presented. In this case, in addition to the issues manifested during the development of the test, we propose the analysis of the mission created in each of the versions. Thus, by separating the missions into two blocks, one for the web implementation and another for the AR implementation, we can analyze the degree of diversion with respect to the indications given by the facilitator. Let us remind that users were told the position and details of each of the inspections by means of a physical map (Figure 3.14). Thus, Figure 3.17 gathers the average results obtained for each implementation as compared with the provided details. Figure 3.18 shows the same information in 3D, adding the average altitude for each point. The color code is the same as for Figure 3.17: the original specification in blue color, the average web results in red and the average AR results in green.

In Figure 3.17 the initial specifications are marked with blue marks. The average positions obtained for the web implementation are represented with red marks and for the Mixed Reality implementation with green marks. The Figure gives the feeling that the results diverge more for the Mixed Reality implementation. As can be seen, the average position for the line points has a higher disparity than the web version. In other words, the point-and-click method used for defining the line segment in the web implementation is more accurate than the MR method. Moreover, the takeoff and landing points are very close, indicating similar performance in both

Figure 3.17: Comparison of the specifications given to users (blue marks) with respect to the average data obtained for each implementation: web (red marks) and AR (green marks).

implementations.

Tables 3.3 and 3.4 shows the error measured in meters made by users for each implementation and parameter of inspections. When calculating it, the following steps have been followed. First, the intermediate position for each parameter has been calculated by user, taking into account the average position between the features in the individual guided tasks (T1, T2 and T4) and the global unguided task (T4). Second, the distance between that position and the one indicated by the facilitator has been calculated. In the table, *cyl bot* and *cyl top* represent respectively the center of the bottom and top circles of the cylinder.

A T-test of paired samples has been carried out to determine if there is a statistically significant difference between the error committed by the participants for each parameter of the mission in each implementation. The data used to perform this test is the one presented in Tables 3.3 and 3.4. Thus, given a certain user, the total error in meters for a certain parameter of the mission is the difference between

Figure 3.18: Comparison of the specifications given to users (blue marks) with respect to the average data obtained for each implementation: web (red marks) and AR (green marks) in 3D.

Error in meters for the Web implementation						
User	line start	line end	cyl bot	cyl top	takeoff	landing
U1	2.26	4.07	0.888	2.653	6.859	6.956
U2	6.481	11.001	1.398	2.864	41.596	44.021
U3	1.51	2.325	0.888	2.653	10.067	9.611
U4	2.98	3.122	2.637	5.07	6.413	5.72
U6	0.476	2.305	3.689	3.689	5.668	6.273
U7	0.848	2.325	0.888	2.653	6.506	6.758
U8	5.634	7.252	2.572	2.572	7.075	6.7
U9	0.651	3.09	2.516	5.008	5.866	7.751

Table 3.3: Error measured in meters for the WEB implementation that compares the positions asked to the participants and the introduced one. For each user and parameter, the average position between the individual tasks (T1, T2 and T3) and the global task (T4) has been calculated.

the target position (as indicated by the mediator) and the position introduced in each task and each implementation. Hence, our null hypothesis is that there is no

Error in meters for the Mixed Reality implementation						
User	line start	line end	cyl bot	cyl top	takeoff	landing
U1	1.724	4.824	0.956	0.929	4.94	4.86
U2	40.61	43.026	0.614	3.2	8.754	9.721
U3	4.38	6.684	0.791	2.511	8.803	8.52
U4	82.804	84.136	0.889	2.476	6.333	9.582
U6	2.409	3.865	1.112	2.426	5.883	3.004
U7	4.063	3.691	0.869	0.962	4.245	6.007
U8	8.337	11.827	2.002	1.442	9.272	7.106
U9	2.821	5.094	0.877	2.631	5.197	0.452

Table 3.4: Error measured in meters for the MR implementation that compares the positions asked to the participants and the introduced one. For each user and parameter, the average position between the individual tasks (T1, T2 and T3) and the global task (T4) has been calculated.

difference in the mean errors ($\alpha = 0.01$).

Regarding the line inspection, *start* and *end* positions of the segment presents similar values for the T-test: ($t = 1.582$; $p - value = 0.15$; $p < <$) for the start and ($t = 1.59$; $p - value = 0.15$; $p < <$) for the end. Thereby, we cannot reject the null hypothesis and we can conclude that there is actually a statistically significant difference of the errors. Let us remind that the T-test compares the web and MR implementation given a certain parameter of the mission.

Regarding the parameters of the cylinder, the t-test results differ. For bot circle of the cylinder ($t = -2.6913$; $p - value = 0.03$), the null hypothesis cannot be rejected. This implies that there is not a significant difference in the errors of each implementation. However, with respect to the top circle ($t = -3.687$; $p - value = 0.0077$; $p < <$), the null hypothesis can be rejected and hence there is a statistically significant difference between the error obtained for both implementations. As an overview, it seems that errors for the bot side of the cylinder are more homogeneous in both implementations while top side presents more disparity. This may be related with the use of the 3-Axis manipulator, especially on the web implementation. Thus, the more distance the users has to displace the parameter, the more disparity may occur, and the precision could suffer.

Finally, regarding takeoff and landing positions, both parameters offer similar results for the T-test, which is coherent since both are defined in the same way. For the takeoff parameter ($t = -1.1254$; $p - value = 0.29$) the null hypothesis cannot be rejected and hence, there is a statistically significant different in the committed errors between both implementations. Second, the landing parameter ($t = -1.3085$; $p - value = 0.032$) has an equivalent behavior. In summary, even

if the *point-and-click* function of the web implementation was adapted to the *gaze-and-tap* of the MR implementation one, the mean committed errors are not the same for both implementations. A deeper study focused on the precision task could be performed to better understand this situation.

Once the errors have been analyzed, in the next Section we will present the details of the User Experience analysis and the results of the fatigue test.

3.5.2. User Experience and Fatigue during the validation

This Section has been divided in two different components. First, the results of the User Experience Questionnaire (UEQ) (Schrepp et al., 2017) will be presented and analyzed. then we will present the results of the fatigue test that was presented to the users. In particular, this fatigue test has been focused on the MR implementation.

Regarding the User Experience Questionnaire, the results will be presented as a comparison between both implementations used for the validation: the web-based interface and the Mixed Reality based one. After completing each block of tasks, users were asked to fill the associated UEQ. This short survey (2-3 minutes) serves to evaluate the user experience quality (UX) of a product or system. To perform this evaluation, a set of 26 pairs of contrary adjectives were presented, related with six different scales: *Attractiveness*, *Efficiency*, *Dependability*, *Novelty*, *Perspicuity* and *Stimulation*. Users were asked to choose a value from 1 to 7 for each one of the adjective pairs. Gathered data was injected into the analysis tool available online (ueq, 2020) and obtained results are as follows.

The first step that the analysis tool performs, consists of translating the values from [1,7] range to a [-3, 3] range. Thus, the scale mean values per person can be presented as in Table 3.5, where these values are presented per user with both implementations side by side. Mean and Standard deviation values are presented in the bottom side of the table.

	Web implementation						MR implementation					
	Attr	Pers	Effi	Depe	Stim	Nove	Attr	Pers	Effi	Depe	Stim	Nove
U1	1.50	1.50	1.50	1.50	0.75	0.75	1.33	1.25	1.75	1.25	1.50	2.25
U2	0.67	1.00	1.25	1.50	1.25	1.50	0.67	1.00	-0.75	0.00	1.00	2.00
U3	0.83	0.25	1.75	1.50	0.00	-0.25	1.67	1.50	2.50	1.50	1.00	0.50
U4	1.50	2.00	1.25	1.25	1.75	1.75	2.33	1.75	1.50	1.75	2.25	2.25
U6	1.33	2.00	1.75	1.75	1.00	0.00	1.00	1.00	0.75	1.25	1.00	1.25
U7	2.17	3.00	2.75	1.75	1.75	1.75	1.83	1.75	0.75	2.25	2.25	2.50
U8	1.67	2.75	3.00	2.50	0.50	0.00	1.67	2.00	1.50	1.50	2.00	1.50
U9	1.83	2.75	2.00	2.00	0.75	0.75	1.17	1.25	0.00	1.00	1.75	2.00
μ	1.43	1.90	1.90	1.71	0.96	0.78	1.45	1.4	1	1.31	1.59	1.78
σ	0.49	0.95	0.65	0.38	0.60	0.81	0.52	0.37	1.03	0.65	0.54	0.66

Table 3.5: UEQ scale means per participant and implementation. Values are in range [-3,3]. Attractiveness (Attr), Perspicuity (Pers), Efficiency (Effi), Dependency (Depe), Stimulation (Stim), Novelty (Nove)

Following the same line of thought as until now, a T-test of paired samples has been carried out to analyze whether there is a statistically significant difference between the UEQ results of both implementations. For all the UEQ parameters, the null hypothesis is that no difference exists ($\alpha = 0.01$). In order to facilitate the reading, Table 3.6 shows the results of the T-test for each UEQ parameter. Let us remind that the web implementation is compared to the MR one.

Attribute	T-test results ($\alpha = 0.01$)
Attractiveness	$(t = 0.11; p - value = 0.915)$
Perspicuity	$(t = -1.52; p - value = 0.1716)$
Dependability	$(t = -1.57; p - value = 0.16)$
Efficiency	$(t = -2.21; p - value = 0.0622)$
Stimulation	$(t = 3.11; p - value = 0.016)$
Novelty	$(t = 6.69; p - value = 0.00027; p < <)$

Table 3.6: T-test results for the UEQ parameters

Results of Table 3.6 shows that the null hypothesis can only be rejected for the *Novelty* attribute. In that case, there is statistically significant difference between both implementations, which is coherent with the scores given by the users. Thus, the Mixed Reality implementation of the editor seemed more novelty in general for them. Regarding the other parameters and given that the null hypothesis cannot be rejected for them, we can state that in general lines, both implementations had a similar score in each category.

Next, in Table 3.7 we present each pair of adjectives and the mean value as well as the standard deviation for each one of the two implementations. Each pair coincides with the items included in the UEQ survey and they are presented in the same order. Besides that, the category to which each pair belongs to is presented with a visual color indication as presented by the analysis tool.

Item	Left	Right	Scale	Web		MR	
				Mean	σ .	Mean	σ .
1	Annoying	Enjoyable	Attr	1.6	1.3	1.6	0.9
2	Not understandable	Understandable	Pers	2.0	1.3	2.1	0.8
3	Creative	Dull	Nove	1.1	1.1	1.8	1.2
4	Easy to learn	Difficult to learn	Pers	2.3	0.9	1.8	0.7
5	Valuable	Inferior	Stim	1.6	1.1	1.6	0.5
6	Boring	Exciting	Stim	0.3	0.9	1.6	0.7
7	Not interesting	Interesting	Stim	1.1	0.6	1.8	0.9
8	Unpredictable	Predictable	Depe	1.8	0.9	1.4	0.5
9	Fast	Slow	Effi	2.1	0.6	0.4	1.6
10	Inventive	Conventional	Nove	0.9	0.8	1.9	0.6
11	Obstructive	Supportive	Depe	1.6	0.5	0.9	1.4
12	Good	Bad	Attr	2.1	0.6	1.9	0.6
13	Complicated	Easy	Pers	1.8	0.9	0.8	1.3
14	Unlikable	Pleasing	Attr	1.5	0.5	1.1	0.6
15	Usual	Leading edge	Nove	0.5	0.9	1.4	0.7
16	Unpleasant	Pleasant	Attr	1.5	0.5	1.3	0.5
17	Secure	Not secure	Depe	1.4	0.7	1.3	1.0
18	Motivating	Demotivating	Stim	0.9	0.6	1.4	1.2
19	Meets expectations	Does not meet expectations	Depe	2.1	0.4	1.8	0.7
20	Inefficient	Efficient	Effi	2.0	0.8	0.9	1.2
21	Clear	Confusing	Pers	1.6	1.3	1.1	0.8
22	Impractical	Practical	Effi	1.9	0.8	0.8	1.3
23	Organized	Cluttered	Effi	1.6	0.9	2.0	0.8
24	Attractive	Unattractive	Attr	0.8	0.9	1.8	0.9
25	Friendly	Unfriendly	Attr	1.1	1.0	1.1	0.8
26	Conservative	Innovative	Nove	0.6	1.2	2.1	0.8

Table 3.7: Mean and standard deviation values for each pair of adjectives and their related scale.

