University of Deusto

A Comprehensive Human-Computer Interaction Model for Augmented Reality Systems

Dissertation submitted by

Mikel Salazar González

for the Degree of

Doctor of Philosophy in Information Systems

under the supervision of

Dr. Carlos Laorden Gómez

and

Dr. Pablo García Bringas

Author Co-advisor Co-advisor

Bilbao, October 2015

Obstacles are there to signify how bad you want something. Failure is a passageway to success. It's how you respond to failures, or obstacles, or those stumbles, or brick walls that will ultimately determine your results. You can let the failures hit you square in the face and knock you out, for good. Or you can get up and fight back. You will always pass failure on the way to success. It's just about never being content with where you are in the present. Moving forward with an insistent vigor to want more out of yourself. To want to do better. To have that internal fire, to not end up a failure. Do you have it in you?

Anonymous

Abstract

With this dissertation, the PhD. student Mikel Salazar proposes a new interaction model that takes advantage of the new capabilities of Augmented Reality systems to facilitate the construction of user interfaces embedded into the real world. A new paradigm that allows, both developers and end users, to create –and share– more intuitive and efficient user experiences.

To achieve this goal, the proposed interaction model offers a new perspective on the communication between human beings and computer systems. One that, instead of obviating the physical environment in which the interaction takes place, takes advantage of it to seamlessly integrate the components of the user interfaces. In this way, and thanks to a detailed definition of user context, the computer system can adapt the interaction elements to the needs and desires of each individual user (greatly simplifying the labor of user interface designers, that no longer do they have to create different version of their user interfaces for multiple platforms or disabled persons). A new approach that facilitates the collaborative edition of contents an, even, the dynamic discovery of new –and meaningful– interaction components as the users perform their daily activities.

To properly validate the proposed interaction model, the author presents in this dissertation a detailed examination of the current situation of Human-Computer Interaction (analyzing with special attention the limitations of current paradigms) and, later, uses this knowledge to build a usability study with which evaluate the different interaction techniques available. A one hundred participant study which positive results reinforce a research work that has been also corroborated both in academic and industrial environments.

Resumen

Con esta tesis, el doctorando Mikel Salazar propone un nuevo modelo de interacción que aprovecha las nuevas capacidades de los sistemas de Realidad Aumentada para facilitar la construcción de interfaces de usuario tridimensionales integradas en el mundo real. Un nuevo paradigma que permite, tanto a desarrolladores como a usuarios finales, crear —y compartir— experiencias de usuario más intuitivas y eficientes.

Para ello, el modelo de interacción propuesto en esta tesis ofrece una nueva perspectiva de la comunicación entre seres humanos y sistemas informáticos que, en lugar de obviar el entorno físico en el que se produce dicha interacción, lo toma como base para integrar los componentes de las interfaces de usuario. De este modo, y gracias a una detallada definición del contexto del usuario, el sistema informático es capaz de reconstruir dinámicamente los diferentes elementos interactivos, adaptándolos a los gustos y necesidades de cada individuo (simplificando así la labor de los diseñadores de interfaces de usuario, que ya no tendrían que preocuparse de preparar múltiples versiones de sus diseños para diferentes plataformas o para personas con discapacidad). Un nuevo enfoque que facilita la edición colaborativa de los diferentes contenidos e, incluso, el descubrimiento dinámico de nuevos —y relevantes— elementos interactivos mientras el usuario realiza sus actividades cotidianas.

Para validar correctamente este nuevo modelo, el doctorando realiza en su tesis un estudio pormenorizado de la situación actual de la Interacción Humano-Ordenador (analizando con especial atención las limitaciones de los paradigmas actuales) y, posteriormente, emplea este conocimiento para construir un estudio de usabilidad con el que evaluar debidamente las diferentes técnicas de interacción disponibles. Un estudio en el que participaron cien personas y cuyos positivos resultados permiten afianzar un trabajo de investigación que ha sido corroborado tanto en entornos científicos como industriales.

Acknoledgements

This dissertation would not have been possible without the support of the people of S3lab (DeustoTech Computing), who supported the author during the years of hard work that this research project required.

Special gratitude goes to my supervisors, Dr. Carlos Laorden and Dr. Pablo G. Bringas, without whose support –and *infinite* patience– this work would not have been possible. Additionally, the author would like to extend his sincere appreciation to Dr. Igor Santos and Dr. Borja Sanz for their continuous encouragement and patience over the years.

It is also important to note that the research project described in this dissertation is based on the contributions and expert feedback received from many great persons both in the scientific community and industrial sector. As Issac Newton famously wrote *"If I have seen further, it is by standing on the shoulders of giants"*.

In this regard, the author would like to express his gratitude to the chairs of the ISMAR 2012 Doctoral Consortium session, Bruce H. Thomas and Si Jung Kim, for their help in defining the scope of the research work. An important support that has been later complemented with the suggestions of many academic researchers, most notably, those provided by Steven K. Feiner, Valentin Markus Josef Heun and Fridolin Wild.

Nevertheless, it is also worth noting that the final result of this research project also takes into consideration the comments provided by industry experts such as Neil Trevett (nVidia Corporation), Martin Lechner (Wikitude GmbH), Marius Preda (Institut Telecom Sud-Paris) and George Percivall (Open Geospatial Consortium). An invaluable expert feedback that was obtained in the AR Community Meetings organized by Chistine Perey (Perey Research & Consulting), to whom the author wishes to express his gratitude.

Last, but not least, the author wants to convey his special thanks to his family for their continuous support over the many years of work that were required to bring this research to fruition.

The present document has been created entirely with Open Source software. In particular, the author would like to thank the communities behind the development of the LaTeX typesetting system, TeXstudio, Blender, the GNU Image Manipulation Program (GIMP) and MakeHuman.

Contents

"Everywhere is within walking distance...
if you have the time."

Steven Wright

1

Introduction

In recent years, important advancements in computer vision and mobile computing have enabled researchers to explore novel technologies that close the gap between real and virtual worlds in meaningful ways. These new technologies, commonly referred to as *Augmented Reality*, allow users to perceive computer-generated objects integrated seamlessly into real-world environments, as if they have an actual physical presence in them. Unfortunately, the lack of a proper Human-Computer Interaction (HCI) model that developers can employ to construct intuitive user interfaces has, so far, hindered the potential of the research field (to the point of being often considered a mere *gimmick*).

With the research project presented in this document, the author seeks to offer a solution to this situation by providing not only a comprehensive logical framework that facilitates the creation of interaction techniques, but also a communication model that allows the exchange of interaction elements between different computer platforms.

To properly explain this research project, section 1.1 presents the motivation behind it and section 1.2 the terminology necessary to gain a full understanding of its scope. Later, section 1.3 states the main hypothesis in which this doctoral dissertation lies and decomposes it into specific and operational objectives. Finally, section 1.4 explains the research methodology employed and section 1.5 presents the outline of this dissertation.

1.1. Background and Motivation for the Study

In the past decades, many sci-fi works have explored the possibilities of interacting with computer systems through fully immersive, three-dimensional user interfaces (to the point that it is often considered a recurrent trope of the genre). Either through projections on transparent surfaces or holographic displays, these fictional user interfaces allow the characters –and the audience– to interpret complex narrative scenarios in a simple and visual way. Unsurprisingly, these interaction systems have served as a source of inspiration to many User Experience designers and Human-Computer Interaction researchers, that have even taken advantage of these entertainment products to establish a "common language" with less tech-savvy users (even at the cost of being subjected to unjust comparisons) [1].

Until recently, technological limitations have made the actual realization of these user experiences an impractical endeavor. The large investments in hardware equipment required to construct such interaction spaces exceeded the capabilities of all but the most well-financed government agencies or military organizations [2]. However, recent advancements in tracking technology [3], coupled with the progressive evolution of the mobile hardware platforms (including Head-Mounted Displays, or HMDs), have enabled the development of a new generation of user interfaces, where the different widgets are integrated in the physical environment and can be operated through intuitive body gestures (instead of relying on metaphors and iconography that might not be appropriate for the circumstances of each user).

Sadly, the main issue still persists. While there is no shortage of research projects that take advantage of these new interaction mechanisms to create compelling user experiences [4], they are often restricted to very specific application domains and cannot be easily adapted to other use cases. There is no overarching framework that enables the easy development of additional widgets or extensions, no navigation system to switch between multiple use cases; not even a proper way to control the actual execution of the operating system. Consequently, current Augmented Reality-based computer platforms have to rely on interaction mechanisms inherited from previous paradigms, which, apart from limiting the capabilities of the system, forces manufacturers to include additional input devices into the hardware, increasing its cost and weight.

Aware of this situation, the scientific community has made conscious efforts [5] to delineate the research topic and establish a preliminary task force to advance towards its resolution. However, these efforts conflict with the interests of several key industry players that propose models centered on the transmission of media contents through highly controlled channels (digital distribution platforms generally known as "AR Browsers") [6].

Against this background, this dissertation aims to establish a novel interaction model that provides the necessary tools for third-party developers to easily create immersive user interfaces while also enabling users to control the global experience and adapt it to their personal situation (needs, long time goals, etc.). A *covenant* that extends beyond the formal definition of the theoretical framework, permeating the entire communication model and ensuring the reliability and security of the different interactions with the computer system (without being locked into a specific *tech ecosystem*).

While current hardware technologies impose significant limitations, the different interaction techniques that can be specified with the proposed model are based on physics-simulations. In this way, it is possible to create platform-independent user interfaces that achieve a high usability level while also opening the way for the formulation of alternative –accessible– interaction mechanisms (compensating for any disability that the end-user may have, without affecting the definition of the user interface itself).

Nevertheless, the final objective that this research project strives for is the creation of an interaction model that, as all successful technologies, becomes *transparent* to the user.

Figure 1.1: A photo montage showing a researcher operating a three-dimensional user interface that can be constructed using the Human-Computer Interaction model proposed in this dissertation.

1.2. Basic Terminology

Due to the large amount of –interconnected– concepts employed in the present dissertation, this section is dedicated to provide the reader with a comprehensive list of the main terms (and their corresponding acronyms) used throughout the document. This examination of the terminology (and the graphical representations that identify the boundaries and relationships between the concepts) will allow the reader to construct a solid conceptual framework over which to incorporate the new theoretical ideas discussed in later chapters.

1.2.1. Human-Computer Interaction

Formally, Human-Computer Interaction (HCI) can be defined as *the academic research field focused on the study of the communication between human beings and computer systems.* However, taking a closer look at this definition reveals several important aspects that have to be taken into consideration:

- **Restricted to the academic context:** While, in academia, the study associated to this scientific field is the responsibility of HCI researchers, the actual industrial application of the resulting knowledge is often considered a separate field (referred to as *Interaction Design* or *IxD*).
- **Complex communication model:** The radical differences in the nature of the entities, makes the study of their communication a multidisciplinary endeavor (involving such dissimilar research areas as computer science, psychology and information design).
- **Limited to heterogeneous communications:** Although there are numerous studies that focus on the interaction between computer systems and other living beings [7, 8], due to the complexity of current computer-related tasks, the bidirectional communication with human beings is generally necessary to complete them. Additionally, it is important to note that the area of study is restricted to the communications that involve participants of each type (interactions between human beings and between machines are, in theory, not taken into account).

The main goal of HCI is to facilitate the communication between human beings (generally referred to as *Users* in this context) and computer systems (or, simply, *Computers*) through the construction of a hardware-software system denominated *Interface*, that acts as an intermediary in said communication. The software components of these interfaces are called *Widgets* (or Controls), while the external hardware devices required for input/output operations often receive the name of *Peripherals* (although, over time, the significance of the term has actually broadened to encompass a large selection of devices, such as external storage or networking subsystems).

To avoid possible confusions with similar terms in other areas of computer science, the aforementioned communication channels are usually referred to as *User Interfaces (UIs)*. Due to the diversity of peripherals available (and the large number of possible combinations), multiple taxonomies have been developed over the years to differentiate between the many types of UIs. Although section 2.1 of this document offers an extensive analysis of the different HCI paradigms from a historic perspective, in the current context, it is important to highlight the recent appearance of *Spatial User Interfaces (SUIs)*; a new type of three-dimensional UIs where the users are not just external entities but active participants within the interaction space.

Independently of the type of UI employed, its underlying HCI model must provide a consistent set of interaction techniques adapted to the different tasks of a particular Use Case (because it is not the same to develop a UI for a basic application than for an operating system). To achieve this goal, each UI has to be validated through a *Usability Study*, where each component is evaluated in terms of effectiveness, ease of use and learnability.

Finally, one needs to be fully aware that HCI models are always holistic in nature. Conformed with the contributions of multiple scientific disciplines (many of whose can be seen in figure 1.2), HCI paradigms must offer different interaction mechanisms that can be modified to fit each situation. However,it is important to remark that they are not meant to be complete products, but foundations over whose third parties can easily construct their own communication channels with the users. An all-encompassing concept often referred to as *User Experience Design* (*UXD* or *UED*).

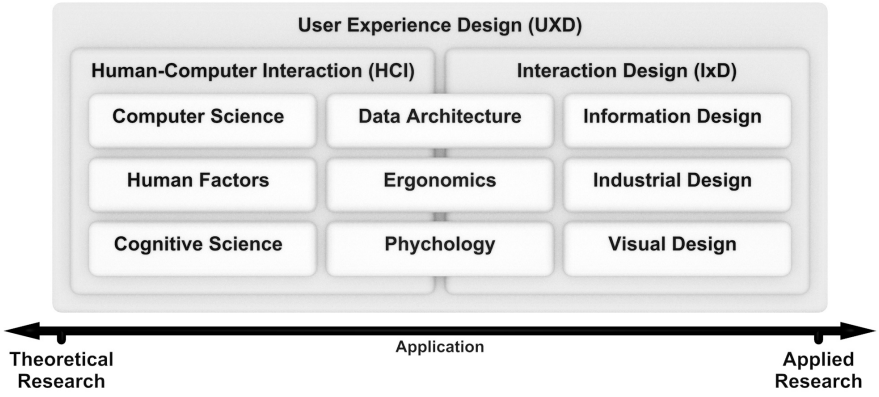

Figure 1.2: Framed within a larger logical construct know as User Experience Design, many scientific disciplines contribute to the creation of better UIs. However, due to the complexity of the whole problem, its resolution is often divided into two parts, with academic researchers focusing on the theoretical side (HCI) while industrial experts try to solve more practical issues (IxD).

1.2.2. Augmented Reality

Due to its popularization in recent years, the term *Augmented Reality* (AR) has been –over–used to describe a large set of technologies that combine virtual and real objects [9, 10]. Consequently, the establishment of a universally accepted definition for this term has become a very difficult problem for researchers and industry experts alike [11]. Nevertheless, by applying the User-Centered Design (UCD) principles in which this research is based, it is possible to formalize the definition of AR as *the integration of virtual objects into the user's perception of the physical world through the mediation of a computer system.*

A through analysis of this definition uncovers three relevant implications:

- **Integration of virtual objects:** The main goal of AR-based systems is to embed virtual elements into the real-world environment as seamlessly as possible. A process that normally involves the constant analysis of the physical surroundings and the real-time recreation of the digital asset.
- **Perception of the user:** Although the use of the word "Reality" in the term might suggest the opposite, to successfully sustain a successful AR experience it is not necessary to alter the physical environment; just the user's perception of it.
- **Mediation of a computer system:** While there are multiple ways to alter the perception of reality in a controller manner [12, 13], the use of modern computer systems offers significant benefits in terms of usability, portability and reliability.

Within a AR experience, each sensory (visual, auditory, tactile and/or olfactory) representation of the virtual object is called an *Augmentation*. These augmentations can be positioned relative to the view of the user (typically, in the corners of the screen), associated to a particular object in the environment(a *Fiducial Marker*) or linked to specific geolocation coordinates (commonly referred to as *Point Of Interest or POI*).

To successfully sustain AR-based experiences it is necessary to perform a real-time analysis of the physical environment to determinate the position of the relevant interaction components. This process, called *tracking*, usually involves the estimation of the relative position and orientation of the computer system with respect to a fiducial marker using the video feed from a camera. However, as three-dimensional scanning hardware and software solutions increase in capabilities and become more ubiquitous, it is possible to analyze natural objects (such as the body of the user) and incorporate them into the interaction space, facilitating the creation of *Spatial Navigation* mechanisms.

While the inclusion of virtual objects over real-world environments can be traced back to the World War II (when naval fire-control systems started including radar technology and mechanical calculators [14]) and has been a common element in many military *Heads-Up Displays (HUDs)* ever since, it was not until 1990 that the term was formally coined by Tom Caudell [15, 9]. Moreover, it took 5 years until the concept was properly delimited within a taxonomy, commonly referred to as the Reality-Virtuality Continuum [16].

This continuous scale introduces the idea of *Mixed Reality (MR)*, an entity that encompasses the complete spectrum of combinations between the physical world and a completely virtual environment. Within this continuum, AR is defined as the subspace that incorporates virtual components over the physical environment (as opposed to *Augmented Virtuality* or *AV*, that employs a virtual setting and integrates real objects into it).

This taxonomy was later expanden into a two dimensional plane with the incorporation of an additional axis to signify the degree of alteration (or *mediality*) to which the base environment is subjected to [17]. As can be seen in figure 1.3, this new taxonomic dimension allows the creation od configurations that where not previously contemplated. Most notably, *Modulated Reality* techniques offer the possibility of incorporating aspects of the environment that the users can not normally perceive (e.g., frequencies of the electromagnetic spectrum outside the range of visible light, sound frequencies outside the human hearing range, etc.), while *Diminished Reality* techniques discard elements that the user prefers not to perceive (an interesting approach for the treatment of phobias [18, 19] and to help overcome specific disabilities [20, 21]).

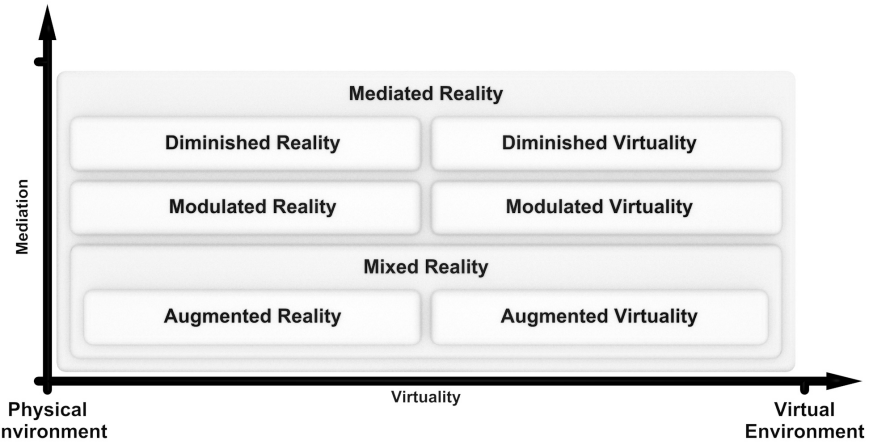

Figure 1.3: The Reality-Virtuality Continuum (horizontal axis) allowed the definition of a logical boundary for the concept of AR, framing it within a larger entity called Mixed Reality. Later, the definition of a "mediation" dimension (vertical axis) was introduced to include other types of perception alteration mechanisms.

1.3. Hypothesis and Objectives

Before elucidating the methodology employed in the research, it is necessary to enunciate its main hypothesis and the specific objectives that can be derived from it.

1.3.1. Main Hypothesis

The logical foundation over which the entire research effort is based on can be summarized in the following statement:

> ***It is possible to construct a comprehensive Human-Computer Interaction model for Augmented Reality systems, based on Spatial User Interfaces.***

By analyzing the three different concepts that conform this hypothesis it is possible to gain a better understanding of the scope –and boundaries– of the research work.

- **Comprehensive HCI Model:** Defines the main result of the research project; a logical structure that, rather than providing a complete set of widgets for every situation (a rather impossible goal to achieve), offers an extensive and extensible set of interaction techniques that can be easily adapted to the user context.
- **Augmented Reality system:** Describes the type of computer platforms over whose the resulting HCI model must operate. Due to the special nature of AR-based interaction spaces (in which real and virtual objects coexist), the application of the proposed model implies the use of specialized hardware to recognize the physical environment.
- **Spatial User Interface:** The practical outcome of this research is a novel three-dimensional UI system in which the interactive objects are –perceived as– embedded into the real world. A solution that offers a more mobile and immersive way to interact with computer systems.

Albeit this conceptual segmentation is useful to discern the different components that conform the research work, it is important to note that, due to the holistic nature of an interaction model, it does not comprise its entire scope. For example, it does not reflect the challenges of integrating the virtual objects of the user interface in the physical environment perceived by the Augmented Reality System, nor addresses how the HCI model should be adapted to the user context. All these considerations will be examined in detail in subsequent chapters.

1.3.2. Objectives

To properly conduct this research project, it is paramount to translate the aforementioned hypothesis into measurable, well-defined goals that encompass the research areas to explore. This transformation results into the following three *specific objectives*:

- ☐ **Examination of the state of the art:** Prior to present any new contribution, it is essential to study the different paradigms that have been developed over time (including the hardware and software solutions derived from them). Special attention must be paid to those interaction techniques that can be extrapolated to SUIs.
- ☐ **Formulation of an interaction model:** As the main result of the research, the proposed communication archetype necessitates a solid foundation. An objective that can only be achieved through the combination of different views of the problem domain; from the purely theoretical to its translation into software data structures. Additionally, this model has to contemplate the information exchange with other computer systems and define a common description language to facilitate it.
- ☐ **Validation of the proposed model:** With all the pieces in place, the entire model must be evaluated to ensure that it complies with the initial specification. However, while some requirements can be directly ratified by the very definition of the model itself (i.e., those that stipulate the support of a specific feature), many provisions need to be validated through the proper study of different usability metrics.

The practical results of these objectives can, in turn, be translated into the following *operational objectives*:

1. **Creation of a comprehensive set of interaction mechanisms:** The main product of the literature review is a list of interaction mechanisms that the proposed HCI model must support –or, at least emulate–. An enumeration that places particular emphasis on the new interaction possibilities inherent to AR systems (i.e., three-dimensional detection and tracking of real objects, including the body of the user).
2. **Implementation of the interaction model:** After its formal definition, the proposed model has to be adequately evaluated on a custom-made software platform. One capable of parsing the aforementioned definition language into user interfaces and providing the necessary tools to test each subset of interaction techniques independently.
3. **Design and execution of a usability study:** Once the interaction model proves to be robust enough, it has to be validated through the conduction of extensive usability tests.

1.4. Research Methodology

The large number of scientific disciplines associated with HCI and the rapid hardware evolution of AR systems led the author of this dissertation to employ an iterative and incremental methodology for this research project. Illustrated in Figure 1.4, this approach combines the Agile Software development principles [22] with the scientific method to facilitate the progressive exploration of the different subjects (without becoming overwhelmed by the abundance of interrelated concepts).

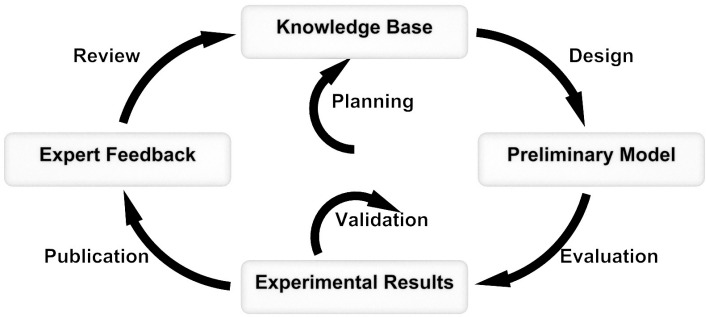

Figure 1.4: The methodology employed in the research project.

The main phases of this methodology are:

1. **Planning:** After gaining the necessary knowledge to fully understand the problem, the research project begins with the formulation of the main hypothesis and the creation of an initial plan to validate it.
2. **Design:** Each iteration of the methodology (re)starts with the selection of the new concepts that have to be incorporated into the preliminary HCI model (even if it requires the recreation of its entire structure).
3. **Evaluation:** Once the theoretical foundations of the preliminary model are solid enough, they must be tested through several evaluation methods to determine their validity (both individually and combined).
4. **Publication:** Independently of the degree of completeness of the preliminary model, the experimental results should be presented regularly to the scientific and industrial community to obtain expert feedback.
5. **Review:** The information obtained from the experiments and expert feedback must be closely examined to identify new concepts (and other research fields) that need to be incorporated into the knowledge base.
6. **Validation:** If the experimental results satisfies all the conditions established in the planning phase, the hypothesis will be considered valid and, therefore, the research project will come to a successful conclusion.

1.5. Document Outline

To facilitate its reading, this dissertation is divided into five chapters:

Chapter 1: Introduction

After presenting the context that motivated this research project, this introductory chapter explains the basic concepts over whose it is based upon and the methodology that was applied throughout the entire study. But, most important, it also contains the formal definition of the main hypothesis of the research and the associated objectives that need to be achieved to validate it.

Chapter 2: State of the Art

To allow the reader to gain a deeper understanding of the current state of HCI, this chapter reviews the different paradigms that have been developed over the last decades. This leads to another section that examines the current limitations while showcasing expected advancements that might help to overcome them in the near future. Afterwards, two sections are dedicated to analyze the current interaction mechanisms from both hardware and software-centered perspectives.

Chapter 3: Interaction Model

The proposed HCI model is presented in the third chapter of this dissertation, where each of its three sections focus on one of its representations. The first one introduces the model from a purely theoretical perspective, establishing its different components and the possible interaction configurations and techniques that can be generated from them. The next section proposes an object representation that can be used to properly implement the theoretical model and, finally, the last section presents a communication scheme that allows the information exchange in a reliable and secure way.

Chapter 4: Validation

This chapter describes the evaluation system constructed to properly validate the main hypothesis of the dissertation. The first section provides a comprehensive view of the methodology employed, while the next two sections summarize the planning and design phases of the usability study. Then, the chapter concludes with the detailed enumeration and the meticulous analysis of the experimental results.

Chapter 5: Conclusions

The last chapter of this dissertation reviews the entire research project (including the related contributions and the expert feedback they received) and presets the venues of future work that can derive from it. The final section remarks several issues that have to be taken into consideration when applying the proposed HCI model in the contemporary society.

"Wanderer, there is no path;
the path is made by walking.
By walking one makes the path,
and, upon glancing behind,
one witness the path
that never will be trod again.
Wanderer, there is no path;
only wakes upon the sea."

Antonio Machado

2

State of the Art

Before attempting the construction of an interaction model, it is of the utmost importance to review the history of HCI to gain a deeper understanding of the concepts and motivations that have driven this scientific field to its current state. Due to the practical –technological– applications associated to this research area, however, such study should not be limited to the associated literature, but also take into consideration the actual hardware/software products and analyze them independently.

This realization constitutes the starting point of this chapter. Section 2.1 reviews the evolution of HCI through the different generations of user interfaces, detailing the historical context in which they originated. A broader perspective facilitates the identification of the current limitations in section 2.2. Later, section 2.3 evaluates the current interaction techniques, while section 2.4 describes the input/output hardware devices that are commonly used to perform them.

Additionally, it is worth noting that the historical perspective that this chapter offers is not only intended to serve as a reference frame for the research project at the time of writing this document (October 2015), but also allow readers from a distant future to better comprehend the situation and motivations behind this endeavor.

2.1. The Evolution of HCI

Throughout the relatively short history of computer science, one of the research fields that has undergone more –and more radical– changes is Human-Computer Interaction. And, although most hardware and software related concepts remain unaffected by the passage of time, the study of communication between human beings and computer systems has experienced multiple paradigm shifts over the last decades [23].

In its continuous quest for interaction techniques with better usability/performance/cost ratios, HCI has been incorporating advancements from numerous disciplines: computer science, information visualization, psychology, human factors (cognitive/physical ergonomics) and industrial design among many others. A conceptual amalgamation that has led to a "stepped" (punctuation equilibrium-based [24]) evolution in which a new branch appeared regularly, as the different research lines converged to overcome the limitations of previous interaction models.

This stepped evolution, with a paradigm swift nearly every 20 years, can be clearly seen in figure 2.1. It is relevant to note, however, that this graph does not reflect the –exponential– increment in number of users that accompanied the adoption of each new HCI model. While they are not completely replaced, the progressive simplification of the interaction techniques facilitated the introduction of computer systems in more and more diverse environments. A progression that can be observed in figures 2.2, 2.4, 2.6 and 2.8, starting at military institutions and universities (with a user base limited to a select few experts) and currently reaching most homes and businesses across the globe (with thousands of millions of potential users).

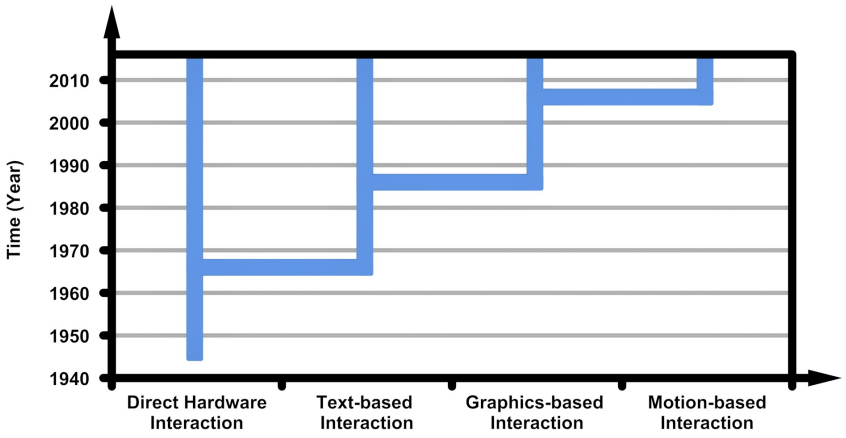

Figure 2.1: The evolution of HCI follows a stepped pattern, with a new model appears approximately every twenty years.

2.1.1. Direct Hardware Interaction

Devices to aid computation have existed for thousands of years. Abacus have been used for arithmetic calculations since, at least, 2400 BC and the discovery of the Antikythera mechanism prove that analog computers (for the calculation of astronomical positions) were already in use *circa* 100 BC [25]. Posterior advancements in mathematics and manufacturing allowing the creation of more sophisticated calculating instruments [26]. However, it was not until the early 19th century that mechanical computers were designed for more than a particular activity [27].

After working on its now-renowned Difference Engine, Charles Baggage proposed in 1837 the Analytical Engine, the first mechanical general-purpose computer. Although it was not actually built until the twentieth century, its design greatly influenced the creation of the first modern computers [28]; most notably, the (electromechanical, programmable) Z1 in 1938, the (electric, programmable) Colossus and the (digital, non-programmable) ABC in 1943 and the (digital, programmable) ENIAC in 1946.

As a result of this influence, interacting with these devices generally implied manipulating the hardware itself, either directly or through the use of external mechanical devices. As can be seen in Figure 2.2, these mechanisms include switch panels to program the system and contrived devices to introduce the data and obtain the results. In fact, the term *computer* was originally coined to describe the job of the women responsible of operating the ENIAC (a difficult labor awarded with the 2008 IEEE Computer Pioneer Award [29]).

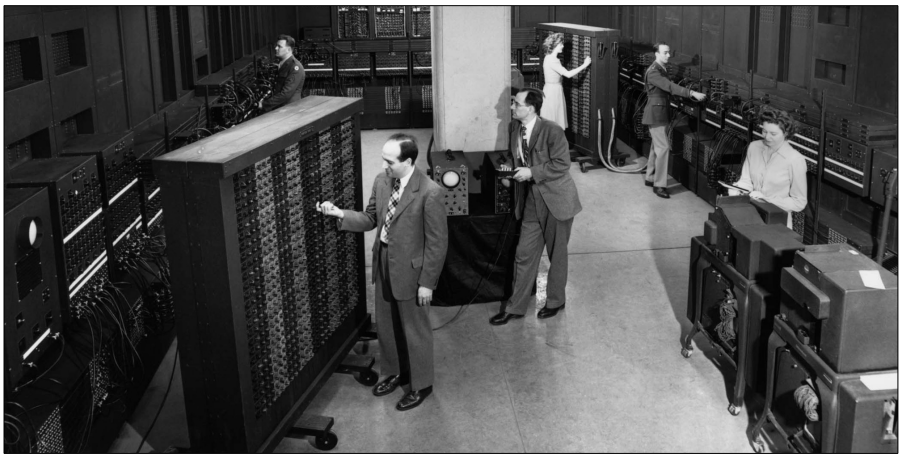

Figure 2.2: J. Presper Eckert and J.W. Mauchly working on the ENIAC, at the University of Pennsylvania in January, 1946. The Portable Function Table (featured in the foreground) contained a series of switches that allowed the operators to modify the functioning of the machine. The punch card reader and writer (bottom-right corner) allowed the data input/output operations.

Since then, computer systems have evolved into extremely complex machines that cannot be directly manipulated (to prevent electrical errors and protect the data transfer). Nowadays, all operating systems provide a *Hardware Abstraction Layer* (HAL) subsystem to shield applications developers from this complexity, providing them with a generic set of functions that allow an easy access to the different devices.

Nevertheless, the design principles from that era still remain in many specialized I/O devices. Although many industrial and consumer electronics manufacturers progressively introduce additional mechanisms to interact with their products (e.g., multi-touch screens, voice recognition, motion tracking, etc.), a significant portion of the user base still prefers to employ external input devices (like those present in Figure 2.3) to perform the most demanding tasks, due to the increased tactile feedback and reliability that physical controls (buttons, switches, rotary dials, joysticks, etc.) provide.

However, as the number of interaction possibilities increase, many of these external devices become so complex and convoluted that many new users are reluctant to take full advantage of them or, in extreme cases, refuse to interact with the computer system entirely (a conduct that has contributed to worsen the effect known *Digital Divide* between different age groups and economic sectors). A situation that manufactures have been trying to mitigate by increasing the general ergonomy, reducing the number of specialized controls, and, even, integrating multi-touch screens in the external devices.

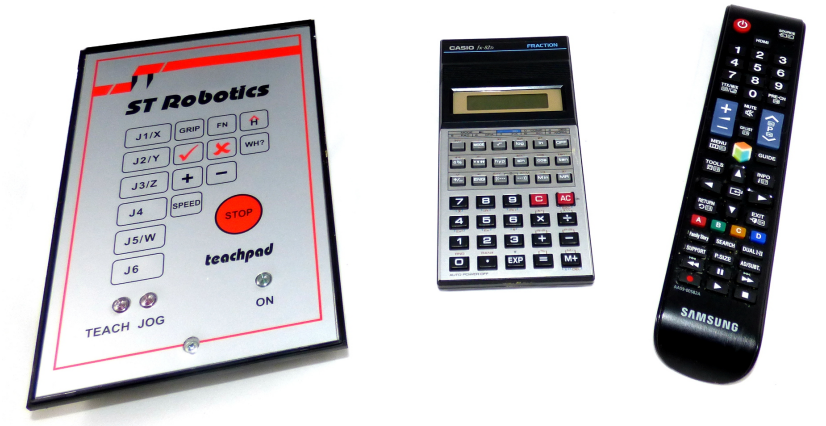

Figure 2.3: Many of the current controllers for industrial machinery (left), small electronic devices (center) and home appliances (right) share some of the interaction mechanisms with the input devices of the first computer systems. Please note that, as the complexity of the device increases, the manufacturers modify the size and color of the buttons in an attempt to provide a better visual and tactile feedback.

2.1.2. Text-based Interaction

While the use of keyboards for data input predates even the ENIAC (the Z1 mechanical computer already included a special set of keys to input alphanumeric values in 1938 [30]), its usage as input devices was only popularized after the commercialization of –moderately prized– electromechanical teleprinters in the early 1960s. Most notably, the introduction of the Teletype Corporation's Model 33 in 1963 represented a turning point for many companies, that, over the course of the following years, replaced the punch card and punched paper-based devices for these types of terminals to communicate with their computer systems.

This new input/output device, coupled with the time-sharing mechanisms that the operating systems of this time began to incorporate, allowed the creation of a new interaction model; one that tried to replicate the verbal communication between human beings. By typing simple commands in the form of verb+object, users could perform a large set of tasks and receive instant feedback printed on paper, in a human-readable format.

However, it was not until the early 1970s, when the popularization of the first computer monitors (also known as "Glass Teletypes" or "Visual Display Units" at the time) allowed a really interactive user experience. Apart from basic –albeit extremely powerful– Command-Line Interfaces (CLI), it was possible to manipulate the output to develop full-screen interaction spaces using ASCII symbols that simulated control panels. Text-based menu interfaces that enabled the users to easily navigate between different controls and operate the respective tasks without having to learn the specific terms and parameter lists of the commands. Later advancements in display technology allowed the use of a palette of multiple colors and higher refresh rates.