These same average values presented in table 3.7 are represented now in Figure 3.19 in a visual manner. The results for both implementations are shown at the same time. Again, the web-based tool is represented by the yellow color and the Mixed Reality implementation with blue color. In this case each pair of adjectives has been reordered to show the *bad* one at the left and the *good* one at the right.

Figure 3.19: Comparison of the mean value per item in both implementations

As Figure 3.19 shows, all pairs have positive values on average for both implementations, which indicates a good performance. Apart from that, there are certain

features of the Figure that stand out. For example, for the pair of adjectives *Boring-Exciting*, users have indicated on average that the AR implementation is much more exciting, which coincides with the comments they made during the tests. On the contrary, in the *Slow-Fast* category, users have indicated that the Web version is much faster than the AR implementation. As a last comment, with respect to the *Conservative-innovative* category, the implementation that raises from the comparison is the Mixed Reality implementation. Thus, this version seems much more innovative to users. According to the comments made by the users during the test, it is something different from what they are not used to.

Figures 3.20 and 3.21 gathers the mean and deviation values for each category as presented by the UEQ analysis tool. The interpretation of these graphs is as follows. For a certain category if the mean value and the deviation range fits in the green area, the analyzed system is below the average systems that was used for the creation of the UEQ survey. If one indicator belongs to the yellow area, the system is on the average. As can be seen in both Figures, the Mixed Reality implementation excels in the categories of *Attractiveness*, *Perspicuity*, *Stimulation* and *Novelty*. However, regarding *Efficiency*, the result is not so remarkable. The web implementation on its behalf presents a much better *Efficiency* and *Dependability* results. These observations coincide with the original hypothesis that users would perform better in the Web version. Hence, this version would be taken as the baseline. The details of the original systems used for the creation of the UEQ survey and against each system is faced to, are not well defined. Therefore, these comparisons should be taken from an illustrative point of view.

Figure 3.20: Mean values and deviation for each UEQ category in the Web implementation

Figure 3.21: Mean values and deviation for each UEQ category in the Mixed Reality-based implementation

To conclude with this Section, the results of the fatigue test will be presented here. As a brief reminder, this test was provided to the participants after completing the four tasks of the Mixed Reality implementation and before the UEQ questionnaire. The fatigue survey is based on the questions of the ISO 9241-9:2000, in particular the annex C. Users were asked to rate between 1 and 7 a set of categories, being 1 associated with worse results and 7 with good ones. Figure 3.22 gathers the average values given by the participants to each category, as well as one standard deviation for each one of them.

Figure 3.22: Average and standard deviation values for each category of the fatigue test. Values range from 1 (if the users experienced strong problems in that category) to 7 (where they had no problem at all)

As can be seen in Figure 3.22, there have not been many fatigue problems among participants. The results seem to indicate that there is a great discrepancy regarding neck fatigue, which coincides with the comments made by users during the development of the tests. Finally, the accuracy of the interaction in the AR implementation has medium-low values. This coincides with the issues expressed by some participants during the test, such as that they had to repeat some gestures so that they could be detected well or that they were not able to select a small inspection with the gazing.

3.5.3. Comments on technology preference and strengths and weaknesses of both systems

After having confronted users with both versions of the tool, the following statements were done by them. First, they were asked about which platform did they think that helped the most to understand the 3D results. There is a general consensus that the AR version helped *absolutely more (5 points over 5, as indicated by four over eight participants)* and that it helped *some more (4/5)* than the web-UI (4 other participants). Thus, none of the participants indicated that the web-UI would help them more to understand the 3D results.

When asked about which of the two versions they found the most *user friendly*, it seems that neither platform stood out. Four of the participants did not opt for any of the options (3/5). Three users found the web implementation *somewhat more* user friendly than the AR version and the last participant indicated that the AR version was *somewhat (4/5)* more user-friendly than the other.

Another question that was asked to the participants was related to the relative ease of use between the two versions. Which of the two platforms did they find the easiest to use? In this case, the web implementation was easier for users in general. One participant indicated that it was *absolutely the easiest (1/5)* to use and other five said it was *some more (2/5)* than the AR version. Only one user said he did not find differences between the two versions (3/5) and only one found the AR version *some more (4/5)* easier to use than the web-UI. These results coincide with the design hypothesis of taking the web implementation as the baseline, since in general participants are more accustomed to working in conventional environments such as the web.

However, the following asked question contrasts with these results. Which approach did they prefer, the AR implementation or the web implementation? The Mixed Reality-based implementation has been chosen by 5 of the participants as the preferred one. Among the arguments given by these users is that it is more novel, “*it helps to understand the space*” and that “*it allows you to get a better idea of the defined mission*”. The main reason for the other 3 users to choose the web version is that it was faster and more accurate than the AR one. Note the comment of one of the participants who indicated that “*in fact he prefers web for creating missions (higher accuracy and less fatigue) and the AR system to visualize the results*”.

Among the features that participants liked most about the web implementation are for example that *it is easy to use* and *very easy to understand*. For these participants, each step of the whole process was also simple and self-contained, even obvious after the procedure has been explained to them. On the other hand, three participants stated issues like it was not easy for them to undo a mistake. Therefore, it is proposed to add this feature in future system implementations. Two participants also stated slight difficulties at the time of setting the height of the inspection

points by using the three-axis manipulator. This is because according to them this controller is not as precise as directly entering the required value in a text field. In general, the integrated 3D view was found very visual for all the participants. However, one of them stated that it was not clear when to use the 2D view and when to use the 3D view. After all participants have finished the whole test, three of them asked for the 2D map and the 3D view of the web implementation to be integrated in a similar way as in the MR-based version. That statement implies that although they are used to work with 2D maps like Google’s one, there is an open door for other different possibilities.

Regarding the Mixed Reality implementation, a set of questions about the interaction with AR content was presented to the participants. First, they were asked about the process of gazing to select. Two participants found that process *very comfortable* (5/5) and four others *comfortable* (4/5). Remaining two participants found it just *normal* (3/5). That indicates that in general users did not have problems during the gazing process. Among the given reasons is that the gazing process replicates the natural movement of the head towards looking for an object in the vision range. However, three participants have stated some accuracy problems when selecting small 3D elements on the map. These users in particular did not use the map controls to change the map zoom level and they worked always with the same one.

The next question was related with the *Tap* gesture since it is a vital action for the whole process. In that direction, 6 participants found the gesture *comfortable* (4/5), one other just *normal* (3/5) and the last one found it *very uncomfortable* (1/5). Those participants who found the *Tap* gesture comfortable stated that it was easy to use and that it is a good metaphor of the process of clicking with the mouse. Those participants that were not comfortable with the gesture had problems regarding the gesture not being properly detected by the AR device (the HoloLens). One of them also said that he found it uncomfortable because it forces to have the arm raised during long periods of time, which he did not find natural.

Next, participants were asked about the *drag* gesture. Five users found it just *normal* (3/5), two others found it comfortable (4/5) and the last one found it *very comfortable* (5/5). In favor of the *drag* gesture is the fact that it is intuitive once explained. One of the users was faced with the problem that when moving the hand while making the gesture, it went outside the detection range of the AR device, which caused unwanted movements. With this in mind, it is possible to integrate some type of control into the application that warns the user that it is going out of the interaction range and that it blocks movement.

Next, a set of questions about the arrangement of AR elements in space was proposed to the participants. These questions have been asked for the AR version and not for the web, since in the first case the elements can be located anywhere in the physical world, that is, they are not constrained to a window. In particular,

a question about the inspection palette was asked. In the web implementation, the user had to choose the inspection to be added from a drop-down menu. However, in the AR implementation that element is substituted by a *palette* that was presented in front of the participants. That palette shows the inspections in 3D, so the user could directly see the shape of each inspection. The palette also follows the user gaze during the inspection adding process until any inspection is selected. When asked if they found it useful, 4 users have indicated that it is *very useful* (5/5) and 2 users just *helpful* (4/5). One user indicated that the palette was *not useful* (2/5) and another that it was *completely useless* (1/5). In favor of the palette, three users indicated that it is *intuitive* and four participants said that it was easy to use since its behavior was similar to a drop-down. From these comments we can verify that although there are certain elements that users have well learned from their experience with traditional systems, they can be adapted to the AR context taking advantage of the visibility provided by this technology. In this case a 2D list of inspection names has been translated to a set of 3D figures that the user can interpret at a glance.

In addition to the inspection palette, users were asked about the layout of the buttons and actions. Five participants did prefer that the buttons appear on an already positioned, fixed panel, as in the case of the gray one of Figure 3.10. One participant preferred the buttons to be floating in the air and another of them find both ways useful. Between the reason to show them floating in the air are that this mode is novel and that 2D panels look like “*aliens*” in the context of Augmented and Mixed Reality, that is, somewhat out of place. Supporters of grouping the buttons on a panel said that inside the panel, the elements are better organized and it is easier to find what they were looking for than if they have to go looking in the air and remember where everything was.

The features of the Mixed Reality implementation that participants liked most are that the 3D experience was *impressive* (5 users) and *enjoyable* (7 users). In general, what they liked most is the three-dimensional component of the tool. However, what they liked least about this version is generally related with the HMD device in use (the HoloLens). Three users indicated that it was *not comfortable* and two that it had a *very small field of vision*. They also indicated that if gestures were not handled correctly, the experience became slower and worsened. In general, the AR version seemed less accurate than the Web implementation because although they had the textual information of the inspections in front (i.e. the numerical values of the positions and altitudes), the relationship between gestures and distances was different than in the web version. As a general conclusion, the vast majority of problems identified by users can be related with technology limitations: *reduced field of view*, *high weight of the AR device*, and issues during the *gesture detection*).

3.6. Summary

In this Chapter, we have presented our study on the effectiveness of interaction in wearable AR systems. This has been done based on a particular use case: the definition of drone flights. Thus, two implementations of an editor with equivalent functionalities have been presented. First, the web implementation, which has been used as a representative of conventional platforms and which has been considered as a baseline (due to the familiarity that user present with this type of interface). Second, Holo-MIS, an adaptation of the editor to the wearable AR context, particularly for HoloLens, has been unveiled. Having presented the details of both implementations, the next step has been to show the results of the validation with users. This validation has been carried out in a comparative way, so that the breakdown into Tasks associated with each implementation was comparable. From the obtained results, we can make the following statements.

First, despite the technical limitations of the AR implementation of the editor and the efficiency reduction, multiple users have expressed their preference for this version. Those technical limitations to which we refer are mainly related with the restrictions on the display of AR content presented by the device, as well as its size.

Second, the analysis of the times associated to the Tasks of each implementation indicates that, in general, those Tasks of the AR implementation have required more time to be completed. However, throughout the tests, there has been an improvement in terms of performance for some of the participants. This leads us to the conclusion that with continued use by the participant, the different in times between both implementations could be reduced.

Third, regarding the precision of the Task carried out by the users, the web implementation has been the one with the best results. This in turn is closely related to the way of interaction and data entry preferred by the users. Thus, when manipulating the elements that defines the flights, there are two forms for doing so: a) by *textually* entering the values and b) by using visual manipulators that allow the participant to move the associated elements. Thereby, when using the web implementation, users have preferred in general to introduce the data by *textual* means when following the instructions of the facilitator. Hence, the precision is greater. However, in the AR implementation, textual data entry is relegated to second place in favor of visual manipulation. In this case, the participants must hand gestures to displace the elements to the desired positions, which together with the intrinsic 3D perspective of the AR visualization has made the made errors somewhat greater.

Fourth and regarding the User Experience, there is not a great significant difference between both implementations. By analyzing the data from the UEQ survey carried out, it can be seen how both implementations stand out in some category,

keeping both in general on a par. For example, users found in general more *exciting* to use the AR version than the web one. However, the web implementation has been *faster* to use. Additionally, the MR implementation has been more *novelty* for the participants than the web one, which the users have found more *conventional*. As a last note regarding user Experience, on average users found the web implementation more efficient, and both in general have been equally attractive.

Fifth and related with the preference of the users towards one implementation or the other, the participants have indicated in general that the AR version was the one that most helped them to understand the 3D results. This fact is of great importance, since it implies that those applications with high 3D content load used nowadays in conventional platforms, they may benefit from the wearable-AR features. A possible example of this are 3D editors like Blender (project, 2020) or Autodesk Maya (Autodesk, 2020).

Sixth, users found the web version easier to use, which coincides with our design hypothesis. However, as was mentioned earlier, a great number of participants have stated their preference towards the MR implementation.