Figure 2.4: An ASR-33 teletype modified to interact with a time-sharing computer system (left) and a DEC VT100 (right), one of the first –dumb– terminals that supported ANSI escape codes (useful for text-based menu navigation).

At the moment of writing this document, command-line and text-based menu interfaces are still present in many work environments –albeit usually emulated over a graphic system– due to their reliability, powerfulness and minimal memory footprint. However, the steep learning curve associated with many CLIs (that often require the precise usage of large sets of one-letter parameters), problematic localization processes [31] and the general lack of aesthetically pleasing widgets are common reasons to justify the progressive substitution of these systems. Furthermore, the need to include a keyboard poses important usability problems: physical keyboards have fixed key layouts and their continuous use can lead to serious physical problems (i.e., muscular fatigue and carpal tunnel syndrome). On the other hand, virtual –soft– keyboards offer a wide range of interesting features (from special key layouts for emoticons, to integrated spell checkers and word completion systems) but their poor tactile feedback, forces the user to pay constant attention to the keyboard (and not to the text itself).

Nowadays, text-based interfaces are generally created as backup systems such as debug consoles and Basic Input Output System (BIOS) menus. Nevertheless, as Figure 2.5 shows, the legacy of this interaction model is still present in current web search engines and personal assistants. In fact, as Natural Language Processing and Voice Recognition technologies evolve and become more prevalent in society, it is expected that more computer systems incorporate interfaces that combine text and voice input/output, following the original notion of employing common –human– conversational patterns to communicate with computers.

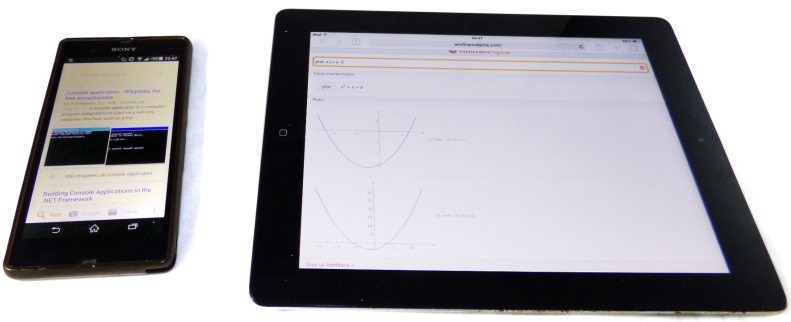

Figure 2.5: Search engines and personal assistant applications employ a sintax reminiscent of text-based interaction models.

2.1.3. Graphics-based Interaction

The rapid evolution that experimented visual displays in the early 1970s motivated the research of interaction mechanisms that were able to take full advantage of this technology to present graphical information in a more compelling way. An effort that culminated in the development of the Xerox Alto in 1973, a revolutionary computer that was not only one of the first systems to incorporate a mouse-driven Graphic User Interface (GUI), but also introduced the *desktop* metaphor [32].

Even though the Xerox Alto was not sold commercially, its influence can be traced to this day. As one of the first general-purpose computer systems designed to be operated directly by the end-user, it helped to popularize the concept of "personal computing" (at a time when time-sharing mainframes were the norm), but it was the advancements in software design that made this system so influential. Thanks to its capability to display raster graphics, it enabled the user to easily create new contents in WYSIWYG (What You See Is What You Get) applications.

Nevertheless, it was not until the introduction of the Xerox Star in 1981 that the general public was able to experiment with the IO hardware configuration and metaphor-based interaction that would come to characterize *personal computers* and *workstations* for the following decades [33]. In fact, the mouse went from being considered an optional peripheral to the main input device to perform selection and navigation tasks within the system (through the indirect control of a bidimensional arrow called *cursor* or *pointer*). This simpler control system, coupled with the ability to transmit concepts trough visual representations (i.e., icons), facilitated the creation of more attractive and efficient user interfaces, which, in turn, spurred the adoption of this type of computers in all kinds of environments at the end of the last century.

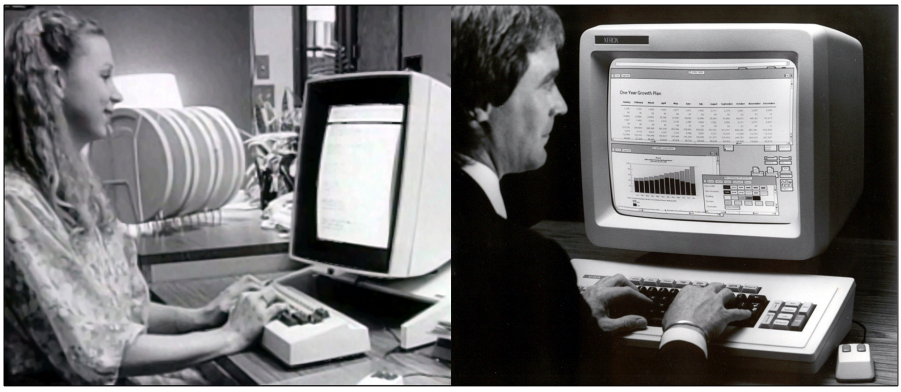

Figure 2.6: The introduction of Xerox Alto (left) in 1973 marked a turning point for HCI, while its succesor, the Xerox Star (right), redefined the standard for *personal computers* in 1981.

Due to the limited resolution of the displays, the user interfaces were initially designed in a similar way to text-based menus, taking the entire screen and necessitating of special mechanisms to switch between applications (and, worse yet, between different modes of the same application). However, as the screen technology –and computation capabilities– improved, multiple interaction elements (often referred to with the acronym WIMP, for *Window, Icon, Menu and Pointer*) were progressively standardized and included within the operating system code, greatly enhancing the effectiveness and consistency of the user interfaces.

In this model, *windows* are special widgets that not only contain the user interface elements associated with a specific program, but also control its very execution (to the point that it is assumed that "closing" a window implies the termination of the related process). And although their location and size can often be adjusted to avoid superpositions, for complex tasks that demand the simultaneous use of several applications, it is rather common that entire windows end completely hidden from the users' view. For these occasions, the operating system must provide the user with mechanisms (generally, an icon bar with an item for each running program) to reorder the different virtual planes in whose the windows are placed (and, in fact, this partition of the screen space has become so prevalent in current operating systems that the entire user interaction is handled by a subsystem called *Window Manager*).

As can be seen in Figure 2.7, more complex applications can employ modal windows and tabbed sections to further divide the screen space depending on the present context of the user. A difficult and ever-changing equilibrium between functionality and usability for which, sometimes, the only viable solution is to acquire another monitor.

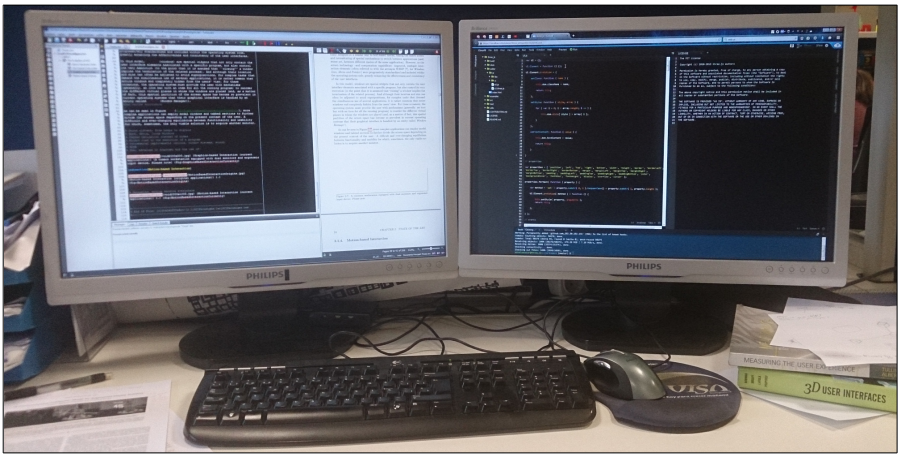

Figure 2.7: A common workstation equipped with dual monitors and ergonomic input device. Please note the use of tabs and partitions to divide the available window space into different sections.

2.1.4. Motion-based Interaction

Although the massive adoption of window-based user interfaces allowed the creation of a successful *digital ecosystem* (a sustainable self-organization of hardware manufacturers, software developers and end users), its success ultimately resulted in a period of relative stagnation for the HCI field. While the research was still being conducted at a good pace, the technological advancements that actually reached the market were few and far between, focusing more in minor improvements in ergonomy and display technology rather than actually improving the interaction mechanisms. Even the structure of web pages (initially designed to contain static documents), was progressively modified to mimic the "*look&feel*" of window-based user interfaces.

All this changed with the introduction of the Nintendo Wii motion enabled controller (colloquially known as *Wiimote*) in 2006 and the presentation of the Apple iPhone a year later. The former featured a streamlined design that eliminated analog joysticks in favor of motion-sensing capabilities, providing a simple way to interact with(in) tree-dimensional environments [34]. Meanwhile, the iPhone featured a multi-touch screen as its main interaction mechanism, completely removing the physical keyboards prominent in previous smartphone designs.

Although similar approaches had been attempted before [35, 36], it were the interaction techniques that accompanied these hardware devices what ultimately determined their commercial success (and the reason for why they represent turning points for their respective industries). Thanks to their simplicity and intuitiveness, these interaction models layered the foundation for a new interaction paradigm based on the natural motion of the human body, allowing the participation of a larger sector of the population (as illustrated in Figure 2.8).

Figure 2.8: The interaction models based on the natural motion of the body are capable of eliminating many of the barriers that prevented the operation of users of all ages and abilities.

The success of both approaches resulted in their almost universal adoption in the following years [37]. Even the products of the two companies that caused this paradigm swift suffered from crossbreeding; with the iPhone including motion sensors since its fourth iteration and Nintendo integrating a touchscreen in the gamepad of their next videogame console.

Moreover, the good reception that these innovations had in the market sparked a new technological race between the main players in the industry, sparing no effort in search of more intuitive interaction techniques. As a result, many of the related fields of study experienced a rapid growth and many of the, until then, obscure research projects were brought to light. Most notably, the launch of Microsoft Kinect in 2010 popularized the use of depth sensing cameras to register the full-body motion of the users (something that the high cost of these devices had prevented until that point).

Although there have been several attempts to standardize these new interaction mechanisms [38] fo facilitate the creation of user interfaces, due to the diversity and diversification of these devices, these efforts showed mixed results. While the basic collection of multi-touch gestures introduced with the first iPhone have become a de-facto standard and is widely employed [39], the complexity of the human body has, so far, prevented the establishment of a common set of recognizable motion-based inputs (forcing developers to devise a different scheme for each application and, consequently, burdening the end users with learning processes) [40]. Some manufacturers have tried to address the issue by attempting to translate the multi-touch gesture system to a three-dimensional space but have only found limited success [41, 42].

Another important issue that plays against the use of these kinds of interaction mechanisms is that, due to the physical limitations of the human body, they do not provide the precision required for the most demanding use cases (i.e., Computer-Aided Design applications).

Figure 2.9: With the introduction of Kinect (top), Microsoft experimented to create more intuitive and immersive user experiences by tracking the body of the users. Meanwhile, its rival Sony tried to emulate the success of other platforms by integrating motion-sensing capabilies and, later, touchpads into their game controllers.

2.2. Current Situation

As history shows, the general acceptance of new HCI models is a complicated and long process. While their "origin" can be traced back to the introduction of a disruptive product in the market, their actual inception is based on many years of research and their widespread application often requires large transition periods (in which manufacturers and software developers look for better ways to implement previous tasks with new interaction techniques.

At the time of writing this document (October 2015), the software industry is still in the middle of the transition process between graphics-based and motion-based interaction models, with more sophisticated applications becoming available for smartphones and tablets each passing month [43]. However, the difficulty in translating the most demanding interaction techniques (in terms of precision and time dedication required) to tactile screens and body motion recognition systems still makes these new computer platforms unfit for many industrial applications (i.e., Computer-Assisted Design, numerical control and, most important, software development).

This situation has led to a –not always harmonious– coexistence of multiple interaction paradigms, each one offering a different approach for the same task. And, while there have been many attempts to combine those approaches into a single computer platform (some more successful than others [44]), most end-users prefer to employ different specialized devices whenever the tasks they have to perform maintain a strong logical cohesion. Nowadays, it is a common practice –in the First World– to employ desktop or portable computers to work efficiently (graphic-based interaction), smartphones or tablets for web browsing and telecommunication (motion-based interaction) and game consoles or smart TVs to consume media contents (a mixture of multiple interaction paradigms).

This section, much like a SWOT analysis, offers a general view of the current status quo and attempts to identify and evaluate the most important circumstances that define it.

2.2.1. Fragmented User Experiences

The large diversity of computer platforms currently available in the market has resulted in the creation of a myriad of single-purpose programs that, more often than not, offer very limited user experiences. Worse yet, many software developers (in an attempt to secure their user base and/or to impose a paid subscription model), employ custom data formats and communication protocols, making their applications not only incompatible with those of their competitors, but also between different versions of the applications (and, even, between different computer platforms).

Needless to say, these practices are very inconvenient for many end-users, whom, in order to complete specific tasks, are often forced to switch constantly between different applications and employ external utilities (i.e., clipboard managers and format converters). A process that gets even more cumbersome when dealing with different computer platforms, because their interaction mechanisms and functionality can vary drastically from one another.

Fortunately, nowadays, most desktop applications incorporate import/export options that greatly improve their interoperability and there are many online services specialized in the conversion between data structures. Nevertheless, these processes are far from perfect and often introduce their own series of incompatibility problems. Furthermore, the degree of user experience fragmentation keeps increasing as many developers are compelled to divide the functionality of their software suites into smaller, self-contained programs (commonly referred as *Apps*) in order to compete in mobile platforms.

Aware of these issues, many operating system developers are starting to include application interfaces (APIs) to enable the easy creation of small and reusable interactive elements (called *Widgets*) that end users can employ to enhance the functionality of the individual apps. However, the limited scope of the APIs and the poor reception among third-party developers has resulted in very restricted interoperability, which, in turn, has lead the end users to disregard them as mere *gimmicks*. Consequently, and with the notable exception of 3D manipulators and virtual keyboards in mobile platforms, widgets have been relegated to little more than animated icons to decorate the desktop/launcher.

2.2.2. Sobresaturated Software Distribution Platforms

The general increase in availability and speed of current networks connections (specially, in mobile platforms) have allowed the proliferation of new software distribution channels. Either trough web servers or specialized virtual stores, end users can access an almost incommensurable –and ever-growing– number of media contents (web pages, books, movies, etc.) and applications.

Contrary to the popular belief that "more options is always better" [45], the extremely large number of choices available at any time makes practically impossible for human beings to locate any specific content without relying on a distributed naming system. And even then, the number of elements to identify can exceed the capabilities of the naming system. One clear example of this problem is the Domain Name System employed to provide unique identifiers to computer systems (i.e., web servers) over the Internet. With nearly one billion registered websites at the time of writing this document [46], the Internet Corporation for Assigned Names and Numbers (ICANN) has had to extend multiple times the number of top-level domains available and change the text encoding from ASCII to UTF-8 [47], in order to cope with the demand of domain names (that can be easily memorized).

Nevertheless, due to the rigid naming conventions that these systems employ, most end-users prefer to use classification services and filtering mechanisms know as search engines to locate the desired resources. In fact, in most cultures that do not employ the latin alphabet, advertisements for websites and mobile apps tend to feature specific search terms rather that the actual identifier provided by the naming systems because they are easier to remember for most people.

These alternative mechanisms are not exempt from problems, however. Even without the controversial use of Search Engine Optimization (SEO) techniques, many unscrupulous developers take advantage of specific search terms to promote their software solutions (often employing similar names and graphic elements to confuse their victims into download malware). A problem that, in conjunction with the existence of multiple versions of the same app (full, demo, ad-supported, free to play, etc.) often makes the distribution systems in mobile platforms very difficult to navigate and find a meaningful user experience.

All this problems lead to a very hostile environment in which developers are often forced to launch new software continuously (to appear in the "featured" list in the different mobile stores) or invest heavily in the promotion of their products.

2.3. Current Interaction Techniques

In order to successfully operate computer systems, users must have at their disposal a simple but powerful set of interaction techniques. A combination of hardware mechanisms (input/output devices) and software components (generally called *user controls* or *widgets*) that create an interaction space where the communication between users and computer systems can take place in a fast and comfortable way.

To achieve this goal, the interaction space must be sustained over multiple abstraction layers, each providing the necessary logical separation to enable users to perform the different tasks with the minimum –physical and psychological– effort possible. In this way, it is possible to define a reduced, hardware-independent and, most importantly, *consistent* set of interaction techniques that can be used to complete each and every task.

Nevertheless, the definition of such logical architecture is far from being an easy task in itself. Several factors (e.g. number of individual commands, input speed, visual feedback, ease of learning, extension possibilities, etc.) need to be considered before attempting to specify the interaction techniques. Furthermore, in the case of AR-based systems, the current hardware limitations and the inherent complexity of the interaction space requires the reevaluation —and redefinition– of most of the mechanisms that have been applied in previous HCI models.

To properly address these issues, this section is dedicated to analyze the current interaction techniques and their possible translations into the physical, three-dimensional environments within which AR systems operate. Following the classification proposed by Doug Bowman [48], the interaction techniques presented in this section is divided into five different categories: selection, manipulation, navigation, symbolic input and system control. However, it is important to note that most user interfaces do not provide a single way to define the interaction mechanisms and that, depending on the combination of input/output devices that the computer system has, the usability, reliability and efficiency of the different techniques might vary greatly.

2.3.1. Selection

Although the popularization of motion-based interaction systems (specifically, those based on multi-touch screens) simplify the creation of interaction scenarios based on *Direct Manipulation* techniques, a significant number of current use cases still require that users *select* the elements that they want to interact with before allowing them to perform any other action. Beyond reducing the possibility of interacting with unintended entities, these selection techniques allow the user to specify, visualize and group the elements at different times and from different viewpoints. An essential mechanism for tasks that require working within complex interaction spaces (e.g., three-dimensional modeling, indoor route planning, etc.).

At the time of writing this dissertation, the vast majority of the UI systems display the different interactive elements in an orthogonal view, with bi-dimensional layouts that disregard –or directly deny– the possibility of superimposing them. This model makes selection a rather simple task that can be completed either by pointing directly towards the desired item (in the case of touch-screens and 3D scanners) or by defining a selection box/cursor over it (using a mouse or keyboard). Multiple selection is possible with additional input mechanism (e.g. dragging the mouse to draw a bigger selection area or pressing keys to switch between advanced selection modes). Figure 2.10 shows several examples of this kind of selection mechanisms.

Nevertheless, this system presents several limitations that make it inadequate for three-dimensional interaction space; the use of orthographic views do not allow for a realistic positioning of virtual elements within the physical environment (a process that requires additional perspective correction and sorting algorithms) and, since most user interface layouts have been designed to make the most out of a limited and static two-dimensional space on screen, they can easily hide –or make inaccessible– important interaction components situated behind them.

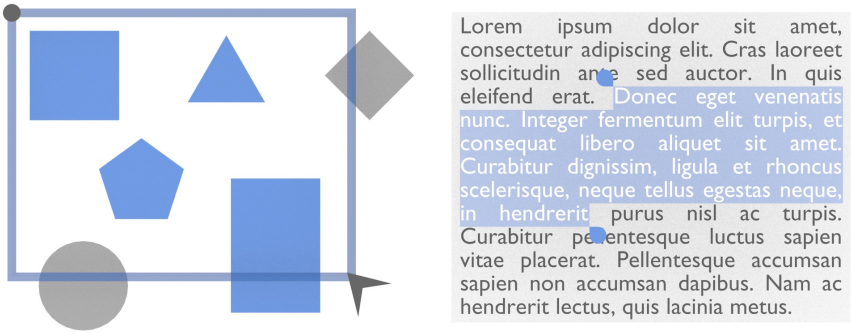

Figure 2.10: The most common selection techniques operate on a two-dimensional space (or an orthographic projection of a three-dimensional space).

This issue has been extensively studied in the literature and, while several solutions have been proposed [49], there is not a single interaction technique that can cover all possible situations. For this reason, instead of favoring one approach over the others, the interaction model must provide a flexible system that allows the user experience designers to define different –but consistent– selection techniques.

Even though this approach offers great versatility and adaptability, it is based on a rather simple technique called *ray-casting*. A basic pointing mechanism that takes two three-dimensional points specified by the user (either with a gaze/hand tracking system or, indirectly, with a 3d mouse) and generates a virtual ray that, once projected onto the virtual scene, determines what objects are to be selected (based on the collision points between the ray and the objects).

Using this pointing mechanism as a starting point, it is relatively easy to specify more complex selection techniques. By simply adding an angular value, it is possible to define a flashlight-based selection technique [50] and, by including another 3D pointing device, users can adjust the length and curvature of the ray [51], allowing the selection of partially obscured objects. A second pointer that can also be used as a nonisomorphic 3D rotation device in order to implement a World-In-Miniature [52] and Voodoo Doll-based [53] mechanisms that facilitate the selection of individual elements within complex aggregations.

To further facilitate the selection of nearby objects, the model must allow the definition of "virtual hand" techniques. In this way, the users will be directly select and manipulate virtual objects with a cursor that replicates the movements of the hand (or, in case of non-human users, a similar extremity). This virtual cursor can also be projected into the distance to reach objects situated far away from the user.

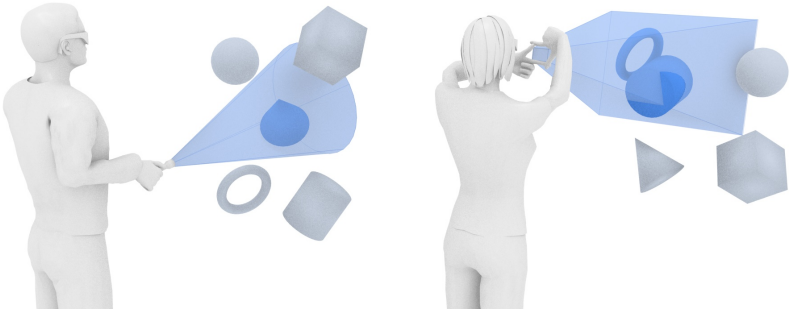

Figure 2.11: Either using a 3D mouse or by tracking the hands of the user, it is possible to create advanced interaction mechanisms that allow the selection of the interactive elements in a SUI.

2.3.2. Manipulation

Once the selection process is completed, users can perform multiple manipulation operations on the virtual objects (including the virtual representations of real entities) to alter their properties. Nevertheless, -and unlike selection- the modifications resulting from manipulation operations persist in time and, thus, their application requires special considerations (mainly, the implementation of undo/redo mechanisms and consistency maintenance tools for collaborative edition environments).

Although there are innumerable variations and combinations, all manipulation task can be reduced to tree basic operations: *translation*, *rotation* and *scale*. The advantage of this approach is that, since these three operations can be easily applied to the transformation matrix of each interactive element (hence their name; *transformation operations*), they can also be stored, transmitted and, when necessary, reverted. However, the actual definition of each operation is not always as straightforward.

While basic translation and rotation operations –using the theoretical center of mass of the virtual object as the pivot point– can be directly mapped to the movement of a 3D mouse, more advanced operations require the use of additional user controls. These manipulation widgets are usually displayed superimposed over the actual object, with different icons to indicate the associated transformation operation and color-coded axes to provide the user with a better spatial reference. As Figure 2.12 shows, there are two main types of manipulation widgets: those that are centered on an –independently defined– pivot point and those that are based on the bounding box.

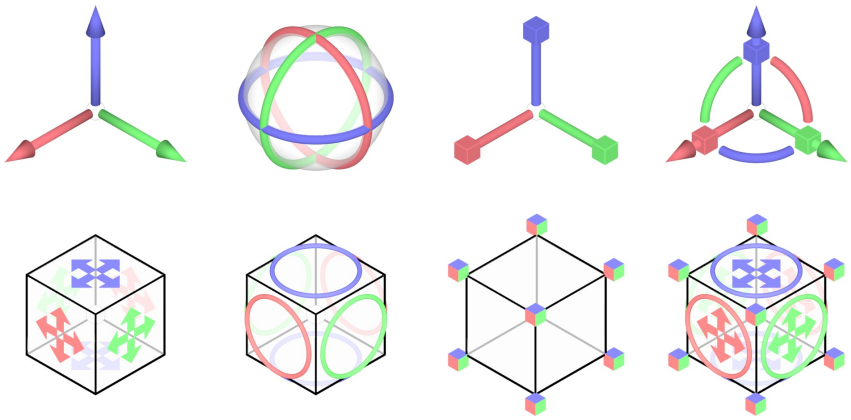

Figure 2.12: Manipulation widgets for transformation operations (from left to right: translation, rotation, scale and *combined*). The upper row shows pivot point-based widgets, whereas the widgets in the lower use the bounding box of the object.

The transformation widgets based on pivot points are, by far, the most used in current 3D modeling applications due to their simplicity and the degree of control over individual objects. Meanwhile, the bounding box-based widgets are the most common system in 2D design applications (including almost all UI editors), since they facilitate the alignment of visual elements and, thus, the organization of the contents into clear spatial layouts. In general terms, pivot point-based manipulation is a better option when dealing with a large number of small objects, whereas bounding box-based manipulation streamlines the management of a small amount of complex objects.

Since both manipulation styles have their own advantages and disadvantages, interaction models should not restrict the definition of manipulation techniques to a single option. Instead, user experience designers should be able to create manipulation mechanisms (with or without visible widgets) that combine both approaches.

Another important feature that an interaction model should include is the possibility of defining grids and smart guides to facilitate the spatial positioning of multiple objects (both physical and virtual in nature) while they are being manipulated. As figure 2.13 shows, this feature can be extremely useful for CAD applications (freeing users from having to manually input the exact values for each entity).

It is important to note that these manipulation techniques can also be applied to individual components of the virtual objects (vertices, faces, control curves, etc.). In addition, by combining this approach with advanced selection techniques (primarily, soft selection values), it is possible to define intuitive widgets for complex tasks such as 3D sculpting or terrain deformation.

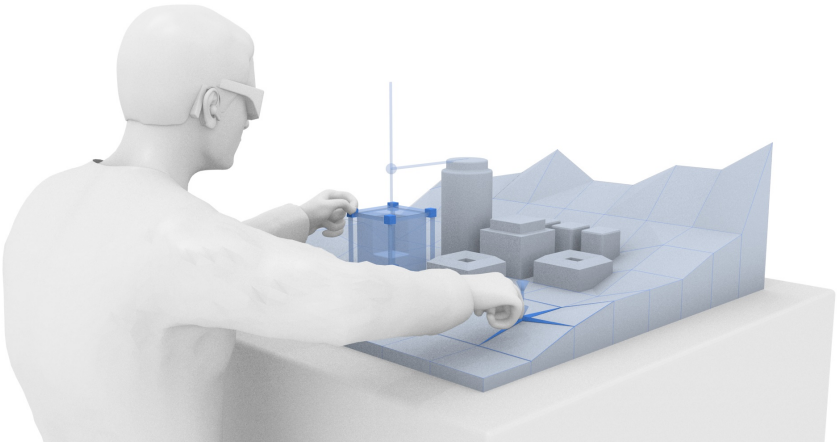

Figure 2.13: An example of a CAD use case where the user employs both manipulation styles to alter the properties of the virtual objects.

2.3.3. Navigation

In most common use cases, navigation is a simple task that only involves the movement of the user within a purely virtual space (although it can be argued that the point of view of the user remains static and are, in fact, the virtual objects that move around it). In the context of AR-based interaction spaces, however, the access to particular sections of the UI usually requires for the user to be located in a specific area within the physical environment. This approach, while greatly reducing the amount of information that the users need to process at any given time, also necessitates the implementation of additional navigation techniques. For this reason, it ois several *physical navigation* techniques that not only take into account the cognitive factors of the task (*wayfinding*), but also the associated motor component (*travel*).

In an AR-based experiences, wayfinding mechanisms must analyze the physical surroundings of the user and retrieve any information that can be useful to understand the context of the user (including his/her global position). This process can be initiated either automatically or manually by the user and, if a meaningful interactive element is located, its relative location should be displayed on a visual representation of the surroundings (a 2D/3D map centered on the user), along with options to filter the results and establish the best route to the destination.

To improve the suggested route, the wayfinding techniques should also take into consideration the preferences and personal background of the users (e.g. the number of times they have travel a similar route or whether the user has a physical disability that limits his/her mobility), the different means of transport at their disposal, and the physical factors of the environment (including weather and traffic conditions).

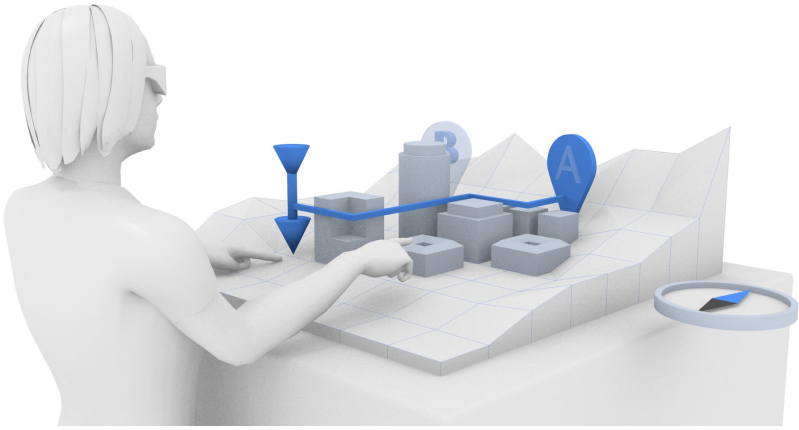

Figure 2.14: To adequately navigate in large three-dimensional spaces, it is necessary to provide the users with a wide variety of wayfinding techniques.

Once a route has been specified, the travel mechanisms can be activated. These navigation techniques assist the users during their movement, either by simply displaying the next waypoint or by pointing the user towards it. In the first case, a spatially integrated indicator may suffice but, if the physical location is outside the field of view of the user (occluded by other real objects, too far away or outside the viewing angle of the camera), additional navigation widgets might be required.

These physical navigation widgets may include diegenic elements that show the path to the next waypoint (such as virtual arrows on road intersections, footsteps on the floor or ghostly figures traveling along the path) and non-diegenic elements that simply indicate its general direction (e.g. compasses, 3D arrows or off-screen indicators). Other options include the creation of virtual sound sources to guide the users toward -or away from- one location and the use of haptic devices that send a signal to the users whenever they stray from the predefined path.

One important design consideration that should be taken into account when defining navigation widgets is that, as with any other element of the UI, they should not negatively interfere with the task itself. Since the very action of navigating the environment may put the physical security of users at risk, navigation widgets should be designed to be as unintrusive as possible, with a minimalistic design that only highlights the most relevant information (such as direction, distance or relative velocity), even disappearing completely when they are no longer needed. Whenever possible, this information should be conveyed using simple signals (color-coded shapes, monotone sounds, basic haptic feedback, etc.), rather than relaying on alphanumeric representations than require higher cognitive effort to be correctly interpreted by the user.

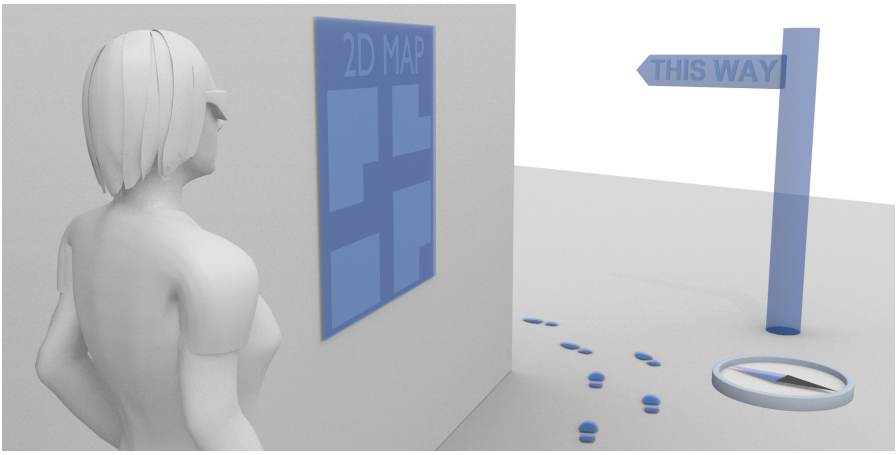

Figure 2.15: Diegenic widgets (e.g., street signs, footsteps and map) and not-diegenic elements (compass) can be combined to guide the users –safely– to their destination.

Symbolic Input

While the interaction techniques described up until this point provide an intuitive and meaningful way to communicate with computer systems, any information transmission model exclusively based on them will present clear limitations in terms of versatility and precision. Even with the most sophisticated three-dimensional positioning mechanisms and construction models, it is extremely difficult to convey real meaning and intentionality behind most communications by using only spatial information.

For this reason, along their history, most cultures have defined different codification systems to facilitate the transfer of knowledge between individuals. Although of very different nature (depending on the physical medium where the communication takes place), all of these codification systems rely on standardized sets of symbols that greatly simplify the information exchanges, while also reducing the occurrence of ambiguous or conflicting messages. Thanks to their existence, it is possible to express, transfer and store complex information (ideas, thoughts, feelings, emotions, etc.) in a relatively easy way.

Nevertheless, since computer systems do not –generally– work with these symbolic representations directly, additional hardware/software layers are required in order to adequately acquire and process them. When handling alphanumeric data, for example, the most practical solution is to present the user with a keyboard (either a physical device, a "soft" version on a tactile screen or a completely virtual recreation) that contains an individual key for every valid character/grapheme in that specific context. An input mechanism that can be complimented with writing assists such as spell and grammar checkers, thesaurus and automatic punctuation systems.

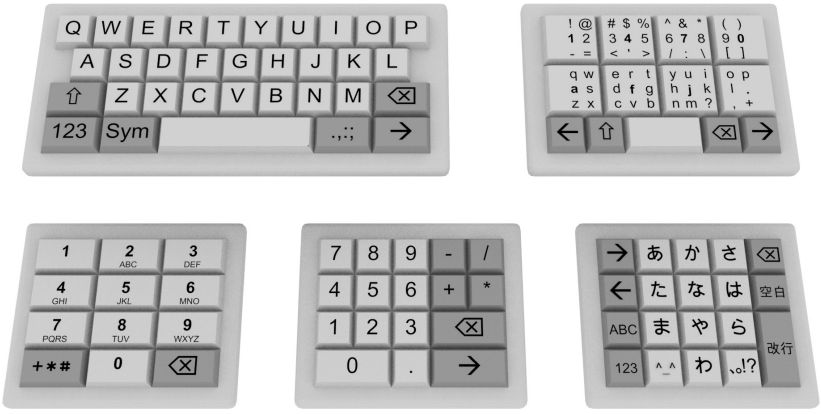

Figure 2.16: Several examples of "soft" keyboards, designed to facilitate alphanumeric input on mobile platforms.

Symbolic input interaction techniques, however, are not restricted to just alphanumeric values. By mixing selection and manipulation techniques, it is possible to construct widgets that ease the definition of complex data structures. This is achieved through the assemblage of basic interactive elements (sliders, spinners, pie menus, drop-down list, etc.) to specify the different data components. An important consideration in applications that require real-time responses since, by providing the users with a limited number of values for each individual data component, they have to easily input the necessary information without being overwhelmed by the large number of options at their disposal.

While many of these symbolic input methods are based on basic, standardized manipulation tasks (pushing a button, rotating a handle, etc.), they can also benefit from the inclusion of supplementary mechanisms that take into account alternative forms of expression. Either by recognizing hand gestures (including language signs and handwriting movements), human speech or corporal expressions, these mechanisms allow the users to input symbolic data without having to physically interact with the representation of the widgets.

Furthermore, by taking advantage of the sensors (i.e.,high resolution cameras, accelerometers, gyroscopes, etc.) and processing power of current mobile platforms, it is possible to create widgets capable of extracting symbolic data from the real world. Beyond performing visual recognition to identify machine-readable optical labels (as illustrated in figure 2.17), these mechanisms can also be used to extract three dimensional data from the environment (e.g., analyzing the face of the user to create a realistic avatar or to validate the identity of the user) and, even, to establish new communication channels with other users (e.g., using the flash of the camera in to transmit Morse-encoded distress calls over long distances).