Seventh, some users have stated problems while using hand gestures to operate with the AR implementation. In general, this was due to the HoloLens not detecting their hand properly. For these participants, the bad experience with this type of control has had a negative impact on their perception of the MR implementation.

Eighth, facing future implementations of applications based on wearable AR, the arrangement of digital content in the physical space has been studied. On the one hand, some traditional elements like dropdowns, can be improved with the integration of 3D AR assets. On the other hand, most users preferred that the buttons, labels and other AR elements appear grouped in the same element (a panel), so that they could find them at a glance without having to look around and remember where each one was.

Chapter 4

AUGMENTED REALITY BASED COLLABORATIVE EXPERIENCES: ANALYSIS AND TAXONOMY

Taxonomy is described sometimes
as a science and sometimes as an
art, but really it's a battleground.

Bill Bryson, A short history of
nearly everything

4.1. Context and Motivation

The aim of this Chapter is two-fold. On the one side, we will present a theoretical analysis of the collaboration in Augmented and Mixed Reality. Thus, a definition of collaboration and an analysis of its main characteristics will be presented first. On top of both elements we will present the concept of *collaboration layers* which, when combined, will shape the AR and MR experiences. This analysis has been performed by taking into account the current state-of-the-art of collaborative AR systems.

The second objective of this Chapter is to present a taxonomy to classify collaborative AR and MR systems and experiences. Some authors have already focused on the analysis of collaborative AR systems. For example, (de Belen et al., 2019)

presents a systematic review of the current state of collaborative Mixed Reality technologies. In order to characterize the systems, the authors rely on three features: i) the scope of the systems (e.g. industry, gaming, teaching ...), ii) the type of device used to display the AR content and iii) the location of the participants (collocated, remote or variable).

The taxonomy presented in this Chapter would serve both, as a tool for characterizing systems and as a reference guide for future developments. Furthermore, since there are many systems proposed in the literature, we believe that the presented taxonomy can speed up the search for references.

4.2. Analysis of the Collaboration in AR and MR Context

4.2.1. Collaboration definition

A possible meaning of “to collaborate” is “to work jointly with others or together” (mer, 2019). By applying this definition to the AR context, the following conclusions can be drawn. First and most obvious, in order to perform a collaboration, multiple users are required. Second, this collaboration implies a purpose, i.e. the participants have one or many objectives that must be satisfied. These objectives may be common for all the participants or they may be individual to each of them. Third, some actions must be performed by the users to achieve the common goal. Fourth and finally, AR assets will be involved into the collaboration, which will show visual elements to users and may be interactive. In this Section, we aim at analyzing the concept of collaboration in Augmented and Mixed Reality to define a set of features or characteristics that have to be managed in an AR-oriented collaboration architecture.

4.2.2. Collaboration characteristics

Apart from “multiple users”, “individual vs. common objectives” or “course of actions”, there are some other aspects that need to be considered when characterizing collaborative Augmented Reality and Mixed Reality experiences. In this Section we present in more detail those aspects and how they influence the design of the experiences. Three are the main characteristics of the collaboration:

Number of participants in the collaboration: This aspect is one of the easiest to reference given that the number of these participants may affect the design of collaborative AR systems.

Individual vs. common service goals: Depending on the application, there may be one or several objectives common to all users. However, it is also possible that each user has one or more personal objectives, which may not coincide with the common ones. As an example, let us suppose a multiplayer game based on AR. Players in a team may be willing to win, while each player individually may be targeting to improve his personal score. Another example could be a collaborative BIM-based (Building Information Modeling) tool in the architecture context. Several participants may be working together to design a building, with each user focused on different components (e.g. one may arrange the electricity system while another may place the piping system). In general, each of them can work independently in their own copy of the building. However, in specific occasions they may have to work together and even reach agreements. The scenario of competition can be seen as a sub-case of the collaboration one, in which the individual objectives of the users are faced.

Operational features to consider when implementing collaboration: To achieve the collaboration goal, some actions have to be performed by the users. Those actions are mainly related with the interaction with the AR content and the communication with the other participants. The implementation of courses of action in collaborative AR require to determine aspects such as time and space management, interaction methods, visualization and, all in all, the collaboration model used for orchestrating all the collaboration. Next, we expose each one of these operational features in detail:

- *Real time vs. shift collaboration:* Time management restrictions are important when defining collaborative AR applications. It is possible for example that all the users are working with the AR content at the same time, which could be seen like *real time*. However, it is also possible that they work *in shifts* and that they do not even coincide in time. In any case, the state of the AR assets must be consistent for all the participants.
- *Shared physical space vs. remote:* Whenever all participants of the AR collaboration experiences are located in the same physical space, it is usually said that they are co-located, and they are working “locally”. On the contrary, if the participants do not share the same space, it is usually said that they work “remotely”. It is also possible to have a mixed situation in which some users are co-located and they are working on remote with another one.
- *Interaction methods:* According to the AR platform, interaction with the virtual content is carried out in a different manner. For example, in hand-held devices like smartphones, the main interaction method is touch-based. However, in case the device is an AR Head Mounted Display (HMD), interaction is

usually carried out by hand gestures and/or voice commands. Thus, the type of interaction will condition the design of the means to implement the course of action.

- *Information exchange channel*: In order to collaborate, participants need to exchange information. Visual means, including annotations or check lists, speech or combined strategies may be used.
- *Content alignment and profiled visualization*: Digital elements may have different views tailored to the specificities of different users. Thus, the visual characteristics presented by AR assets may not be exactly the same for all users. Some examples of these characteristics are the object position, scale or rotation and more advanced ones like color or shape. For example, let us consider a 3D cube located on a real, physical table. If the AR content is *aligned*, the cube will be seen by all participants in the same exact position (setting aside possible drift errors) and with the same features. On the contrary, if the AR content is *not aligned*, each user will see that same cube with the same features but located on a different position. For example, one may see it on the table while another user may see it on the floor. However, both representations will refer to the same cube. In this case, being aligned or not refers not only to the position, but also to characteristics such as rotation and scale (e.g. it is possible that each user sees the cube in the same position but with different sizes). There is another possibility: that the AR content exposed to each user is different, even if they must work together. In this situation we say that the AR content has been *profiled* to the user; in other words, that it has been adapted to the user profile. Somewhere in the middle, it is possible for several users to share some *aligned* AR elements while only a subset of them can see some profiled content.
- *Collaboration model*: The collaboration model defines how users organize themselves to visualize and interact with AR content. This organization may come from themselves (they agree by speech), due to implemented control mechanisms (e.g. if the element of interaction has to be transferred) or by high-level rules (e.g. it is defined that they have to work in shifts). Due to the importance of this concept for the present within the context of this Thesis, Section 4.2.3 will analyze how the models are integrated into the collaboration as well as some illustrative examples of them.

4.2.3. Collaboration layers

Based on the characteristics presented in the previous Section, we next abstract three service layers where collaboration constructs are managed. Thus, from less to

more enriched interactive scenarios, the layers are: i) the visualization of the AR content, ii) the interaction with the AR elements and the iii) high level orchestration of the others two. Figure 4.1 shows an outline of the three layers and how they would be instantiated within an example application: the voxel-based collaborative game that will be later detailed in Section 5.6.2. Below, each layer is explained in detail.

Figure 4.1: Collaboration Layers

Layer 0: Visualization. This is the most basic level, as we understand that there cannot be any AR collaboration if users cannot share AR digital elements, hence we have labeled it as layer 0. In this layer, both *information sharing* and *content alignment* of the assets are involved. First of all, it is possible for all users to share exactly the same AR elements, that is, that everyone can see exactly the same. However, it is also possible for the sharing process to be *filtered* according to some rules (e.g. based on hierarchical roles where the user who is at the top of the hierarchy has access to all the information levels while others can only see specific levels depending on the permissions allowed to their position in the hierarchy). Second, it is also possible that the *shared* AR elements present the same characteristics (e.g. position in the real world, rotation or scale) for all the users, or on the contrary that they are profiled for each one. At this point, the organization of the collaboration and the AR components fully relies on the participants. Blocking and interaction have to be accorded by the users during their own collaboration. In this case, speech may be a useful way to regulate collaboration. In other words, the collaboration model is not restricted.

Layer 1: Interaction. Once the digital elements are shared among participants, the next step is to allow the users to *interact* with them. With that, we understand that the AR elements are able to detect users' inputs and to react according to them. Let us suppose an AR cube that is positioned in front of the user, who is using a smartphone to visualize it. When the user *touches* the cube on the screen, it should be able to detect that touch and if necessary, to execute some application logic such as changing its color. It is also possible that instead of touching, the user performs a *drag* gesture which could have other associated logic, such as

making the cube rotate on itself. As can be seen, when using the same AR device there can be different ways of interacting with the AR elements. Likewise, for each type of AR device, the forms of interaction with the AR content will be adapted to them. As was the case at layer 0, the collaboration model does not present any restrictions. Users will have to organize themselves to regulate the interaction. To automatically orchestrate the collaboration, the next layer emerges.

Layer 2: Collaboration Management. With layer 1, participants are given the ability to interact with AR content. It may be the case that several users decide to interact with the same element at the same time. In principle, there is nothing that prohibits or regulates this situation. How should a hypothetical AR cube react when one user is moving it to go up and another user is commanding it down? In this case the logic of the application is wrapped with an external element, a *Collaboration Model* (CM). A *Collaboration Model* is compounded by a set of rules that control all the elements of the collaboration, which can be both visualization and interaction capabilities. The most basic *Collaboration Model* would be the *Unconstrained* one, in which no rules are specified. Thus, even in the case that the collaboration management is not explicitly done, that situation can be associated to a *Collaboration model*. Another example may be the use of a *Turn-based* model. In this case, the interaction with the AR elements is restricted only to the user whose turn it is.

In summary, by defining the rules that shape a CM, it is possible to control the flow of an application without modifying its basic logic. An important fact is that the same set of rules (i.e. the same Collaboration Model) can be applied to completely different applications. In other words, *collaboration models* are application independent and can be reused. Below we propose five types of high-level collaboration models extracted from the literature to illustrate this concept.

- I) *Turn-based model*: The interaction with the AR elements is managed by using a token. Only the user who has the token (therefore in her turn) can interact with the virtual objects. The interactions of other users with the AR content are in principle discarded.
- II) *Ownership-based model*: In this collaboration model, each digital element has an associated owner. Thus, the visibility and interaction with each element may be restricted only to its owner. It is also possible that the elements are visible for all or a group of users with only the interaction being limited.
- III) *Layer-based model*: This collaboration model is characterized by having one or many *layers*, to which the AR objects are linked to. One digital element may be present in several of those layers. Users on their side will have access to

certain layers and they will be able to visualize and interact with its associated digital elements.

- iv) *Hierarchy-based model*: In this case users are organized in a tree-based hierarchical structure. On one hand, the user at the top of the hierarchy has access to all AR elements, both for interaction and visualization. Changes performed by that user will always prevail over those initiated at a lower level of the hierarchy. On the other hand, users located in the lower branches of the tree will have limited both their visibility and their ability to interact. It is reasonable to assume that it will be the participants themselves who define these restrictions for their subordinates.
- v) *Unconstrained model*: In this collaboration model there are no restrictions. All user interactions are processed and no control over the visibility of AR elements is established. In this case a FIFO (First In, First Out) policy will be applied.

It is important to note that these five models gather the main and common collaboration schemes we have identified, and that there may be more possibilities, since the rules themselves are what define each of the models.

4.3. Presenting the Taxonomy

Having presented the main components of the collaborative AR applications and experiences, it is the turn of presenting an analysis framework to taxonomize those systems. The main objective is being able of characterizing any system that fits to the definition exposed in Section 4.2.1 by its main features, to facilitate comparison. Additionally, this taxonomy could be seen as a reference guide on systems to consult for future developments. The categories of the taxonomy are listed next:

- Limit number of participants
- Location of the participants
- Collaboration model
- AR content visualization Platform
- Interaction Method
- Method for aligning the AR content
- Integration of real-world information

In the following subsections we will explain each one of these aspects in detail. Later, in the last part of this Chapter we combine all these features in the form of a table in which the researches mentioned in the state of the art of Chapter 2 are analyzed.

4.3.1. Limit number of participants

This feature is the most direct to analyze since only one of two possibilities must be considered. Either the experience has been designed a) to involve only one user or b) to involve several participants. In the first case, we understand that there would be no collaboration since by definition several users are required for it to exist. In the event that the limit number of users is greater than one, the theoretical limit that could be reached should be indicated.

4.3.2. Location of the participants

This characteristic considers the physical locations of the participants involved into the collaborative experience. Following the same nomenclature used by (Lukosch et al., 2015), mainly the collaboration can be divided in the following groups: *face-to-face*, *remote* and *mixed*. Figure 4.2 shows a scheme of the three groups and their details will be presented below.

Figure 4.2: Visual representation of local, remote and mixed locations

- (a) Face-to-Face: In this group are included those systems in which all the users share the same physical space, even if the same digital information is not

accessible for all of them. Some examples of this kind of collaboration can be found in the VITA project, by (Benko et al., 2004) and DARCC, by (Hammad et al., 2009). In this second example, users are co-located near the construction site and they can control a virtual crane while they can see what other users do.

- (b) Remote: Those systems in which the users do not share the same physical space are grouped here. Some works reference two main stakeholders involved in the collaboration process, a *local user*’ and a *remote user*’. Whenever both types of participants interact at the same time with the system and they are located in different places, which one is the remote and which one is the local user? For most authors, the local user usually is the one that interacts with the system, while the remote one communicates real time with the user on the field. But nevertheless, when both users interact with the system, this definition falls short. For us, the *local user* is the one who provides the information of the physical environment to the collaboration (which makes even more sense in the Mixed Reality context). But nevertheless, if all the participants contribute with information of the physical environment, all of them would be in their own *local context*, with the other users located remotely. In this group, unlike the face-to-face scenario, (in general) users will not be able to see what others do. They will only see the results of the manipulations to the virtual content. Additionally, a communication may be established among the participants. (Poelman et al., 2012), (Datcu et al., 2014a) and (Gauglitz et al., 2014) are some examples of remote AR collaboration scenarios.
- (c) Mixed location: In the two previous cases (*local* and *remote* collaboration), there is a consensus on how all the users communicate through the system. Either all of them share the physical location (*local* or face-to-face) or each one of them are located in different physical spaces (*remote*). In the *Mixed* location some users are collaborating face-to-face and some of them are located remotely. For example, two ground users could be inspecting a physical element annotated via AR assets while other remote worker is watching a digital representation of that element and the same AR content. In this case we must bear in mind that the infrastructure for the face-to-face may present a different performance than the infrastructure for the remote systems (e.g. different latency times and different deployment technologies). There are no examples of this type of collaboration in the literature, however, we believe that in the future more and more systems with this type of collaboration will appear.

4.3.3. Collaboration Model

The collaboration model used for orchestrating the collaboration can be implemented in different ways. On the one hand, the interaction method itself can influence the implemented model. An example of this situation is the use of a single physical device to interact with the system, such as a joystick. In this case, users would have to give up the device to interact *in turns*, so that several of them could not interact at the same time.

On the other hand, the collaboration model could be managed from a high-level point of view, being able to apply one type of model or another depending on the circumstances. For example, the Studierstube system (Schmalstieg et al., 2002) implements a model based on information layers. As already indicated in Section 4.2.3, this model could be used as a basis to implement others such as hierarchical or property.

Apart from that, the chosen collaboration model manages the symmetry between the capabilities of the participants in the system. Thus, it could be possible to give more visualization or interaction capabilities to certain users based on their privileges (asymmetric) or to grant the same capabilities to all of them (symmetric).

Given that the possible collaboration models are theoretically infinite, when categorizing systems using this taxonomy we are going to stick to the five models exposed in Section 4.2.3: *unconstrained*, *turn-based*, *layer-based*, *ownership-based* and *hierarchy-based*

4.3.4. AR content visualization platform

This characteristic of the collaboration refers to the device used to present the digital information to the users. In this case it is possible for a system to offer more than one type of device at the same time. Next, a group of the most common visualization platforms are presented:

- (a) Head Mounted Display (HMD): This kind of devices are worn by the users in their heads which keeps the user's hand free. Microsoft HoloLens (hol, 2018) are nowadays one of the most popular devices in this group, although they are not alone. Meta 2 (met) and Magic Leap (mag) come stomping and are presented as a more accessible alternative than the HoloLens in terms of costs (\$1,495 for the Meta 2 and \$2,295 for the Magic leap One Creator Edition facing the \$3,000 of the HoloLens). More recently, the HoloLens 2 have been presented, having a launch prize of \$3,500. Meanwhile HoloLens are a standalone device, Meta 2 requires to be connected to a traditional PC and Magic Leap has his own small hip-mounted computer. More details about these devices can be found in Chapter 2. This kind of devices tend to offer the interaction with the AR content by means of gestures and voice commands.

- (b) Hand Held: Smartphones and tablets are representative examples of this group. In this case the user holds the display with his hands. Thus, the digital information appears superimposed to a video feed obtained with the device's camera in the device. Usually, this kind of devices offers interaction by means of touches on the screen. However, other functionalities are also offered, like shaking the device or tilting it.
- (c) Spatially Augmented Reality (SAR): In this kind of systems, the digital information is projected over physical elements. The core principle of SAR is that the projected information is decoupled from the users of the system. Hence, SAR scales naturally up to group of users, which enables collocated collaboration between them.
- (d) Hybrid or Multi-device: There may be some cases in which all the participants of the collaboration are not using the same AR device. Some of them for example may use smartphones and some others may use HMD. This group is of special importance at the present time, as it is increasingly sought to integrate different types of devices in collaborative systems. Ideally, as an example, an Android device and an iOS one should be able to work together without problems. This gives the system a greater potential of reaching more users and platforms, since each platform offers different opportunities and advantages.

4.3.5. Interaction method

This characteristic refers to the main interaction method used by the participants to interact with the system. In principle, it is closely related to the device used, but this is not mandatory. Besides that, is also possible to combine multiple devices or techniques to provide a multi-modal interaction. The most common methods present in the literature are:

- (a) Touchable screen: The use of tactile devices is usually related with hand-held displays. This kind of interaction is easy to use for the users since they are already used to working that way. The most basic example of this is selecting an item by tapping on the screen or moving a digital element by dragging.
- (b) Hand gestures: Some AR platforms allows the user to use gestures to manipulate the digital elements. As an example, interaction with HoloLens heavily relies on the use of gestures such as tapping, which is understood as a “click” metaphor. There are other possibilities like pinch and drag and to detect the opening/closing gesture of the hand (this is called bloom gesture in the HoloLens system). Other systems rely on other more natural gestures such as

pointing.

Hand gestures can be combined in some scenarios with haptic feedback to improve the interaction. This can be used, for example, to associate physical sensations with a virtual object or providing haptic clues the user with haptic patterns.

- (c) Joysticks: This kind of interaction had a higher popularity in the earlier AR platforms. Nowadays there is a tendency to allow interaction without needing specific hardware, as in the case of hand gestures and integrated touch screens. The functionality offered by these devices in general has been integrated in the form of graphical interfaces, which explains their downward trend.
- (d) Voice commands: The last kind of interaction method allows the user to make use of voice commands to transmit simple orders to the system. This is highly convenient in AR system that use HMD since the combination of gestures and voice is slightly natural to the user.

4.3.6. Method for aligning the AR content

This characteristic refers to the process by which the virtual information gets aligned between the users. This parameter focuses on the position, rotation and scale of AR assets. Does the same digital element appear in the same location for all the users? On the contrary, does each user view the digital information with different parameters? Six different techniques can be used to perform (or not) this alignment.

- (a) Not aligned: This is the most basic option. In this case, all the digital content shown to a certain user is positioned with respect to the device the user is carrying on. Thus, the same digital content can be shown to multiple users, but that content will not be positioned nor oriented in the same way for all the users.
- (b) Marker based alignment: This process is based on the use of one or several physical markers or images situated on the real world. The basic concept is that the AR device detects those images and estimates its relative position, size and rotation using image analysis techniques. Once those markers have been detected, the digital content can be attached relative to that estimated point in the space. This strategy is one of the oldest in AR environments, since it does not require external hardware other than the marks. It only requires processing capabilities, of which new devices have more and more. According to the limits of the alignment we split this technique into two categories.

- (I) Mark-restricted world: The digital information is shown directly *attached* to the mark (generally over it). By manipulating the mark, the user can rotate and move the digital asset. This way of alignment is the most direct, however, it presents a major problem. Once the mark is no longer being detected, AR content tends to drift. Usually devices that perform this process combines image analysis and position estimation based on sensors such as the accelerometer and the gyroscope. These sensors tend to accumulate errors that are corrected thanks to the image analysis when the mark is detected. Without that correction process, AR content moves away from the “real” position. In few words, this process generates a local coordinate system over the mark.
- (II) Physic world alignment: In this case several marks are placed across the physical world (that can range from room-sized to all world-sized). Those markers are detected by the system as said before, but in this case the digital asset is not “attached” to the mark. Thus, the information from the marks is combined to generate a shared coordinate system between the digital world and the real world. The more marks the system can detect, the better results it provides. This fact is due to once the marks are not visible, the whole process is only driven by the sensors with the problem mentioned above. However, if more marks are present, it is easier to detect one of them and correcting drift errors.
- (c) Environmental Point Cloud or SLAM map: In this case the digital information is *attached* to a set of 3D points on the space. The group of 3D points that represents the physical space is known as *point cloud* and it can be generated using infrared images or image processing techniques, among others. Some frameworks offer the possibility to detect objects and use them as if they were markers. The biggest problem with this strategy is that the generation of the 3D point cloud depends on the framework and the device used. Hence, it is not an easy process how to align point clouds from different sources and share them among different platforms. This technique is also related to the similarity of the real-life environments in which the system is used. In a face-to-face scenario, this does not have a great importance, since necessarily the space will be the same for all the users. Nevertheless, in those scenarios in which users are not co-located, each space of each user will present different physical characteristics. When developing systems that make use of this technique, this factor must be taken into account. For example, when establishing a position with respect to the original point cloud (the one obtained by the device that initiates the collaboration), it is better to relate that position with a point on a flat surface. This is due to it is very likely that any user can have access to a table or the floor.

- (d) Electronic tracking: Some systems use specific hardware to perform the alignment. The tracked elements are located on a well-known space (e.g. a room) which really limits the interaction possibilities. In addition, the specific hardware makes it harder for the vast majority of users to accept the system. This kind of tracking was used for example in the Studierstube system (Schmalstieg et al., 1996).
- (e) GPS based: In this case the system uses GPS on the client devices and latitude/longitude as the common coordinate system. This approach presents two main issues. First, due to GPS errors, accuracy is quite bad, which means that two users do not see the digital content exactly in the same position. Second, this approach is not viable indoors due to the loss of GPS signal inside buildings. However, in AR/MR applications where the alignment does not have to be highly accurate, this approach is the easiest choice to implement, since it only requires a GPS connection.
- (f) Free election: This technique can be seen in fact as a meta-technique, that is, we relegate the alignment to each participating device, which can make use of any of the aforementioned techniques. More in detail, in this case the system allows the user to choose an arbitrary point in the real world to which the digital AR content will be attached. This real-world point will be tracked by the AR framework and will adapt to the user movements (which presents the same problems as the environmental point cloud strategy). If two or more user choose the same point and the same alignment, they would see the digital content aligned in a similar way as in the markers case.

4.3.7. Integration of real-world information

This characteristic refers to the fact whether information about elements of the real, physical world are introduced into the experience. Thus, there are two possibilities. Either a) the AR content is decoupled from the real world or b) the AR content relates to information from the physical world. In this second case there are several possibilities depending on the type of integrated information:

- (a) Geolocation: A geographical position is used to influence the information presented to the user (e.g. display an information panel only when in front of a certain element). Thus, the content stream depends on the location of the device.
- (b) Environment: AR content *collides* or *interacts* with physical objects (e.g. can perch on them). In this situation, the flow of AR content depends on the physical environment around the user and the device. AR assets could be

transferred from one environment A to another device in another environment B. However, if the relationships between the assets and environment A are maintained, the perceived result in environment B could be conflicting or lose meaning. Thus, for example, if an element is visible and within the reach of the user in the A environment, the situation could arise that when transferring it to environment B, it would be behind a wall or inside a physical object, thus being unreachable. Regarding the loss of meaning, it can be seen in the example of an AR element bouncing on top of a table in environment A, which when transferred to environment B would bounce in the air as there was no table in that same position.