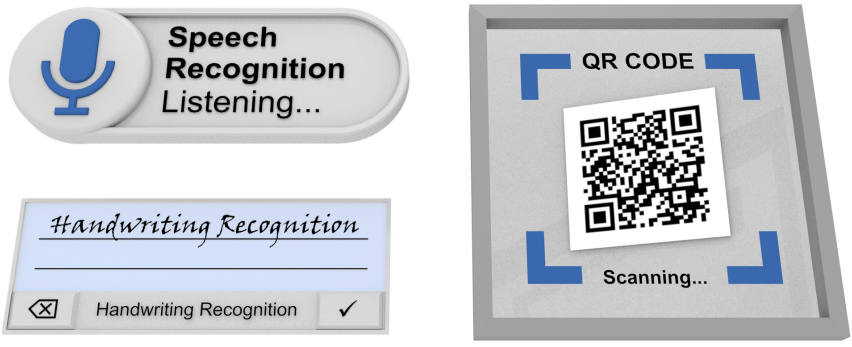

Figure 2.17: Several examples of widgets capable of extracting symbolic data from the physical world.

System Control

In order to provide a satisfactory user experience, most operating systems incorporate special interaction mechanisms that allow the users to manage the execution of the system; from switching between different application contexts (through a specific key combination, hand gesture or voice command) to managing the different hardware devices that make up the computer system (via a series of complex menus). While these mechanisms often require a certain degree of expertise to use them correctly, their inclusion at the operating system level is of critical importance (specially, when they become the only option to recover from an application error that block any other interaction mechanism).

Sadly, one of the main challenges associated with the design of AR-based reality-based UIs is the definition of the system control elements. This is due to the contradictory requirement of providing the users with an easy access to a wide range of functionalities of the operating system without interfering with the interaction task itself. Since, in this kind of computer systems, the users perceive the physical reality through the same devices that present the augmentations, the spatial positioning of the interactive virtual elements can easily distract the attention of the users or, directly, occlude elements of the environment that are critical for the successful completion of the task. A problem further aggravated by the relatively small resolution and FOV of many current displays.

The most common solution to this problem is to position the system control widgets within the peripheral regions of the user's perception (the corners of the screen), leaving the rest of the visual space for the users to directly interact with the augmented environment. Only when a contextual option is directly requested by the user (e.g. showing a virtual keyboard to input text messages, displaying a context-sensitive menu upon selecting a real object, etc.) or an important interaction requires the full attention of the users (such as a response to an impending danger to their physical security), the widgets may position themselves in a more prominent location.

In any case, the user interface manager should be able to determinate what elements are relevant for the user at any given time, rearranging them whenever necessary to provide an easy access to the main functions. Additionally, depending on the input devices available, different shortcut mechanisms (key combinations, hand gestures, voice commands, etc.) should be provided for expert users.

2.4. Current Interaction Hardware

As the first section of this chapter shows, even at the advent of modern computing, the presence of specialized I/O hardware mechanisms allowed users to interact with the system more efficiently and comfortably. A concept that was later labeled as "peripheral" and redefined to encompass any *external device that could be used to get information from or put information into a computer system.*

Over the course of the last decades, as the original punch card readers/writers and switch panels were been relegated to oblivion, the introduction of new and more versatile peripherals have enabled the development of more intuitive an efficient interaction techniques. A technological evolution that, nowadays, has resulted in software platforms that can be controlled simply with body motions or –a limited subset of– human speech.

However, it is important to realize that these advancements do not happen by themselves, but are the result of extensive research and engineering. That, behind the design of each peripheral, there is a lot of applied knowledge that needs to be carefully studied to fully understand the circumstances that motivated its inception (and those of their introduction in the market).

Nevertheless, there are many factors to take into consideration in this analysis, including the following:

- **Modality:** The type of information that each device provides to or retrieves from the computer system.
- **Ergonomy:** Indicates how the users physically interact with the device (in terms of operational speed and the amount of effort required to operate it).
- **Portability:** Specifies whether the users can carry the hardware device with them or not (and if so, if it has to be connected trough a cable or a wireless communication can be established).
- **Reliability:** The conditions (either internal or external) that might negatively affect the operation of the hardware device.

As peripherals become progressively more complex and diverse (to the point that the separation line between input and output devices gets fuzzier every day) these factors are not only relevant by themselves but, as will be shown throughout this section, are necessary to establish a proper classification.

2.4.1. Input Devices

A computer system must provide the users with –at least– one specialized hardware mechanism that enables them to control its operation and input data structures. And although most times this communication will be discrete in nature, to sustain a good (interactive) user experience, the physical mechanisms must always register the different signals accurately and transmit them as fast as possible to the Central Processing Unit.

Because of their importance, the design of more efficient and ergonomic input devices has become one of the main focuses of HCI research. An effort that has resulted in both improvements in common peripherals and the creation of novel ways to interact with computers [54]. Yet, many important issues still remain unsolved [55].

While, in recent years, there have been many successful attempts to simplify the design of input devices (to reduce the learning process and, overall, create more intuitive interaction techniques) [37], most manufacturers tend to fill their products with –often redundant– physical mechanisms to differentiate themselves in a glutted market. And although some of these *advanced controls* might make sense for specific purposes (e.g., *gaming* and graphic design), their presence in common computer systems is often counterproductive and might even widen the *Digital Divide*.

On the other hand, the standardization of communication interfaces [56, 57] has allowed the creation of many interoperable input devices and the simplification of the external appearance of the computer system (because there is no need to include different connectors for each input device). Nevertheless, these solutions have not been able to completely solve endemic problems such as input lag and data transfer limits.

Another important point to consider when analyzing input devices is the culture in which they where originally created. Apart from socio-economic circumstances that might have an impact on the external appearance of the device (the materials and colors of the casing, the labeling of the buttons/keys, etc.) the symbols/metaphors that the designers employ and the physiology of the testes can greatly influence the design –and, ultimately, the functionality– of an input device. And nowhere is it more evident that in hand-held devices, many of which have to be modified for different markets because of the differences in hand size between human population groups (usually defined by age, gender, phenotypical traits, etc.).

Keyboards

With a design reminiscent of old typewriters (due to the aforementioned coevolution during the 20th century), keyboards are one of the most simple ways to input symbolic data, navigate complex menus and control the system.

Each key in a keyboard is a binary switch that, once enough pressure is applied upon it, transmits a specific signal. This signal is then translated into a character code or function call using a predefined key mapping process (part of which is often performed by the micro-controller embedded in most current keyboards). Nevertheless, while the text encoding process is relatively straightforward in the case of most indo-european languages (because it is possible to link a key to each of the symbols of their alphabet), for those with more complex writing systems (e.g., Chinese, Japanese, etc.) additional keys and software mechanisms have to be employed.

Although often discredited for not providing as much precision as analog mechanisms, keyboards are also used as a viable way to navigate through bi- and three-dimensional environments (using the WASD key configuration).

As for system control techniques, desktop keyboards often offer dedicated keys that can be resigned for each context (often referred to as *function keys*). Moreover, most applications contemplate the use of *keyboard shortcuts* (specific key combinations) to facilitate the access to the most popular functions.

Figure 2.18: Different keyboard types (for specific tasks): general-purpose (top), numeric input (left), –portable– alphanumeric input (bottom left) three-dimensional navigation (bottom center) and system control (bottom right).

2D Pointing Devices

This type of input devices are designed to obtain, at least, one stream of bi-dimensional values. The resulting information can be used to –indirectly– position a 2D cursor and, through additional input mechanisms (buttons, wheels, other streams etc.), perform selection, manipulation and navigation tasks.

Until the popularization of multi-touch screens, the best exponent of this category was the *mouse*; a hand-held device that registers bi-dimensional motion relative to the surface on which it is dragged. Originally envisioned in 1960 as a box with two wheels in direct contact with the surface [58], its design has experimented continuous evolution, including additional controls (buttons and wheels) bellow the fingers of the users, replacing the motion-detection mechanism with an optical system and ,overall, greatly improving the ergonomy of its exterior. Currently, it is relatively common to find variations that operate through wireless communication and/or obtain the relative positioning through motion sensors (more commonly known as *air mouses*).

An alternative that has gained wide acceptance in recent years is to employ a touch-sensitive surface that registers the motion as the users drags objects over it (typically, their fingers or special stylus). While the absolute positioning employed in these devices often limits their precision, the possibility of detecting the pressure of the touch and, more importantly, the ability to track multiple objects has allowed the emergence of more intuitive interaction techniques in recent years [59, 60].

Figure 2.19: Different kinds of 2D pointing devices: a mouse with advanced functions (top left), a touchpad common in laptop computers (bottom left) and a graphics tablet that can be used with a special stylus or as a multi-touch surface (right).

3D Pointing Devices

In a similar fashion to their bi-dimensional counterparts, the main objective of these devices is the indirect determination of a virtual cursor in space (and with the help of additional mechanisms, perform selection, manipulation and navigation task). However, in these instances, this operation not only requires the computation of the position in the three spatial axes, but also the orientation of the device; a three-dimensional rotation often expressed as euler angles or quaternions (to avoid the Gimbal-lock effect). A new approach motivated by the practical impossibility to access the entirety of the available space (in contrast with the screen space), thus forcing the employment of projection-based interaction techniques for almost all tasks.

This requirement is reflected in the design of desktop 3D mouses, that, as can be seen in figure 2.20, provide the necessary six Degrees-of-Freedom (DoF) by mixing the concept of their 2D counterparts with that of joysticks. Nevertheless, the inherent complexity of these devices forces their users to go though a difficult learning process before being able to operate them correctly.

On the other hand –quite literally–, many current motion-based controllers include mechanisms to detect their absolute position and orientation. While the concept of tracking the user hand has been present in the market for decades (even the infamous Nintendo Power Glove included a pair of ultrasonic transmitters for this purpose [61]), it was not until the beginning of the 21th century when the use of infrared light and magnetic field technologies were developed enough to allow the detection with the necessary precision for CAD applications.

Figure 2.20: On top of providing six DoF, current 3D mouses (left) include additional controls to facilitate navigation and system control tasks. Meanwhile, the widespread acceptance of the Wiimote (center) between researchers, encouraged manufacturers to develop their own 3D pointing devices (right).

(Remote) Controllers and Gamepads

As their name implies, controllers are devices specifically designed to perform system control tasks. However, as computer systems have grown in complexity over time, so too has the design of controllers, integrating additional input mechanisms to navigate through multidimensional menus, select the appropriate option and manipulate the associated values.

Furthermore, when including these physical controls into the computer system is not viable (either for functionality or design reasons), these devices are commonly redesigned to operate as external peripherals (generally referred to as *remote controllers* or, simply, *remotes*). Originally, these devices were nothing more than an extension of the circuitry, but nowadays they have become computer systems in their own right. A rapid evolution that becomes evident in the design of current game controllers (or *gamepads*).

Although often reviled due to their association with game consoles (to the point of being labeled as mere "toys"), current gamepads are nonetheless *technological marvels* that incorporate all kinds of input –and output– subsystems in very ergonomic form factors. In fact, the number, variety and complexity of the physical mechanisms can become a barrier of entry for many users (that often consider these devices an example in over-engineering [62, 63]). A situation that the main manufacturers are well aware of, and one in which they spare no expense [64].

Nowadays, the versatility and reliability of gamepads has reached such a point that they have become the primary input device in many research projects, art projects and industrial applications. Even defense organizations regularly employ them to remotely operate military robots and vehicles [65, 66].

Figure 2.21: Current remote controllers and gamepads share many button layouts and ergonomic design features. A convergence motivated by the need to standardize navigation mechanisms in desktop computer systems (top left), video game consoles (top right) and, even, mobile platforms (bottom center).

Camera Systems

By far the most computationally intensive input device (both in terms of data size and the resources required to process it), the use of camera systems has become intrinsically associated with AR applications. This is due to the fact that the visual sense is the main channel that human beings employ to perceive their surroundings and, consequently, video data is required as the base for most augmentations.

Due to the complexity of the real world, the visual information obtained from the camera systems can not be directly employed to perform any interaction task. Instead, the image streams must first be analyzed –generally, in real time– through optimized computer vision algorithms to extract *key points* and other relevant visual features [2]. This information can then be processed through *registration* and *tracking* algorithms [67] to determinate the relative position, orientation and motion of the device and/or other physical objects in the environment (most notably, fiducial markers and parts of the users' anatomy). Only after these processes are complete is possible to use the video feed to perform selection and manipulation tasks. In the case of mobile computer systems, this techniques can also be used for spatial navigation tasks.

At the time of writing this document, many key technological companies are promoting the integration of infrared (IR) based depth sensors into the camera systems to obtain a three-dimensional representation of the physical environment. In this way, it is possible to optimize the aforementioned video analysis processes (mainly, by depth-based background-removal techniques).

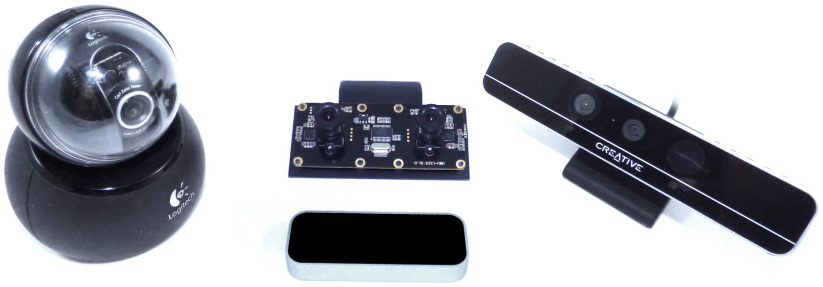

Figure 2.22: Although most computer platforms nowadays include *webcams* to allow users to take photos/vídeos and participate in video-based telecommunications, there is a large amount of camera systems in the market; from robotized cameras with head- tracking functionalities (left) to estereoscopic systems for immersive AR (top center) and IR sensors that accurately register the hands of the user (bottom center). Some manufacturers are even mixing both types of sensors to create 3D scanners small enough to be embedded into mobile platforms(right).

Other Input Devices

In an effort to provide more intuitive and accessible interaction mechanisms to all kinds of end users, multiple venues of research in HCI are oriented towards the creation of new input devices. Some of the most successful are:

- **Speech input:** Aside from telecommunications, capturing human voice through acoustic-to-electric sensors has become a viable way to input symbolic information and control computer systems. Thanks to recent advancements in speech recognition [68, 69] and natural language processing [70, 71], it is nowadays possible –usual, even– to employ *voice commands* to operate mobile devices when the user can not freely move their hands (e.g., while driving a vehicle or holding utensils).
- **Bioelectric input:** With the recent rise in popularity of health-oriented sensor technology, the idea of employing the bioelectric signals generated by the central nervous system to better interact with computers has gained increased acceptance. Beyond detecting the pulse and muscle nerve signals, current research focuses on the development of successful Brain-Computer Interfaces (BCIs) to allow the direct communication with computers (thus solving many of the issues associated with AR-based interaction).
- **Multi-sensor input:** One of the biggest challenges that *wearable computer system* manufacturers currently face is the difficulty to include functional and accessible input mechanisms in platforms that are designed to be as compact and weightless as possible. Many manufactures employ sensor fusion-based solutions [72] to make the user interfaces as intuitive as possible, but others opt for approaches based on external, wireless controllers [73].

Figure 2.23: Common microphones (left) can be used to register human speech. However, to detect bioelectric input, it is often necessary to employ *wearable* devices (center) with special sensors in direct contact with the users' skin. Other external devices (right) can also be fitted with additional sensors to provide complementary control mechanisms.

2.4.2. Output Devices

To properly convey the information generated by computer systems to their users, it is essential to have a deep understanding of both the technologies behind current output devices (also commonly referred to as *displays*) and the human sensory system that has to recognize and interpret the signals. Moreover, it is important to properly identify the different medium conditions and disabilities that might alter their proper perception.

As with its input counterparts, the massive popularization of mobile platforms has greatly accelerated the development –and miniaturization– of new display technologies. In fact, current high-end smartphones feature screens with higher pixel density and color accuracy than most desktop monitors and, although surround sound systems usually offer better audio quality overall, a large percentage of users prefer to employ headphones when interacting with virtual environments because the binaural cues these audio systems provide enable them to easily locate the sound sources in three-dimensional space [74, 75].

In recent years, this trend has resulted in the emergence of a new type of mobile hardware devices, often referred to as *wearable computers* due to their ergonomic design, that users carry with them at all times. A new paradigm that not only allows the users to interact with computer systems in very diverse situations but also defines new mechanisms to capture the movements of the body and provide feedback via haptic signals.

A clear example of this technological trend is the resurgence of stereoscopic Head-Mounted Displays (more commonly called *VR headsets*) as a viable output device for common multimedia and gaming applications. Previously relegated to obscurity due to the resounding commercial failure of the Virtual Boy [76], the aforementioned advancements in graphical displays have allowed the creation of new interaction mechanisms –and, even, entire computer platforms– centered around them.

Nevertheless, it is important to note that, due to the higher refresh and data transfer rates required to display multimedia resources, most output devices have to rely on wired connections to properly transmit the data to the users. A restriction that, nowadays, either limits the interaction space (i.e., the users have to be seated) or forces the users to carry the entire computer system (including the power source) with them.

Visual Displays

As aforementioned, the visual sense is the main way human beings obtain information about their physical surroundings. Consequently, to communicate through this channel, the computer system must be equipped with a display capable of representing a large amount of graphical information. Furthermore, to adequately present an *augmented version of the real world* trough the display, it has to constantly assess the relative location of the device and redraw the virtual objects according to the location of each user. An extremely convoluted process that many manufactures try to simplify by making their visual displays see-through (so that they don't have to include high-resolution cameras and screens to compensate) and assigning one display for each user.

While the use of *light-emitting surface-based devices* (or, simply, screens) continues to be the main way to convey visual information to the end users, many other approaches have been suggested. Most notably, many research works focus on the possibilities of light projection-based devices to display virtual scenes over multiple static surfaces (i.e., CAVE systems [77]), the hands of the users (portable projection systems [78]) or, directly, reflect the visual data onto their retinas (virtual retinal displays [79] and see-though HMDs). However, the occlusions generated by the entities present in the environment, the possibility of interferences between multiple projections and, above all, the inability to "subtract" light originated from other sources –specifically, the Sun– makes this solution not viable except in very specific environments.

Figure 2.24: Recent advancements in display technology have allowed the creation of high resolution screen for mobile platforms (bottom left), stereoscopic screens for game consoles (bottom center) and hand-held projectors (bottom right). A rapid evolution has also had an important impact in costumer-grade HMDs (top).

Sound Systems

Often overlooked in UI design, auditory displays can, nevertheless, be a powerful tool to communicate information to the user. Some of the most common applications of these systems include associating simple sounds to certain actions to provide additional feedback (e.g., a clicking sound when pressing a virtual button or a loud noise when an error occurs) and to use a text-to-speech system to transmit complex messages. However, these systems can also provide many important benefits to SUI designers.

In a three-dimensional environment, the inclusion of auditory displays can be used to generate localized audio cues that guide the user through the space or broadcast information that it is only relevant within a specific region. Beyond serving as mere ambient effects, these 3D audio cues can also be tied to a logical behavior, allowing the interactive communication of complex data through variations in volume, pitch and tone of a sound (a process known as *sonification* [74]). Furthermore, by allowing users to record their own voice (and position it in a three-dimensional space), it is possible to create an annotation tool that facilitates successful collaboration experiences (even over asynchronous telematic communications).

There are many techniques to generate three-dimensional sound. However, as with visual displays, the location of the sound emitters is an important consideration that might profoundly impact the experience. If the users are expected to remain static or move inside a small spatial region, the use of a multi-speaker system might be the most appropriate –and comfortable– solution. If, as often is the case with AR-based experiences, the user is expected to move in an non-controlled environment, the use of stereophonic headphones is generally a better option since it is possible to regulate the sound that users perceive through each ear and, thus, provide better binaural cues.

Figure 2.25: While wired headphones (top left) and speakers (top right) are still a viable solution in many interaction scenarios, their wireless counterparts (bottom) generally offer a more simple and seamless user experience. Furthermore, the use of noise isolated headphones (center) might be necessary in multi user scenarios.

Haptic Systems

Although the term *haptic* is used so often to describe those systems that provide *tactile* feedback that they are usually employed as interchangeable terms, haptic devices also take into consideration other perception mechanisms of the human body to convey meaningful information to the user. These mechanisms include thermoreceptors (perception of temperature), nociceptors (perception of pain) and, most importantly, the propioceptors present in the skeletal straited muscles, tendons and joints. Thanks to these later sensors, the human brain can recognize kinesthetic cues and maintain a –relatively low-resolution– model of the different parts of the body.

While, most of the times, end users are not consciously aware of them, taking advantage of these perception mechanisms is an important design consideration when designing a physical controls. In fact, those peripherals that do not provide enough haptic feedback (e.g., analog sticks with slippery surfaces, membrane keyboards with small key travel, etc.) are generally discarded by their users.

Nowadays, most gamepads feature analog controls with excellent tactile feedback and often include several vibrators to transmit complex kinesthetic cues. However, because these peripherals are designed to be hand-held, the amount of force that they can exert is relatively small. For this reason, more advanced simulations in training and medical applications require the use of peripherals that can be grounded to a surface (a desktop, wall, ceiling or floor) and have actuators with enough torque to provide a realistic *force feedback*.

Another important venue of research that has been explored for several decades is the possibility of employing body-referenced haptic devices (often referred to as *powered exoskeletons*, *exomusculatures* or *exosuits*) to allow a mobile user interaction. However, the weight, size and power requirements of the actuators normally hinders the experience and might even make the users lose their balance.

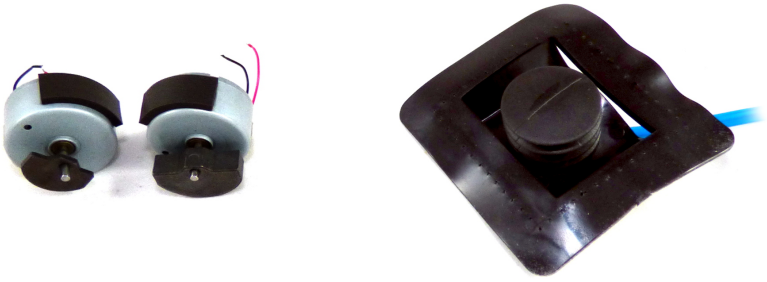

Figure 2.26: Nowadays, most gamepads include vibration motors (left) to convey important information to the user. Some manufactures have experimented with embedding neumatic actuators (right) into vests to provide a more realistic experience.

"If I am walking with two other men,
each of them will serve as my teacher.
I will pick out the good points of the one
and imitate them, and the bad points of
the other and correct them in myself."

Confucius

3

Interaction Model

Due to the numerous aspects that need to be taken into consideration, constructing a new HCI model is a very complex task. One that requires to analyze in great detail not only the individual elements that conform it, but also the relationships that they can form between each other. An endeavor that, in order to provide a meaningful AR-based interaction to the end users, is further complicated by the need to appropriately describe the physical environment that surrounds them.

After an extensive research process, the interaction model presented in this chapter offers a comprehensive solution to this conundrum. A user-centric approach that attempts to encompass the different interaction elements into a –relatively– simple logical structure from which to facilitate their management and, ultimately, create a communication system that allows their exchange with other computer systems (thus facilitating the creation of new user experiences and software ecosystems around them).

To provide the reader with a better understanding of the proposed solution, the present chapter provides three different but complementary perspectives (or representations) of the interaction model. Section 3.1 defines its different elements from a purely theoretical standpoint. Afterwards, section 3.2 translates these interaction components into data structures that computer systems can manage properly. Finally, section 3.3 provides the necessary mechanisms to securely exchange these interaction elements with other computer systems.

3.1. Theoretical Model

Before delving into the inner workings of the proposed interaction model, it is necessary to study the logical premises that serve as its theoretical foundation. In this way, not only the reader will be able to attain a detailed overview of the model itself (including the differences with previous HCI paradigms), but also will obtain the knowledge required to apply it correctly and easily extend it to construct their own user experiences.

To achieve these goals, this section formally defines the conceptual framework over which the proposed interaction model is based on. First, it presents a high-level view of its most distinctive features (while providing additional explanations to detail the benefits of their inclusion). Later, an exhaustive study on the interaction model is introduced, detailing the different participants that take part on it, as well as their roles and idiosyncrasies. A classification that is further analyzed in the next subsections, dedicated to the interaction scenarios and techniques that can be derived from it.

Nevertheless, it is important to clarify that the main motivation behind this revision of the conceptual basis of Human-Computer Interaction is not just eagerness to design new mechanisms to improve the information exchange –although it is a welcomed side effect– but, rather, the need to satisfy the new technological requirements resulted from the need to properly understand the users' context. Since Augmented Reality-based experiences aim to seamlessly integrate virtual elements into the users' perception of the physical world, the interaction model must provide the tools to analyze the surrounding environment (including the body of the users), incorporate the components of the user interface and display the modified result to the final users. An extremely complex task that can not be completely solved with current archetypes.

For this reason, the proposed theoretical model reexamines several axioms that current paradigms take for granted (e.g. the assumption that only human beings can be considered proper end users or that a constant gravitational force is always present in the environment) and performs a complete reevaluation of the communication model itself. However, due to the different natures of the participants in the information exchange and the complexity of the medium in which it takes place, a great number of new requirements have to be taken into consideration.

Throughout the following subsections, all these new conditions will be explored in detail, conforming a comprehensive logical framework that will serve as the theoretical foundation for the rest of the chapter.

3.1.1. Main Design Principles

Due to its holistic and comprehensive nature, the proposed interaction model cannot be completely described using a simple list of key features. However, there are several distinctive characteristics that can provide a better understanding of the theoretical framework as a whole.

User-centric Approach

In this research project, the idea of centering the interaction around the user goes beyond a mere design principle for user interfaces. This decision has a significative impact on the definition of the entire model.

While previous models define virtual spaces within the boundaries of the graphical displays (forcing users to create several layers of mental abstraction to find, reach and operate the desired interaction element), the proposed interaction model allows the definition of the user interface components (Widgets) in a boundless space centered around the user. This relative positioning-based solution offers several important advantages:

- **Simplicity:** Instead of having to manage heterogeneous data structures to handle numerous interactive objects in large, complex environments, it is possible to employ a single and –relatively– simple data structure to store and update them whenever necessary (when their associated resources are modified or while the user navigates the environment).
- **Optimization:** By having a smaller set of elements to manage (and by being able to determine the extent to which they are perceived by the user) it is possible to significantly reduce the amount of computation required to adequately computate the simulation.
- **Intuitiveness:** Since this approach is based on the point of view of the user, the interaction techniques build upon it do not require additional cognitive processing (perspective transformations, mental self-projections, etc.). This leads to easier exchanges of information and, ultimately, a more intuitive user experience.

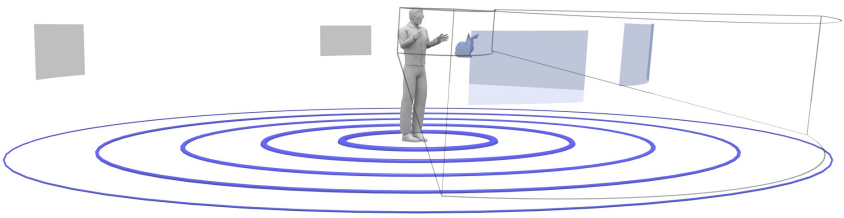

Figure 3.1: In the proposed model, the interaction components are positioned in relation to the users' physical presence, greatly facilitating their management.

Comprehensive Framework

The scope of the user experience is not limited by the individual tasks performed by the users but, rather, it encompasses all communications with the computer system (from its very activation to the moment its turned off, and even through multiple sessions). Consequently, an interaction model must not only provide a framework for application developers to design the different user interfaces, but also a software environment for the end users to manage their execution at an operating system level. This includes obtaining a list of available applications, showing/initializing the appropriate user interfaces for each given context (either established directly by the user or proposed by the computer system for the present situation) and hiding/finishing them when they are no longer deemed necessary.

The proposed interaction model satisfies this requirement by offering the users an extensive set of tools to customize their experiences (to the point where there is no clear division between developers and end users). This approach, in direct contrast to current trends (that greatly restrict the customization options), allows the redefinition of the entire user interface of the operating system by means of a simple description language (further explained in the last section of this chapter).

Nevertheless, it is important to note that the usage of the adjective "comprehensive" to describe this approach is not adventitious but, in fact, adequately expresses the extensive nature of the model. Rather than providing a complete –but closed– set of tools to describe the user interfaces, this model presents a large set of basic interactive elements (from simple 3d objects to multi-state behaviors) that can be customized and combined into advanced widgets. With this modular structure, it is possible to define really complex user interfaces (even for an entire operating system), and divide them into multiple levels of complexity. In this way, the end users are empowered to adapt the experience to their needs and desires, without having to deal with the actual definition of the widgets.

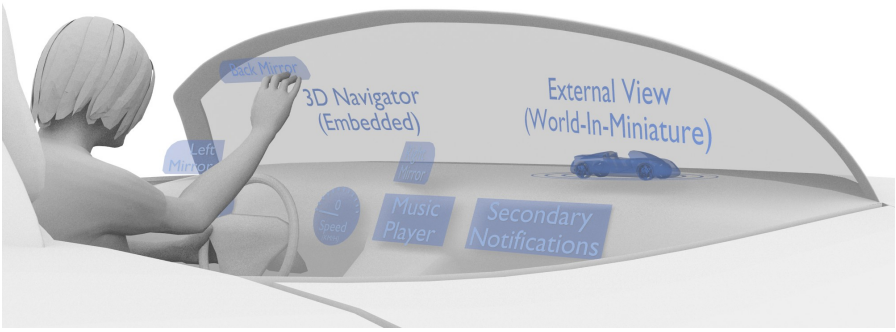

Figure 3.2: Taking advantage of the provided tools to customize the user interface of an AR-based system –exclusively– on the proposed interaction model.

Situational Awareness

One of the ultimate goals of user experience design is to create an interface system so efficient and intuitive that the user does not realize its very existence. As Andries van Dam expressed in his influential article Post-WIMP User Interfaces [23]: *"[...] A more feasible goal to strive for is the computer as perfect butler, à la Jeeves, who knows my context, tastes, and idiosyncrasies and discreetly does my bidding by anticipating my needs without needing explicit direction".*

Following this line of reasoning, the proposed interaction model has been designed in a way that can easily store and manage the contextual information of the user at any given time. However, this imposes a new requirement that the platforms that implement the proposed framework have to abide by: the development of a dedicated subsystem for the ascertainment of the user context.

A software solution that gathers all the available situational information (directly introduced by the users or automatically extracted from the input devices or from predictions based on the user actions and schedule) and analyzes it to determine the general user context –and subcontexts–.

Either defined by the operating system, the end users or the applications themselves, these context data structures describe the conditions that the interaction model must satisfy in each situation. By taking into account this information, it is possible to adapt the contents of user interfaces in real time, offering the users an easy access to meaningful contents without having to manually input keywords or perform search queries.

Additionally, this solution also allows the assessment of the risk that an action may entail for the physical security of the user (including collisions with other elements in the environment or entering dangerous zones). In this way, the users can focus their attention on the task at hand, with the assurance that the computer system will prevent any potentially perilous situation. Thanks to this mechanism, it is possible to create –and maintain– completely mobile user experiences.

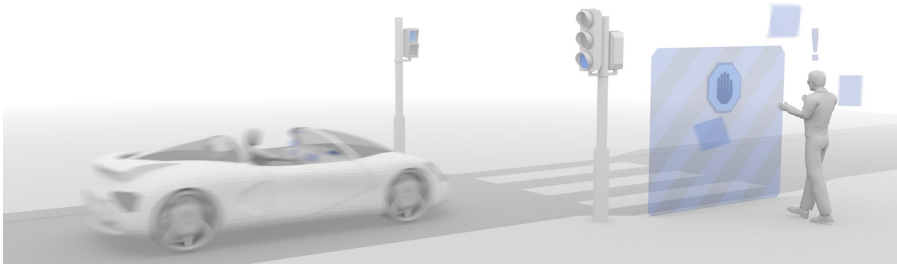

Figure 3.3: By taking into consideration the situational information it is possible to modify the contents of the user interface to avoid potentially dangerous situations.

Collaborative Edition

Although earlier in this section it was stated that it is recommended that each user interacts with a single computer system, it does not mean that the interaction model is restricted to these two agents. Rather, the proposed model provides mechanisms to facilitate the exchange of information between an unlimited number of users through its very communication architecture.

Beyond sharing static data structures, the users are able to exchange dynamic content in real-time, ranging from simple voice messages to entire sections of the user interface. This enables the users to participate in large scale projects in whose individuals can cooperate to achieve a common goal.

A collaboration system further reinforced by the possibility of projecting a virtual representation of the user (an avatar) that other participants can interact with. An approach that can be used for telepresence communications, allowing users to coordinate their actions intuitively and execute them synchronously (even across time and space).

Additionally, the architecture of the interaction model has been designed to solve one of the recurring problems associated with collaborative edition environments: the preservation of the user intention in concurrent edition applications. An issue that arises from the ability to overwrite other users' actions without their approval, and one that sometimes results in the loss of hours of work.

While this issue can be solved with careful work planning or by restricting the edition to certain sections of the shared workspace, these solutions do not address the core of the problem and, more often than not, they unnecessarily limit the interaction capabilities. Instead, the proposed communication architecture takes advantage of operational transformations to ensure that all users can edit the entire workspace (from different views), with the certainty that, even if they work on the same section, their actions will not be affected negatively between them (and even then, they can be reverted until a valid state is achieved again).

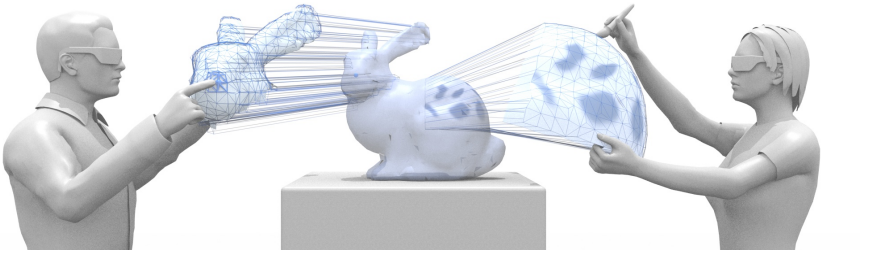

Figure 3.4: An example of a modeling application that uses the proposed interaction model to allow multiple users to work collaboratively on a three-dimensional model.

Advanced Spatial Positioning

In order to provide successful user experiences in Augmented Reality applications, the virtual elements of the user interface must be perceived as an integral part of the physical world. To achieve this goal, it is not enough to simply superimpose the virtual object over the video feed provided by the camera. A more advanced system is required to tackle the issues that arise from the positioning of virtual objects into the real world.

Apart from being positioned relative to the physical presence of the user, the virtual elements of the user interfaces can be globally positioned within the physical environment by deploying fiducial objects (such as AR markers or easily recognizable symbols) or by defining additional coordinate spaces (from basic tracking mechanisms to global positioning systems). Nevertheless, from the user point of view, these virtual objects can be occluded by other elements of the environment (or other components of the user interface) and the system has to be capable of solving these problems in real time. While it may seem a straightforward task, this issue becomes rapidly more complicated when phenomena such as transparency, reflection, refraction or sound reverberation have to be taken into consideration.

Another important element that has a significant impact on the perceived spatial position of the virtual objects is the lighting. Even through, to achieve a believable shading on the object itself, it is usually enough to replicate the light sources present in the physical environment (an information that can be easily transmitted or, even, calculated in real time), to provide a more realistic illumination, additional processes such as shadow projection, subsurface scattering or ambient occlusion have to be adequately processed.

To solve the aforementioned issues, the interaction model proposed in this document allows the recreation of the physical environment. Either obtained through the available sensors or –preferably– through the communication with specialized computer systems, the simplified representation of the surroundings of the user provides the necessary information to seamlessly integrate the different interaction elements into the reality perceived by the user. Figure 3.5, showcases an example of this mechanism.

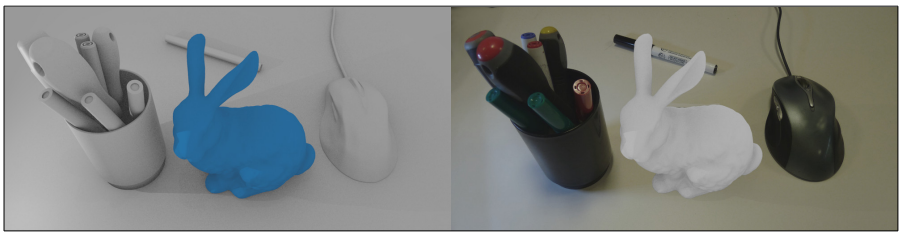

Figure 3.5: By allowing the acquisition of three-dimensional representation of the physical environment (left), it is possible to integrate virtual objects into it (right). Please note the shadow projections between the different elements of the scene.

Physics Simulation

Even if the components of the user interface seem perfectly integrated within the environment, the sense of immersion can be significantly undermined if they do not react realistically to physical forces (generated either by the users' body or by other real objects). To avoid these situations, the proposed model contemplates a subsystem that replicates the physical forces present in the environment and calculates a realistic physical behavior for the virtual objects.