4.4. Applying the Taxonomy to the State-Of-The-Art Systems

In this section we make an initial classification of the systems presented in the state of the art based on the presented taxonomy. The result is presented in the form of a table in which the analyzed papers represent the columns. The rows of the table represent the categories of the taxonomy. For each combination of system and characteristic, an X represents a match while the question symbol represents our guessing in case of doubts. In the event that after analyzing the paper the classification for one characteristic has not been clear, we indicate with a question mark the one that we believe most closely matches.

	Rekimoto 1996	Schmalstieg 1996	Szalavári 1998	Carlson 2002	Kaufmann 2003	Fuhrmann 2001	Benko 2004
Location of the participants							
Face-to-Face	x	x	x	x	x	x	x
Remote							
Mixed							
Collaboration Model							
Unconstrained	x						x
Turn-based							
Ownership-based						x	
Layer-based		x	x	x	x	x	x
Hierarchy-based					x	x	
Unspecified							
AR Visualization Platform							
Head Mounted Displays		x	x	x	x	x	x
Hand held	x						x
Spatially Augmented Reality							
Hybrid							x
2D Screen							
Interaction method							
Interactive 2D surface	x	PIP	PIP	PIP	PIP	PIP	
Hand Gestures							Glove tracked
Joystick							
Voice commands							
Touchable Screen							
Graspable device							
Method for aligning the AR content							
Not aligned							
Environmental Point Cloud							
Electronic trackers	x	x	x	x	x	x	x
Free election							
Physical markers							
GPS based							
Unspecified							
Integration of real-world information							
Not integrated		x	x	x	x	x	x
Geolocation							
Environment	x						

Table 4.1: Applying the taxonomy to the state-of-the-art literature. Part 1

	Nilsson 2009	Billinghurst 1998	Poelman 2012	Reitmayr 2001	Huang 2013	Dactu 2014	Dong 2011	Butz 1999	Huynh 2009
Location of the participants									
Face-to-Face	x			x			x	x	x
Remote		x	x		x	x			
Mixed									
Collaboration Model									
Unconstrained		x		?					
Turn-based	x								
Ownership-based						x		x	
Layer-based			?				x	x	x
Hierarchy-based			?		x				x
Unspecified				x		?		?	
AR Visualization Platform									
Head Mounted Displays	x	x	x	x		x	x	x	
Hand held									x
Spatially Augmented Reality									
Hybrid									
2D Screen					x	x		x	
Interaction method									
Interactive 2D surface						x	x	x	
Hand Gestures			x		x				
Joystick	x	x							
Voice commands									
Touchable Screen									x
Graspable device				x				x	
Method for aligning the AR content									
Not aligned		x			x				
Environmental Point Cloud						x			
Electronic trackers	?		x						
Free election									
Physical markers				x			x	x	x
GPS based									
Unspecified									
Integration of real world information									
Not integrated	x	x		x	x		x	x	x
Geolocation									
Environment			x			x			

Table 4.2: Applying the taxonomy to the state-of-the-art literature. Part 2

4.5. Summary

The contributions of this Chapter can be divided in two main components. On the one side, we have presented an analysis of the characteristics of the collaboration itself, with special emphasis on AR systems. On the other side, we have presented a taxonomy that serves to classify collaborative Augmented and Mixed Reality systems. The combination of both contributions allows both to establish a theoretical framework for the study of collaboration and to facilitate the classification of AR experiences based on their characteristics.

Chapter 5

SARA: AN ARCHITECTURE FOR SHARED-AUGMENTED REALITY EXPERIENCES AND APPLICATIONS

Bow ties are cool.

The Eleventh Doctor

5.1. Context and motivation

Augmented Reality (AR) and Mixed Reality (MR) technologies are becoming more and more accessible each day. New devices that make use of different development platforms come to market every little time. As an example, at the time of writing these lines the two most important mobile technology platforms (Android and iOS) and one of the most popular Mixed Reality Head Mounted Display (HoloLens) uses different development environment, different programming language and different 3D framework (iOS uses swift and SceneKit, Android uses Java with Sceneform and HoloLens uses Unity with the Mixed Reality Framework (MRTK)). Unity itself offers a solution that can be adapted to the platforms indicated above. However, it forces developers to create applications using their libraries and their 3D framework, in addition to having a lower performance compared against native applications.

At the moment of developing AR-based collaborative experiences, there are two main issues to face. First, with several users involved in the collaboration, the

experience development gets complicated. Thus, the synchronization between users and the event management as well as the the communication between the devices are some of the tasks to be considered during the development stages. Second, the most common way of programming multi-user AR applications is to either use a proposal such as Unity (write once, run everywhere) or implementing the application logic for all types of devices that might be involved.

In this context, SARA emerges, being initially presented in a journal publication (Vaquero-Melchor et al., 2020). Thanks to the fact that it has been designed and implemented from scratch to be microservice oriented, it offers flexibility and facilitates network deployment as well as replication capabilities. In addition, it allows the logic of the application to be programmed for a single platform while using its functionality in many others. Finally, thanks to its integration of *collaboration models*, the collaboration type and characteristics can be modified on the fly. In the next Section the main features of the proposed architecture will be presented.

5.2. SARA features

First and as an overview, the solution architecture and its development has been carried out based on the challenges identified of the literature presented in Chapter 2 and the collaboration study presented in Chapter 4. SARA's name comes from "*an architecture for Shared-Augmented Reality experiences and Applications*" and its characteristics are presented below.

1. **Multi-user management:** Since SARA's main objective is to facilitate the development of collaborative AR and MR experiences, multi-user management is a core concept of architecture. With this characteristic we refer both, user access control (e.g. logins, logouts) as well as the exchange of information between participants in the collaboration.
2. **Session management:** To orchestrate the collaboration, SARA relies on the concept of *Sessions*. Several users may connect to the same Session, each session having a different functional concept and objective (i.e. each one associated to a different collaborative application). The AR content associated with a Session may be shared among the participants in the collaboration. Each session will be treated independently within the system, so the same instance can be used for handling totally different applications at the same time.
3. **AR content visualization management:** SARA is able to manage in a flexible way the visualization of AR assets. By this we mean that it is capable of controlling features as if all users can see a certain AR element (e.g. it

is possible to control that visualization based on permissions) as well as if everyone sees it with the same characteristics. This management can be carried out from the functional logic of a Session or externally through the use of collaboration models.

4. **Device-adapted interaction management:** Each AR device has an inherent form of interaction associated with it. SARA integrates all these interaction types and is able to work with them. As an example, when the used device is a smartphone, the main interaction is performed through its touch screen. On the contrary, when the user is wearing an HMD like the HoloLens, the main interaction is performed by hand gestures. But not only that, the user expects that *a touch* on a digital object and *a tap gesture* to have a similar functional meaning. Moreover, interactions from different device types are translated to a common understanding. Thus, SARA is able to handle all these different interaction methods while providing them with the adequate interaction meaning.
5. **AR platform and development framework independence:** The whole architecture has been designed and implemented as a platform-independent environment. This feature refers the fact that different devices can cooperate over the platform, regardless of its manufacturer, type and operative system. From smartphones to wearable AR devices, going through standard platforms like computers, SARA puts at their disposal a cross-platform information exchange system. Besides that, the creation of new end-point applications for the architecture is platform and framework independent, which allows developers to work with the tools of their preference.
6. **Location independence:** The collaboration between participants can be performed both *remotely* and *locally*. This process is transparent to the user, since she is only asked to select a session to connect to. Then, according to the session details, if required, the alignment process would begin. From that moment on, all AR assets injected into the session would become visible to all participants (taking into account the rules of collaboration models)
7. **AR Content Alignment:** In case that AR content alignment is required, the system establishes it according to the requirements of the session, which are defined by the user who created it. To do so, SARA offers *different strategies*. The basic mode of operation with the current implementation of the platform is intended to be through the use of markers, which are 2D physical images located on the real world. Although it would be ideal, the process of aligning cloud of points (also known as SLAM maps (Dissanayake et al.,

2001)) extracted from the real world has such difficulty that it is proposed to be integrated in a future version of the architecture.

8. **Time decoupling:** Collaboration is allowed to take place both in real time (or near-real time) with all the users collaborating at the same time as well as in shifts, with the participants not being coincident in time. To do so, SARA keeps a central state of the collaborative session and changes on this state can be made at any time. Later, that state can be recovered on demand (e.g. if a participant connects later to the collaboration).
9. **Extensible:** It is possible to add new features to the system easily. This affects both the addition of new services with new functionality to the system, and the adaptation and extension of the associated data model.
10. **Independent of the communication network protocol:** In order to cover as many devices as possible, the system offers a transparent way of establishing the communication. At the moment of truth, there are certain frameworks and platforms that present different difficulties to work with some communication protocols, hence the option of choosing is given to the developer. The use of TCP, UDP, Websockets and MQTT is contemplated in the current implementation.
11. **Logic decoupled and reusable:** Thanks to SARA, all the logic of the application can be implemented for a single device/platform, which we call the *provider*. This provider will inject AR content to a session and will have the application logic. Then, the other participants will send interaction events to relate with that content or trigger certain functionalities. This second group is labeled as *consumers*. The main core of consumers' client application is common to all the applications and hence it can be reused and taken as starting point for future developments. Taking that core as the base, some logic such as interface management may be implemented, but it is not strictly necessary.
12. **Adaptation of non-AR, non-collaborative applications:** By using SARA it is easy to inject the content of applications that were not generated with Augmented Reality in mind and expose them to users as AR ones. It is also possible to update applications that were not initially developed as collaborative to allow that functionality.

5.3.1. The Communication Service (CS)

This component is in charge of everything related with the communication, mainly it is in charge of message forwarding (events) between all the clients involved in the collaboration. Besides that, it keeps a central state for each Session and applies the collaboration model over it (more details about this in Section 5.3.3). These three tasks make the *Communication Service* (CS) a critical element of the architecture. This is the one exception mentioned above, regarding the fail tolerance of the architecture. In the event that this service is not operational, collaboration can not be performed. However, thanks to being microservice based, it is easy to establish replicas of this component and for example to redirect the communication to another instance. Next, the elements that make up the CS are presented. Figure 5.2 shows them in more detail.

Figure 5.2: Communication Service details

The *Communication Service* is composed by the following elements. First, the *Network Manager*, which is the main entry point to the system, since it is responsible for offering an access point to the clients. As said in feature 10 of Section 5.2, we sought to make SARA communication protocol independent. In order to achieve this, all messages received by the network manager are processed in the proxy. There, the content is extracted to be later processed by the *Event Manager*. Once the users are connected to a certain Session by using a certain communication protocol, by default that same protocol will be used to forwarding them the events. SARA currently offers four possibilities: MQTT, TCP, UDP y WebSockets. As it can be seen, two of them are low-level (TCP, UDP) and the others are high level (MQTT and WebSockets). From this moment we will refer these messages as *Events*, referring to the content of the messages exchanged by the participants of the collaboration.

The main task of the *Event Manager* component is to identify the received

events and to react as required. These events may be about users connecting to sessions, updates of a session state or interaction events among others. For the development of this thesis, the set of events implemented can be seen in Section 5.4.2. Some of them may be for the consumption of the *Communication Service* (e.g. the *NewUserConnection* one) while others will be forwarded to the clients (e.g. the *IncrementalUpdate*).

In case that the event processed by the *Event Manager* is one that connects a user to a session, the *Clients Manager* component is responsible for processing and updating the connection details. It is also in charge of closing the connection with users if necessary (e.g. if she has been too much time idle). The credentials of the users and their handling are not treated in this component.

The *Session Status Manager* component stores the information of all sessions handled in the system at all times. Each time that some session element is modified, the new changes are applied to this central state and forwarded to the clients so they can do the same. By doing this it is possible at any time to recover the central state (e.g. when a participant connects later to the session or when the collaboration is performed by shifts).

Given the wide range of variability of 3D frameworks, the *Session Status Manager* can use the *Transformation Interpreter* component. This element has been though to be used in those situations in which a coordinate system operation is required. Let us use an example in which a participant/client uses the Unity platform while another one is using ARKit. In the first platform, Z axis points from the center of the world towards the opposite side of the user, moving away from her, while in the second platform it points towards the user. However, regardless of the coordinate system used on each platform, the AR content is expected to be shown the same in both. If the coordinate system is not corrected, that would not happen. When any client ask for the session status, the 3D models, position, rotations and so on are tailored to the target platform. The same happens with update events. Which coordinate system should be used is specified when the user connects to the Session (by knowing the used device, the 3D framework can be guessed).

The last component of the *Communication Service* is the *Interaction Interpreter*. It is in charge of reacting to interaction events and to process them in a centralized way. Once processed, those events can be redirected to the pertinent Session.