This physic simulation has major implications for the entire interaction model. Not only all the user interface components have to be defined using materials with physical properties (e.g., mass, color, speed, hardness, electrical conductivity, elasticity or density) but all the interaction techniques have to be specified using physical forces (either by generating them directly or by creating proxy virtual objects).

Additionally, the system must be able to obtain the properties of the current physical medium and the forces present within it. While most of those attributes can be obtained directly using the input devices (gravity, viscosity, luminosity, etc.), more complex and variable medium properties (such as weather, wind, ocean currents, etc.) can be obtained from external systems and transmitted directly into the simplified model of the environment where the physical simulation takes place. In this way, it is possible to create widgets that behave in a similar way to their real-world counterparts (e.g., buttons, switches or door handles), increasing the intuitiveness of the user interface. Nevertheless, the designer must take into consideration the properties of the medium so that those widgets can operate correctly under different conditions (for instance, in underwater environments, in the vacuum of space or on the surface of high-gravity planets).

However, while the recreation of physical forces in the virtual world is a relatively simple process, the inverse is not possible without complex force-feedback hardware systems. For this reason, it is recommended to define the user interface components in such a way that can not interfere with the normal operation of the physical elements where they are spatially positioned.

Figure 3.6: Several examples of how the physics simulation applied to the user interface components can enhance the user experience (or damper it, if used incorrectly).

3.1.2. Interaction Components

In current HCI paradigms, the communication model only involves two entities: the computer system and the user that operates it. Thanks to this level of abstraction, user interaction designers can focus their attention on the definition of the general workflow for each task, without having to specify in detail the underlying mechanisms that make it possible.

However, this simple model proves to be unsuitable for Augmented Reality-based applications. Because the components of the user interface can be spatially positioned, the interaction model has to account for any occlusion and superposition issue that may arise. Furthermore, in order to integrate these virtual elements into the reality perceived by the users as harmoniously as possible, additional factors of the physical surroundings (lighting properties, atmospheric conditions, etc.) have to be incorporated into the representation.

As a solution for this problem, the proposed interaction model establishes a clear distinction between physical and non-physical entities. As figure 3.7 shows, this decision results in the introduction of a new interaction component situated between the computer system and the final user. A new entity that not only represents the physical medium in which the communication takes place but it also takes part in it (e.g., an element of the user interface can be set to react to a change in the environment, regardless of whether the user caused it or not).

Figure 3.7: The main components of the interaction model.

Nevertheless, the implications of the introduction of this new layer in the model go far beyond the mere representation of the physical elements of the environment. Since there is a conceptual separation between the psychological and physiological aspects of the users –similar to the distinction between hardware and software in computer system– it is possible to define interaction mechanisms that take into account their physical presence, cognitive capabilities and personal preferences. This level of control over the user context, along with the environment information, allows the definition of situational awareness user interfaces. Thanks to this holistic approach, it is possible to create more intuitive and immersive user experiences without having to resort to complex logical structures.

End-User

Generally speaking, the term *user* refers to "any person that uses or operates something" [80]. However, within the boundaries of the proposed interaction model, this statement requires several clarifications in order to provide a clear and precise definition:

1. The connection with the computer platform must not be disregarded. Even if the interaction model contemplates the possibility of tracking elements within the physical environment (that may or may not have an effect on the communication), the main goal continues to be the establishment of a solid link between the user and the computer system.

2. The interaction model does not accept intermediary entities. While nothing prevents the communication outside the specified domain (users can converse with their peers and, separately, computer systems can connect with other devices to exchange information), any element that tries to mediate between the user and the computer system will not be taken into account (or, at most, it will be considered as another element of the environment). In other words, all users are *end-users*.

3. Both the physiological and psychological nature of the user needs to be adequately analyzed to determinate the current interaction context. While the present model does not restrict the user role to only human beings (due to recent studies[7, 8] proving that other living beings are able to successfully interact with computer systems), a basic level of cognition and self-concept –a personhood– is required to intentionality take part in the communication process and be considered a proper user.

Once gathered, these conditions provide a concise definition of what constitutes a valid user within the present model, as well as a high-level specification of the associated interaction mechanisms (the physical presence that user's body exerts in the surrounding environment and the sensory system responsible of the user's perception). Additionally, this clear demarcation prevents the apparition of malicious entities that can have a negative effect on the communication (such as software agents posing as legitimate users or intermediate elements that modify the message).

Furthermore, this comprehensive definition allows the extraction of detailed descriptions of the user context (and subcontexts). Important data that the computer system can process to assess the global situation and adapt the contents of the user interface to fit their needs and capabilities.

This contextual information can be divided into the following elements:

- **Identities:** Represent how the users present themselves to the computer systems (and, through them, to other users). Apart from the one required for the basic communication with the operating system, each user can define multiple identities for different interaction contexts.
    - Id: Unequivocally identifies the relationship between the user within the computer system. It may include additional security information (i.e. password, cryptographic keys, biometric data, etc.).
    - Name: The name the user likes to be addressed or known by. It does not have to be the real name, but it should be protected to avoid potential identity thefts.
    - Avatar: A projection of the physical presence of the user. Although it is strongly suggested that it matches the actual physiology of the user, it can be modified to fit each particular interaction context.
- **Personas:** Widely employed in marketing and user experience design [81, 82], the concept of *persona* (from the Greek word for mask, $\pi\rho\acute{o}\sigma\omega\pi o$) allows an easy description of a social group [83]. Within the proposed interaction model, however, the term is applied at an individual level, conveying the psychological state of the user in a format that the computer system can easily interpret and employ to adapt the contents of the user interface accordingly.
    - Personal data: Basic information about the user (such as real name, gender or age) that helps to further personalize the experience.
    - Actions: Simple –but interrelated– descriptions of the different tasks that the users perform during the interaction process (e.g., "walking", "talking", "driving on the highway", etc.).
    - Needs: Based on the classic Maslow's hierarchy of needs [84], this logical structure describes the needs of the users (in order to provide them with suggestions that are relevant for their interests).
    - Capabilities: Describes the physical and psychological capabilities –and disabilities– of the user.
    - Goals: The objectives that the user tries to achieve.
    - Interests: The motivations behind the user's actions.
    - Relationships: Interpersonal connections with other users.
- **Preferences:** Althrough the previous elements provide enough information to properly accommodate the user experience to each context, ultimately, the system must provide alternative mechanisms to adjust the individual parameters manually. These options should be presented in a hierarchical structure to facilitate the modification of simple values as well as complex data types (culture, input configurations, etc.)

Environment

As stated before, within the present interaction model, the physical environment is an entity that cannot be disregarded. Beyond serving as the medium in which the communication between the end user and the computer system takes place, its correct specification is essential to maintain meaningful Augmented Reality-based user experiences.

Nevertheless, the complexity and variability of the natural world makes its complete definition an almost impossible task. Even in enclosed, artificial environments, the number of elements that can be included into the physics simulation easily surpasses the computational capabilities of most current platforms (more so when the software system also has to take the physical presence of the users into consideration, in real time).

For this reason, instead of focusing on trying to achieve a perfect recreation of the users' surroundings, the interaction model provides a framework to define simplified representations of the physical environment. In this way, it is possible to define the interaction spaces for the user interface while accommodating it to the capabilities of current mobile platforms.

The acquisition of these simplified representations can be achieved through two complementary mechanisms: using the input devices of the computer system or, preferably, through the communication with dedicated hardware platforms situated in the users' surroundings (computer systems equipped with sensors arrays that periodically analyze the real world in order to maintain an updated virtual recreation of it). Since the framework supports multiple levels of detail for individual objects, users can combine both mechanisms to interactively obtain a more accurate representation of the physical environment while transversing it. Additionally, users can take advantage of the aforementioned collaborative edition tools to coordinate their movements and share their partial representations, allowing a more precise and complete result.

It is important to remark that, this interaction component is not limited to describe the physical surroundings of the users, but it can also be employed to contain the virtual elements of the user interface (or, in the case of telepresence-based interactions, the avatars of other users). In fact, it is recommended to divide the definition of the interaction space into multiple environment entities. In this way, it is possible to define multiple "reality layers" with different attributes (i.e., whether the entities it contains are real or virtual in nature) and establish different relationships between them (thus allowing the creation of non-diegenic virtual objects that can only be viewed or manipulated by particular users).

Apart from these attributes, the definition of an *environment* component can contain the following elements:

- **Regions:** Define the spatio-temporal boundaries of the environment. Their specification allows a better management of the entities contained within them.
    - Location: The position and dimensions of the region, defined in relation to an absolute point in space and time. In this way, the objects can be situated using a more intuitive (local) coordinate system.
    - Zones: Spatial subdivisions of the region that, apart from facilitating the management of the computational resources, may provide additional contextual and navigational information.
    - Media: The properties of the environment from the standpoint of interaction, to establish the best communication channel available depending on the physical medium (i.e., radio communication systems in the vacuum of space).
    - Markers: Fiducial markers strategically disseminated throughout the physical environment to provide a more accurate spatial positioning.
- **Entities:** The individual elements that are present in the environment. A good analysis and classification of the different objects (i.e. inanimate/animate, inoperable/operable, etc.) may open the door to better interactions and, even, the prediction of their future status.
    - Shapes: The external boundary of the different parts of the object.
    - Materials: The material properties of each part of the object, including their optical, mechanical, thermal and electric properties.
    - Behavior: Additional information that enables the system to simulate the physical interactions with other objects (specially, the body of the end users).
- **Forces:** A force represents an influence that causes an object to undergo a change in its physical properties. These include fundamental forces such as gravity and electromagnetic iterations but also others derived from them (including visible light, wind or tidal forces).
    - Effects: The physical properties modified by the application of the force (velocity, angular momentum, temperature, luminance, etc).
    - Fields: Defined either with vector functions (with origin, direction and magnitude parameters) or with emitter-attractor systems, a field specifies the spatio-temporal regions where the force exerts its effect and the values in each point.

Computer System

Within the context of an interaction model, a computer system is not only an essential participant of the communication, but also the entity responsible for its management. It provides both the physical (hardware) and logical (software) elements that the user can operate to perform a given task and achieve the intended result.

Often referred to simply as "computers", these entities are, in fact, extremely complex systems that have long surpassed the original goal of simply transform the given information according to a predefined program. Nowadays, computer systems are composed of a multitude of specialized modules with a high number of interconnections, making the establishment of a clear delimitation of their boundaries a difficult task (something especially true in many current mobile platforms, whose operating systems even rely on online services for important tasks such as data storage or graphical computation).

While their heterogeneous nature prevents any attempt of establishing a formal definition, from the standpoint of user interaction, computer systems only have to define the necessary mechanisms to sustain the interaction with the user. To that effect, they have to provide the physical devices and logical structures to retrieve the input data (either directly from the user or through the environment), process it and output the result (in a format that the user can easily understand). Combined, these three procedures constitute the base of the user interfaces.

The basic goal of the user interfaces is to provide a virtual interaction space where the operators can effectively control the computer systems to perform complex tasks over an extended period of time. More advanced user interfaces feature modular architectures to facilitate their creation and understanding, mechanisms to aid the user in learning and making operational decisions and audiovisual stimuli specially designed to enhance the global experience.

These concepts become even more important within the proposed interaction model, where the user interface is no longer a secondary element in software development, but the centerpiece of the entire system operation. In fact, in this new model, there is but a single user interface, to which each application may incorporate their own *widgets*; reusable interactive elements that provide the necessary virtual entities and logical behaviors to manage the operation of the system. An approach that, in conjunction with the aforementioned context management subsystem, allows the definition of powerful but intuitive user interfaces.

The specification of computer systems contains the following elements:

- **Devices:** The different parts in which the computer system is divided into (from an interaction standpoint).
  - Functionality: The computational capabilities of the device (and the best purpose it can serve).
  - Setting: Allows the specification of the appropriate operational mode for each given situation.
  - Representation: The presence of the device in the physical world. In conjunction with the definition of the user's avatar, its proper specification greatly facilitates the simulation of physical interactions.
- **Contents:** The media information (texts, images, videos, etc.) employed in the creation of the UIs. To facilitate their access and sharing, contents can be grouped into larger content elements.
  - Data: The digital resources/documents of the media content.
  - Metadata: Additional information about the content data (i.e. file name, language, creation time, etc.). It can also include licensing information.
  - Codification: Additional information required to properly decode the content data.
- **Widgets:** Interaction elements that can be used as building blocks to create interoperable UIs. As with content elements, each widget can also serve as a container of other (child) widgets, greatly facilitating the development and reutilization of highly interactive objects.
  - Behavior: The interaction techniques and (business) logic associated to the widget. Represented as a sequence of actions-reactions, these behaviors constitute the base of the entire interaction model.
  - Resources: Specifies the content elements that are displayed and/or modified by the widget.
  - Variables: To facilitate the creation of complex interaction mechanisms, it is possible to specify specific parameters for each widget.

3.1.3. Interaction Scenarios

Since the inception of the term *Augmented Reality*, several taxonomies have been proposed in an attempt to classify the different applications that take advantage of this technology. A task that, due to the increasingly complex implementations and the wider scope of the application domain, is still far from reaching a definite and complete state.

While several of the taxonomies focus on the visual display technology [85] or the tracking requirements [86], other authors [87, 88] suggest a more user-centric approach. In these typologies, the main classification criteria is not the augmentation procedure but rather how the user perceives it. As a result, this approach allows the specification of interaction techniques in a hardware-independent manner, making it a perfect fit for the proposed interaction model.

By using this approach in conjunction with the interaction components explained in the previous subsection, it is possible to define three distinct scenarios. As figure 3.8 illustrates, these interaction scenarios always present the users at the center and the physical environment surrounding them. However, what sets them apart its the location of the computer system (through which users perceives the *augmented* physical world). In a *direct* interaction scenario, the physical world is completely perceived through the sensors of the computer system, while the other scenarios allow the interaction with real objects –including the body of the user– without the possibility of applying any augmentation process to them (which may be an intended behavior). Additionally, if the different hardware devices of the computer system are embedded into the surroundings objects, an *indirect* interaction scenario becomes an *integrated* one. An important distinction with serious repercussions from the point of view of the user experience.

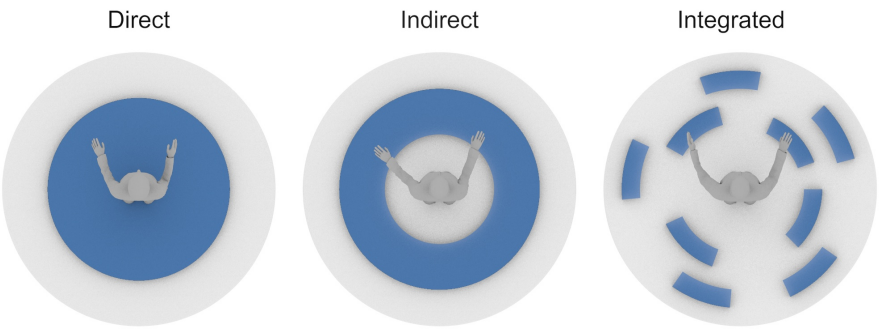

Figure 3.8: The three different types of scenarios that can be created from the aforementioned interaction component separation.

Direct Interaction

The introduction of noise is an endemic problem of any communication system that takes place within a physical medium. To minimize this unwanted effect, the most straightforward approach is to reduce the physical distance between the participants. This can be achieved through the design of peripheral devices in direct contact with the body of the users (ideally, directly connected to the sensory system).

While, at the time of writing this document, the peripherals designed for AR systems –even those that are marketed as *wearable*– are not yet ready to be used for long periods of time (due to their size, weight and energy consumption), current technological advancements make almost certain the apparition of viable solutions in a near future. In fact, several companies have expressed their intention to pursue this research effort [89, 90].

This interest is motivated by the advantages associated to this interaction scenario in terms of mobility for the final user. With a HMD system, the user does not have to perform any additional action to participate in the AR-based experience. Either through screens situated in front of the eyes, half-mirror systems or virtual retinal displays, the user can perceive a visual augmentation at all times, with a consistent level of quality and without occlusions.

Combined with a microphone-headphones systems to provide auditory augmentation and several tracking mechanisms for detecting the relative position and rotation of the head (as well as additional peripherals such as data gloves, motion controllers or three-dimensional mice), these hardware devices provide an excellent sense of immersion.

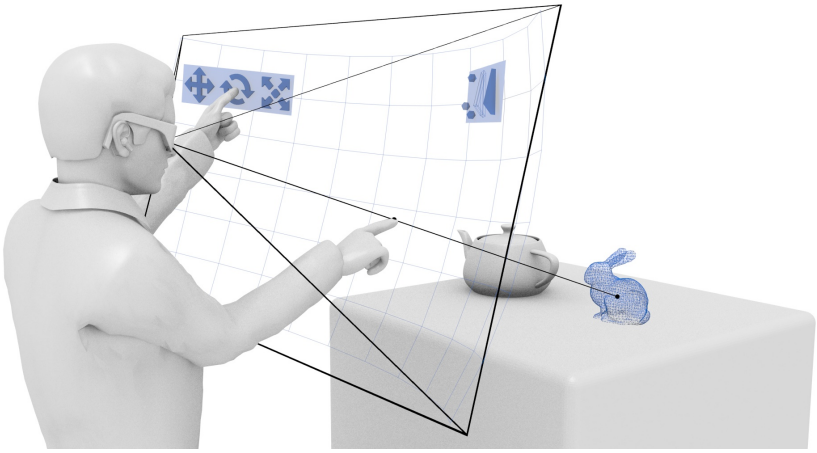

Figure 3.9: A direct interaction experience achieved through the use of a see-through HMD and a bend-sensing data glove.

Nevertheless, this level of immersion can easily be shattered by the lack of haptic feedback associated with direct interactions with virtual objects. An endemic problem that does not have a simple solution. While current haptic technologies allow for the delivery of pressure sensibility and variable sensation of friction, the main issue still remains: it is not –yet– possible to provide an adequate level of force feedback without anchoring the system to a fixed surface (since, otherwise, the generated force may affect the balance of the user's body and make him/her fall).

One possible workaround to this problem is to develop additional interaction techniques that allow the users to manipulate the elements of the interface without having to move their physical body. This can be achieved through eye-gaze tracking or bioelectric input systems and, while it introduces a logical distinction that goes against the principle of aiming for a seamless integration of virtual objects into the real world, it may provide a simpler and faster method to perform complex tasks. Additionally, these new interaction mechanisms can greatly limit the "mime effect" (i.e., the negative view that the users project when interacting with elements that others can not see), increase the physical security (by avoiding collisions with real objects) and allow users with physical disabilities to operate the system with a similar level of proficiency.

Another important consideration about this interaction scenario is that, while they do not have to have the augmentation process active at all times, the hardware devices may be constantly present (more so, if they are implanted into the body of the user), and thus, may have a significant impact on the life of the user. For this reason, the AR systems that employ this approach must include at least one –hardware or software– switch to enable or disable the augmentation, as well as a clear way to determine the current status and distinguish the virtual elements from the real environment (so that the user can immediately identify potential dangers).

Finally, the development of the hardware devices for this interaction scenario must also take into account how their use may affect interpersonal relationships. Since being able to look into the eyes of other persons and gesticulate to convey emotional nuances is considered an essential part of the communication between human beings, augmented reality devices must be constructed to avoid creating interferences. For example, HMDs should be transparent –from the outside, at least– to allow an easy reading of facial expressions, while data gloves should be designed to enable users to touch other persons (from shaking hands with them to punching them) without having a negative effect on the interaction or damaging the hardware equipment. Moreover, these devices may include mechanisms that actually enhance the interpersonal communication. This can be achieved by simply allow the users to personalize the exterior of the device to express their personality or, for long distance communications, by including sensors that can read and transmit the current emotional state.

Indirect Interaction

As explained in the previous section, the current technological limitations of HMDs do not yet allow the general adoption of AR-based experiences based on the direct interaction scenario. However, the massive popularization of consumer electronics (mainly, smartphones and tablets, but also laptop computers and game consoles) equipped with cameras and advanced graphic processing units, have led to the establishment of a different interaction scenario; one where the augmentation takes place through a computer system situated between the user and the physical environment. Furthermore, this scenario can have two different configurations:

See-through (or "Magic lens"):

In this configuration, the computer system (generally, a mobile platform) takes the image obtained through a camera system situated in the back of the display to present the users with an augmented version of the physical environment situated in front of them, creating the illusion that the device is transparent. Due to the relatively small size of the screen, the users can –and, sometimes, must– constantly reorient the display and/or their heads to see the entirety of the virtual scene. Since most of the computer systems do not have a stand to support their own weight, the user has to hold them with at least one extremity, which hinders the experience due to muscular fatigue.

Another recurrent issue associated to this configuration is the use of touchscreens as their main input mechanism. Although it provides a relatively good haptic feedback, having to use the fingers to interact with the system results in partial occlusions and unwanted displacements of the screen (due to the force exerted on its surface), resulting in a lack of precision that makes manipulating small or distant objects a frustrating task.

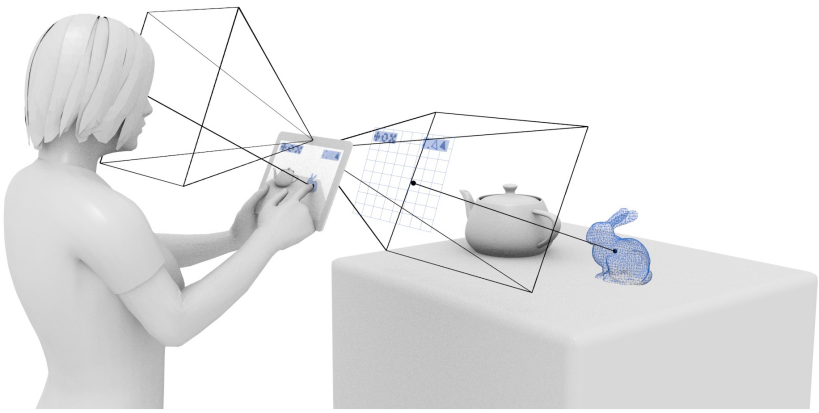

Figure 3.10: See-through indirect interaction scenario.

Mirrored (or "Magic Mirror"):

Usually reserved for fixed computer systems, this configuration makes use of a front-facing camera system situated next to –when not directly embedded into– the screen. In this way, the camera system can take images not only of the physical environment but also the users themselves, allowing interaction mechanisms based on the natural movement of their entire bodies. Furthermore, because these hardware devices do not have to be supported, users can move freely as long as they remain inside the vision cone of the camera system.

The augmentation process in these fixed systems is generally of higher quality (since they don't have to deal with the size, weight or energy consumption limitations associated to mobile platforms), but since the resulting image reflects the image obtained from the camera system, the users have to correctly identify themselves in the reflection (which can be a problem for persons suffering from dyslexia and even an impossible task for animals unable to pass the mirror test).

The main problem associated with this configuration is that, since the users have to be situated far from the screen in order for their image to be captured by the camera system, they cannot touch the screen and, consequently, they cannot use it as an input device or receive haptic feedback from it. Several solutions to this problem have been proposed, and while they usually address it, they require a physical controllers or complex image recognition algorithms (that do not provide haptic feedback and generally force the users to maintain a pose for several seconds to properly recognize it). Moreover, many of these solutions increase the occurrence of unwanted occlusions, because the projection of a virtual object can be occluded both from the point of view of the user and from the perspective of the camera system.

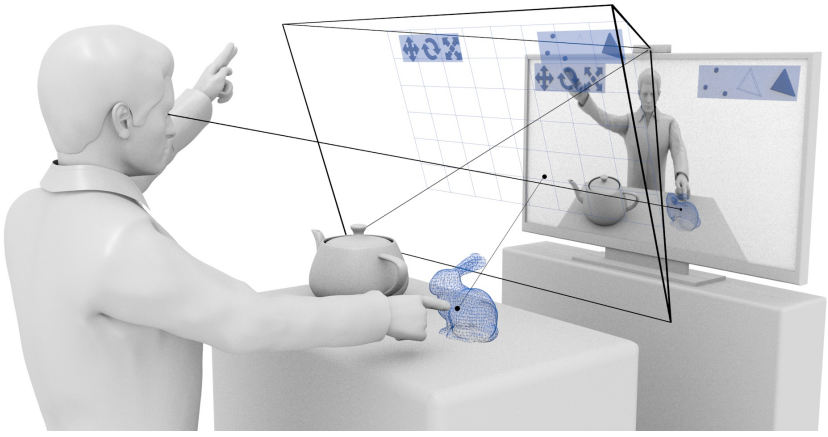

Figure 3.11: Mirrored indirect interaction scenario.

Integrated Interaction

Closely related to concepts such as *Ubiquitous Computing* [91], *Internet of Things* [92] or *Ambient Intelligence* [93] this interaction scenario encompasses all hardware configurations that perform the augmentation process over multiple physical locations at the same time. This process may be managed by a single computer system or, in more advanced models, a close network of independent robots (small computer systems with both sensors and actuators to operate in the real world).

One important aspect to note is that, while the previous definition excludes the singular application of the previously mentioned interaction scenarios, it does not deny the combination of several devices based on them to implement an integrated interaction scenario. What differentiates this last classification is the requirement of maintaining a constantly updated model of the surrounding physical elements (specially, the hardware devices of the computer system and the bodies of the users) so that that all users can perceive the augmentations without unwanted occlusions or overlapping virtual elements.

Generally, in this interaction scenario, the virtual elements that the augmentation process generates are not displayed on dedicated screens. Instead, the resulting images are projected onto the surface of real objects, making them seem like the augmentation is an integral part of the object itself (hence, the name of the interaction scenario). The advantages of this model are numerous; the users can perceive and interact with the real objects directly (without the limitations that a screen introduces in terms of visual resolution and haptic feedback), the virtual elements can be displayed with varying degrees of detail and multiple users can collaboratively operate the system without requiring any additional computational processing.

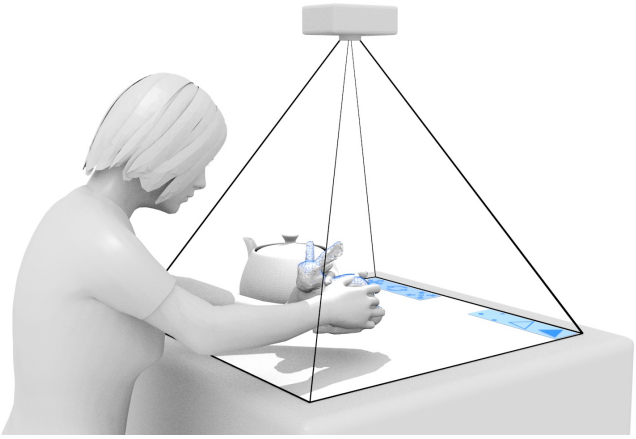

Figure 3.12: 2D projection-based interaction scenario. Note that the interaction is tied to the manipulation of physical objects, which is both intuitive and obtrusive.

Nevertheless, these advantages can only materialize in very controlled physical environments. Due to the application of 2D projectors as the main way to present the augmentations, all physical objects –including the body of the users– must be analyzed in real-time to adapt the projected image to the different surfaces. This is a complex situation that gets even more difficult to handle the more physical objects are introduced, because the system does not only have to process the light projection cone and the shadows the objects generate, but also the properties of the different materials (so that the projector can compensate the difference between the desired and the perceived color reflected on the surface). And, while this can be partially solved using multiple fixed projectors and a three-dimensional scanner [94], the most common approach is to minimize the number of objects in the interaction space and set the projection cones in a way that the users can not intersect with them (retroprojector-based CAVE systems).

Another approach to this problem is to assume the limitations of the model and, instead of constraining the users to an enclosed space, equip them with a portable projector. Attached to a wearable computer (mounted either on the chest [78] or on the head [95]), this projector displays the information over the surface of nearby objects –including their hands–. Coupled with a portable three-dimensional scanner and advanced image processing algorithms, these systems can provide more intuitive interaction mechanisms.

Furthermore, in the foreseeable future, advancements in three-dimensional holographic displays will –predictably– overcome the limitations of current projection technologies and allow users to interact with virtual elements positioned in the three-dimensional space. And, while this new techniques may introduce additional quandaries (absence of tactile feedback, privacy concerns, etc.), they will also allow much more natural and immersive user experience.

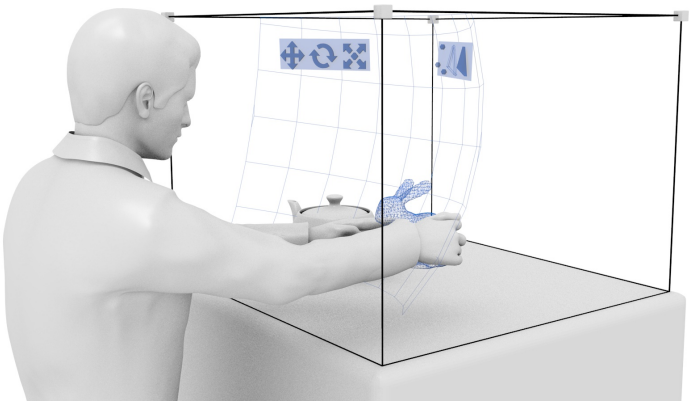

Figure 3.13: A 3D hologram-based interaction scenario. In this configuration, the interaction is performed through completely virtual objects

3.2. Object Model

Once the theoretical model has been established, the next step is to translate it into a logical -software- architecture that a computer system can easily process. A task that not only requires a deep knowledge of the different relationships between the components of the model, but also how should it be applied to real-world situations.

To achieve this ambitious goal, the only viable solution involves the application of the principles of Software Engineering. In this way, it is possible to divide this complex problem into more manageable components, enabling the creation of a solid object model that can account for multiple recovery scenarios.

Furthermore, because this new model will serve as the foundation of very heterogeneous implementations, it must be designed to be both language and platform independent. In this way, third-party developers are able to create new interaction techniques that take advantage of the capabilities of each computer platform (and define fallback mechanisms in case that the system does not provide a certain functionality). Figure 3.14 provides a good overview of the resulting object model.

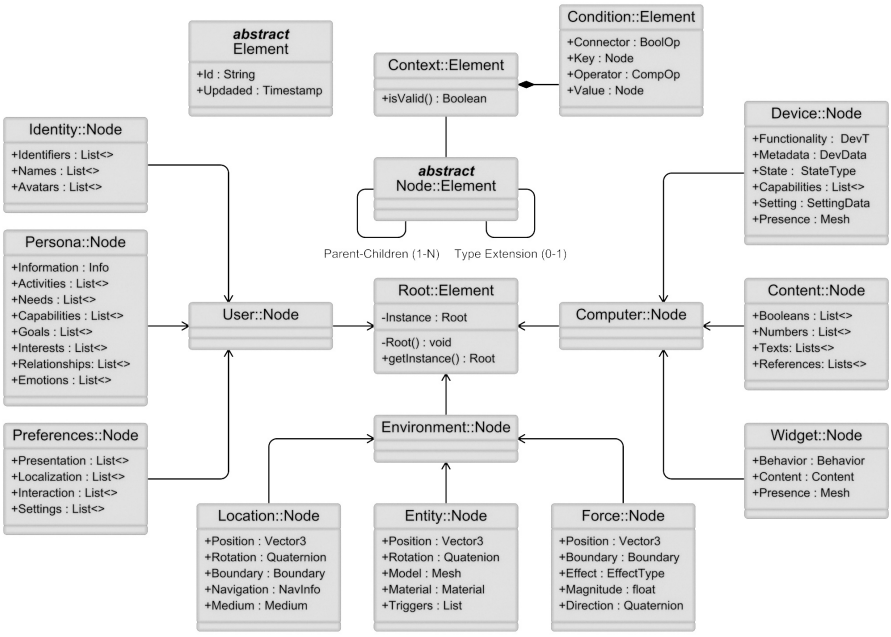

Figure 3.14: A simplified UML Class Diagram, showcasing the basic logical architecture of the object model.

3.2.1. Node Structure

The complex and dynamic nature of the software architecture required to support the proposed interaction model necessitates the creation of a simple but powerful data structure from which all other objects inherit from. A data type with the methods required to communicate with other instances in a meaningful way, allowing developers to easily navigate between the different representations of the logical elements described in the theoretical model.

In Software Engineering, this basic data type is commonly referred to as "Nodes", following the nomenclature employed in Graph Theory [96]. Each instance of a node contains a value (or a relatively simple set of values) and references to other nodes, making it possible to create sophisticated information structures without having to define complicated methods or inheritance mechanisms. While the actual implementation details of the node class will be discussed later in this section, it is important to note that the definition of the node class must contain the following members:

- **Unique identifier:** A mechanism that allow to unequivocally identify the node in the entire structure. In order to facilitate the serialization –and debugging– processes, it is recommended to employ alphanumeric values. In this way, it is possible to reflect relative and absolute paths within the node structure (by combining the node identifiers).
- **Content Metadata:** Although, in most cases, it may be possible to infer the type of the value contained within the node by its very definition (e.g. a Position node can only contain a three-dimensional location), certain nodes may require additional information about the size or the format required to correctly interpret it (e.g., the number of items in a data collection, the scale associated to a temperature value or the format of a Color node).
- **Connection List:** A collection of references to other node objects. This may be specified either as a generic data structure (child list) or part of the definition of the class itself (parent node, subnodes of a certain type, etc.)
- **State Management:** As the node contents are accessed, modified or removed, it is highly recommended to define additional fields and event handlers within the node class implementation, so that the system can manage the state of each instance properly. Therefore, it is possible to minimize the transfer data size (i.e., by only sharing the nodes that have been updated) or recover from an unwanted situation (by reverting to a previous state).
- **Context Management:** Each node must include situational data to determine its validity in a given context. This information may be provided by a subnode and may also be applied to other subnodes.

The result of applying this node definition to the theoretical model components is a data structure with high hierarchical connections. Excluding the secondary references between the main components (since, while useful, are optional), the resulting abstract data type can be categorized as an *unordered, non-binary tree structure*. Compared to a purely relational model, this framework offers significant benefits:

- **Simple Navigation:** Once established that *every node –except the root node– must have a valid parent reference*, it becomes much easier to navigate through the entire node structure. A feature specially valuable for classes with recursive associations, because it allows the constant modification of linked values (e.g., a widget adapts its contents according to the size of its parent).
- **Update Propagation:** When a node update is requested, the state change only has to be propagated upwards in the hierarchy. Thanks to this solution, the update function can accurately trace the nodes affected by the modification.
- **Direct Serialization:** As will be explained later in this chapter, the use of a tree structure for the object model greatly facilitates its conversion to other representations (such as XML or JSON).

Nevertheless, this reconstruction of the Object Model imposes new restrictions that have to be taken into consideration. Most notably, the inclusion of a new node that serves as the *Root* of the architecture and the identification of the secondary connections.

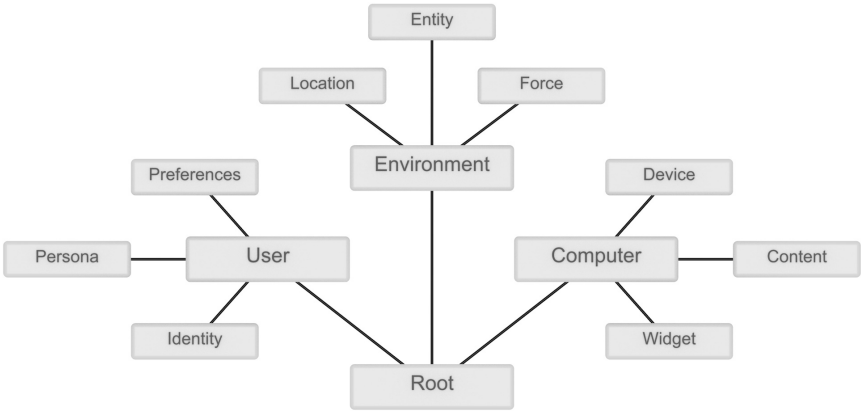

Figure 3.15: By recreating the object model as a tree structure, it is possible to easily navigate between the nodes that compose it (thus, simplifying the propagation of update messages and their serialization).

3.2.2. Main Nodes

The aforementioned structure comprises numerous nodes of very diverse nature. Because many of these nodes contain simple values or are very implementation-dependent, this section will only focus on those of special relevance to the proposed interaction model.

Root

The transformation of the object model into a tree structure necessitates the creation of a new object to serve as the *root* from which all other connections can be reconstructed. Nevertheless, this new node does not only act as a container for User, Environment and Computer nodes. because it is, by definition, a singleton object, it also includes several methods and operators that facilitate the management of the entire structure.

Context

As explained in the previous section, all nodes –except the root node– contain a context node reference to indicate under what conditions they can be considered valid (and, conversely, when they should be ignored). A state change that is then propagated down the node structure until the evaluation of an interior context node determines otherwise.