5.3.2. The SARA Client and Content Generators

A SARA Client is the access point to the system. It represents the application that is executed in a certain device by a user or autonomously, as will be explained later. By combining several of these clients the collaboration will happen. The first point to highlight is that SARA Clients are configurable and they can be adapted according to the requirements. Thus, it can be seen that the clients may have

different implementations. Below are the main components that make up the SARA Clients and we will later indicate how they can be combined for more specific cases. Figure 5.3 shows a detail of the architecture diagram focused on the clients and their relationship with the content providers.

Figure 5.3: SARA Client and Content Generators in detail

The main core of every SARA Client is the *Session Manager* component. This component is in charge of communicating with the *Communication Service* and to receive information from it. More specifically, it is a set of standard functionalities for managing network communication as well as sending and receiving events. This component will be common for all applications, and once implemented for a certain platform, it can always be reused.

If the *Session Manager* receives an event which is related to the state of a Session (e.g. if the 3D asset catalog has been modified), the *AR View Component* is in charge of processing it. To do so, it creates the 3D model using the local AR library and adds it to the AR view or updates its properties. It is possible for the SARA Client to not be used for viewing AR content and, in that case, this component is not required. Thus, we have that there will be Clients that are able to generate the status of a session (e.g. to create a 3D model and to associate it with the state) without an AR framework being required.

All the digital AR content is arranged and organized around one spatial point: the coordinate origin. If this origin gets moved, all the AR content will move with it. The *World Center Manager* represents the functionality that allows the user to select a certain point in the real world to use it as the coordinate origin. In general, AR frameworks select as the coordinate origin the position occupied by the AR device at the time of launching the application. It is possible then, that the user can select another point manually (e.g. by touching the surface of a table in the AR view) or automatically (e.g. by using 2D physical marks). Obviously, if the client is not used for displaying AR content, this functionality will be ignored.

The next component has been labeled as *App logic* and it represents the specific operations performed in the Client that may or may not affect the Session Status. To

illustrate this component, let us make use of the example exposed in Section 5.6.2, the collaborative voxel-based game. Assuming the simplest collaboration case, there will only be two involved users. One of them is implemented in Unity and the other one in ARKit. The Unity client will generate the voxel world and will apply the users' actions on it. The iOS client will receive that world and it will generate an AR representation. The *App Logic* in the case of the Unity client will be the implementation of the voxel world management (i.e. adding or to deleting them, or changing their type). The iOS client does *not* have that logic implemented; It will only receive the world state (i.e. the Session) and will generate interaction events when the user touches the voxels. However, this iOS client implements a palette view for selecting tools and voxel types that is not present in the Unity client. It has been specifically generated for the mobile device and adapted to its screen.

In summary, depending on the role adopted by the clients within the collaboration, its implementation will change. Those clients that inject content into the Session and update its state are labeled as *providers*. On the contrary, those clients whose main functionality is to receive updates of the Session and to generate interaction events are labeled as *consumers*. In advanced situations it is possible to have several providers within the same Session, always taking into account that this may cause conflicts with the status of the Session and complicate the development stage.

In the aforementioned example, the iOS client generated interaction events whenever the user touched the voxels of the game. The *Interaction Manager* component is in charge of detecting when the user has interacted with a voxel and it generates the appropriate event. Finally, this component redirects the event to the *Session Manager*. In mobile context, to *touch* a voxel implies that the user literally use her finger to contact with any point on the screen. However, in Head Mounted Display context and HoloLens in particular, touching or interacting with an AR element implies pointing with the head (gazing) and making a confirmation gesture. As a last example, assuming a conventional interface (e.g. web-UI), to interact with a cube the user would have to *click* with the mouse in the position that it occupies on the screen. With those three examples, it can be seen how the *Interaction Manager* component has to be adapted to the involved device. In case that interaction is not expected to happen, this module can be ignored.

Until this point, we have described the inner functionality of the SARA Clients. Thanks to the *Content Provider Services* (CP), clients are able to incorporate external information into the collaboration. Moreover, it is enough for one of them to make this addition so that all collaboration participants can access to it. The functionality of these providers may vary. An example can be a Content Provided that generates a 3D model from a set of parameters, as the ground 3D service exposed in Appendix B. Any client can contact that service to request for the Digital Surface Model (DSM) of a piece of land. After receiving its response, the client can generate an actual 3D mesh and inject it into the session.

Finally, it has been decided to take into own consideration the *Content Generator* services. This has been done to differentiate between those services that really create consumable information and those that provide access to the information to the previous ones. To illustrate this division, let us take as an example the IOT (Internet Of Things) scenario. In that context, a *Content Generator* may just be a system whose main task is to read the room temperature and storing it into a data base. A possible *Content Provider* service would be one that accepts request from a client and returns information. In this case it can be as simple as a plain number that represents the temperature. However, it may be something more advance, like a 3D model generated from the digits of the temperature.

5.3.3. The Collaboration Manager Services

The last group of services is formed by those that are in charge of handling the collaboration from a higher-level point of view.

Figure 5.4: Collaboration management services in detail

The first one in importance order is the *Collaboration Model Manager Service* (CMMS). This service is in charge of applying the *Collaboration Models* to the Sessions. Each Collaboration Model is formed by a set of rules that defines in which way the collaboration between users has to be managed. In Section 5.4.3 more details are given regarding how those models are defined in SARA.

Fundamentally, each Session has one active Collaboration Model (and hence a set of rules) at any time. Each time the *Communication Service* receives an event from the clients, it checks whether that event is valid for the session. To do so, it redirects the event to the CMMS, where the actual validation will be performed and waits for the response. Then, the CMMS faces it against *all* the rules defined by the model. If the event successfully exceeds all the rules, the CMMS returns a positive result and the Communication service proceeds to apply it. Otherwise, by default the event is discarded.

This service has the following features. First, SARA offers five collaboration models ready to be applied to the collaboration (more details of the models can be found in Section 5.4.3). Second, it is possible to change the collaboration model for a session on-the-fly *without being required to change the application logic*. Third, it is completely feasible to modify the provided collaboration models to add or remove rules from them. If at any time the Communication Service can not communicate with the Collaboration Model Manager, it has been decided that all events will be accepted by default. This situation may be caused by failures or service outages or by explicit decision of the developer.

5.4. Data model

5.4.1. Sessions

The main concept of SARA’s data model is based on *Sessions*. A *session* has a conceptual meaning similar to a *scene*, which is the most common concept among the main development frameworks. Multiple participants can join to the same session, which will have its own objective. Additionally, each session will allow the participants to share AR content while interacting with it in a manner adapted to the type of AR device used. Finally, SARA is able of handling multiple sessions tightly at the same time. Figure 5.5 shows a class diagram of the data model associated to the sessions. The “SARA” prefix has been added to those classes that are characteristic of the architecture, with the aim of distinguishing it from the names used by the main frameworks.

A *Session* has an associated *Session Status*, which holds all the AR content associated to that session. Each *Session Status* is defined by *Nodes*, being the one at the top of the hierarchy the root one. Nodes have a reference to its parent and they also may present a reference to a group of children nodes. This structure has been selected to be similar to the one used by the main 3D frameworks (e.g. *GameObjects* in Unity and *SCNNodes* in ARKit). Thus, by combining several of these nodes, a hierarchical tree-like structure appears. *Nodes* have an associated *Transform*, which stores the position, rotation and scale of the Node. This concept

have associated. This definition ranges from the basic definition of “painting” the model (i.e. setting the color) to more advanced features (e.g. shininess, reflectance or transparency). In those frameworks, it is an usual practice to have multiple materials associated to the same element. However, it was decided to keep that count to only one for developing SARA (however, this featured could be easily added in the future). In the SARA context, each Material has an associated base color, as well as an emmissive color, a metalness value and a roughness one, which present a simplified version of PBR materials (Physically-Based Rendering). Thus, the rendering software calculates the appearance of the material taking into account the properties specified. However, it is quite common in most popular 3D frameworks to offer other types of materials. Figure 5.6 shows an example of how the Blender 3D editor offers various types of materials: one to give the appearance of glass, another to make the element look polished (glossy) and one called *principled BSDF*. The latter is a more advanced version of the material defined in SARA, and its initials mean *Bidirectional Scattering Distribution Function*. In summary, this type of material takes into account the parameters shown to define how light is scattered around 3D elements.

Figure 5.6: Blender node editor where various types of material and their properties can be seen. Material called *Principled BSDF* is an advanced version of the material representation used in SARA.

5.4.2. Events

In the previous Section it has been exposed how the information inside Sessions is defined within the architecture. Now, in this Section we present the data model that permits the exchange of information between the different components and services of SARA. Within SARA, this exchange of information is done by means of *Events* sending. Figure 5.7 shows a class diagram relative to the Events and its subclasses.

Figure 5.7: Class diagram of the data model relative to Events

As can be seen in the upper, central side of Figure 5.7, the main entity is the *SARAEvent*. As has been done up to this point in the document, the prefix “SARA” has been added to all classes in the class diagram to distinguish them from those used by the main development frameworks and languages. From now on, we will ignore that prefix to facilitate reading. Each Event has always five basic properties:

an unique *id* (used for tracing the events), a *timestamp* (in milliseconds), a *sender user* (which includes the id of the user who sent the event), a *type* (which is used to distinguish the different event types and is used as an enumeration) and a *targetSessionID* (which stores to which session should be propagated). Just below the Event class, two classes can be seen that inherit from it, which represent the two main types of events that SARA will handle: *Information events* and *Interaction events*.

On the one hand, *Information* events encapsulates those events that update the information of the session as well as the connections. Below, the main sub types of Information events are presented as well as their purpose.

- Whenever a client application requires to connect to the SARA ecosystem, it must generate and propagate a *NewUserConnection* event. In this event the id of the client that will be connected must be specified, as well as the *Connection Method*. As will be detailed later, the system allows the network connection to be done by using different network protocols such as TCP, UDP, WebSockets and MQTT. Once the client has connected with the system, it can be easily known which of these protocols was used to establish the connection. However, in order to keep the data model clear, it has been decided to indicate this information within the event itself. At this moment, the connection information of the user has been stored, but she has not been connected to any *Session* yet. In order to do this, a *ConnectToSession* event is required.
- A *ConnectToSession* event contains the identifier of the session to which the client is going to connect, the identifier of that same client and the *status reception format*. The reason of the last field is closely related to another of the types of Information events, the *SetSessionState*.
- The *SetSessionState* event is used to completely replace the state of a session as well as initializing it for the first time. As it can be appreciated, it has a *format* field and a *new session state* one. The reason of the *format* field is as follows. With the aim of maximizing the compatibility with as many platforms (or 3D frameworks) as possible, the whole status of the session can be encoded with different formats which are well known and pretty common in most 3D development tools. Some of them, such as OBJ or COLLADA, are the most popular on all platforms. Each of these formats has its own advantages and disadvantages. For example, OBJ is human readable and very easy to parse. However, it is very verbose and it has limitations. With the aim of allowing a greater clarity to the events exchange, to offer a greater flexibility and to have a greater control over the content of the event, we introduce the use a custom format based on JSON. The idea is to export the status of the sessions (i.e. the tree-like structure based on nodes) to one of

these formats and to encode the result to a base64 string. Since each client indicates its own connection method, different users may receive the same *SessionStatus* encoded in different formats. It can be argued that this type of communication is not the most efficient. Facing future implementations of SARA one improvement may be to encode the entire model as flat bytes, which would reduce the amount of information moved within the system (since the JSON format introduces redundant characters) as well as the communication times (the smaller the messages, the shorter the sending time).