As simple as this mechanism may seem, it is a powerful tool that allows the real time management of extremely complex structures (without having to actually modify their contents). This system is further improved by the possibility of combining multiple context nodes into a single instance by simply referencing them.

Each context node is defined by a set of condition subnodes, each of which contain the following child nodes:

- **Connector:** A boolean operator (*and*, *or*, *not* and combinations of them) that establishes how the current evaluation result should be joined with previous ones.
- **Key:** The reference to the node which value needs to be evaluated.
- **Operator:** A comparison operator ($=$, $\neq$ $<$, $\leq$, $>$ or $\geq$) that defines the evaluation process. It is important to note that there may be comparison operators that are not applicable to all data types (e.g., string values only accept equality and inequality comparison operators).
- **Value:** The value that the referenced node will be evaluated against. This value can be provided either as a constant value or using another node reference.

(User) Identity

This node contains all the identification and representation data associated with the end users of the computer system. As explained later, this information is of critical importance to sustain the interaction model in multi-user scenarios, but, at the same time, it must be cautiously specified to avoid exposing data that can be exploited against the legitimate user's interests. For this reason, the present node definition encapsulates the relevant information in a way that minimizes the risk of over-sharing it.

Furthermore, the specification of this node allows the creation of multiple identities for a single user. These instances can be crafted in real-time depending on the situation (defined by the context nodes) and may contain temporal values that expire after a certain time span or when the associated user logs off from the system.

Additionally, any implementation of this node must account for the possibility of incomplete or invalid definitions. As long as an instance contains the adequate security credentials, it should be included into the structure and, to the greatest extend possible, repair the invalid nodes using provisional values.

The identity node contains the following three child nodes:

- **Identifier:** Unequivocally identifies the user within the system. Apart from a local id value, the node may reference subnodes containing global id data and secure login information.

- **Name:** The alphabetic value that the computer system must employ during the communications with the users (including those in whose it acts as an intermediary). While it may suffice to use a single alphabetic value (full name) in most situations, it is recommended to divide the information in several subnodes (name, surname, title, nickname, etc.) so it can be easily adapted to different contexts and cultures.

- **Avatar:** Describes the representation of the user in the environment. Because, in an augmented reality-based scenario, the interaction between the user and the computer is closely linked to their physical presence, it is strongly suggested that the avatar representation (or, at least, its collision mesh) matches the actual body of the user. This node must contain both the static, three-dimensional model of the avatar and the skeletal information of the body of the user, so that the avatar can be inserted within the different environments and updated frequently. Note that the actual representation of the avatar is threated as another entity of the environment so it can be shared with other computer systems without having to share the actual user node with them.

(User) Persona

As described previously in the previous section, in the proposed interaction model, a persona is a container for all the information regarding the psychological state of the user. This is an extremely powerful tool that enables the system to adapt the user interaction to the actual needs of the user in real time. Conversely, the sensitive nature of this information requires extreme security measures to safeguard the user's privacy. Aware of this fact, the object model stores this information inside a separated *persona* node to which only the computer system has access and whose contents should only be modified by user authorized devices.

Although there are innumerable ways to specify the psychological state of the user, the following subnodes offer a comprehensive view of it:

- **Information:** Contains different data values regarding the real-life circumstances of the user. These include data such as name, age (both as a numeric value and as a birth date), gender, address and phone.
- **Activity:** Describes an action that the user performs. These nodes can be defined either as a static collection of boolean values or as a dynamic list of actions currently in execution (thus, implicating that the absence of a specific action marks it as invalid within the actual context).
- **Need:** Defines the physiological and psychological needs of the user in the current situation. Unlike the action node set, each need node must include a numeric value to facilitate the creation of a priority list.
- **Capability:** Any special ability or disability that the user may have must be included within this subnode. In this way, the computer system can adapt user interface to the user's cognitive and physical capabilities.
- **Goal:** Identifies general goals that the user is currently trying to achieve through the use of the computer system. These can be as simple as fulfilling a given set of tasks (i.e., *TODO* lists) or complex long-term goals with multiple steps.
- **Interest:** These nodes reflect the preferences of the user in terms of "things that likes or dislikes". For each node, a numeric value should be included to allow direct comparisons between items. Negative values can be used to indicate those items that may cause a negative reaction for the user (allergies, phobias, etc.)
- **Relationship:** Defines social relationships with other users (regardless whether they are present in the current situation or not). This information can be assigned individually or through the creation of user groups.
- **Emotion:** Determines the current emotional state of the user.

(User) Preferences

While the idea of grouping the customization options into a single entity is common in user interface design (visual themes, style sheets, etc.), these options rarely go beyond the visual representation of the objects. Within the model proposed in this chapter, however, the *Preferences* node allows the complete redefinition of the interaction techniques (and the widgets associated with them). In this way, each individual user can manually adapt their experience to their needs and desires without having to modify the widgets themselves.

Instead of being a static structure, this node contains multiple subnodes that act as dynamic libraries from which the system can extract the necessary data to recreate basic *Content* and *Widget* nodes. These child nodes include:

- **Presentation:** Contains a dynamic collection of parameters that define the general appearance of the contents of the user interface. These parameters include relatively simple data structures such as colors or animations, but also references to external multimedia files (images, sounds, videos, 3d models, etc.).
- **Localization:** Different cultures have distinct ways to present the same information; informal languages, visual/auditory iconography, punctuation signs, date formats, metric systems, keyboard distributions.. Even the witting direction can change between languages (e.g., Arabic, Japanese, etc.) The localization node provides a way to easily describe how the information should be displayed and/or interpreted in each case.
- **Interaction:** Enumerates the different interaction techniques that the users can perform. Each node must define the physical context in which it takes place (action) and the logic response to it(reaction). Additionally, it can specify input shortcuts for expert users and basic content associations (animations, sounds, etc.) for each action.
- **Settings:** Contains a list of device settings that the user prefers. While different platforms may not be able to fully comply with these requirements, they must try to satisfy them to the greatest extend possible.

It is important to note that, while other *Persona* and *Widget* nodes can temporally modify (i.e. expand upon) individual interaction techniques, ultimately, it is the data contained within the *Preference* nodes that manages the entire user interaction. This offers the user an unprecedented level of control over the general experience, even allowing him/her to establish the conditions under which specific user interactions can be used (and to extent).

(Environment) Location

The location node provides a way to divide an environment into three-dimensional regions. This greatly simplifies the definition of interaction spaces by constructing new local coordinate systems where the entities can be easily positioned. Furthermore, by allowing the specification of spatial boundaries (a technique usually referred to as *geofencing* [97]), this node enables developers to vastly optimize the user experience. A critical feature for AR-based experiences on mobile platforms.

Although this node has been specially designed to facilitate the inclusion of real world spaces into the physical simulation, it is not restricted to this function. For example, this same node can be used to specify different sections within the screen space, making the definition of HUDs a much easier task.

It is possible to further subdivide the spatial location through recursive references (i.e., location subnodes), allowing the definition of complex or non-static locations (such as buildings, vehicles, etc.). However, additional care must be taken to ensure that inner locations are contained within the upper limits and do not unintentionally overlap with each other.

The required subnodes of a location node are:

- **Position:** Defines the three-dimensional displacement of the new coordinate system. This data can be specified either as a global value (using geolocation standard systems [98] or fiducidal markers) or, in the case of sublocations, with a value relative to the parent location. While not mandatory, it is strongly recommended to use the center of the location as the position value to minimize (floating-point) errors.

- **Rotation:** Optional rotation value that should be applied to the new coordinate system (and all entities associated with it).

- **Boundary:** Specifies the spatial boundary of the location. It can be defined as a sphere (radius value), a box (three-dimensional size values), or a complex shape (geometry node).

- **Navigation:** Additional information to successfully navigate through the environment, including navigation points/meshes and portals (connections) to other locations.

- **Medium:** The properties of the physical medium contained within the location. These include: air pressure, temperature, humidity, etc. Beyond being useful for a more accurate physical simulation, this information can determine the best interaction technique for that situation.

(Environment) Entity

An *Entity* node contains the information of a three-dimensional object within an environment. This object (not to be confused with a class instance) may represent a real world counterpart, a user's avatar or a component of the user interface. Whichever the case, this node stores the necessary data to adequately simulate and display it.

While each instance of the *Entity* node can only have a single representation, it may be present in more than one location (or environment, if they are correctly *linked*) at the same time, allowing better occlusion management and advanced physical interactions. For example, this approach grants the user the ability to employ physical objects as virtual pointers to select or, even, "press" on widgets.

As *Location* nodes, *Entity* nodes can have recursive references to subdivide complex or non-static objects. This approach can also be used to provide different levels of detail (geometric/collision meshes, textures, etc.) for the same object.

Excluding secondary connections to *Avatar*, *Location* and *Widget* nodes, the *Entity* node may contain the following subnodes:

- **Position:** The three-dimensional placement of the entity within the location. If the object belongs to several locations, a different position for each one must be provided.
- **Rotation:** The orientation of the entity. It can be specified using Euler angles, a quaternion or by providing a reference to another entity (usually, the user's point of view) towards which the current entity must be reorientated.
- **Model:** The three-dimensional model that be used to represent the specific entity. If the entity belongs to the physical world, only its collision mesh needs to be taken into account.
- **Material:** Defines the mechanical and chemical properties of the entity, allowing its precise physical simulation.
- **Trigger:** Usually generated from widget behaviors, trigger nodes specify the type of *action* associated to a modification of the entity properties. In this way, the system can detect significant changes in the environment (regardless of whether the user initiated it or not) and activate the adequate response to them.

(Environment) Force

To simplify the physical simulation of the entities of the environment, physical forces can be defined using *Force* nodes. These nodes allow a precise definition of the physical forces operating within a real-world environment, without having to constantly analyze them. For example, while gravitational forces can be obtained using accelerometers, its effect can be considered constant within an small interaction space (and, thus, the accelerometer data can be used to better estimate the user movement).

In a similar way to *Entity* nodes, *Force* nodes may be shared between multiple locations (and linked environments). However, since translating the physical forces generated in a virtual environment into the real world require the presence of special devices (force-feedback systems), any implementation of the proposed model must be able to recognize when its actual application should be discarded.

As previously explained, a physical force can be defined either with a vector function or as a emitter-attractor system. This entails the creation of multiple subnodes (many of whose may not always be required):

- **Position:** The relative position of the force within the environment.
- **Boundary:** Limits the application range of the force to a spatial region. It can be defined as a sphere, a non-axis aligned box, or a geometric shape.
- **Effect:** Indicates the physical properties affected by the force. Each property may have a different scale factor.
- **Magnitude:** The degree to which the entities affected by the force will be modified. These measures can be expressed either as constant values, as linear equations or exponential functions.
- **Direction:** The direction towards the force is applied. If no value is defined, the force is assumed to be omni-directional.
- **Emitter:** Describes the position, rotation, size, shape, dispersion angle and magnitude of a force emitter. Attenuation/decay rates can be defined directly or inferred from other values or from the properties of the medium (described in the *Location* node).
- **Attractor:** Similar to the previous subnode but, due to the nature of attractors, the magnitude value is reversed.

(Computer) Device

As its name implies, a computer system is an aggregation of devices that operate together to perform a given computational operation. Each device is highly specialized in a given task, so in order to achieve complex goals they have to exchange data between themselves.

While the diversity of computer devices makes its precise specification an impossible task, the device node provides a common framework that allows their basic management and the definition of contextual information based on them.

To facilitate their access, device nodes are indexed not by a custom identifier but by the functionality they provide (General computation, Graphic display, Sound system, external connection, etc.). In case that more than one device provide the same functionality, they can be grouped within a single device node and receive an internal numeric index. In this way, contextual conditions are much easier to define.

The relevant information of a device is divided into the following subnodes:

- **Functionality:** The computation functions that the device can provide (input/output mechanisms, general computing, network communication, etc.).
- **Metadata:** Contains general information about the device, including serial number, vendor name and driver version.
- **State:** The current status values of the given device. If no state is defined, the device should be marked as disabled.
- **Capability:** Enumerates the operational capabilities of the device. These include: resolution, refresh rates, available sensors, etc.
- **Setting:** Describes the current operation mode of the device.
- **Presence:** In a similar way to the *Avatar* node (associated to the user identity), this node defines the physical presence of the computer device. The position of the resulting entity can be defined relative to another device, a physical location or, in the case of wearable devices, to the entities associated with the *Avatar* node.

It is important to note that devices do not have to be integrated within a computer platform in order for them to correctly operate. The proposed model allows the definition of *Device* nodes that represent analog mechanisms (such as mercury-in-glass thermometers or analog multimeters) or biological systems (e.g., human body reactions, sunflower orientation, etc.) that can be analyzed by the computer system to extract situational information.

(Computer) Content

Content nodes include the data structures required to sustain the user interaction. Usually, these nodes specify the symbolic information displayed by the widgets (static resources such as texts, images and sound), but they can also store dynamic, user-generated information and serve as contextual variables that control the user interaction (e.g., the slide number in a presentation or the operator values of a calculator widget).

This logical division between the data definition and its actual representation enables users to easily update and share this information without the need of external mechanisms. Additionally, this approach allows the creation of new contents in real time and the seamless definition of collaborative edition spaces.

Another important feature of the *Content* nodes is the possibility of using recursive references to subdivide the information into smaller data structures. This not only results in optimized communication with other computer systems (by grouping items that update regularly into their own subnodes), but also allows the replication of the widget structure, making the recreation of the association connections a much easier task. Furthermore, this approach offers multiple ways to localize specific contents (without having to recreate the entire content node for each culture or language).

Formally, *Content* subnodes should accommodate all kinds of data types. However, because the inclusion complex data types and multimedia BLOBs (Binary Large OBject) can negatively affect the global understanding of the object model status -and, thus, the creation of contextual conditions-, it is strongly suggested to limit the size of data structures and create external mechanisms to appropriately manage them (leaving only a reference subnode to maintain the connection with the actual object instance). These data structures can be constructed using the following subnodes:

- **Boolean:** Contains a single boolean value that can be used for the definition of contextual conditions.
- **Number:** Describes a numerical value. The actual data type used to store the value (integer, floating-point or fixed-point) can be directly specified or inferred from the value itself.
- **Text:** Contains an alphanumeric string that can be used as a text resource. By using additional string serialization techniques (such as CSV or JSON) this node can be used to store multiple and/or more complex data structures (a useful technique to create advanced widgets).
- **Reference:** Contains a reference to a multimedia element that is managed outside the object model. Additional metadata must be included to indicate the media type (image, audio, video, 3d model, etc.) and how to interpret it correctly.

(Computer) Widget

As previously explained, widgets (also commonly referred to as *user controls*) are the building blocks from which UIs are made. In the proposed model, *Widget* nodes try to encompass all the interaction logic in a hierarchical structure that simplifies the definition of interaction task.

While most UI design systems establish a clear distinction between common widgets (labels, buttons, checkboxes, etc.) and containers (windows and panels), the proposed object model does not differentiate between them. Each Widget node can have recursive references to other instances, allowing the definition of complex widgets from the combination of simpler ones. In this regard, the concepts of *application* and *plugin* lose their meaning, thus allowing an easy reutilization of complete sections of a user interface.

This compartmentalization concept extends even further by the ability to create copy instances of other widgets in real time. Either by using the type subnode or by employing the extension mechanisms of the transmission language (later explained in this chapter), user interface designers can have a extensive and extensible widget library at their disposal.

The *Widget* node exposes the following subnodes:

- **Behavior:** Enumerates the sequences of reactions associated with each action that can be performed on a widget, including -but not limited to- initialization, activation/deactivation, selection and manipulation. Advanced functionality (beyond basic management of *Content* items), can be defined by including ECMAScript code within the Reaction nodes.
- **Content:** The reference to the *Content* node(s) that contain(s) the data structures associated with the widget. While it is possible to directly specify the *Content* node data, it is highly discouraged. Instead, it is recommended to employ concatenated, relative references that replicate the *Content* node hierarchy.
- **Presence:** Describes the presence of the widget in the interaction environment(s). In an analogous manner to *Avatar* nodes, this subnode manages the *Entity* nodes associated with the widget. In most cases, these entities will be virtual objects (3d geometries, texts, etc.) that should be created and destroyed according to the user context, but they can also be actual objects present in the real environment (allowing a much more intuitive way to interact with the user interface). Additionally, this subnode also contains the required information to adequately position the different entities (specified with an specific location, an absolute position vector or a relative value based on the parent widget).

3.2.3. Implementation Considerations

Translating the proposed object model into actual code is a more difficult task than it might initially appear. Any software engineer/architect that attempts to implement the aforementioned node structure must first understand the implications of developing a middleware solution that manages the entire user interaction system in real time.

Node Class

As the base from which all other objects are derived, the implementation of the node class must provide the fields and methods to manage it within the global structure. In actuality, this can be achieved just by attentively handling the reference to the parent node instance, ensuring that all parent-child relationship are correctly updated.

However, due to this relative simplistic approach, the node class has to be defined with the utmost care to avoid creating "god objects" that take too much memory space and become very hard to expand upon. If possible, it is recommended to minimize the amount of class members that derived classes must inherit (or, even worse, implement). In certain instances, it might be a better solution to code structure-related operations as static methods or in external manager objects, rather than over-complicate the definition of the base node class. For example, this may be applied in single-user interactions by providing a static reference to the associated user node.

Another important consideration regarding the node class implementation is how to specify the collection of references to child node instances. While a simple dynamic list may be enough for most situations, to facilitate future access to subnode instances, it is recommended to use dictionary data types that allow their indexation through their identifier. In this way, it is also easier to detect duplicated references and ensure that all identifiers are unique.

Finally, if the programming language supports generic type definition, it is advisable to implement a generic version of the Node class to simplify the creation of nodes whose only function is to group subnodes of the same type. This approach streamlines the implementation of new nodes and their correct integration within the hierarchy (because it inherently negates the possibility of linking references of an invalid type).

Data Type Nodes

Beyond the node classes discussed in the previous section, there are numerous node –derived– types that should be taken into account when attempting to implement the proposed model. These nodes contain basic data types that can be referenced from multiple points in the node hierarchy (although, as stated before, they must always have a single, valid parent reference).

Primitive data types such as numbers (i.e, boolean, integer and real values) or alphanumeric strings can be contained within the node object. More advanced data types, can have their components divided into several child nodes and use the main node to provide additional format information. Examples of this complex nodes are:

- **Temperature:** While it is recommended to use a floating-point number in the Kelvin scale to indicate a temperate measure, different data types or scales can be employed (as long as their usage is indicated in the main node).
- **Time:** Although in most situations it might be enough to use a single primitive value in accordance with the ISO 8601 standard [99], the definition of time nodes may also include the time zone or date format.
- **Color:** Color values can be defined either using generic names (*blue*, *green*, *white*, etc.) or by dividing their color components into several subnodes. In the latter case, is advisable to also indicate the color model/space in the main node (RGBA, CYMK, HSV, HSL, CIELAB, YUV, etc.)
- **Currency:** Apart from a fixed-point numeric value, this node must indicate the currency and should also include a timestamp value to easily adjust the value to other spatial-temporal conditions.
- **Vector:** Specifies a point in a three-dimensional space. If the components do not follow the standard XYZ order, it must be reported in the main node.
- **Geometry:** One of the more complex data types employed regularly in the node structure, the geometry node provides the spatial definition of a three-dimensional entity. There are multiple formats to codify and organize this information (Constructive Solid Geometry, volumetric meshes, etc.), but the most common solution is to use polygonal meshes to define the limits of the model. These meshes are generally specified with a vertex array and a list of indexes to arrange these vertices into triangles. Additional information can be included to indicate the normal vectors, color/material indexes, bone weights and animations associated to each vertex.

Code Portability and Interoperability

When translating the components of the object model into actual code, it may be tempting to include platform-dependent instructions within the node classes. Nevertheless, it is strongly suggested to maintain this functionality separated, creating an abstraction layer that offers significant benefits in terms of code portability and reusability, while having a minimal impact on general performance.

In practice, this approach implies the development of two separate software subprojects. One that contains the definition of the data types and manages the node structure and another that handles its actual representation on a given platform (whether through the use of a middleware solution or not). In this way, it is possible to maintain and upgrade each subproject -almost- independently, while also taking full advantage of the capabilities of each platform.

One effective way to achieve this goal is to include a void reference to an abstract class (i.e., *Representation* class) that each implementation can use as a link to the node instance (and has an entry point to the entire node structure). Furthermore, by using inheritance mechanisms, developers can expand upon the original functionality and define enhanced representations for specific nodes.

Nevertheless, it is important to understand that the main motivation for employing this programming scheme is to ensure the interoperability of the interaction components between different hardware platforms. Beyond providing good backwards compatibility mechanisms and easing the creation of ad-hoc collaborative user experiences, maintaining the implementation of the proposed object separated from the platform-dependent code creates a "leveled playing field" in which all developers can create their software solutions (with the assurance that they will work as intended) and compete in equal terms. Meanwhile, restricted by the definition of the proposed interaction model, hardware manufacturers can focus their efforts on improving the technical capabilities of their products (instead of being forced to employ anti-consumer practices to try to gain a larger market share). All this results in a more sustainable *digital ecosystem*, one that benefits all involved parties (including the end-users) and one that can grow organically, without the need to forcefully advocate for the adoption of the proposed interaction model.

3.3. Network Model

Beyond the confinements of research centers, the creation of an open and distributed communication system is an indispensable requisite to construct a successful and sustainable software ecosystem. As good as the initial set of interaction elements may be, if there is not an easy way to expand upon it and share these modifications with others, the platform will soon become obsolete.

In other words, to successfully support the interaction techniques described in the previous sections (specially, those that take advantage of the collaborative edition or the physical simulation systems) in the long run, it is necessary to define a *network model*. A software-hardware system capable of establishing dynamic connections between multiple computer platforms and, thus, facilitates the exchange of information.

While not technically part of the classic HCI formalization (because human beings are not directly involved in the network communications) the accurate definition of this representation of the proposed model is, nonetheless, crucial for the creation of meaningful user experiences. In the first instance, the formulation of a proper network model enables users to have access to additional interaction components –including new contents and widgets– without having to resort to external services (e.g., software distribution systems or search engines). Moreover, thanks to the all-encompassing nature of the underlying object model, it is possible to easily share the definition of the interaction space (i.e., physical entities and public user data) and even –temporally– assume direct control over shared computer devices.

Furthermore, it is important to note that the information exchange within this network model does not have to be limited to one direction, nor constrained by time/space restrictions. Thanks to its hybrid architecture, the proposed model is able to distribute the resources across the entire network, ensuring their availability without the need of additional publish-subscribe systems (although the option still exists).

Throughout the present chapter, the reader will be able to gain a complete understanding of this new network model; from the aforementioned logical architecture to the mechanisms that facilitate the exchange of interaction elements with other network nodes.

3.3.1. Hybrid Architecture

Once the general objectives of the proposed model have been properly established, it is necessary to define the underlying network architecture that it is based on. A complex process that requires the careful balancing of numerous –and often conflicting– design requirements [100].

Due to the dynamic nature of the network topologies necessary for the creation of collaborative edition spaces, the application of the principles of *Peer-to-Peer* (P2P) architectures might seem like a good idea at first. However, this unstructured approach presents several issues in terms of scalability and information propagation (a critical factor for search/discovery mechanisms, as will be explained in the following section). Conversely, client-server architectures generally allow an easier and faster search but, if the structure is highly centralized, it might be vulnerable to Denial of Service (DoS) type attacks and do not usually allow the dynamic linking that highly mobile platforms require.

Under these circumstances, the proposed model employs a hybrid framework that combines the advantages of the aforementioned architectures, while also minimizing their disadvantages. Figure 3.16 shows an general overview of this architecture, showcasing the features that will be individually explained in the next subsections. Please note that the network presented in this illustration (and replicated in figures 3.17, 3.18 and 3.19) is only a simplified representation of the proposed architecture and that many of its features have been designed for large scale networks (with hundred of thousands of interconnected nodes).

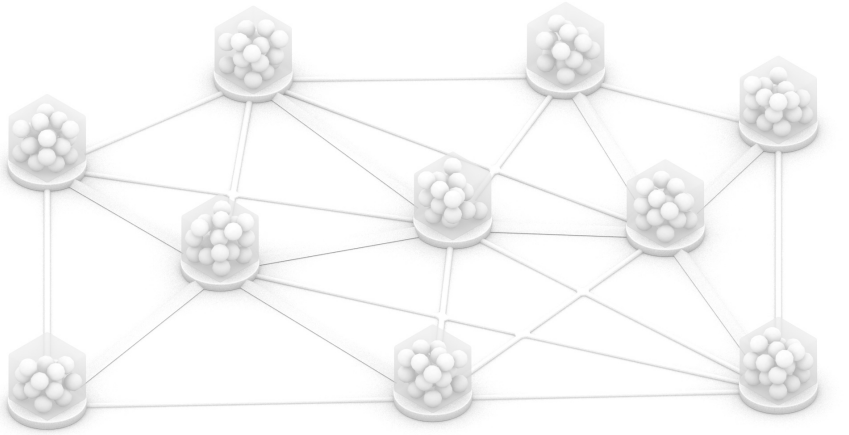

Figure 3.16: Independently of the underlying physical connections, the proposed hybrid architecture facilitates the exchange interaction elements between the different network nodes and, thus, the definition of collaborative user experiences.

Equipotent Participation

Following the P2P schema, the proposed model allows all network participants to communicate with each other in a direct manner, without the use of an external administrative system. The framework provides a –software– overlay that enables the free exchange of resources between the different participants, independently of the type of the computer system (single-user, dedicated, etc.), the network topology, or the communication technology (wired or wireless) employed. An approach that greatly facilitates the creation of ad-hoc networks for mobile, collaborative interactions.

However, it is important to note that this does not mean that all network nodes are equally privileged in all situations (i.e., they are not strictly *peers*). Although any computer can initiate the information exchange at any time, it does not mean that the receiver has to accept it unconditionally. As will be discussed later in this section, a participant may have to authenticate itself (i.e., provide valid credential data) and/or define a special communication channel in order to establish a login session.

While these mechanisms are restrictive in comparison with the open resource exchange in many P2P networks, they allow the easy implementation of security measures at a low level. Beyond providing a good protection against the most common types of attacks, this approach allows the arbitration of the information exchange and facilitates the creation of leveled play-fields for collaborative edition applications (e.g., interaction designers can easily designate specific network nodes to serve as "referees" in a multi-player game).

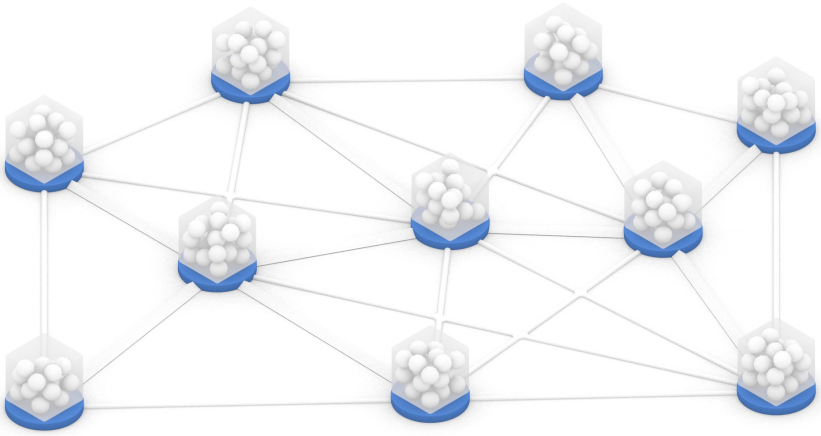

Figure 3.17: In principle, all network nodes can establish direct communication channels between them (the most basic rule of P2P networks). However, to facilitate the creation of *Webs of Trust* (an important factor in the definition of robust security mechanisms), the implicated nodes should always be able to negotiate the conditions of the information exchange.

Decentralized Structures

As mentioned before, network architectures with centralized structures generally allow a faster search time but exhibit performance bottlenecks and single points of failure. Furthermore, the rigid connections generated by the client-server relationships impose serious limitations for the users in terms of mobility and collaboration.

On the other hand, the unstructured nature of –pure– P2P architectures allow network nodes to dynamically establish links between themselves, resulting in a better load-balancing (due to the distribution of the search queries) and higher fault tolerance (thanks to multi-path routing methods). However, as the network grows in scale, the lack of a proper structure makes the search for non-popular resources increasingly harder to find (and, depending on the dynamic query termination method employed, some resources might even become unattainable for part of the network).

To overcome these limitations, the proposed model supports (and encourages) the generation of decentralized structures over a P2P architecture; stable arrangements of closely linked, dedicated systems that facilitate the communication throughout the entire network, even if it scales to a global –planetary– level. Although these nodes can also exchange interaction elements (e.g., to establish a login session, returning cached responses, etc.), the main function of these computer systems is to *serve* as a part of distributed directory system, storing categorized routing links to other nodes, in order to make them –and the interaction elements that they contain– easily accessible. As for the categorization system itself, it is recommended to employ dynamic mechanisms that take into account the contextual information contained within the search queries to balance the workload between different nodes more efficiently.

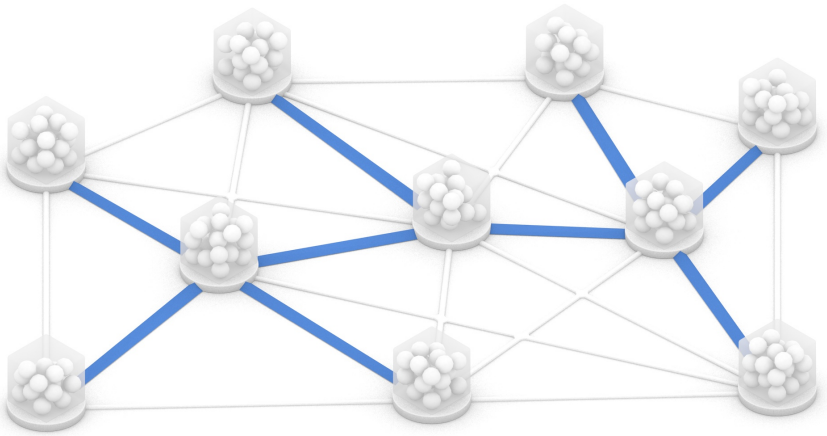

Figure 3.18: The proposed architecture facilitate the autonomous generation of decentralized structures within the network.

Distributed Resources

In the proposed framework, resources can –and often should– be distributed among multiple nodes across the entire network, reducing the search times and ensuring their availability (even for distant nodes or in the case of disconnection of the original source). However, because distributing the entire resource may involve caching (and, thus, replicating) a large volume of data, multiple factors must be taken into consideration.

First, to facilitate the search and avoid replicating the same resource more times than necessary, a proper naming mechanism should be employed to unequivocally identify and index each distinct resource within the network. This can be achieved through simple directory-based naming techniques [101] but, to be completely certain that there are not duplicated identifiers throughout a decentralized structure, it is advisable to complement it by making use of *Distributed Hash Table* (DHT) mechanisms [102]. These systems employ hasing algorithms to generate a universal identifier from the actual data of the resource, ensuring that each result is unique and can be used for indexing the resources in a decentralized structure.

Nevertheless, as the network grows in size, relying only on naming mechanisms is not enough to guarantee the efficient access to all resources; they have to be distributed adequately through the different sectors of the networks. To achieve this goal, this model applies *clustering* techniques, to construct a hybrid system that facilitates resource search processes while also keeping the beneficial properties of a "loose" architecture. These techniques take into account the actual topology of the network (the physical location of the nodes and the quality of their connections) and existing decentralized structures, to divide the network into progressively smaller logical regions.

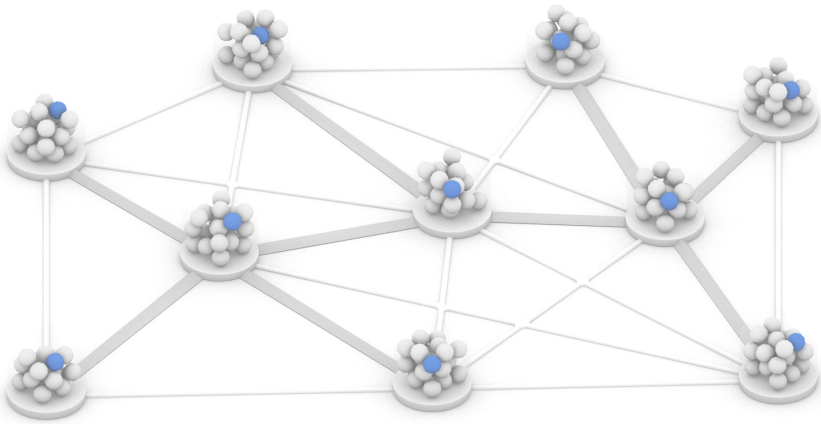

Figure 3.19: The hybrid network architecture provides the necessary mechanisms to distribute the interaction elements across the entire network.

3.3.2. Discovery Mechanisms

In any mature software ecosystem, the number of available interaction options increases over time, making a mechanism to select the appropriate one(s) –for the current user context– an essential tool for its correct operation. In an initial phase, a simple menu that lists all the possible choices can be enough to efficiently interact with a computer platform (i.e., the *ls*, *info* and *help* commands in Unix-like operating systems). However, as the complexity of a system grows and more interaction components are available, more advanced mechanisms are required.

Current software distribution platforms employ a combination of naming systems, categorization models and search engines to allow the user to actively look for the desired option. Nevertheless, neither of these search techniques are free of issues: naming systems (such as DNS [103]) require the prior knowledge of the –exact– identifier associated to each option, categorization models with many hierarchy levels can be difficult to navigate and search engines often suggest invalid results. In any case, when the number of available choices vastly exceeds the cognitive capabilities of the users (e.g., the main app stores/markets for mobile platforms or the World Wide Web), neither of these search mechanisms offer a reliable solution.

In an attempt to overcome these limitations, in recent years, many search engines have started to take into consideration several aspects of the user context to optimize their *suggestions*. However, the data provided by the computer system (mainly, geospatial locations and basic user preferences) and the predictions based on past behavior only allow a limited refinement of the results (filtering and ordering them by relevance or time range). Ultimately, it is the user who has to input the correct search terms and choose the appropriate option.

With a wider and more precise definition of the user context that the proposed interaction model introduces, the situation changes drastically. Because each node of the object model presented in the previous section already provides the contextual metadata necessary to indicate the conditions in which it (and all its subnodes) should be included into the node hierarchy, it is possible to take advantage of this very concept to constantly search and obtain new –and meaningful– interaction elements (including widgets, contents, devices, environment entities, etc.) without the direct involvement of the user. This automatic search process, denominated *discovery*, that takes into consideration past choices, current circumstances and, even, predictions of future events to provide a more direct and seamless user experience each time.

Within the proposed interaction model, the discovery mechanism operates as a background process, constantly monitoring the entire node hierarchy. Whenever the global context changes significantly (e.g., when the user enters a specific location or a certain amount of time passes), the discovery mechanism must automatically look for new *compatible* computer systems through the available networks. A procedure that can be performed both by actively broadcasting an initial message (and awaiting a response) or by passively listening to regular broadcast messages of other network elements (a preferable behavior in the case of shared devices).

If the initial response does not explicitly disallow it, the discovery subsystem must then attempt to initiate a login session with the new computer platform. After the authentication process is completed, both network nodes must negotiate the communication parameters to ensure the sustainability of the information exchange (i.e., version of the transfer protocol, error recovery controls, data formats that each one accepts, etc.).

Once the connection is properly established, the nodes can exchange request messages to discover new interaction elements. Thanks to the contextual information (extracted from the user context at the time or generated using prediction of future user behavior) included within these request messages, the receiver can take advantage of the very same context condition mechanism present in the object model to determine the interaction elements that the requester node is searching for.

It is important to note that, unless explicitly stated in the request, the associated response message does not have to include the complete definition of all the interaction elements valid for the specified context. Depending on the network conditions, the user preferences and the actual size of the data transfer, it might be more convenient to enter a negotiation phase where both participants attempt to narrow down the list of interaction elements to share (requesting with progressively more detailed contexts and responding with more complete definitions). This allows network nodes to access, reference, cache or store a much larger number of interaction elements and constitutes the basis for the autonomous generation of the aforementioned decentralized structures within the network architecture.

Nevertheless, the communication between network nodes is not restricted to request-response cycles. At any time, a network node may broadcast special messages to announce the addition/update/removal of an important interaction element or to warn about a critical change that can affect the user context (possible disconnections from the network, threats to the physical security of the user, etc.). This proactive approach ensures the availability of interaction resources without the need for additional publish-subscribe mechanisms.