- As opposed to the *SetSessionState* whose intention is to replace the old state (i.e. the nodes) of a session entirely, the *IncrementalUpdate* event is used to update specific properties of the nodes such as the position, the rotation or more advanced ones. To do so, it is required to specify the id of the target node, as well as the *property path*. Let us put an example to illustrate this concept. Suppose a Node whose mesh has the form of a cube. Within one client, the application logic is changing the rotation of the node in such a way that the cube is spinning on itself. When that change is detected on the client application (as will be explained later), an *IncrementalUpdate* event is generated with the following information. First, the id of the node is the one of our example. Then, the *property path* has to indicate the property that did change and then it is encoded in a String. To do that, we concatenate the path of hierarchical properties towards the property that was changed. In our example, this *property path* would have a value of “*transform.rotation*”. On its side, the *new value* field will have the new value of the rotation property, for example $\{0, 275, 0\}$. In the case that it is the position of an object that changes, the path would be “*transform.position*” and in the case of the scale it would be “*transform.scale*”. By using this event, selective changes can be made without requiring to override the whole state of the *Session*.
- Another type of Information event is the *AskForStatus*. This event is raised from a client right after connecting to a Session and, as its name says, it serves to indicate the *Communication Service* that it wants to receive the status of a Session. Once received by the *Communication Service*, it sends the current state of the session to the client who asked for it. It would be possible to discard this type of event and perform this same function when it is detected that the user has connected to the session. However, it has been decided to keep this event since it allows a finer control of this functionality, and that it is not strictly necessary that this event is always sent.
- The *NodesAdded* and *NodesRemoved* events has a similar functionality. They are created and propagated when a client (which will usually be a *provider*)

detects that a node has been added or removed to the hierarchical tree. Periodically and from time to time, the current structure of the session status is compared with a previous snapshot. In the event that additions or deletions are detected, the relevant events are created and in the case of the *Node-Added* event, the relevant information is added to the list of nodes. On the other hand, when indicating the deletions of the tree, it is only necessary to indicate the ID of the nodes that have disappeared, therefore only a list with their identifiers is transmitted.

- The last type of event that inherits from the *Information* class, is the *SetNode-Material*. This event is generated and propagated when a change on a node material is performed. Later, in Section 5.5.1 this detection procedure will be addressed in more detail. For the moment, it is sufficient to indicate that when a change has been detected, the new material is sent to replace the old one. It would be possible to use the *IncrementalUpdate* event to change the material of a node by specifying the appropriate path. However, it has been decided to keep this event separate for the sake of clarity. This event can be also used to assign a Material to a node that does not have one.

On the other hand, there is another subclass of the *Event* class, the *Interaction Event*. This type of events are used to encapsulate information of the interaction from the user with the digital AR elements. As an initial set of *Interaction Events* we propose two: the *InteractionEventClick* and the *InteractionEventDrag*.

The first one has to be propagated whenever a user *touches* or *makes tap on* an AR element. It is important to note that this event will be generated from any client that has interaction capabilities. When the client receives that propagated event, it is parsed to later apply that event in the within the logic of the application. The *InteractionEventDrag* has a similar behavior but, in addition it will have information of the direction of that drag gesture (it would also be possible to add information like acceleration).

5.4.3. Collaboration Models

The third key concept in which the data model of SARA is based is the *Collaboration Model*. In Chapter 4 five different models were identified: *Unconstrained*, *Turn-based*, *Ownership-based*, *Hierarchy-based* and *Layer-based*. Figure 5.8 shows a class diagram of these models, as they are handled by SARA.

The main entity in the class diagram is the *CollaborationModel*. Each model has a unique *id*, a *name* and a textual *description*. These fields are used to store the collaboration models and provide a readable description to future users who may use them. Additionally to those fields, each collaboration model has a set of *rules*. The

Figure 5.8: Class diagram of the data model relative to the Collaboration Models

main concept on which collaboration models are based is that they can be asked for the validation of events. Thus, every implementation of the *CollaborationModel* class will have to implement that method. During the validation process, the received *Event* will be faced against all the rules associated to the collaboration model. There are two possibilities for the result of that validation: either *True* or *False*. In the *True* case, the event has managed to pass the rule and it can be faced against the next one. This process is repeated until one of the following two things happen: either the event has passed all the rules and it is *valid* or it has failed one of them. The result of this validation is the redirected to the *Communication Service*.

We would like to clarify the concept of *rule*. When implementing this classes, there are two possibilities for the defining the rules. The first one consists of using a rule-based external inference engine. In this case, the system can take advantage of the possibilities that these types of engines grant. Among others, they show flexibility and the rules are usually easy to define for users.

The second possibility consists of directly implementing the logic in code. This technique will be easier to implement in those situations where it is required to manage certain information to control events. For example, the *Turn-based* model requires to know whose turn is it. In the class diagram, the *current user turn* stores the id of the user who is currently the leader. For the validation process of this Collaboration model, only one thing is required: to know if it is the turn of the user who sent the event (i.e. to compare the sender of the event and the current leader).

In case there is a coincidence, the result would be *True*. A *False* result would be returned otherwise.

Regarding the *Layer-based* model, only one piece of information is required: a list of Layers, each one of them identified by a unique *id*. Each layer on its own has two fields. First, it stores a list of the *nodes ids* of the nodes associated with that layer. Second, it keeps a list of the ids of those users who has access to the layer. Thus, by adding or removing the users ids from that list, it is possible to control the access to the layers. The *id* of a node may be included in more than one layer. In that case, that node may be handled by more than one layer. The same situation occurs with the users. The *id* of one user may appear in different layers, meaning that he will have access to those layers

The last two Collaboration Models, *Hierarchy-based* and *Ownership-based* are of a higher level than the previous ones. This means that, although both models of collaboration are quite common in the literature, they have a peculiarity when implementing them on SARA. In the first one, there is a tree-like hierarchy of users. These users may or may not have access to the nodes (either in terms of visualization or interaction). In the *Ownership-based* model, only the user who “has” the node will have access to it. In reality, when implementing both collaboration models, it is possible to use the same principles that in the case of the layer-based model. For example, in the *Hierarchy-based* model there could be a layer associated only to the user located at the top of the hierarchy, with all nodes being accessible. Then, there could be another layer for their subordinates, or one layer for each of them.

In summary, with the implementation of the layer-based model and controlling the logic of access to these layers, it is possible to offer more advanced collaboration models.

Which *Collaboration Model* to use entirely depends on the service goal of the experience or application to be developed. In other words, SARA does not impose any model but she enable the developer to choose or combine the most adequate ones for the target application. An example of how multiple collaboration models suit different applications can be seen in Table 5.1. In that table, the first column of the table holds some tips for choosing the model, while the other two columns are associated with two examples.

On the one hand, we propose the voxel-based game prototype presented in Section 5.6.2. For the purpose of this Section, it is enough to indicate that multiple users can cooperate for manipulating a piece of terrain made by cubes (i.e. the voxels). That can be done in such a way that they can add or remove those cubes for creating building and landscapes. All players will have identical permissions and all of them will be able to add and remove cube. There is only one AR resource, the terrain, whose appearance will change as the results of the interactions.

On the other hand, we propose a prototype application for the visualization and management of smart city resources: the SmARt city viewer presented in Section

5.6.1. In its most basics, this prototype allow different users (e.g. the municipality, service providers or operators), to inspect injected data sources on a 3D representation of the terrain. In this case, there may be different AR resources, such as the 3D representation of the terrain as well as the AR representation of the injected data. The key concept behind this prototype is that the visualization of the AR assets will be based on the roles assigned to the users. Thereby, Table 5.1 summarizes how each *collaboration model* may fit or not for both applications.

	Use case	Voxel-based prototype	SmARt city
Unconstrained	When parties coordination is not needed or relies on human means	A choice if every player is empowered to add and remove cubes simultaneously and coordination relies on peer-to-peer communication	Not adequate, as it is assumed that the number of services service providers and resources in the city will be large enough to need at least information filtering services for visualization
Turn based	When resources need to be univocally controlled by a single user at a given moment in time	A choice if every player is empowered to add and remove cubes and coordination needs to be orchestrated to avoid conflicts and organize interaction	A choice in case two or more service providers share a common resource and may have control over it: e.g. a signage panel in which alerts can be presented may be admitting new information in case the control token is free
Layer based	When resources need to be grouped and the access to those groups has to be managed	It does not apply, as this application only contains a single node: the terrain mesh that players will modify	Layers will be associated to a given service (cleaning, air quality, traffic controlling. . .). Each layer will contain a set of nodes in it and will be secured by applying role-based access. Users authorised for each layer will be able to visualize the resources on it.
Ownership based	When a user owns one or more given nodes with full control over them. These resources could be within the same layer or not	Not applicable, as this application only requires a single resource for common interaction: the terrain mesh that players will modify	Applicable to determine e.g. which operators can access and visualize the state of specific resources.
Hierarchy based	When a hierarchy of roles is needed to control the different resources	The basic application is built on players with the same role, thus the hierarchy-based model does not fit. In case hierarchies are implemented, it is assumed that a user with upper role (game master) would have at least to approve the initiatives of the others	A hierarchical model does fit in this application, as there will be users with different roles that will be authorized to access functionalities depending on their role, e.g. municipality (overall control), service providers (overall monitoring of the service, interaction with shared resources), service managers (tactical decisions in a service layer and over owned resources) and operators (visualization and information generation over authorized resources)

We would like to highlight the fact that an application could integrate different collaboration models. For example, in the SmARt city scenario, the municipality could establish a *hierarchy-based* model over service layers to force and control specific aspects of the city management. At the same time, urban service operators working on the same level of the hierarchy could own and control their network of resources using a *ownership-based* model. Figure 5.9 illustrates an example of the combination of multiple models.

Figure 5.9: Combination of multiple collaboration models in the SmARt city prototype

As can be seen in Figure 5.9, a *layer-based* model is present, with three managed layers: *Traffic Lights*, *Cleaning Services* and *Air Quality Data*. Access to the *Traffic Lights* layer is granted to a Traffic Light Manager, which owns the resources associated to that elements. This manager should receive the state of the lights and update the AR representation associated to them.

At the same time, access to the *Air Quality Data* layer is granted to multiple clients, which will generate and own the resources associated to different weather stations. Those clients can represent institutions or citizens. All of them can would see all the resources stored on the layer, but only the client with the ownership of the station representation could manipulate it.

Also at the same time, a *hierarchy-based* model may be used to establish the control over the elements of the *Cleaning services* layer. The members further down the hierarchy would have access to a subset of the vehicle fleet, while the supervisor

would have access to the entire set.

In case of combining multiple *Collaboration Models* at the same time, conflicts could arise. For example, let us suppose that a user has access to a layer with a certain resource belonging to a different user. In that case, should she be able to see and interact with it? Or should he only be able to see it? In that case, the collaboration models should be tailored to the details of the experience and the five mentioned above would be insufficient as they are.

5.4.4. Orchestrating the communication between system components

The aim of this Section is to illustrate the process of the communication between the components of the architecture. Figure 5.10 shows an overview of the general communication process.

In the example of Figure 5.10 there are four entities involved: the *Communication Service* and three *Clients*. One of these clients (Client 1) will act as a *Provider* and will inject elements to the Session, while the others two (Client 2 and Client 3) will act as *Consumers*. Additionally, Client 1 and Client 2 will connect at the same time, while Client 3 will connect later. The interpretation of the sequence diagram is as follows.

First, Client 1 and Client 2 get connected to a certain Session, by sending an event to the *Communication Service*. Once done that, both clients will be connected to the Session. Upon connecting, both clients will receive the status of the stored session (if any) as indicated by the purple rectangle. At this point, both clients would have received the same status and they would have generated the AR representation from them.

Later, the *provider* of the session (Client 1) adds a node to the session status. This node is sent from Client 1 to the *Communication Service*, from where it would be propagated to all the connected clients at that moment. The *Communication Service* will also apply that event to its local copy, in order to keep it consistent with the other ones.

Then, the green rectangle illustrates the situation in which a third client connects later to the collaboration session. At that point, she would receive the session status as stored in the *Communication Service*, with the node added by Client 1.

Finally, the orange rectangle illustrate the interaction process. Thus, Client 2 would interact with a certain node of the session status. In that Client application, an Interaction event would be generated and sent to the *Communication Service*. From there, that event would be propagated to the other clients. The client that receives the interaction event may react to them. For example, the *provider* client

Figure 5.10: Sequence diagram of the overview of the collaboration process

may apply that interaction event to the application logic and trigger a certain functionality. It is also possible that the other clients provide visual or audio feedback about that event (e.g. a click sound effect).

Here concludes the explanation of the collaboration process overview. Next, we will proceed to expose each of these steps in more detail, starting from the incremental updates.

5.4.5. Incremental updates and Interaction management

To illustrate the workflow of incremental updates, let us suppose the scenario of Figure 5.11. In this example there are two clients involved (Client 1 and Client 2) and the *Communication Service*. Client 1 would act as *provider* and Client 2

would act as consumer. For this example, the Incremental Update events will not be validated by the *Collaboration Model Manager Service*.