3.3.3. Security Measures

Up until this point, the specification of the interaction model has not contemplated the possibility of errors or inconsistencies in its internal definition. However, once the network model allows the exchange of interaction elements with other computer systems, it is imperative to define several security measures to protect it against network errors and potential attacks. After all, security and privacy concerns can easily undermine any attempt to introduce a system that relies so heavily on the understanding of the user context.

Nevertheless, it is important to note that, in computer science, *security* is a complex concept that involves many different disciplines; from electrical engineering (to detect wiretapping), to psychology (to avoid scams based on *social engineering*). And, as heterogeneous as their components can be, security measures must permeate the entire definition of the interaction model to be effective. As often stated, "*security is like a chain; it is only as strong as its weakest link*".

While a complete protection may not feasible –or, even, preferable– in each and every situation, it is essential to implement an *Information Security Management System* (ISMS). An independent process that constantly monitors the communications to prevent the leakage of important information, stop possible attacks (or, at least, minimize their negative effects) and, overall, ensure the integrity of the internal object structure. Three distinct issues that will be examined in further detail throughout this section.

In any case, it is of critical importance to realize that it is not possible to confer a sense of safety without actually providing the end users with the tools –and knowledge– necessary to maintain an effective control over the aforementioned security measures. Therefore, any ISMS implementation must be accompanied with a *control panel* that enables the users to define the security policies with different degrees of complexity (*enticing* the users to take better responsibility for their security by giving them more options as they learn to take full advantage of them). Furthermore, the definition of these security preferences might also contain contextual conditions to better adapt them to the circumstances of the interaction context (e.g., if the user is a minor, the system might require the express content of his tutor before accessing certain contents).

Structural Integrity

Before addressing the security issues related to network communication, it is necessary to define the inner mechanisms that ensure the structural integrity of the individual network nodes. After all, *security is a shared responsibility* and, therefore, if these mechanisms are not robust enough, they can compromise the security of the entire user base.

As explained in the previous section, each computer system that implements the proposed model employs a node structure to manage the different interaction elements. And while this greatly simplifies transfer of these elements between multiple computer systems, if they are not correctly defined, it can negatively affect the integrity of the whole node structure (mainly, due to the need to recreate secondary connections between the different parts of the object model). A circumstance that can be exploited by possible attackers to inject *malware* (malicious software) or, even, take control of the computer system.

Needless to say, it is of crucial importance to incorporate mechanisms to prevent this kind of situations (or at least, to recover from them as soon as possible).

First, to guarantee the integrity of the interaction elements shared between the different network nodes the first step is to devise a naming system to uniquely identify them across the network and provide and verify that they have not been altered. A goal can be accomplished through the usage of the aforementioned DHT mechanisms (as long the hash values are generated using the complete definition of the interaction elements), complemented with simple reference maps (i.e., string dictionaries that provides aliases/translations for the hash values) for easier access. Nevertheless, it is also recommended to include recovery systems that contemplate the possibility of network errors and invalid node references, detecting and solving any possible incongruence behind the scenes.

Additionally, this mechanism can also be used to easily establish access restrictions to specific nodes. Instead of protecting the creation of secondary connections withing the object model with complex authentication systems (passwords, encryption, etc.), the different nodes of the hierarchy can specify restrictions through the same hash table and naming systems. In this way, those users requiring additional protection for the resources they share (e.g., media content providers, financial institutions, etc.) can do so without having to resort to encryption mechanisms or external *Digital Right Management* (DRM) solutions. Furthermore, by allowing the use of *wildcard characters* and *regular expressions*, these restrictions can be applied to a large set of nodes while maintaining the extension capabilities of the model.

Nevertheless, to prevent the possible misuse of interaction elements in phishing attacks (a type of scam that consist on replicating legitimate user interfaces to mislead users into providing sensitive information), any implementation of the proposed model must also provide the tools required to selectively disable the extension mechanism (when is deemed to be unnecessary). A solution that can be as simple as allowing the user interface designers to label certain nodes as "not extensible" or "not usable" outside a specific context (in a similar way that many current programming languages include keywords to restrict class inheritance). Furthermore, this system also allows the definition of template nodes (similar in concept to abstract classes or interfaces) without the need of a special context condition.

Another possible attack point that the developers responsible for implementing the proposed model should be fully aware of is the definition of the widget behavior. As explained in previous sections, the interactions with the users are based on physical simulations (mainly, through collisions between the users' avatars and the virtual entities generated by the widgets). This opens the door to a myriad of possible exploits, from creating "invisible" widgets over soft keyboards that record the individual strokes (a technique often referred to as *keylogging*) to generate widgets with the intention to hide critical information from the users (even putting their physical security at risk). A situation that grows even more complex when allowing the inclusion of scripting languages such as Python, Ruby or Javascript.

Against this situation, the best approach is to create an independent subsystem dedicated to analyze the widget behavior and ensure that the associated environment entities do not occlude each other (or an important element of the physical world). Beyond acting as an anti-malware measure, this system can also be used to detect any possible threat to the user physical security, activate a warning message to alert the user of the dangers ahead and provide suggestions to avoid them (specially, if an immediate action is required).

Finally, another security measure that should be implemented is one that restricts the concurrent access to specific nodes to guarantee that no deadlocks ever occur. In certain situations, it might even be a viable option to create a temporal copy of these nodes and redirect the associated connections.

Secure Communications

Formally speaking, Communications Security is a field that focuses its attention squarely on *preventing the unauthorized access to telecommunications*. However, securing the communications within a computer network is a much more complex endeavor (even more so when it features the aforementioned hybrid architecture). Taking the Open Systems Interconnection (OSI) model [104] as the theoretical foundation of the network system, securing communications involves the protection of seven logical layers: physical medium, data link, network structure, transmission of data segments, communication sessions, data presentation and application-related services.

Fortunately, extensive research –and standardization– in telecommunication technologies already provides the means to construct secure P2P network structures [105, 106]. Therefore, the following security measures have been designed to facilitate the management of secure sessions and ensure the exchange of interaction elements.

Once the data link has been properly established between two network nodes, the first objective is to create a secure session (providing an encrypted channel and error recovery mechanisms). To achieve it, the first step is to send an initial message containing the *minimal* amount of information required to unequivocally identify both nodes within the network (i.e., NAT or IP address), the P2P routing algorithm and a list of the supported encryption systems. If the receiver node accepts the communication through the proposed address system, it must send an acceptance message (indicating the selected communication parameters, if there are more than one). Otherwise, the receiver node must respond with another initial message, proposing a different address, routing and/or encryption systems. Unless there is a security policy that explicitly denies it (for instance, to minimize the effects of a possible Denial of Service attack), this negotiation must continue until both parts agree on a common communication scheme.

With a –more– secure channel to exchange information, the next step is for both parties to authenticate themselves. Although this can be solved through mutual authentication systems [107] (using the data contained within the *identity* nodes for this purpose), it is strongly recommended to also employ third-party authentication mechanisms [108] to prevent Man-in-the-middle attacks. An approach that implies the creation of a message protocol to obtain identity certificates from dedicated network nodes (*Certificate Authorities* or CA) or to exchange them with other peers whose identities have already been verified (a scheme often known as *Web of Trust* [109]).

After both sides of the link have been authenticated, they can finally establish the session parameters to ensure a reliable communication. These include fields such as preferred data formats, compression mechanisms, maximum waiting time (before the connection needs to be rechecked), etc.

Once the session has been adequately created, the efforts to secure the communications must be aimed towards protecting the transmission of interaction elements. An endeavor that necessitates the construction of mechanisms to protect the access and the transfer of the object nodes associated to these contents.

The access to the different components of the internal object structure must be protected with a permission system that can restrict the reading and/or modification of specific object nodes. Usually, the definition of these restrictions might only require the access rights for individual users or user groups, but it might also be interesting to include the possibility to restrict the access to certain object nodes based on the connection parameters (e.g., same origin, IP address range, current traffic load, etc.). And while this can be integrated in the aforementioned contextual condition system, it might be preferable to use a separate object subnode to store the definition of these restrictions (so that it does not interfere with the aforementioned discovery mechanisms).

Meanwhile, to adequately transfer the interaction elements (especially in collaborative edition environments), both computer nodes have to apply serialization and deserialization mechanisms that not only support the partial redefinition of an existing object node but also do it in a secure way. Although the serialized data transfer can be efficiently resolved through the use of *operational transformation* mechanisms [110, 111] (to ensure the data conservation and the time synchronization), it is also necessary to establish a subsystem to validate the received data (before leaving it to the aforementioned structural security system to integrate it correctly into the object hierarchy). A relatively costly *arbitration process* that, nevertheless, can be delegated to a mutually trusted third-party.

Lastly, to properly finish the communication, both network nodes must agree on ending the session (through the standard exchange of BYE messages [112]). Because maintaining a session requires dedicating processing capabilities, if a specific amount of time (defined by the *maximum waiting time* session parameter) passes without receiving a message from the other part, a network node must send a connection check message. If no response is received soon thereafter, the communication must be considered *lost* and the computer must then release the computing resources associated to this session. While this approach may seem too strict, it prevents any attack based on the exploitation of session recovery.

Privacy Protection

Despite not being often addressed explicitly in the definition of information security systems, *guaranteeing the privacy of the user data must always be considered a primary goal for any implementation of the proposed interaction model*. Due to the large amount of personal data that is required for this system to offer meaningful user experiences, multiple mechanisms must be set in place to avoid leaking this information –whether intentionally or not– to third parties. Failing to do so might not only result in a more ineffective communication (i.e., direct marketing SPAM), but also might open innumerable attack vectors threatening the very wellbeing of the users (from mere scams to direct threats to their physical security). Moreover, the negative consequences of these oversights might extend far beyond the individual users, affecting the social and commercial groups they belong to (e.g., endangering corporate reputations, easing industrial espionage or, even, altering the outcome of general elections [113]).

Conscious of the importance of this issue, the very theoretical foundations of the model have been designed to prevent the transmission of sensitive information. Thanks to compartmentalization of the user data in specialized nodes, any implementation of the proposed model can establish different access policies to manage the nodes containing the *identity*, the *persona* and the *preferences* of each individual user. And, although it is highly recommended to firmly restrict the access to these three main nodes (specially, to the description of the psychological state of the user, contained within *persona* nodes), it is always possible for the end users to establish exceptions for extreme situations (e.g., medical emergencies, kidnappings, natural disasters, etc.). Furthermore, the model allows the definition of multiple instances of these nodes (and subnodes), enabling the creation of –partial– personal profiles for different contexts and, thus, greatly reducing the amount of user data that needs to be shared between computer systems.

Another major advantage of this approach is the fact that, because the computer system already "knows" the sensitive data that should not be transmitted, it can prevent the user from exposing it unknowingly. By reviewing all communications with the exterior, the system can warn the users before they share an interaction element that can potentially compromise their privacy. Although the actual responsibility for privacy management ultimately lies with the end users, the system has to provide them with the necessary tools to explain the possible repercussions of sharing a particular element before starting the information exchange and to visualize the personal data that other parties have access to during the communication. More advanced mechanisms can also keep track of the personal information shared with different computer systems and attempt to limit its propagation beyond the intended receivers (by negotiating a trust relationship with the other part and/or by explicitly stating the legal framework that applies to the shared data).

To further protect the privacy of the users, any implementation of the proposed interaction model must include a subsystem dedicated to globally evaluate the communications with other network nodes to warn the user about any potential leaks of personal information through the abuse of the contextual condition system. While the use of situational data is necessary to determine the suitable widgets for the present situation, it is important to monitor the messages sent by the discovery mechanism to avoid disclosing sensitive data (directly or by omission). Depending on the size of the transfer and the network conditions, it might be preferable to request and download a higher number of interaction elements and perform the filtering internally rather than exposing personal information that might be used against the end user.

In more unreliable environments, this privacy protection measure can be complemented with a mechanism that generates *shadow users*; fake –but convincing– replicas of the users, specially designed to mask their actual presence. At the cost of increasing the network traffic (and the processing power necessary to simulate the behavior of the replicas over time), this proactive approach seeks to frustrate any attempt to extract –reliable– personal information of the users without their consent. And while, at an individual scale, it might be impossible to conceal all aspects of the user context (because they can be extracted from external sources), the degree of uncertainty derived from the inclusion of such system might prevent the secretive extraction of private information for data mining purposes.

Finally, and although it might seem paradoxical, it is also important to consider the possibility of relying on external computer systems to provide a better privacy protection. For example, *proxy systems* can be used to conceal the physical location of the user (greatly reducing the possibility of a physical attack) while specialized network nodes can provide *anonymization* and *third-party authentication* services to facilitate the access to specific resources without exposing the users to identity theft schemes [114]. However, one novel concept that is having a significant impact in recent years is that of *privacy brokers* [115], network entities that, with the explicit acquiescence of the users, gathers their private information to not only protect it but also to provides anonymized statistics about it with Big Data companies in exchange for additional services.

3.3.4. Data Transmission

Culminating the description of the network model, this section provides the necessary components to construct a communication channel that can be used to successfully exchange interaction elements between the nodes of the network. Operating on the highest layers of the OSI model, the protocols that conform this communication channel facilitate the translation of the object instances into a data representation that can be shared with other computer systems (guaranteeing its correct delivery and posterior reconstruction).

Although, to preserve the platform independence of the model, this section does not provide an actual specification of the exchange language, it addresses all the key issues that can arise from its creation. Furthermore, it proposes several extension mechanisms to enhance the syntax –and semantics– of any resulting language, making it not only a way to transfer data between computers but for developers to create viable user experiences directly with it.

Another critical point that this section addresses is how to transfer the information of the interaction elements in a way that can be easily incorporated within the object hierarchy of the receiver. Contrary to other document-based approaches (where the user interfaces are kept separated), the inclusion of new items within the proposed model has to be properly specified to ensure their interoperability with existing object nodes.

Lastly, to regulate the transfer of information between two computers, a communication protocol must be defined. A simple but robust set of messages that, in conjunction with previously described security measures, enable the transfer of the associated data while also providing synchronization and error recovery mechanisms.

As a whole, these subsystems constitute the basis of a framework that facilitates the creation of open and interoperable user experiences. A common language that enables the exchange of interaction elements independently of the type of computer system or development platform employed.

Serialization Formats

To communicate interaction elements through a computer network, the emitter must first convert (or *serialize*) the associated object nodes into a stream of bytes in a way that the receiver can properly interpret (or *deserialize*). This double translation process, generally known simply as *serialization*, involves both the transformation of the memory representation of an object to a data format suitable for transmission and the subsequent recreation of the original object instance. It is important to note, however, that this procedure does not require the destruction of the original memory representation (i.e., inherently, it is not a *transference* but a *duplication* process) and that, as long as both communication participants known the object definition, they only have to exchange the modified state variables for both of them to maintain a synchronized copy of the object instance.

After clarifying its theoretical principles, the next step is to establish the appropriate data format for the serialization process; a decision for which there is not always a single, optimal solution.

The most basic and effective way to serialize interaction elements is to simply extract the complete memory representation of the associated nodes in binary form. However, if the computer that receives the binary stream has a different hardware architecture or its implementation is not exactly the same that the one of the receiver, it will be necessary to perform an extremely complex conversion process (one that needs to account for different programming languages, source versions and encoding formats).

Another, more widely used approach is to employ a text-based representation for the exchange of interaction elements. While not as efficient as binary counterparts [116], the use of human readable (UTF-8 text-based) formats has become the primary medium to transmit documents over the networks due to its relative simplicity, extensibility capabilities and easy debugging. More importantly, the standardization of many of these formats has increased the *interoperability* of the contents, facilitating their massive adoption.

Although the proposed interaction model is –programming– language-agnostic, it is highly recommended to employ human-readable formats to exchange the object node definitions (and state variables) while restricting the use of binary formats for the transmission of the large data structures (particularly, multimedia content such 3D models, sounds, textures, or videos, that can be encoded/compressed using standard formats). In this way, developers can employ the serialization language not just for debugging purposes, but also as a viable *User Interface Markup Language* (UIML) with which generate new interaction elements, either directly or through external mechanisms. Moreover, by using the aforementioned extension mechanisms and default values for serialization allows the definition of complex object nodes with relatively small number of characters.

When selecting a human-readable format for data serialization, there are several factors to take into consideration:

- **Adoption rate:** While it might be tempting to create a new markup language from the ground up, there are already numerous data serialization formats that have been standardized and extensively tested. Furthermore, many development frameworks include libraries and tools (editors, parsers, validators, etc.) for the most popular formats.
- **Representational similarity:** By minimizing the syntactical and structural differences between the representations, the serialization process can be greatly optimized (both in terms of time and resources required)
- **Resulting size:** Although the serialization result might be encrypted, compressed and/or divided during the transference process, minimizing its original size is important to reduce the network traffic.
- **Reference support:** If the data format provides a reference system, it can be used to reconstruct secondary connections between –locally defined– object nodes without relying on external naming systems.
- **Extensible design:** By allowing the extension of its syntax, a data format can be used to create forward-compatible resources (thus, avoiding the need to update stored object representations with each iteration).
- **Internationalization features:** Apart from specifying their associated encoding formats and language codes, the data format should provide methods to aid in the localization of specific values (measurement units, dates, etc.) to different languages and cultures.

At the time of writing this document, the best option is XML (eXtensible Markup Language) because of its simple but powerful structure. Nevertheless, other data formats such as JSON (JavaScript Object Notation) and YAML (YAML Ain't Markup Language) are viable alternatives due to their representational similarity with popular programming languages (JavaScript and Python, respectivelly).

Dynamic Structure

In contrast with current AR browsers models, in whose user experiences are compartmentalized by the static nature of the definition documents [117], the proposed model presents a single hierarchal object node structure where the interaction models are dynamically integrated. This solution, in conjunction with the aforementioned contextual condition system, makes possible the *seamless transition and interoperation between user experiences.*

In this new approach, the concept of document still exists, but just as a mere container of serialized data to facilitate the transfer and storage of interaction elements. And while documents can include metadata to facilitate the deserialization and object node integration processes, ultimately, the decision lies firmly in the aforementioned *structural integrity* mechanism.

Nevertheless, this approach presents several issues that need to be addressed to ensure its correct functionality in the vast majority of use cases:

- **Inclusion points:** While the object node structure establishes a clear distinction between the main components of the interaction model, some of them (i.e., *location*, *entity*, *resource* and *widget* object nodes) can have child nodes of the same kind. Therefore, interaction elements that must be contained within others must explicitly indicate the point of the hierarchy where they should be included.

- **Update mechanisms:** In cases where specific object nodes might experience modifications so infrequently that it is not reasonable to maintain an open communication with the other party to obtain regular updates (e.g. news broadcast, discussion forums, etc.), the emitter should be able to identify the nodes with a value to indicate when the receiver can ask for its renewal (i.e., defining an *expiration date* or, preferably, a *update rate* value).

- **Partial definitions:** As explained in section 3.3.2, to optimize the exchange of large number interaction elements required by the discovery mechanism, the object node structure must be able to –temporally– store incomplete definitions.

- **Access redirection:** If a network node does not have the complete definition of the requested interaction elements (or security policies prevent their transmission), it can redirect the petitioner to other network node from with obtain the actual interaction elements.

Transfer Protocols

To adequately exchange interaction elements between network nodes, it is paramount to define a transfer protocol; a small but robust set of messages that facilitate the data transfer in the OSI application layer.

Over the years, many protocol specifications have been proposed to facilitate the exchange of structured information over large networks [118]. Unfortunately, they were designed with a clear focus on *web services* over –rigid– server-client architectures and, therefore, can not be directly applied to the interaction model proposed in this document. Nevertheless, many of their core principles are still valid in this new paradigm:

- **Stateless exchange:** Although, as aforementioned in section 3.3.3, it is necessary to establish a session to provide a secure communication between two network nodes in a P2P architecture, this does not mean that –at the application level– they have to define state variables to know the transfer state. While it is recommended to obtain statistics about the access to certain object nodes (to identify strange patterns that might signal a potential attack), no context values should be stored during the exchange of messages.

- **Self-descriptive messages:** Each message must contain all the information necessary for its correct processing at the end. Even if the definitions of the interaction models contained in the messages are incomplete (to minimize the network traffic during discovery process), the message must include all the necessary metadata to correctly interpret its contents.

- **Cacheable contents:** To ensure the access to interaction elements in large networks and to efficiently distribute the traffic load, the object nodes contained within the messages can be temporally stored in a particular node (although to access them, they have to be visible and identifiable via the aforementioned naming systems and DHT mechanisms).

These simple architectural rules, common in REST (Representational State Transfer) interfaces, enables the *spontaneous generation* of scalable and reliable networks. Furthermore, by reducing the complexity of the connector semantics it is possible to incorporate into the network all manner of computer systems (e.g. public devices, sharing hubs, anonymization proxies, etc.) that expand the possibilities of the model.

"Movement is proven by walking."
Diogenes of Sinope

4

Validation

One of the biggest challenges of this research was the creation of a experimental methodology to properly validate the proposed HCI model. While the hypothesis on which it is based states with great precision the primary goals that need to be verified to be considered *valid*, due to the great diversity of human factors involved in their evaluation (i.e., anthropometry, biomechanics, cognitive psychology, etc.), a more direct, down-to-earth approach was also necessary.

Following the advice contained in the quotation heading this chapter, the validation procedure was revised to include a usability study in which a significant number of persons –of diverse backgrounds– could test a partial implementation of the HCI model. Through specially designed use scenarios, the participants evaluate the different interaction techniques (Selection, Manipulation, Symbolic Input Navigation and System Control), providing the feedback data by means of usability metrics.

This chapter examines the entire validation process. From the conceptualization of the evaluation method to the analysis of the results, including the design of the usability study and the technical considerations that led to its final form.

4.1. Methodology

As discussed in the introductory chapter, the validation of the main hypothesis of this research requires the evaluation of the resulting interaction model. An endeavor that entails more than the mere verification of the supported interaction techniques; it implies the definition of a complete experimentation environment in which the entire model must be thoroughly tested. However, this intention reveals one of the biggest challenges that this research has had to face: the nonexistence of a similar paradigm to compare to.

Although, over the past decades, there has been no shortage of publications showcasing novel interaction techniques [119, 120], none of them have presented a comprehensive enough model to center the entire interaction with a computer system around it. During this time, the responsibility of integrating these techniques into a cohesive whole has fallen on the shoulders of the operating system developers. While understandable from an industrial perspective, this situation has forced many HCI researchers to perform actions akin to "reverse engineering" to merely comprehend the underlying paradigms and being able to construct over them [121].

Against this background, the original intention of creating a experimentation environment based on an existing paradigm was abandoned, opting instead to develop a completely new system that showcases the main innovations of the proposed HCI model (such as spatial positioning and situational awareness). In this way, it is possible to obtain a more exhaustive evaluation of the whole composition and, thus, draw more realistic conclusions.

Nevertheless, while this decision prevents making direct comparisons with previous HCI models, it does not negate the legitimacy of the analytic mechanisms that are commonly used in their evaluation. In fact, due to the encompassing nature of usability studies, the application of this methodology necessitates a broader approach. As will be shown if later sections of this chapter, the combination of different types of usability metrics and the definition of a standard experimental design offers an excellent way to put the proposed model to the test.

This section presents the different approaches that were taken into consideration during the definition of the evaluation methodology and, despite not all of them were employed in the research work covered in this document, it is important to take them into consideration for future venues of work.

4.1.1. Evaluation Methods

Since the publication of the famous paper "Heuristics evaluation of user interfaces" [122] in 1990 (and similar works [123, 124]), usability testing has come a long way. While heuristics and expert knowledge are as relevant as ever in the design phases of GUIs, nowadays, the evaluation of the usability does not merely contemplate the interaction with the computer, but the entire user experience. A rapid evolution that has reached the point where many computer systems are designed –and thoughtfully tested– to appeal to the users even before they open the shipment box. [125]

As a result of this conceptual amalgamation, many new aspects have to be taken into consideration when designing a usability study:

- **Location:** For formal studies, it is recommended to reserve laboratory space to allow the participants to perform their actions in a calm, controlled environment. Nevertheless, to obtain more realistic –albeit less precise– data, it is possible to perform the test remotely, in the real-life scenarios where the users are accustomed to operate with similar systems.
- **Equipment:** Usability Testing labs should be outfitted with multiple audiovisual hardware systems to properly record the experiments and extract behavioral and physiological metric data. As for the experiment platform itself, it is usually provided by the evaluation team (mainly, to exclude the possibility of hardware problems, making the resulting data more reliable). Nevertheless, in online tests –specially those that analyze the usability of web sites–, it is common to allow the participants to use their own computer platforms because it provides relevant information about their usage in different situations.
- **Participants:** The process of selecting/recruiting candidates for a usability study is, more often than not, a challenging endeavor. While there are many guidelines about the suitable sample size for each type of test, it is always difficult to define what constitutes a representative and diverse enough sample (or what the target demographic is at all).
- **Time:** During the planning phase, it is of great importance to define the maximum duration of a session (and whether breaks between them are required or not). This facilitates the creation of a simple, flexible schedule that the participants can easily adhere to.
- **Budget:** Last, but certainly not least, the amount of monetary resources invested in a study have a significant impact on the rest of factors. A bigger budget not only results in better equipment, but also allows hiring external experts to moderate the experiments (greatly reducing the time required to conduct the study) and providing the participants with a monetary compensation.

Depending on these factors and the global scope of the usability study, there are several types of test that can be carried out:

- **Expert Review:** Before performing any other kind of evaluation, it is advisable to let a small group of usability experts (no more than ten) scrutinize the entire experimental system. Beyond making sure that the different usability heuristics are adequately considered, their expertise can help to identify previously unseen deficiencies and provide feedback on how to correct them.

- **Lab Testing:** As previously mentioned, lab tests take place in a controlled environment outfitted with special equipment to measure the user reactions (although the presence of moderators is usually preferred, in order to adequately analyze and respond to them). This evaluation method provides reliable results with a relatively small sample size of participants (from ten to one hundred, depending on the amount of techniques to evaluate).

- **Remote Testing:** A common practice in current web and mobile development is the integration of metric collection systems within their products to evaluate their usability in real-life environments (both before and after the release of the product). While this provides a significant amount of statistical information, the irregularity in experimental conditions entails that a very large number –typically thousands– of samples are required to infer a valid conclusion.

- **Online Surveys:** Often integrated in remote testing platforms, online surveys are, nevertheless, an evaluation method that requires special attention. They are specially useful to capture relevant –subjective– information about the general user experience and small design details, making them an important tool in software systems (e.g. web sites) where fast iteration is usually required.

- **Focus Groups:** A traditional market research technique, focus groups are moderated discussions that involve five to ten participants. These debates can provide useful information about the perceived usability of a system (even without actually using it).

- **Contextual Interviews:** In these meetings, the mediator observes the users while they operate with the experimental platform in a real-life environment. While contextual interviews are more informal and do not follow a strict script, the results tend to provide more realistic data.

Needless to say, these approaches are not exclusive. They can be repeated with different participants to evaluate multiple versions or combined to further solidify a previous conclusion.

4.1.2. Data Collection

Independently of the chosen method, the evaluation requires the acquisition of data values in order to draw solid conclusions. However, not all the data that can be obtained from a usability study has the same statistical relevance. In many instances, the variables maintain a strong correlation and, therefore, do not have any meaningful significance.

On account of this fact, one of the most critical tasks during the design of the usability study is to identify the variables that have special importance and establish *usability metrics* from them. With a well-defined range of possible values, these measures enable the evaluators to infer which parts of a system work properly and which ones need improvements.

However, the collection of usability metrics should not be regarded as a mere code dump but rather as a complete subsystem (preferably, integrated with the debug mechanisms to provide an easy access via console). Furthermore, it is highly recommended to include dedicated widgets to obtain and manage additional measurements (specially, self-reported metrics).

One disadvantage of metric-driven design is the fact that they only reflect past results. While this pronouncement may seem obvious, it is of critical importance to be aware of it before defining the tasks that the user has to complete in the different scenarios of the usability study, because it greatly limits the conclusions that can be obtained afterwards.

In view of this circumstance, one possible solution is to store all the available data into the results database and filter it afterwards. However, while this is feasible (and, in fact, the experimental platform does it as a form of fail-safe mechanism), the seer amount of irrelevant data makes it an impractical procedure. Instead, it is a better solution to plan the metric collection before performing the usability study, obtaining much more detailed data and, thus, more solid conclusions. Additionally, thanks to the secure communication mechanisms embedded into the proposed interaction model (the *share* node), this process can be easily automated.

Finally, another important aspect that must be taken into serious consideration during the transmission and storage of the metric data is the privacy policy to which this information is subject to. Aside from guaranteeing its confidentiality, the system must provide mechanisms to comply with international regulations. This goal that can be achieved either through data anonymization procedures (that make sure that the stored information do not contain any item that can identify the associated individual) or by applying cryptographic techniques (that encode the data in a way that only authorized users can access it). However, due to the complex requirements that the legislation introduces when dealing with encryption mechanisms [126], generally, it is a much better option to simply remove the fields that can identify the user to anonymize the data.

Performance-based Metrics

Directly embedded into the code, performance metrics provide valuable information about the execution of an interaction scenario. Not only they provide a way to quantify the effect of each interaction technique but they are also useful to estimate the magnitude of a specific usability issue.[127].

For each task, there are several metrics to take into account:

- **Success ratio:** Stored either with binary (true/false) values or through an enumeration with multiple levels (based on the number of completed subtasks, the degree of assistance required or the strategy employed), this metric measures the percentage of users that successfully complete a given task.
- **Time-on-Task:** Expressed as the number of seconds required by the user to complete a given task, this metric offers a relatively simple way to measure the associated *efficiency* level (without taking into consideration the required mental and physical effort).
- **Execution errors:** Whether due to the user's mistakes or a special condition that the code can not address properly, an error may surface. In these cases, the system must signal it through a specific metric and provide a way to recover to a previous situation.

Due to their objective nature, performance-based metrics are a powerful tool to evaluate different interaction techniques. This is specially useful when defining an interaction scenario because they allow the comparison of different techniques for each task and, even, offer a clear path to further adapt the selected technique to the context at hand.

Furthermore, this kind of metrics also enables the evaluation of the intuitiveness and *learnability* of each technique over time. By including the same performance-based metrics in consecutive use cases (or, better yet, multiple usability studies with the same participants), it is possible to separate the intrinsic problems of the interaction technique from those that appear during the learning process.

Issues-based Metrics

Unlike their performance-based counterparts, usability errors cannot be easily identified because they only surface while exploring each interaction scenario. For in-person lab tests, their detection can be partially delegated to the evaluator (who should pay close attention to the users' verbal and non-verbal communication), but for online tests, it is imperative to define a subsystem that, despite well-defined metrics, identifies a usability issue, annotates the associated circumstances and provides the participants with the means to recover from that situation.

The different usability issues can be divided into the following categories:

- **Navigation issues:** Indicate when the user does not know what options are within his reach (both physically and on a cognitive plane). These issues are relatively common when dealing with complex interaction scenarios and, while it may be impossible to eliminate them completely, their effect can be minimized through mechanisms that guide the user in a non-intrusive way (e.g., highlighting the available options).

- **Terminology issues:** Occur whenever the user does not understand a specific term in the instructional texts. In a lab test, he must communicate the issue to the evaluator, who will annotate it and provide a simpler explanation.

- **Content Issues:** Similar to the previous category, these issues originate when the user does not understand the significance of a specific resource of the user interface (e.g., an icon, a three-dimensional model, a sound, etc..). Again, in the context of a usability study, the only solution involves asking the evaluator about it.

- **Functionality Issues:** They appear as a result of an incomplete understanding of the task at hand. While these issues may be interpreted as an execution error (and, in fact, they are a direct consequence of them), it is important to recognize them as soon as possible, so that they can be corrected before the situation becomes hard to recover from.

To correctly evaluate –and later fix– the different usability issues, the associated metrics must also reflect their severity and the frequency with which they appear (an action that also implies the creation of a rating system for each metric in order to adequately present them). This additional information is of critical importance because it enables the development team to establish a comprehensive planning and focus their efforts on the most relevant issues.

Self-Reported Metrics

To make a proper assessment of the usability of a system, it is inescapable to take into consideration the actual opinion of its users. While subjective, their perception of the entire experience can point out deficiencies in the system that can not be detected in any other way. Moreover, users with experience in similar research areas can suggest workflow improvements and future work lines.

Nevertheless, due to the extreme complexity of human language, it is necessary to establish a way to translate these perceptions into data structures that can be adequately evaluated. A process that relies on the application of different rating scales to obtain numerical values, called self-reported (or subjective) metrics.

There are many different types of rating scales but the most common are:

- **Likert Scales:** Presents an statement to which the responders rate their level of agreement specifying a single choice from a multiple-point scale. Originally [40], this scale employed a 1-7 range to express the degree of agreement (from *Strongly disagree* to *Strongly agree*), nowadays, however, the 5-point version is more commonly employed in the literature.

- **Semantic Differential Scales:** Similar to a Likert Scale, these rating systems display two opposite concepts (usually, adjectives) at either end of a 5-point or 7-point scale. The users are prompted to select the point that more closely matches their opinion.

- **Numeric Interval Scales:** Allow the user to directly associate a concept with a numerical value. The different points of the scale can be displayed as discrete items, over a continuous line or as a single-choice list. Whichever the case, the resulting value must be convertible to a real number so that an arithmetic mean value can be extracted once all the metrics are collected.

While self-reported metrics can be obtained during the use case execution, to avoid affecting the results, its is a common practice to wait until all tasks are completed before asking the user to rate his experience. These post-task rating system are kept relatively brief, usually requesting only a basic *ease of use* value for each task (although there are more sophisticated methods such as After Scenario Questionnaire [128] and the Expectation Measure form [129] that take into consideration additional factors). It is at the end of the whole session when participants are asked to fill a more detailed questionnaire (generally a System Usability Scale form [130], but the Computer System Usability Questionnaire [131] or the Questionnaire for User Interface Satisfaction [132] can also be employed).

Behavioral and Physiological Metrics

Although they are very difficult to measure without dedicated equipment, the behavior and physiologic reactions of the participants during a usability test communicate important information about each interaction technique. For example, certain facial expressions can reveal when a user is having problems with a particular use case, motivating the initialization of the appropriate assistance measures (whether it just implies highlighting a particular component of the user interface or, in a test lab, the response of the evaluator).

Among the different metrics that can be applied to evaluate the behavior and physiology of the user, the most common are:

- **Verbal Communication:** During a lab test, the participants are asked to follow the Think-Aloud protocol, expressing their opinions verbally –as long as it doesn't affect their performance–. Once the test is over, it is also recommendable to record their general opinion (using a Positive / Neutral / Negative scale) so that it can be used afterwards to compare between multiple designs.

- **Facial Expressions:** By analyzing the face of the users with a camera pointed at them, it is possible to determine their emotional state. Although it is recommended to analyze the image feed in real time (to immediately correct any problems that may be detected), it is also possible to record it and analyze it at a later stage.

- **Eye-Tracking:** With a sophisticated video-based system (usually equipped with infrared light sources and sensors) it is possible to constantly track the point of view and the pupillary response of the user. This information is remarkably useful to map the cognitive processes employed while navigating through a user interface.

- **Body Language:** Certain body movements (e.g., fidgeting, rubbing the head, etc.) can tip off the evaluator that the participant is having problems understanding the task at hand. More excessive movements can signal both joy or frustration.

- **Biological Reactions:** Generally referred to as "biometrics", the measurements of physiological characteristics of the human body (such as heart rate or skin conductance) can offer additional information about the context of the user (e.g., physiological stress, health issues, etc.). Nevertheless, the size and weight of the hardware required to constantly monitor the users vital signs usually make its measurement a difficult task.

4.1.3. Data Analysis

After all the metrics are collected, a comprehensive analysis is required to extract meaningful information from the raw data. Beyond directly operating with the obtained values, this process aims to identify logical connections and discrepancies between the different variables to validate –or disprove– the assertions that support the entire study.

Figure 4.1 shows an overview of this process. Starting with the reception of the raw data from the experimental platform and its conversion to a tabular form. Later, this data is analyzed with several statistical procedures to obtain relevant information it and, finally, represent the results in a graphical format.

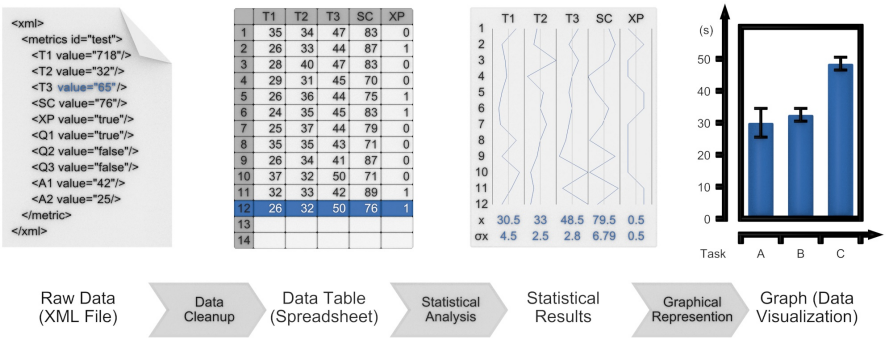

Figure 4.1: An overview of the Data Analysis process.

Data Cleanup

Even after a careful and well-planed data collection, it is a common occurrence that many metrics do not provide enough substantial data to support a solid conclusion. Moreover, in many usability studies, it is a good operating practice to include additional metrics –or to split existing ones– to identify and isolate incongruent results. Consequently, it is advisable to perform basic cleanup operations on the raw metrics data before analyzing it.

This data cleanup process may require *filtering* the values to ensure that external circumstances (i.e., biological problems and environmental conditions) have as little impact as possible in the results, *merging* several metrics into a new variable to increase their usefulness, *checking* the values for consistency (maintaining conditional relationships between linked metrics) and/or *converting* the resulting data into a usable –tabular– format. Nevertheless, it goes without saying that this process has to maintain the data integrity at all times, ensuring that any alteration of the values is not subjected to any kind of bias.

Statistical Analysis

Once all data is in an operable format, the next step is to decide the right statistical procedure to analyze each metric. For nominal and ordinal data, basic procedures such as frequency, crosstab and chi-square analysis [133] are enough in most cases. For interval data (i.e., Likert scale data or System Usability Scale scores), descriptive statistics and correlation analysis offer new insight into the data –and the relationships between the different variables.

Statistical analysis aims to obtain relevant information from the relatively small sample size obtained in the different tests. This goal can be achieved through the measurement of central tendency (the mean and the median) and variability (the range, the variance and the standard deviation) values, which result in a normal distribution that can be applied to a larger population with a certain level of certainty. A confidence interval that can also be calculated once a significance level –generally, 5 percent– has been established.

Once all the individual variables have been analyzed independently, it is possible to study the resulting attributes –specifically, the means– to compare them and extract additional meanings that might be easily distinguishable otherwise. This analysis can be performed using t-test over independent or paired samples; assuming equal variances in the former and hypothesizing a mean difference of 0 in the latter (because the initial assumption should be that there is no difference between the means). For three or more samples, advanced variance analysis techniques (commonly referred to as AMOVAs) can also be applied.

Finally, after all other analysis have been completed, the resulting information can be used to perform correlation analysis that uncover previously unseen relationships between variables (or validate the already established ones). One common way to fulfill this task is by creating a scatterplot graph with two variables and drawing a trend line to visualize the correlation between them. However, it is recommended to also use a mathematical function to calculate the exact correlation coefficient that expresses, within a range of -1 to +1, the fortitude of the relationship (the closer the value is to the extremes of the range, the stronger the correlation).

Fortunately, while all these procedures require complex calculations, there are data analysis packages –many included in popular productivity software solutions– that can shoulder most of the computational burden and, even, facilitate the visual discovery of relationship between variables.

Graphical Representation

Expressed in tabular form, metric data can be very difficult to interpret correctly (more so when the size of the table grows so large that even identifying the row and column associated to a cell becomes cumbersome) and, while many spreadsheet applications include diverse mechanisms to highlight specific variables, drawing verifiable conclusions directly from numerical values is a daunting task. Instead, it is advisable to represent these data in graph form, easing the visualization and comparison of multiple data series and, consequently, greatly facilitating the communication.

Nevertheless, this conversion process is not devoid of pitfalls. While there are a large number of excellent books on the design of effective data graphs [134, 135], the sheer amount of variations and factors that have to be considered, more often than not, result in poorly constructed graphs (or, worse yet, may inadvertently lead to data manipulation). Among the many advices that are usually repeated in the literature, the more important are:

- **Choosing the appropriate type of graph:** While bar charts and line charts may appear to serve the same function, the former are better to express discrete data points while the latter allow a better visual comparison between data sets. Furthermore, pie charts are an excellent way to represent percentage values (as long as all sum 100%).
- **Labeling the axes as units:** Although they may seem obvious, when presenting numeric values on the different axes, it is essential to include the name and the type of unit (percentage, measurement units, etc.) associated with them. These additional texts can be attached to each value, but most of the times it is enough to display on a separate label, oriented along the specific axis.
- **Showing confidence intervals:** When presenting means of participant data (times, ratings, etc.) in line and bar graphs, it is recommended to also show the 90 percent confidence intervals as error bars.
- **Avoiding data misrepresentation:** While adjusting scale of the axes to "zoom" onto a specific part of the data may make sense in time series with relatively small variations between values (case in point, stock market graphs), whenever possible, a regular scale starting at zero should be employed. Additionally, the graph designer has to be fully aware of how colors, scales and perspective may affect the perception between different values before applying these factors.
- **Presenting the right amount of information:** In some instances, integrating multiple data points into a single graph can provide additional insight into the relationships between variables. However, it is usually recommended to strive for visual consistency and clarity rather than include all possible relationships and risk overloading the graph.

4.2. Usability Study Planning

Devising a usability study is always a tall order; there are numerous –human– factors to take into consideration and high-level questions that need to be answered (i.e., the goals of the study itself and those of the user). Furthermore, due to the novel features of the proposed HCI model, there are no previous works that can be used as a valid reference for the present study (nor middleware tools that can be directly applied to it). For this reason, from its inception, the design of the study was primarily oriented towards the evaluation of the model as a whole, instead of focusing on the different interaction techniques that can be created with it to solve a particular task.

The implications of this decision have a significant impact on the definition of the use cases (or interaction scenarios) in which the usability study is divided into. A situation that can be clearly seen in figure 4.2, where the Use Case 0 encompasses the entire study, providing the logic behaviors to sustain a consistent user experience throughout the execution of the different interaction scenarios and the acquisition of self-reported metrics. As illustrated in the same figure (at the bottom right corner), it also provides special widgets that the evaluator can use to manage the execution of the experimental platform, indicate any additional *guidance points* that the participant may require during the experiment and, finally, submit the information to perform the data analysis detailed in the previous section of this chapter.

Each use case is further divided into three main sections, or screens:

1. **Instructions:** An instructional text (which, in use cases 1-3, is accompanied by a explanatory illustration) provides the participants with a general orientation to the tasks they have to perform next, including any restrictions such as order of operations, time required or invalid actions.

2. **Task Execution:** For each interaction scenario, the participants are presented with a virtual environment (integrated within the physical reality, in Use Cases 1-3) populated with a series of widgets specially designed to evaluate the specific tasks associated with each use case. Is in these screens where most of the performance-based, issue-based and behavioral metrics are acquired.

3. **Metrics Reporting:** At the end of each use case, an additional screen with several Likert Scale widgets enable the participants to input their subjective usability evaluation for each task.

To properly present the study, additional screens (title screen, EULA, SUS and Tech Experience Forms, etc.) are included within the Use Case 0.

This relatively simplistic design nevertheless offers an excellent overview of the possibilities of the proposed HCI model, while also maintaining a good balance between data acquisition, user satisfaction and time per session.

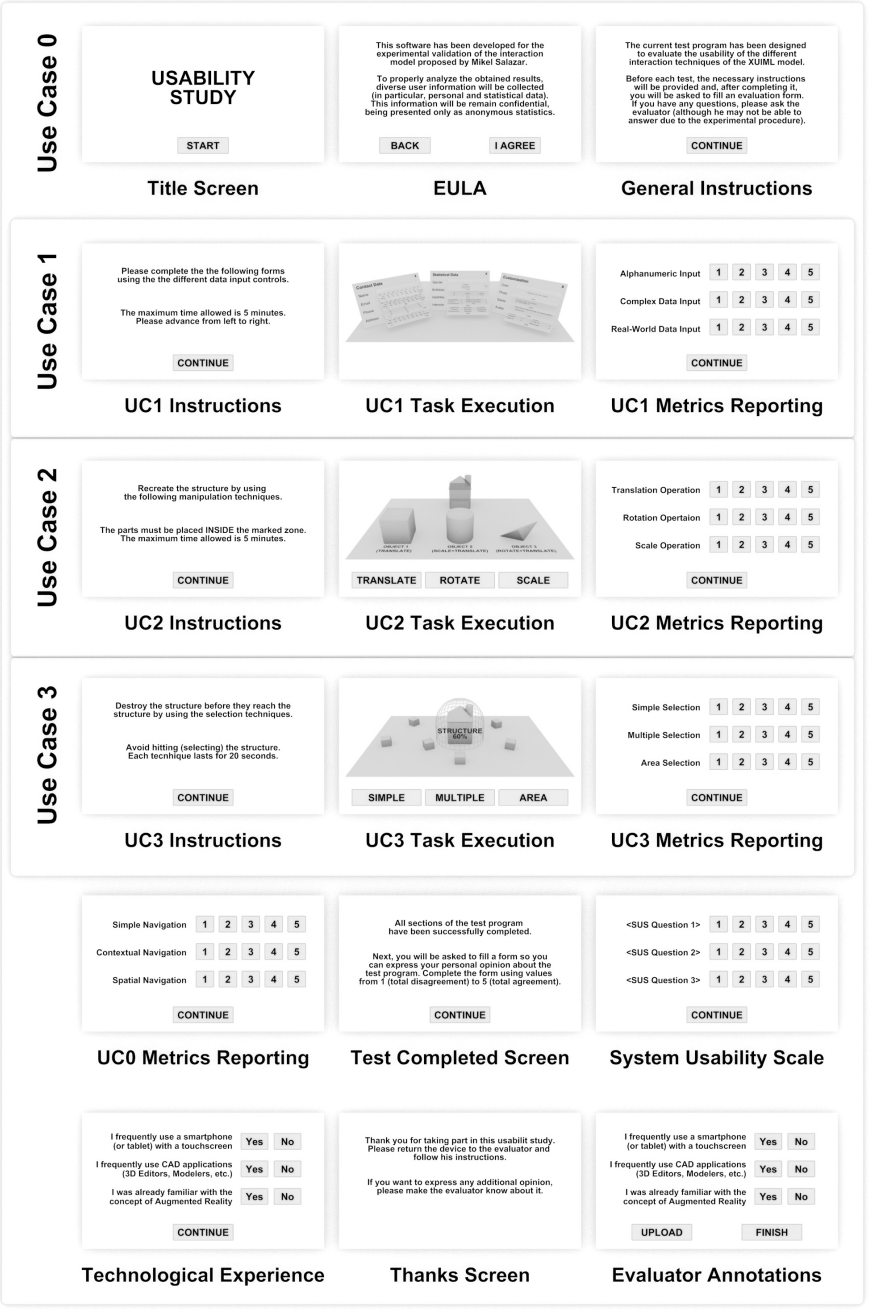

Figure 4.2: A storyboard of the usability study, showcasing the different screens in which the the different user cases are divided.

4.2.1. Use Case 0: Navigation and System Control

To successfully conduct the usability study, it is essential to provide the participants with a logical environment that enables them to perform the different tasks in the proper sequence, displaying the relevant information at all times and allowing them to control the system state to certain extent (although always under the supervision of the evaluator). To achieve this goal, the Use Case 0 presents a series of widgets that facilitate the visualization and manipulation of different system variables, including those that handle the navigation during the other interaction scenarios and the introduction of self-reported metrics. In a sense, this use case takes the place of the operating system, managing the execution of the different "applications".

In contrast to other scenarios, this use case does not attempt to showcase any complex interaction technique, but to provide a consistent construct over which the entire usability study can be carried out as easily as possible. To that end, as can be seen in 4.3 the different widgets of this use case are presented in a non-spatial environment (what it is commonly referred to as a HUD, or *Head-Up Display*), completely detached from the physical world. Additionally, while the widgets maintain their three dimensional definition, they are positioned on a plane orthogonal to the view of the user in an attempt to replicate the bidimensional GUI layouts common on current applications [136] and, thus, minimize the adaptation effort required to operate them correctly.

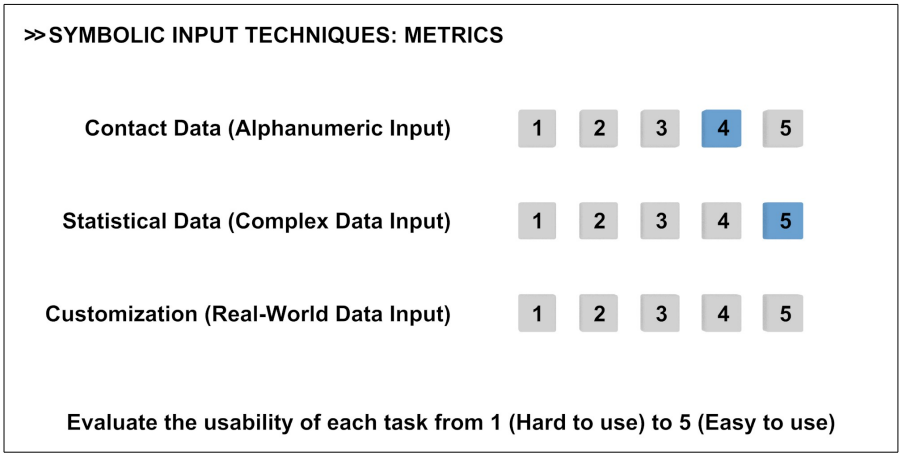

Figure 4.3: Different non-spatial widgets employed in the Use Case 0. Their layout was designed to facilitate the access to the relevant information while also supporting occlusions generated by the hands of the users. The widgets situated in top row display the current system status, while those in the center of the screen contain important instructional texts, allow the introduction of self-reported metrics or facilitate the navigation between the different screens and operation modes.

Tasks

- **Simple Navigation:** To ensure that the participants have enough time to understand the instructional texts, each one is accompanied by a "Next" button. Upon activation, these widgets modify the nodes that manage the screen sequence, indirectly updating the visibility state of the different components of the user interface.
- **Contextual Navigation:** Similarly to the aforementioned widgets, the "Finish" buttons allow users to advance to the next section of the usability study. However, these buttons are tied to contextual conditions, only appearing when all the tasks associated with each use case (including the input of self-reported metrics) are successfully completed.
- **Spatial Navigation:** The integration of fiducial markers within the physical environments allows the precise tracking of the user in real time. This, in turn, offers the possibility of defining behaviors that take into consideration the relative position of the user to show –or hide– different components of the user interface.
- **System Control:** The user interface must include logical mechanisms to allow the participants to recover from erroneous situations and control the execution of the usability study (although, in lab tests, the access to this functionality should be restricted to the evaluators).

Metrics

- **(Total) Completion Time:** The amount of time (in seconds) required to complete the entire usability test.
- **Self-reported Metrics:** Through Likert scale-based widgets, the participants can provide subjective evaluations of the different interaction techniques in a consistent and measurable manner.
- **Evaluator Annotations:** If the usability study is conducted as a lab test, upon completion, the evaluator must access a special screen to indicate if any additional explanation (i.e., guidance point) was required.

Guidance Points

- **Careful Reading Request:** If the participant skips any instructional text screen (whether intentionally or not), the evaluator has to revert to the previous state and ask the participant to read the text carefully.
- **Self-reported Metrics Explanation:** If the user does not know how to input self-reported metric values, the evaluator can point to the explanatory text on screen and provide further clarifications.
- **Privacy Concerns:** If the participant expresses concerns about introducing personal data, the evaluator must remind him about the EULA.

4.2.2. Use Case 1: Symbolic Input

This interaction scenario allows the participants to explore the wide variety of symbolic input mechanisms that the proposed interaction model -literally- puts at their fingertips. Through the use of spatially positioned "soft" keyboards, the participants must introduce their own personal data and preferences as symbolic values. In this way, the participants can perform the tasks with more certainty (because they know the values to input beforehand), while also obtaining the data required to populate the User node (including the Identity, Persona and Preferences submodes), which, in turn, enables the customization of the general user interaction (in this and successive user cases).

To progressively introduce the usability study participants into the complexities of navigating a spatial user interface, the different symbolic input widgets are grouped into three panels (oriented in such a way that they can be easily perceived as individual entities, wordlessly instructing the user to physically navigate between them). As figure 4.4 shows, each panel contains a different type of input widget: alphanumeric keyboards for contact information, option selectors for statistical data and advanced controllers for customization. Furthermore, these widgets are constructed using the information contained within the User node, so changes introduced in one can propagate to others (e.g., the widget where the participant can customize the appearance of their avatar employs the photo and voice sample obtained in the previous widgets to update the model, while the laterality node value is taken into consideration to determinate what hand to raise during the *salute* animation).

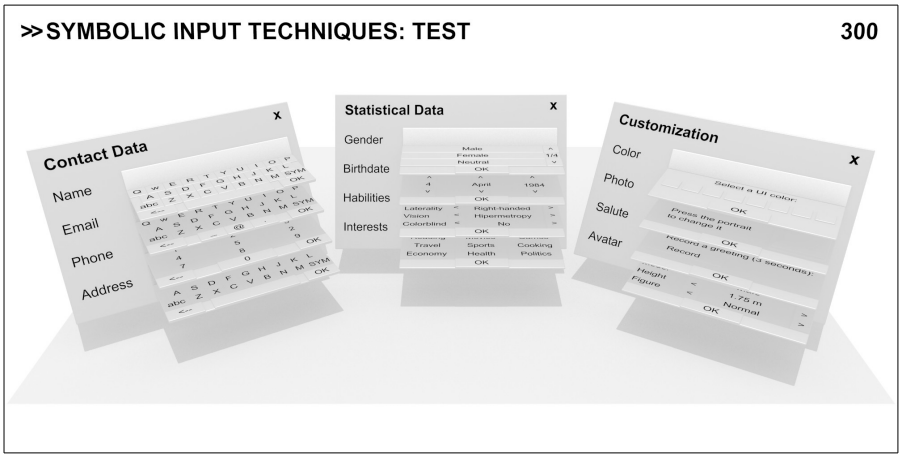

Figure 4.4: The different symbolic input widgets employed in the Use Case 1, presented here as an explosion diagram to facilitate their visualization. During the actual test, only one soft keyboard can be active (visible) at a time for each panel.

Tasks

- **Alphanumeric Data Input:** Using soft keyboards with layouts adapted to specific data type and format, the participants must input their contact information: name (alphabetic value with capitalization), email (alphanumeric value with special characters), phone (numeric value) and address (alphanumeric value).
- **Complex Data Input:** Confronted with large lists of items to choose from, the participant is asked to interact with the different widgets at their disposal to indicate his gender (single-option selector), birth date (date picker), abilities/disabilities (multiple-choice selectors) and personal interests (multiple-choice combobox).
- **Real-World Data Input:** Using the different sensors of the device (front-facing camera and microphone), the participant can capture real-world data to define the face and voice of his avatar. The color and properties of the avatar model are provided using special widgets.

Metrics

- **Completion Time:** The number of seconds required to complete all the tasks of the Use Case.
- **Interaction Errors:** Indicates the number of times the user had to correct a symbolic input operation (including the number of times the "back" key was pressed in the virtual keyboards).

Guidance

- **General Operation:** If the participant does not know where to start, the evaluator must remind him about the recommended sequence.
- **Time Remaining:** When half of the allotted time passes, if the participant has not completed at least two of the tasks, the evaluator must point to the time counter and explain its function.
- **Spatial Positioning:** If the participant has problems pressing keys, the evaluator must remind him about the spatial nature of the user interface.

4.2.3. Use Case 2: Manipulation

The goal in this Use Case is to provide the participants with an interaction space in which experiment with different manipulation techniques. To achieve this goal in a way that can be properly evaluated, the participants are asked to recreate a structure using three pieces, each requiring a different set of manipulation operations to fit within a delimitated area.

Although manipulation tasks can normally be divided into basic transformation operations(translation, rotation and/or scale), it is important to note that the set interaction techniques associated with each operation depends on the hardware available. As explained in the previous chapter, while direct manipulation mechanisms (using 3D mouses or hand recognition systems) allow users to modify the position and rotation of an object, more complex interactions require the creation of additional manipulation widgets (often called *manipulators*). Moreover, to facilitate their utilization, these widgets are usually presented in a different -overlaying- layer centered around the selected objects.

Due to the relative simplicity of this interaction scenario, the transformation operations can be performed through point-based manipulators. However, as shown in figure 4.5, it is recommended to include additional bounding box-based widgets to facilitate the alignment of the objects in the three-dimensional space (specially on platforms that are not equipped with stereoscopic displays).

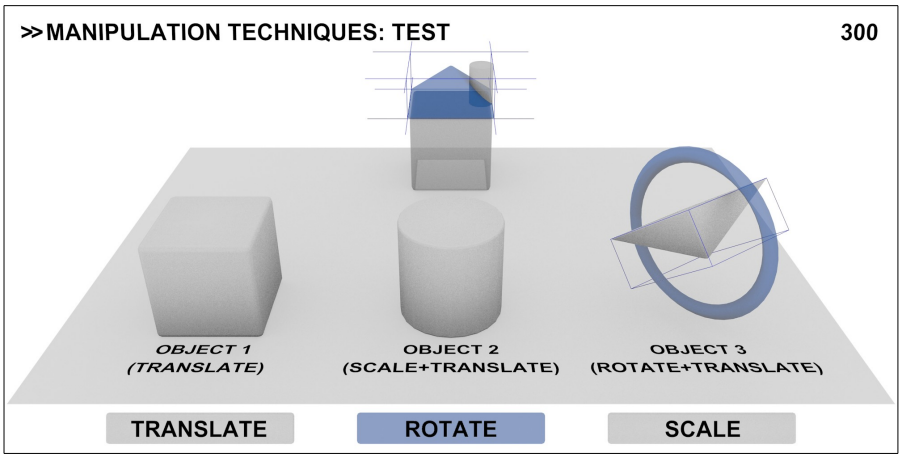

Figure 4.5: A representation of the execution of the Use Case 2, showcasing the different manipulation widgets necessary to perform the transformation operations. Point-based manipulators allow an easy transformation (one axis at a time), while a bounding box-based widgets allow an easier alignment of the pieces with their target position.

Tasks

- **Translation:** Required to position all three pieces of the structure within the outlined region, this task is performed through an axis-aligned manipulator. By dragging one of the six color-coded arrows that compose the manipulator, the study participant can move the selected piece in the associated direction (within a three-dimensional grid).
- **Rotation:** Composed of three color-coded rings, the rotation manipulator allows the participant to visualize and modify the orientation of the selected piece. To facilitate its usage, the manipulator generates a fan of selection regions that extend far beyond the radius of each ring.
- **Scale:** Similar to its translation counterpart, the scale manipulator presents six arrows (with a different termination) to interactively specify the size of the selected piece in each main direction. However, in this case, the manipulator is set to the local coordinate system, providing the user with a better understanding of the applied transformation.

Metrics

- **Completion Time:** The number of seconds invested in the manipulation of each individual piece (due to the different number/types of operations required to correctly place each one).
- **Interaction Errors:** Indicates the pieces that were not successfully integrated within the outlined region in the alloted time.

Guidance

- **General Operation:** When the participant expresses insecurity about the manipulation techniques, the evaluator must explain their operation (and the feedback associated with successfully integrating a piece).
- **Time Remaining:** When half of the allotted time passes, if the participant has not integrated two of the three pieces, the evaluator must point to the time counter and explain its function.
- **Spatial Positioning:** If the angle between the manipulation direction and the device orientation is too small, it can result in interaction errors. Whenever the evaluator detects this problem, he must warn the user about it and request him to modify his point of view, so that is perpendicular to the manipulation plane.

4.2.4. Use Case 3: Selection

In this final Use Case, the participants have to employ the different selection techniques at their disposal to "destroy" the targets before they reach the structure (the same that he/she built in the previous use case). To further enhance the user experience, the structure displays a percentage value that decreases when a target reaches it or when the participants erroneously perform the selection operation over it. Since the targets can appear behind the structure, the participants are forced to reposition themselves -and the computer platform- to appropriately select them while avoiding the structure.

While selection operations are usually considered an almost trivial issue in previous HCI models, when confronted with the unpredictable movement of the camera that AR systems allow, the complexity of the task grows exponentially. Without the possibility of relying on screen-space coordinates, multiple selection operations become a difficult problem that requires the analysis of the point of view of the user, the location of the pointer(s) and the specification of the *selectable* entities within the environment (and whose should not be taking into account; such as glass or other transparent surfaces).

Taking the appearance of a game to instill a sense of urgency on the participants, this use case presents three different selection techniques (disguised as "weapons") that the users have to employ to select the target objects before they reach the structure. These selection techniques, illustrated in figure 4.6, explore the use of ray-casting mechanisms to select multiple mobile objects as fast and accurately as possible.

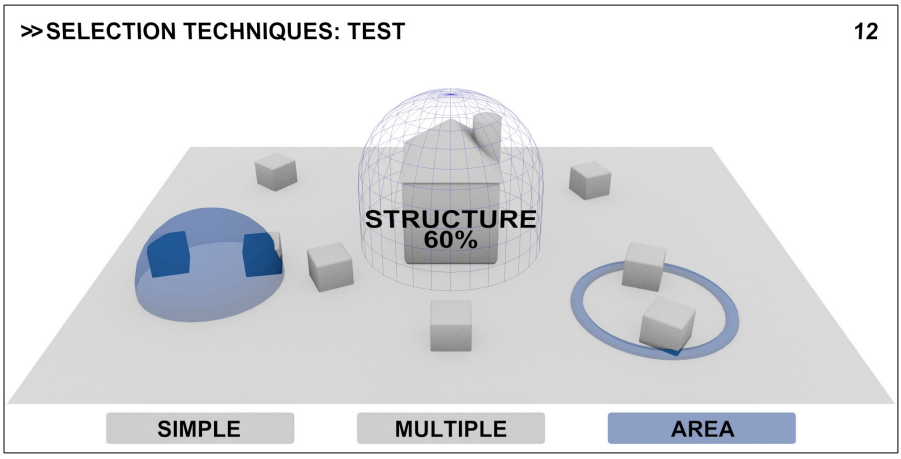

Figure 4.6: During the use Case 3, the participants can experiment with three different interaction techniques to select mobile objects in a three-dimensional space (while moving around to avoid hitting the structure built in the previous scenario).

Tasks

- **Simple Selection:** In this mode, the participant indicates the target to destroy by "clicking" on it (instantaneously, pressing and releasing on the object projection on screen). While simple, this technique only allows the selection of one target at a time, which, in conjunction with the occlusion of part of the screen produced by the hand of the user, makes it inadequate for large groups of targets.
- **Multiple Selection:** After being limited to one single target, the test participant is allowed to select multiple targets by dragging their finger over -their projections on- the screen. However, this ability is limited to a couple seconds each time and it makes more easy to unintentionally hit the structure.
- **Volume Selection:** Finally, the participant is allowed to employ circular selectors to define a bigger selection area. The radius of the selector is fixed but the center can be dragged around until the participant stops pressing the screen. At that moment, all targets contained within the area will be marked as selected (and, thus, destroyed).

Metrics

- **Completion Time:** The number of seconds that the structure maintains a integrity value greater than zero percent.
- **Interaction Errors:** The damage sustained by the structure, either due to collision with targets or as a result of invalid operations.

Guidance

- **General Operation:** If the participant does not know how to use a specific selection technique, the evaluator must explain its operation.
- **Invalid Operation:** The first time the participant hits the structure, he should be reminded that it is an invalid -and penalized- operation.
- **Spatial Positioning:** If the participant does not know how to select targets behind the structure, the evaluator must remind him that he can change his point of view.

4.3. Experimental Design

Once all sections of the usability study have been accurately specified, it is possible to construct the interaction space to adequately conduct the validation procedure. An undertaking that not only encompasses the determination of the physical environment where the experimentation takes place, but also the definition of the computer platform employed by the participants and the procedure followed by the evaluation team.

Nevertheless, hampered by the very novelty of the proposed HCI model and the lack of standardized AR technologies, this –arduous– endeavor required a complex preparation before coming to fruition. Since the public presentation of the theoretical model [137], the experimental design suffered several complete redesigns to allow both the implementation of the interaction techniques and the middleware platform for their proper validation.

During the nearly six months of iterative work (from April of 2014 to October of the same year) that were required to construct its final rendition, the experimental design evolved continuously to accommodate itself to different software and hardware configurations. In the attempt to find the best possible environment for the usability study, many research directions were contemplated (e.g., HMDs, eye-tracking techniques, etc.). While not all of them resulted in an applicable solution, the final experimental planning took advantage of them to achieved a satisfactory compromise between the adherence to the initial requirements and the creation of a consistent user experience.

To further validate this experimental design, a preliminary version of the experiments was presented during the September 2014 AR Community Meeting [138]. A situation that offered the opportunity to conduct several *focus groups* sessions with leading researchers in the field and industry experts, resulting in valuable information that helped shape the final version of the software.

Finally, the experimental design phase concluded with an *expert review* where five hand-picked participants (selected due to their knowledge in the field of User Experience) evaluated the entire system, trying to find the limits of the interaction space. The feedback obtained from this test, although not objective enough to be included in the final results, allowed a better definition –and expansion– of the contextual conditions and usability metrics employed in the final tests.

This section is dedicated to present the different aspects of the experimental planning in its final form, showcasing the different technical challenges that the research team had to overcome and the reasoning behind important design decisions.

4.3.1. Evaluation Procedure

To properly validate the proposed interaction model (and, by extension, the hypothesis on which this dissertation is based on), the usability study was conducted as a *lab test*. This evaluation method, explained in the first section of this chapter, contemplates a controlled environment in which the participants are presented with a series of tasks that they have to perform under constant conditions. As a result, the metrics obtained through this procedure (either related to performance data or self-reported by the participants) are much more reliable and can be used to draw solid conclusions.

Nevertheless, the quality of the results do not only depend on the metrics themselves, but also on the number and diversity of the participants involved. For this reason, the usability study is devised to comprise, at least, one hundred subjects of different backgrounds (including research personnel, undergraduate students and people not affiliated with the University of Deusto). This sample size ensures a high confidence level in the findings extracted from the data aggregation.

As for the conduction of the lab test, the responsibility ultimately lies with the figure of the evaluator (or moderator). An actor charged with providing the means for the participants to complete the different tasks of the usability study. This assignment (illustrated in figures 4.7, 4.8 and 4.9) consist on accompanying each individual through the entire session, ensuring that there are no problems that may undermine the evaluation process. To achieve this goal, before entrusting the participant with the experimental equipment, the evaluator has to ask the participant about any condition that may impact its proper handling (left-handedness, disabilities, previous injuries, etc.) and provide general instructions about the nature of the study and how to properly handle the device (to avoid occluding the camera of the device during the session).

Figure 4.7: At the beginning of each session, the evaluator must restart the experimental platform, explain the conditions of the test to the participants and instruct them in the proper handling of the device.

During the test execution (except for the moments in which the participant inputs self-reported metrics, that must remain confidential to ensure their validity), the evaluator must remain next to the participant, paying close attention both to the experimental equipment and the body of the user. In this way, the evaluator can easily identify any non-verbal cues that may signal a problem and, if necessary, anticipate a solution for it. However, to avoid interfering with the evaluation, these interactions must be kept to a minimum and always follow a pre-established protocol (a set of conditioned responses included in the printed form that the evaluator must fill out for each session).

While it is strongly recommended to limit the communication with the participant to the guidance points established for each use case, ultimately, the evaluator is able to provide additional assistance to compensate for any contingency that may surface (users with disabilities, hardware problems, etc.). In such instances, the evaluator must annotate the reason that motivated the assistance and its actual extent.

Throughout the test, the role of the participants is limited to follow the instructions provided on screen before each use case, trying to complete the different tasks in the alloted time. However, to properly evaluate each interaction technique and provide accurate self-reported metrics, the participants must be presented with situations that enable them to fully explore their functionality. A user goal that not only implies the correct implementation of the different techniques, but also requires the creation of a system that enable the users to recover from error states that may have not been contemplated during the design of the usability study. However, instead of relying on undo-redo commands or the *memento pattern* [139] to restore the entire system to a previous state, the logical behaviors associated to the widgets already include recovery mechanisms that allow the users to correct their own mistakes in an intuitive way (although not without updating the associated error metric, to ensure an adequate evaluation).

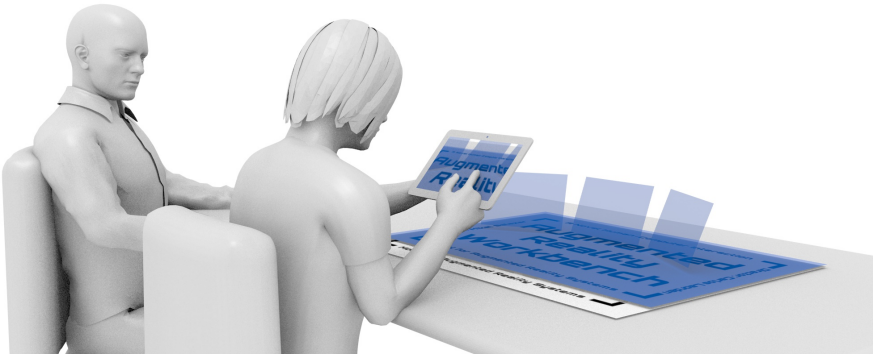

Figure 4.8: During the execution of the different use cases, the evaluator has to pay close attention to the behavior of the participant, respond to any verbal or non-verbal communication and provide guidance –only– when needed.

Once the participants complete the usability study (Use Cases 1-3), they are presented with a System Usability Scale form that also includes three binary questions about the users' own technological experience; whether or not they usually operate with a device with a tactile screen (such as tablets or smartphones), if they are used to work with three-dimensional editors and whether or not they knew in advance the concept of AR. This post-session form provides meaningful information that allows a better understanding of the subjective perception of each individual user, and thus, a more precise data analysis.

Originally, this form was designed to be printed out (on the other side of the sheet paper where the evaluator has previously written down his annotations), so that the data input was not affected by the limitations of the experimental platform. Yet, to ensure the anonymity of the information, this idea was discarded during the preliminary test, opting instead for integrating the form within the Use Case 0, taking advantage of the widgets already defined for the input of other self-reported metrics.

After the participant fills out the form, a message appears on screen thanking him for his collaboration and asking them to return the experimental equipment to the evaluator. At this point, the evaluator has to retrieve the device and offer the participant some sweets as a form of compensation for completing the usability study. Then, while the evaluator accesses the special screen to input into the system the annotations he has previously recorded on paper, he is compelled to initiate a light conversation with the participant, asking him about his general opinion of the experience. Although it may seem superfluous, this review process is important because it can reveal circumstances that the evaluator has not previously aware of (e.g., terminology issues, physical impediments, etc.). Needless to say, this information must also be included in the evaluator annotations and submitted to the server alongside with the metric data obtained during the session.

Figure 4.9: After the participant completes the Use Cases 1-3 and fills out the post-session form, the evaluator must retrieve the device, review the experience with the participant and upload the session data alongside with his own annotations.

4.3.2. Experimental Equipment

One of the major challenges this research project has had to face is the absence of a completely viable hardware platform with which to perform the experimentation. As explained in chapter 2 of this dissertation, the current literature has presented a myriad of different interaction techniques, but the actual hardware systems that are currently available have a limited scope and are not yet capable of presenting a satisfactory user experience (or, at least, one that can be sustained over long stretches of time).

Aware of these limitations, multiple interaction scenarios were contemplated in the early stages of design, including those that involved the creation of custom-made hardware platforms. Specifically, several weeks were dedicated to evaluate the possible implementation of a direct interaction scenario that allowed the user to better understand the capabilities of the proposed model (by incorporating stereoscopic view of the virtual elements and interaction techniques based on the natural motion of the user's body).

As shown in figure 4.10, during this subproject different hardware configurations were tested, using both commercially available products and custom-built alternatives. Nevertheless, the use of Head-Mounted Displays introduced several issues that hindered the general user experience. Beyond their known limitations in terms of resolution and positional tracking, current HMD technologies tend to induce a condition known as "motion sickness" [140]. While there are different techniques to palliate this effect [141] and the associated symptoms tend to dissipate over repeated usage, this circumstance greatly limited the interaction for first-time users (to no more than two minutes, in most cases). This, coupled with the inability to provide satisfactory –and consistent– tactile feedback, forced the decision to abandon this approach in favor of an indirect, see-though scenario.

Figure 4.10: Different prototypes constructed during the initial phase of experimental design. This subproject explored the possibilities of creating a successful AR computing platform by employing commercial hardware and custom-made headsets.

Extensively researched [2] and well established in the industry [142], this interaction scenario presents, however, important disadvantages that had to be overcome in order to create a acceptable hardware platform for the usability study. Mainly, the use of a multi-touch screen as the main input device requires the implementation of a subsystem that translates the presses on the surface into virtual entities that can physically interact with the components of the user interface. A complex procedure that demands more precision than many devices can actually provide and one that conflicts with common interaction techniques in these kind of devices (i.e. the multi-touch gestures that the users are accustomed to employ to move, rotate and scale objects cannot be directly applied to the three-dimensional environment where the spatial user interface exists).