Figure 5.11: Sequence diagram of the process of applying incremental updates without the Collaboration Manager Service

At the top of Figure 5.11, both clients connect to session, as has been illustrated before. Then, Client 1 (the provider) populates the state of the session with nodes and that information is propagated to Client 2. Let us suppose that the application logic implemented in Client 1 is changing the position of one of these nodes over time. Whenever a change on the position is detected, an *IncrementalUpdate* event is generated and sent to the *Communication Service*, from where it will be propagated to the other clients. Two considerations must be taken here. First, this same procedure will happen with other properties of the nodes, such as the *rotation* and the *scale*. The new value for those properties will be encapsulated into the events, so other clients can properly update their local representations. Second, in this example only the *provider* client is updating the properties of the node. However, it is feasible for Client 2 to generate the *IncrementalUpdate* events. Thus, when Client 1 receives those events, they are applied to the session status. If both clients would generate those incremental events at the same time, they would get applied as they arrive

on the *Communication Service*, which may cause conflicts and unexpected results. To overcome that situation, a fourth element gets introduced into the sequence: the *Collaboration Model Manager Service*.

The *Collaboration Model Manager Service* is in charge of validating events against the collaboration model associated to the sessions. Thus, whenever the *Communication Service* receives an event, it is validated by the *Collaboration Model Manager Service*. If the result of that validation is *True*, the event can be applied. The *Communication Service* applies that event to its local state and then it is propagated to the remaining clients. Otherwise, if the result is *False*, the event is discarded, and the client that launched it gets notified. Figure 5.12 illustrates the validation process when applied to incremental updates.

Figure 5.12: Introducing the *Collaboration Model Manager Service* into the process of incremental updates

The same principle is used for validating Interaction Events. Whenever the *Communication Service* receives one of them, it asks the *Collaboration Model Manager Service* for its validation. Figure 5.13 depicts an example in which two clients are involved into the collaboration and the session has a *Turn-based* collaboration model associated.

Figure 5.13: Sequence diagram of the process of propagating an Interaction Event by facing it to the Collaboration Model

As can be seen in step 3 of Figure 5.13, the event is faced against all the rules of the collaboration model associated to the session. Additionally, the turn belongs to Client 1. As the model is *Turn-based*, it is only required to compare the current user whose turn is it and the event sender. Thus, in step 4 the result of the event would be *True* and in step 5 the event would be applied. However, when Client 2 tries to interact with a node, the result of the validation would be *False* because it is not his turn. Thereby, the event would be ignored.

5.5. Practical issues on implementation

5.5.1. Detecting changes in nodes properties

Appendix C provides a detailed example of detecting node changes focused on the Unity environment. However, this process can be easily adapted to all types of development environments.

5.5.2. Adding interaction to the AR elements

The aim of this section is to detail how the interaction process with SARANodes works. To illustrate this process, let us propose the following situation as an example. Suppose there are two clients: a *provider* which will hold the application logic and one *consumer*. Let us also suppose that the status of the session only contains a cube (i.e. there is a SARANode whose mesh has the shape of a cube). The *provider* client may be using a HMD like the HoloLens and the *consumer* client may be using a hand-held device such as an iPhone. How may both user interact with the cube using the capabilities of their own devices? The HoloLens client may be capable of using a gesture (although there may be other possibilities like speech recognition) and the user with the iPhone would expect to *touch* the cube on her screen to trigger the interaction. As a last brushstroke on this scenario, let us suppose that whenever one of the user *interacts* with the cube, it will change its color to a new, random one.

In this scenario, the first step would be to *enable* the interaction to the cube node. Thus, whenever that cube is originally created in code, a new method has to be added, which will change the color of the associated material. This method may have a similar aspect to the following:

```
1 public void ChangeMaterialColor(SaraEvent_Interaction e){  
2     mr.material.color =  
        Random.ColorHSV(0f,1f,1f,1f,0.5f,1.f);  
3 }
```

Listing 5.1: Method that changes the color of a Node

The details of how to change the color in the Unity environment are beyond the intent of this document. Simply grasp the idea that that is going to be the functionality that triggers when users interact with the element.

In the next step, the interaction has to be associated with the cube (which would be a GameObject in the Unity platform). Thereby, when the cube is initialized, a SARAAction is generated to indicate that it will have an associated interaction. This procedure should be similar to the next one:

```
1 void Start(){
2     //Get a reference to the SARAInteractionManager
      singleton (more details later)
3     SARAInteractionManager im =
      GameObject.Find("SARAManager")
4     .GetComponent<SARAInteractionManager>();
5
6     //Get a reference to the SARAManager instance
7     SARAManager man = GameObject.Find("SARAManager")
8     .GetComponent<SARAManager>();
9
10    //Get the associated SARA unique id for this gameobject
11    string mysaraid =
      man.GetSARAIidForGO(transform.gameobject);
12
13    //If the Interaction Manager component is present
14    if(im != null){
15
16        //Create a new SARAAction instance
17        SARAAction a = new SARAAction();
18        a.user = man.client_ID; //Set the interacting user
      to the client ID
19        a.action_name = "change color"; //Set a readable
      description
20        a.nodeid = mysaraid; //Set the nodeid to my id
21        a.action = ChangeMaterial; //Link the action with
      the already created method
22
23        a.type = ActionType.click; //Set the action type as
      "click"
24                                //(which has a similar meaning
      to TAP or touch)
25
26        //Finally, register the action within the
      InteractionManager instance
27        im.RegisterAction(a);
28    }
29 }
```

Listing 5.2: Creating the SARAAction

From now on, the SARAInteractionManager will listen for interaction events from the users and whenever that happens, it will execute the associated method.

This class has the following structure:

```
1 public class SARAInteractionManager : MonoBehaviour
2 {
3     //A list of registered SARAActions is keep
4     private List<SARAAction> registeredActions;
5
6     // Start is called before the first frame update
7     void Start()
8     {
9         registeredActions = new List<SARAAction>();
10        //Here, the action list is initialized
11    }
12
13
14    //By using this public method, any element can register
15    //a new SARAAction
16    public void RegisterAction(SARAAction act)
17    {
18        registeredActions.Add(act);
19    }
20
21
22    //This method will be called from the SARAManager
23    //instance when
24    //a new event of this type is parsed
25    public void
26        OnInteractionEventReceived(SARAEvent_Interaction e)
27    {
28        //Here, we want to know the type of the interaction
29        //event,
30        //and doing the right cast
31        try{
32
33            SARAEvent_InteractionClick click =
34                (SARAEvent_InteractionClick)e;
35
36            if(click.sender_user != null &&
37                click.sender_object != null)
38            {
39                //If the event is of type click and well
40                //formed,
```

```
36         //the following method is called.
37         ClickEventReceived(click);
38     }
39
40     }catch(Exception ex){
41         //Cannot cast to Click type
42     }
43
44     /* The same procedure would be followed for other
45        interaction
46        event types, such us Drag */
47 }
48
49 /*This method receives a click event and go through all
50    the registered actions.
51    If there is any action associated to the nodeid
52    indicated by the event,
53    it is executed.
54 */
55 private void
56 ClickEventReceived(SARAEvent_InteractionClick e)
57 {
58     SARAAction a = null;
59     for(int i = 0; i< registeredActions.Count; i++)
60     {
61         a = registeredActions[i];
62
63         if(a.nodeid == e.sender_object && a.type ==
64            ActionType.click)
65         {
66             a.Execute(e);
67         }
68     }
69 }
```

Listing 5.3: SARAInteractionManager class

The SARAInteractionManager instance receives all the interaction events when the SARAManager instance has parsed them. After that, when a new event has arrived, the system iterates through all the registered actions, comparing the node id stored inside the event and the one stored within the actions. If there is any

match, the functionality gets called. , the interaction event is ignored by default. However, there may be some scenarios in which the user who send the event gets notified.

Thus, going back to the example scenario, when the smartphone client *touches* any 3D AR element on her screen and when the HoloLens user makes a gesture gazing towards any MR element, a new Interaction event is generated and propagated towards the *provider* client.

There are two main advantages of making use of this structure. First, it follows a similar convention for adding interaction to elements as the most popular 3D frameworks do. Thereby, it is easy to adapt the old interaction scheme to the new AR capabilities. Second, it offers the possibility of more advanced scenarios, in which for the same Node, there may be multiple linked actions, associated to different users or roles. As a last note on this Section, the structures and classes presented here have been kept as simple as possible, with the aim of facilitating the content assimilation. Although the system is able to work with this implementation, improvement may be required for more advanced scenarios.

5.6. Creating and deploying applications over SARA: an use case approach

5.6.1. A first Approach. SmARt city: A data source viewer based on hierarchical roles

This section takes a first look at the possibilities of development with SARA. In order to achieve this, we present the *SmARt city prototype*. The functional objective of this prototype is to allow multiple users to inspect a variety of data sources injected on a 3D view terrain. These data inputs would be associated to different entities in the context of a smart city. Additionally, multiple user roles may be defined, such as *cleaning brigades*, *lighting maintenance* or *security forces*. Some examples of the available data may be the traffic light status or the position of public transport vehicles. Thereby, each role may have access to different data sources. For example, a citizen could use the application to see the position of the bus she is waiting for while the security forces coordinator could see the current position of all the security brigades. When implementing the prototype, it has been decided to maintain a set of simulated data sources. Table 5.2 shows the main features of this prototype:

Figure 5.14 shows two screenshots of the prototype. In the first one, a user with a role associated to the security forces can see the position of a police vehicle. In the second one, some traffic lights are shown with their current status.

As indicated in table 5.2, the collaboration model selected to orchestrate the

Common Objective	To inspect data sources on a 3D representation of the terrain
Individual Objectives	Associated to each role.
Number of Users	Unlimited
Location of the Collaboration	Mixed
Involved Devices	Computer, HoloLens and iOS devices
Collaboration Model	Layer-based. With this collaboration model, each role has an associated layer, to which users with that role may access.
Temporality	Data sources may be updated in real time, However, each user would launch the application when required. Thus, in general it will be decoupled in time.
Interaction Capabilities	Disabled. Users are only allowed to inspect the data sources
AR Content Alignment	When multiple users with same roles share a view and make decisions together, physical marks are used

Table 5.2: Features of the SmARt city prototype

collaboration is the *layer-based* one. Although this example has a strong hierarchical component, we decided to select the *layer-based* model due to its versatility. In Section 5.4.3 we stated that the *hierarchy-based* and *ownership-based* models could be implemented based on layers, and this is the perfect example.

Thus, one layer is dedicated to the cleaning services while another one is dedicated to the transport services. Then, each data source would have its dedicated layer. Additionally, in order to generate the hierarchy, mode *sublayers* could be created. Those behave exactly the same as the regular ones with the difference that they only have a subset of the elements included in the others. Thereby, those users with more access privileges would have access to the general layers, while the subordinates would have access to the more specific ones. Below, we present the details of the process of creating a SARA based experience.

5.6.2. Detailing the creation process. A voxel-based cooperative game prototype

This Section details the application development process based on SARA. With this aim we present our prototype, which consists of a collaborative voxel-based construction game, inspired by Minecraft (min, 2019a) and its more recent one implementation, Minecraft Earth (min, 2019b). In the original game, the player

Figure 5.14: Screenshots of the SmARt city prototype

controlled a digital avatar inside a cube-based (voxel) world. Then, she can add or remove these cubes for creating landscapes and structures. In the Minecraft Earth implementation, the voxel world is constrained to a small surface and it can be viewed by means of Augmented Reality. Thus, in our prototype several players will work together on a piece of land, to which they can add or eliminate cubes in a collaborative manner. By default, a simple landscape appears, as a starting point for the users. From there, they can use three simple tools to manipulate the aspect of the world. These three tools are: i) a shovel used to remove the cubes, ii) a brush used to change the type of one cube and iii) a block adder, which will create a block at the desired point.

This small prototype has a self-contained functionality that allows all the SARA capabilities to be easily applied. All the logic of the application, has been implemented using the Unity environment and initially it was only focused to be for a single user, allowing the participant to click on the screen to interact with the world. Thanks to SARA, AR collaboration capabilities can be easily added. To illustrate the developing process, we propose the combination of HoloLens (as a representative of wearable AR) and two iOS devices (an iPhone and an iPad, representative of hand-held ones). Table 5.3 gathers the main features of this prototype.

The development process can be summarized in 5 steps: (1) configure and launch the Communication Service, (2) implement the main logic of the application, (3) create the SARA clients, (4) add specific logic to those clients and (5) selecting and applying the Collaboration Model. Figure 5.15 gives an overview of this process identifying the main tasks in each of the steps. Next, each of those five steps will