Another issue related to the use of multi-touch devices is that their size and weight often hinders proper handling in AR scenarios. Because the point of view of the user is determined by the position of the back-facing camera, the device must be constantly hold in midair, which, even by letting the elbow rest on a planar surface, entails a significant physical burden. A precarious situation that is made even worse by the difficulty of getting a good grip of the device (without partially occluding the screen with the hand) and the unintentional displacements that occur whenever the user presses on the screen surface (specially, in the opposite side of where he is holding the device).

While not completely resolved, the final version of the experimental equipment includes additional components to alleviate these issues. Pictured in figure 4.11, the custom-made handle allows a better grip for both right and left-handed users, while allowing them to physically interact with the device from its center of gravity (which is directly connected to the extended palm). Although this solution slightly increases the weight that the users have to support, the increased mobility that this solution provides more than compensates for it.

Figure 4.11: The final version of the experimental equipment consisted on a Nexus 10 tablet with a custom-made handler. Please note that the back side of the handler included an adjustable Velcro buckle and a hole for the camera.

4.3.3. Physical Environment

Owing to the need to evaluate different physical navigation techniques, a spatial region was demarcated within the real world to properly conduct the usability study. This location encompasses the entire work area assigned to the DeustoTech Computing-S3lab group in the fourth floor of the Faculty of Engineering (University of Deusto, Bilbao, Spain). As can be seen in figure 4.12, the physical environment is divided into multiple rows of tables and is illuminated both by the fluorescent lights in the ceiling and by the sunlight that comews through the windows (a circumstance that turned out to be an important factor in the recognition of the AR marker).

Figure 4.12: The physical environment where the usability testing took place. The left image shows a real photograph of the location, while the right shows the simplified model used to define the environment.

In the initial stages of the experimental design, we contemplated the possibility of creating a use case exclusively focused on physical navigation to showcase the spatial nature of the user interfaces that can be defined with the proposed model. Nevertheless, the complexity of the navigation task itself, coupled with the limitations of current tracking technologies and the risks associated with performing the evaluation in a relatively crammed work environment (physical collisions and possible falls), forced us to restrict the interaction space to a single desk. Illustrated in figure 4.13, this configuration often referred to as an AR Workbench.

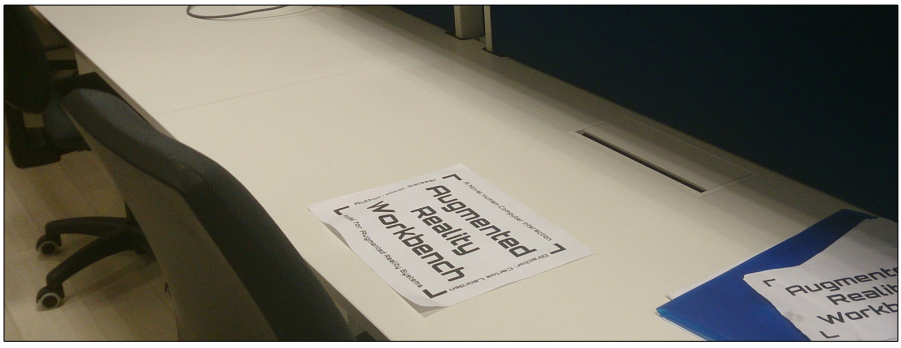

Figure 4.13: The desktop with the AR marker used in the experiments.

This constraint, nevertheless, turned into an useful feature, greatly simplifying the different interaction scenarios and offering the participants with a more comfortable experience overall (since they can perform the different tasks while seated and with their elbows resting on the table). Furthermore, by limiting the movement of the user to a specific location, the virtual objects can be arranged into multiple depth-based layers (which, in turn, allows the use of simple navigation terminology such as "left/right" and "in front/behind").

To facilitate the integration of virtual environments into the physical space, a special fiducial marker was devised. Pictured in figure 4.14, this bidimensional image was designed to feature a large number of high-contrast points, distributed in a pattern that the AR library can track in real time (even when partially occluded) and estimate the relative spatial location and orientation of the device. Initially, the AR marker only contained the limit marks in the corners and the words "Augmented Reality Workbench" in the center, but, to improve the tracking process (specially, when the device was close enough for the camera to only capture a small section of the marker) a longer text was added in the borders.

To facilitate the evaluation process, the final version of the marker was printed on a standard A3 sheet of paper and physically placed on top of the workbench. Although it was originally fixed in place to more closely match the environment definition, after the initial tests the marker was set so that the evaluator can adjust its location to better suit the length of the arm (and physical disabilities) of the participants. Nevertheless, this decision also implied that –intentionally or not– the participants may also move or rotate the marker.

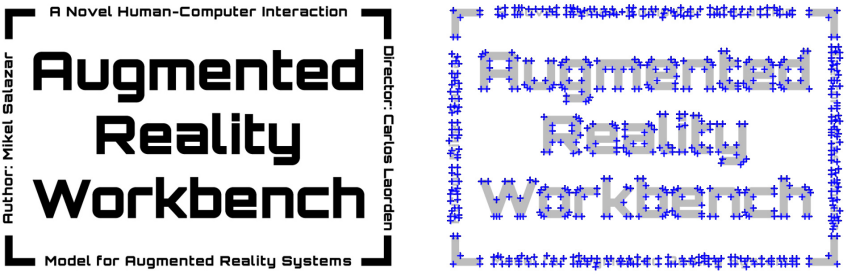

Figure 4.14: The bidimensional image used in the marker (left) and the high-contrast points that the AR library registers and tracks (right).

4.3.4. Software Platform

Due to the large number and diversity of interaction tasks that need to be performed during the usability study, a dedicated software platform is required to successfully complete the different Use Cases. Thanks to the object model described in the previous section, this middleware layer can be implemented over any current three-dimensional engine in a relatively straightforward way (although the input-output management and physical simulation processes may require additional coding).

For the prototype employed in the usability test, multiple development platforms were considered, including the option to create one from the ground up. Ultimately, Unity 4.5 was selected as the main engine due to the relatively simple but powerful API (which allowed a faster iteration time between tests) and the wide selection of AR libraries already available for this environment (of whose Vuforia was selected as the library of choice for the final version of the prototype because of its good performance in mobile platforms and advanced marker tracking systems).

It is important to note that, although the Unity engine includes a sophisticated graphical editor, the –partial– C# implementation of the object model included in the prototype was completely created using the MonoDevelop IDE. Furthermore, as can be seen in figure 4.15, the classes associated with the Unity engine are kept in a separate namespace. In this way, the code –licensed under LGPL v3 terms– can easily be ported to other platforms.

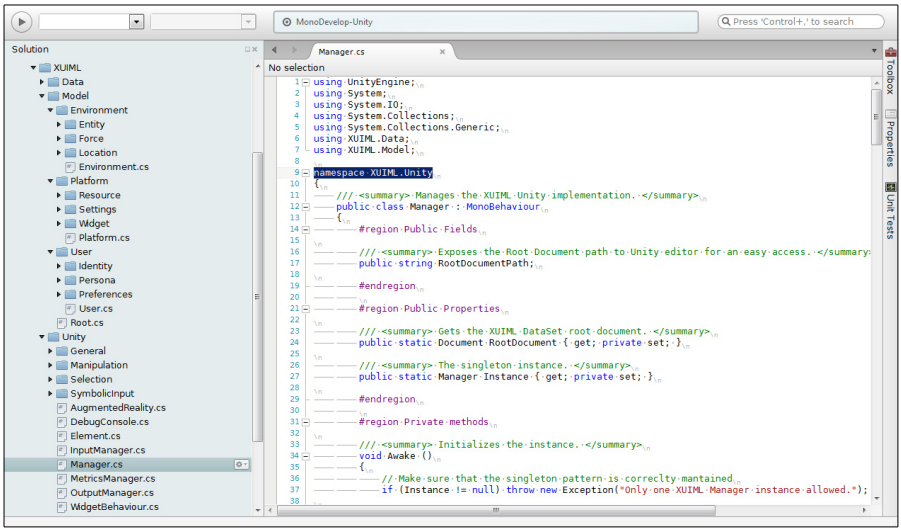

Figure 4.15: An screenshot of the MonoDevelop editor where the C# implementation of the object model was coded. Please note that the files associated with the Unity engine are kept in a different folder to maximize the code portabilitity.

Early in the development of the software platform, the complexity of the code grew to the point that it was not feasible to debug it only from the development system (an Intel i7 920 with 8 GB of RAM memory and a Nvidia GeForce GTX 260). A problem that was averted with the inclusion of a custom debug console that allowed the management of the application execution directly from the experimental equipment. This solution not only enabled a faster correction of logical errors but also provided the usability study evaluators with a tool to check the entire node tree to identify problems within the experiment definition.

In the final stages of development, in order to improve the general performance of the experimental application, each use case was analyzed with the profiler included in the Pro version of the Unity editor. This optimization process achieves an increase in the framerate (almost doubling the mean FPS value, from 15 to 28) and a faster resource load (due to the better compression algorithms), which, in turn, results in a better user experience. Figure 4.16 showcases the elements involved in this process.

Finally, the experimental results are stored into a MySQL database through a custom-made PHP interface. This interface processes a serialized version of the resource node associated to the experiments (that only includes the anonymous data items updated by the user interaction), extracts the relevant metrics and stores them through an SQL Insert operation into the appropriate table of the database.

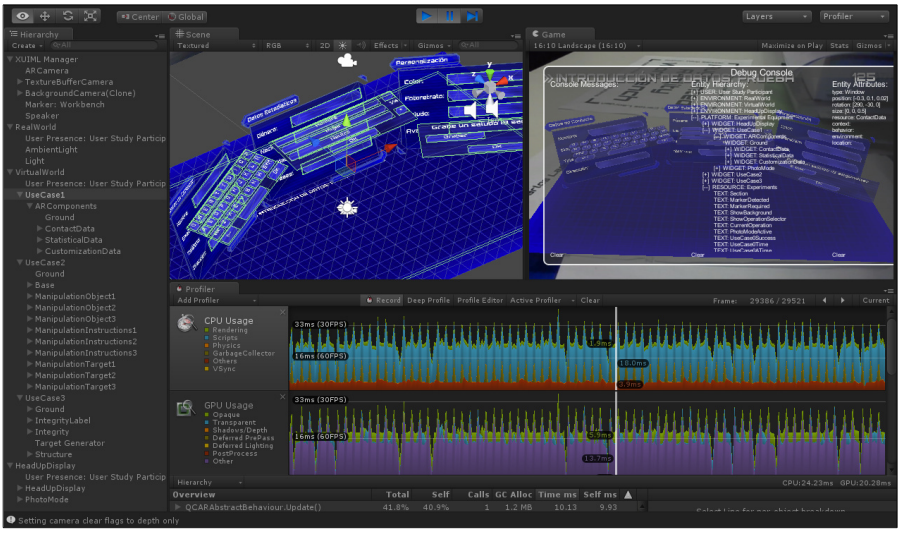

Figure 4.16: An screenshot of the Unity editor in Debug mode (featuring the custom debug console in the top right corner and the profiler tab at the bottom). Except for the "XUIML Manager" object, all elements of the scene hierarchy are recreated in real time.

4.3.5. Visual Representation

During the development of the software platform, several skins –or visual styles– were designed to showcase the diversity of interaction techniques that the proposed model is capable of creating. However, due to the limited graphical capabilities of the experimental equipment and the statistical variations that their inclusion could have induced, the software platform employed in the usability study only contains the *Futuristic* skin. As can be observed in figure 4.17, this visual style gives the widgets a –low-poly– geometric appearance that simplifies the rendering process. Thanks to this representation, the experimental equipment can sustain a fluid framerate (30 frames per second) even at the WQXGA native resolution (2560x1600 pixels).

To enhance the user experience, the visual style also included, particle effects, animations and sounds for the different actions. These additional feedback allowed users to easily identify when a given task have been correctly performed or made a mistake, even when their fingers occluded the widget itself (e.g., when pressing keys or selecting targets). Furthermore, to improve the haptic feedback, the software platform also included a module to take advantage of the vibration motors built into the experimental platform (although, its intensity was toned down in the final version to avoid performance issues).

Figure 4.17: The different widgets employed in the usability study, as displayed in the final version of the experimental platform. The low polygon count, coupled with the smart use of transparent shaders, allows a easier recognition of the different elements and a seamless interaction with them.

4.4. Experimental Results

After ensuring that the experimental platform was robust enough through extensive internal evaluation (including focus groups and expert reviews), the usability study was conducted as a lab test. A challenging enterprise that, albeit was successfully completed, required a large organizational effort and was not achieved without problems.

During the six weeks (from 9 October to 18 November of 2014) that were required to acquire the necessary experimental data, a total of 102 sessions –of approximately 15 minutes each– were conducted; two more than initially intended, due to operating system crashes. Additionally, due to the high energy consumption and excessive heating, the experimental platform had to be left on charge for twenty minutes every three consecutive sessions in order for it to operate in optimal conditions (without a loss of framerate). Moreover, on twelve occasions, network problems hindered the upload of the experimental results (although these situations were later corrected using the included failsafe mechanisms).

Due to time and budget constraints, for this lab test, the majority of the participants were recruited from the Faculty of Engineering (University of Deusto), where the experimental physical environment is situated. Even though great efforts were made to leverage the diversity of the participants (in an attempt to replicate that of the society at large), the degree of variation between the sample values is relatively low.

Despite these shortcomings, the lab test concluded satisfactorily and all the –anonymized– data was recovered from the online database for its throughout examination. A data analysis that, as explained in subsection 4.1.3, involves several cleanup procedures to extract the relevant information in a way that can be later analyzed through statistical methods to find correlations between the different variables. Nevertheless, these intermediate results offer, by themselves, meaningful conclusions about the usability study.

This section is dedicated to present these experimental results, aiming to provide the reader with a more complete understanding of the outcome of the lab test. To this effect, the following subsections showcase the data obtained from the different metrics (both defined in the initial design or derived from the evaluator annotations) and the participants themselves.

4.4.1. Participants Data

During the execution of the usability study, several data items about the individual participants were collected to facilitate posterior analysis procedures. These values include the personal information provided by the participants themselves (properly anonymized to preserve their privacy) and the relevant information that could be extracted from the evaluator annotations.

The variables that store the personal information of the participants were defined during the design phase and the collection of the respective values was integrated within the Use Cases 0 and 1 (by including specialized widgets that allowed the participants to directly input the data). As for the variables extracted from the evaluator annotations, they were identified by reviewing the entire dataset after the conclusion of the usability study.

Main Categories

After performing a general data cleanup process and discarding the variables with extreme dispersion indexes (with either too many or too little samples in each category), three variables were selected as the main categories: the gender, laterality and occupation of the participants. Figure 4.18 shows the distribution for each of these categories.

The gender and laterality variables were specified by the participants though single-choice widgets in the Use Case 1. For the gender variable, participants could choose from a list of 16 items (*Male*, *Female*, *Gender neutral*, *Transexual male*, *Transexual female*, *Bisexual*, *Intersex*, *Transexual*, *Androgynous*, *Asexual* and *Other*) although only the first two options were selected in this study (73% *Male* and 27% *Female*). For the laterality variable, users could select either *Left-handed*, *Ambidextrous* or *Right-handed* (with a result of 86%, 5% and 9%, respectively).

The occupation variable was specified by the evaluators in their annotations and later grouped into three main classes: *Researcher*(50%), *Student*(38%) and *External Personnel*(12%).

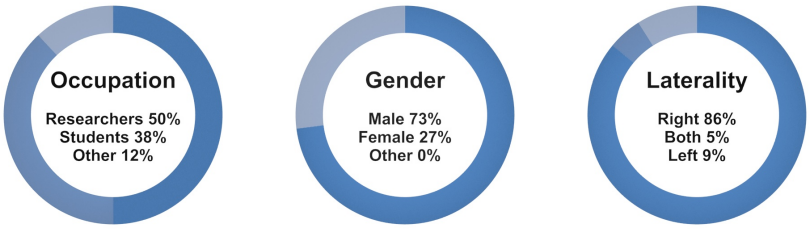

Figure 4.18: The sample, categorized by Gender, Laterality and Occupation.

Age Intervals

Another relevant variable that the participants specify during the Use Case 1 is their current age (using a *datepicker* widget to input their birth date). However, because of the wide range and heterogeneous distribution of the provided values (that can be appreciated in Figure 4.19), they had to be divided into three intervals with a similar number of elements: *less that 21 years* (32%), *between 21 and 28 years* (33%) and *more than 28 years* (35%).

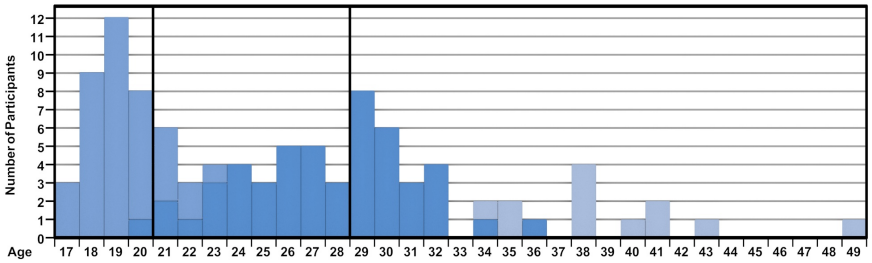

Figure 4.19: The age distribution of the participants, showcasing the different age intervals and the high correlation (88.29%) with the associated occupation.

Technological Experience

To better evaluate the compatibility of the interaction techniques created with the proposed HCI model with the ones that the participants are habituated to use, an additional form was included at the end of the Use Case 0. In this form, the participants are prompted to answer if they had any previous experience with devices equipped with tactile screens, with Computer-assisted Design applications or were familiar with concept of Augmented Reality. As can be seen in Figure 4.20, a 94% of the participants answered affirmatively to the first question, while only a 27% of them did so to to the second question. Finally, an 82% of the total sample admitted to have employed AR before.

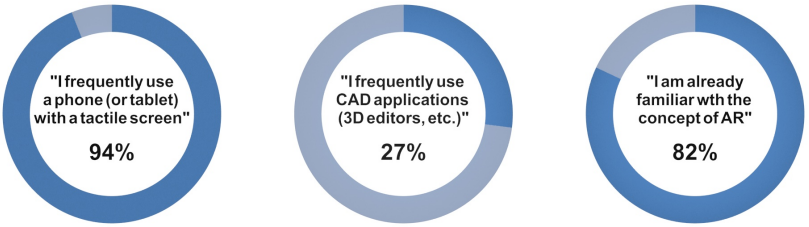

Figure 4.20: The accumulated responses to the three questions regarding the technological experience of the participants.

4.4.2. Performance Metrics Results

As previously explained, performance metrics provide an accurate and objective assessment of how the participants have carried out the different Use Cases -and subtasks- in which the usability study is divided. An excellent viewpoint from where to evaluate the effectiveness and efficiency of the different interaction techniques and estimate the magnitude of any usability issue that may appear.

The collection of the results associated to these metrics requires the creation of a dynamic program analysis subsystem (commonly referred to as a *profiler*) that constantly monitors the the computer system and updates the appropriate variables when needed.

Success Rates

To evaluate the different groups of techniques in a simple way, each Use Case (except Use Case 0) has a single binary metric that describes whether a participant has completed it successfully or not. In Use Cases 1 and 2, this metric checks if all the tasks have been completed in the alloted time (300 seconds). In Use Case 3, because there is a fixed number of seconds for each task, the success is determined by the integrity value associated to the structure that the participants are tasked with protecting (if this value gets below 0, it is considered as a failure).

As figure 4.21 shows, the success rate is very high for all the Use Cases, with average values of 96%, 91% and 86% respectively. The decline in these values can be attributed to the –deliberate– increase in difficulty throughout the different tasks, as well as the weariness that several participants experienced due to the weight of the experimental hardware.

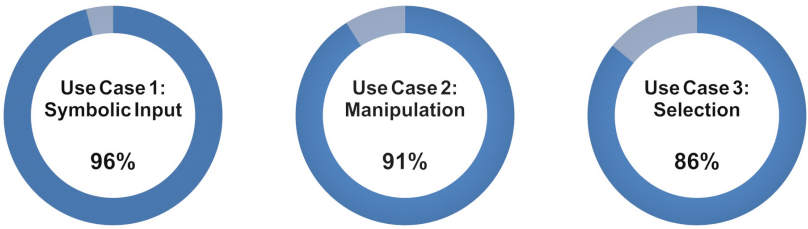

Figure 4.21: The success rates for Use Cases 1-3.

Time-on-Task

Another metric deserving of serious attention is the amount of time that the participants required to complete the different tasks in each Use Case. While it is not the only variable that should be taken into consideration when evaluating the efficiency value of interaction techniques (at least, not without providing the user with enough time to master them), it certainly provides a simple way to measure and compare them.

The results for this metric can be observed in Figure 4.22. Please note that, due to the fixed duration of the tasks of the Use Case 3 (Selection), the associated Time-on-Task results do not present any variation.

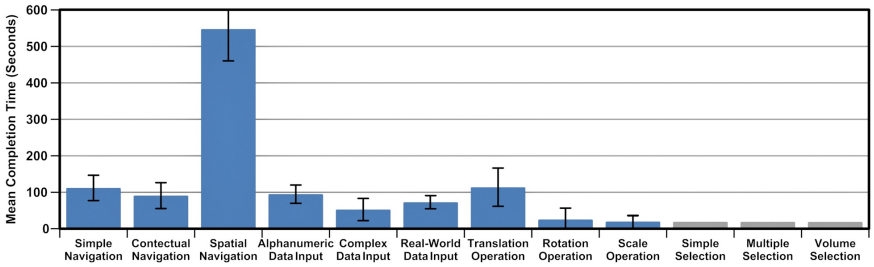

Figure 4.22: The average Time-on-Task values for the different tasks.

Execution Errors

The last performance-based metric included in the profiler measured the number of errors that the participants made while employing the different interaction techniques. As can be seen in figure 4.23, each task had different error conditions and number of error opportunities: in UC0 errors are only contemplated in the Physical Navigation task (when the tracker is lost). In UC1, all erroneous inputs constitute a different error while in UC2 using the counter increases each time an incorrect transformation operation is used. Finally, unintentionally selecting the structure in UC3 is considered an error (and substracts a small portion of the structural integrity value).

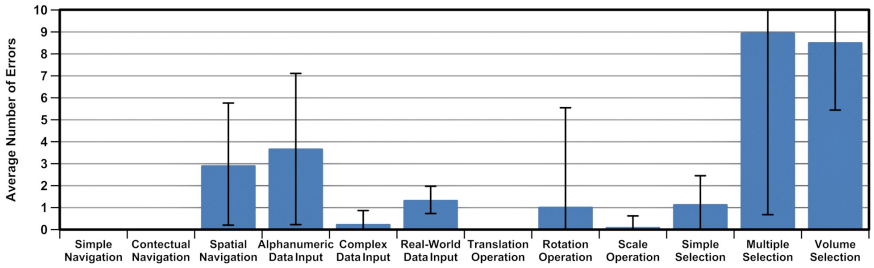

Figure 4.23: The average number of execution errors in each task.

4.4.3. Issues-based Metrics Results

Acknowledging the fact that it is not feasible to implement a user interface system that perfectly matches the usability requirements of every participant, the experimental design included several metrics to identify problems that may arise during each session and, if necessary, solve them before they impede the completion of a task. Because the identification of usability issues is a complex process that requires the constant monitorization of the experimental equipment and the behavior of the user, the collection of these metrics was performed by the evaluators themselves (by manually annotating the values on a separate sheet of paper and, after retrieving the experimental equipment at the end, including these annotations into the data later submitted to the online database).

Guidance Points

Already defined in the design phase, this list twelve items –tree for each Use Case– attempts to enumerate the most predictable issues that the participants might encounter during the sessions. Each item contains a precise description of its activation conditions and the actions that the evaluator must perform to assist the user. Additionally, to facilitate the collection of the associated data, the entire list of guidance points was included in the printouts that the evaluators were instructed to use for their annotations.

As can be seen in the Figure 4.24, the average usage of guidance points reveals that nearly half of the participants required additional help to adequately employ the interaction techniques required for each task. Also, in Use Case 3, 94% of the participants had to be reminded that selecting the structure is an invalid operation.

Another relevant conclusion that can be extracted from these results is that physical navigation techniques were not intuitive (with an 85% of the participants requiring the *Spatial Positioning* explanation), but they were relatively easy to learn (reducing the need of additional explanations to a 70% and 27% in sucesive scenarios).

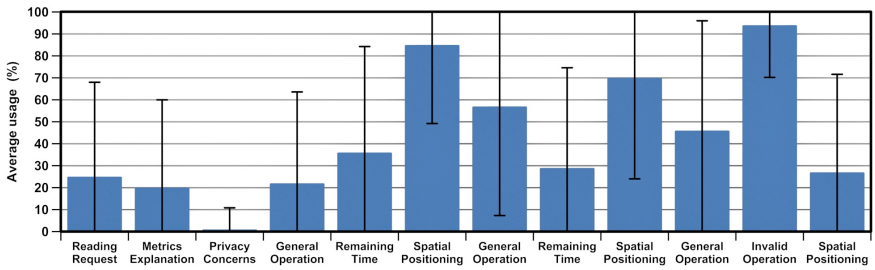

Figure 4.24: The average usage of guidance points.

Usability Issues

Through the careful examination of the annotations provided by the evaluators, it was possible to discover several unforeseen problems related with the usability of the experimental platform:

- **Navigation issues:** At the beginning of Use Case 1, when the participants were required to use physical navigation techniques for the first time, 85% of them required additional assistance to position the device in such a way that they could properly operate the soft keyboards (typically, they started pressing the keys too far away from the relative position of the widgets, resulting in many symbolic input errors). While understandable, this impediment could have been palliated by including a tutorial screen where the users are able to familiarize themselves with the navigation possibilities available in AR environments (before being asked to complete other tasks in a timed scenario).

- **Terminology issues:** In the final version of the experimental system, the use of generic terms such as "target" and "structure" (intentionally selected to describe the same widgets independently of the visual *skin* applied) proved to be a hindrance for, at least, 6% participants. In those occasions, the users expressed doubts at the beginning of the Use Case 2, unable to establish a connection between the instructions and the widgets provided for this scenario. Moreover, n the Use Case 1, many users did not understand the instructional text associated with the second real world analyzer (the one that asked them to provide a photo of them).

- **Content Issues:** Apart from the semantical divergences associated with the aforementioned structure in Use Case 2, many participants found the visual appearance of the widgets confusing (specially, the one that scale operation to). Also, 34% of the participants required several attempts to find the "@" symbol in the soft keyboard designed to input email addresses (even though it was placed at the bottom of it).

- **Functionality Issues:** Although the final version of the experimental platform went through several revisions to ensure the functionality of the interaction techniques, several widgets presented problems in their behaviors. Most notably, (due to the implicit conversion between euler angles and quaternions) the rotation widget could suffer the loss of one degree-of-fredom as a result of a gimbal lock. Additionally, the persistence of the operation buttons at the bottom of the screen (in a non-spatial layer) often resulted in the unintended cancellation of selection operations in Use Case 3.

4.4.4. Self-Reported Metrics Results

As a counterbalance to the aforementioned types, self-reported metrics offer a subjective view of the different interaction scenarios. A –rather hazy– perception that, though the use of Likert scale widgets, can be transformed into concise data types for easier transmission and analysis.

Ease of Use

At the end of each Use Case, the participants were presented with a small form that compelled them to rate the different interaction techniques that they had to employ to compete the given tasks. However, due to the complexity of the term *usability* (that encompasses concepts such as learnability, intuitiveness, ergonomics, etc.), the participants were asked to evaluate each technique on the basis of the perceived difficulty for each task (using the values of "Hard to use" and "Easy to use"for both extremes of the Likert scale).

As Figure 4.25 shows, the averaged result values for all the metrics are strongly leaned towards the "easy" side, with a total –aggregated– mean of 4.12 out of 5. On average, those related with simple tasks such as basic navigation or simple selection have higher values in the scale, whereas those that require more sophisticated inputs receive a lower rating. Special mention deserve the metrics associated with the rotation operation and the input of real-world data; in the first case, the aforementioned issues derived from the *gimbal lock* resulted in a lower evaluation, while the hardware limitations severely hindered the experience while employing the physical environment analyzers. Also, the annotations provided by the evaluators suggest that the lower score associated with the contact data input may be motivated by the problematic implementation of the soft keyboard employed for the introduction of the email address (that substitutes the space bar of the traditional alphanumeric keyboard with the common symbols of the email address syntax).

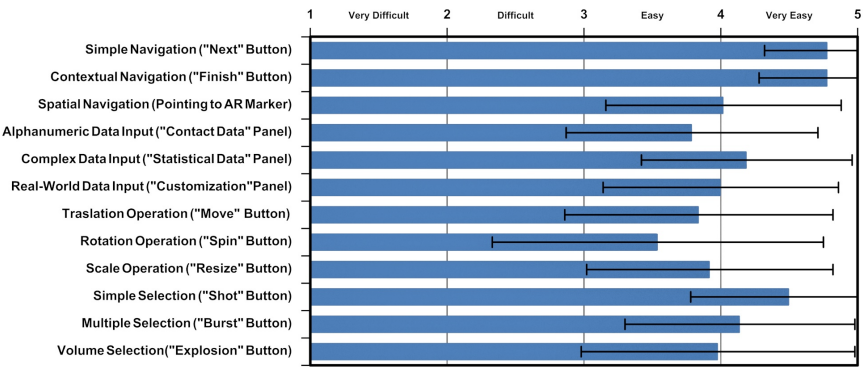

Figure 4.25: The averaged values of the different ease-of-use metrics.

SUS Form

While the System Usability Scale (SUS) form does not provide a very exhaustive evaluation of the interaction techniques employed in the study, it is an excellent way to determinate the global validity of a system. Furthermore, being a standardized method (ISO 9241 Part 11), the results obtained with this form can be compared with that of other usability studies.

With an –aggregated– global score of 75.2, these results can be considered as very positive. As figure 4.26 shows, this score allows the experimental system to be ranked above 72% of tested systems (using data from over 5000 users across 500 different evaluations [143]).

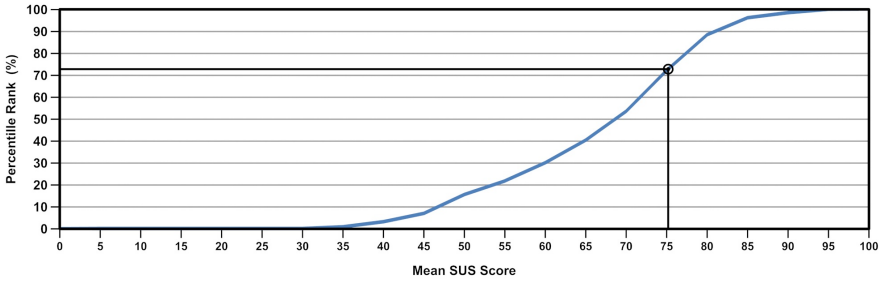

Figure 4.26: The position of the experimental system in the ranking of SUS scores.

Additionally, as can be seen in figure 4.27, the averaged results for the individual points follow a zigzag pattern that matches the polarity (affirmative/negative nature) of the associated explanatory sentence. Among these results, it is worth highlighting that the participants found the different interaction techniques especially intuitive and well integrated. Conversely, these values show that they also consider the entire experimental system a little complex and do not feel very confident with it.

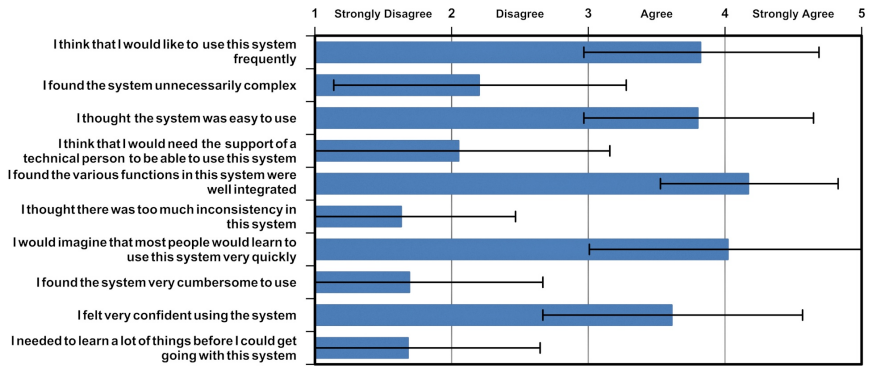

Figure 4.27: The averaged scores for each SUS form item.

4.5. Result Analysis and Discussion

As explained in the Methodology section of this chapter, the extraction of robust conclusions from the experimental data necessitates of an in-depth statistical analysis. In addition to the individual mean values of each metric, this process takes into consideration the relationships between the different variables and calculates their actual significance. In this way, it is possible to identify the most problematic elements of the usability study and evaluate them with increased precision.

It is important to note that, due to the myriad of external factors (i.e., ambient, societal and personal conditions) that can have an effect on the usability measurements, it is not possible to asseverate the statistical results with total certainty (applying the famous dictum "correlation does not imply causation"). Nevertheless, this section provides objective justifications for the most relevant results.

4.5.1. Dependence Analysis

To appropriately conduct the statistical analysis, the variables of the final dataset were divided into two groups based on the dependence between the variables:

- **Independent variables:** This group encompasses the values directly related to the usability study participants: gender, age, laterality, occupation and technological experience. Although their values are inputed during the experiment, these variables are not linked to its execution and, consequently, meaningful conclusions can be inferred from them. A process that, as is the case in this study with the age intervals, might require the grouping of several values to establish classifications with enough items on each category. Additionally, it is recommended to analyze the relationships between these variables to better understand the real effect that each one has over the statistical analysis results (assuming that the sample is heterogeneous enough).

- **Dependent variables:** The performance, issues-based and self-reported metrics collected during the experiment store relevant evaluation data and, thus, are categorized as dependent variables. Because these metrics have been hand-crafted to provide different views of a limited set of interaction scenarios, their values maintain strong correlations (specifically, those that measure the times of the use cases and of their individual tasks). Nevertheless, this degree of redundancy facilitates the discovery of discrepancy and errors that may have remained hidden otherwise.

4.5.2. Statistical Analysis

With the dependency of the variables already established, multiple analysis procedures can be performed. Through the use of descriptive statistics, it is possible to calculate the measures of central tendency and variability for each variable. The resulting means can then be compared by using methods such as the t-test(for two samples) or ANOVA (for three or more samples), obtaining a table of dichotomic values that reflects –the significance of– the relationships between the different variables.

After applying these statistical analysis methods to the experimental data (with a confidence level of 95%, two-tailed tests and assuming unequal variances), the resulting table reveals several relevant correlations. Unfortunately, the large number of dependent variables in the dataset (encompassing a total of 61 metrics) impede the direct inclusion of the entire table in this section.[1] Furthermore, due to the binary nature of the table values, they cannot be easily represented in graphic form (at least, not without employing misleading values such as the internal significance or the standard error measure). Consequently, in the interest of clarity, the significant results are presented the following enumeration, grouped by independent variables (and ordered by their influence):

1. **Technological Experience:** As anticipated, the most determinant factor for the successful completion of the usability study has been proven to be the knowledge and abilities that the participants had acquired before the test. Those participants that answered positively to the first point ("I frequently use a phone -or tablet- with a tactile screen") performed ostensibly better than their counterparts in the last two use cases, both in terms of success rate (with a 9.6% and 14.9% increases, respectively) and mean completion time, and required the additional explanation on how to input self-reported metrics with less frequency. In addition, the participants that declared that they "frequently use 3D CAD applications", completed the use case 2 significantly better (100% success rate), in less time (-23 seconds on average) and with less mistakes. Finally, those that claimed to be "already familiar with the concept of Augmented Reality" did solve the first task faster (-13.9 seconds) and, in general, the evaluator was not required to explain them how to take advantage of the physical navigation techniques.

2. **Occupation:** The occupation group to which each participant belongs is a good indicator of the associated experimental result values. Although there are clear differences between the success rates of researchers, students and external personnel, the analysis of variance test suggest that the most significant metrics are those that store time-on-task values and SUS form answers. By performing additional mean

[1] For a complete view of the results, please visit `http://thesis.mikelsalazar.com`.

comparison tests for each combination of classes, it is possible to discern great differences between them. More concretely, researchers tend to complete the different tasks significantly faster than students (15,7%) and external personnel (17.2%), while students provide the highest combined SUS scores (an average value of 80.2, against the 74.1 of researchers and a 63.4 of external personnel). Also, the evaluators were required to provide additional guidance in the use case 3 to external personnel more frequently (mostly due to inexperience with game scenarios).

3. **Age Interval:** As explained earlier in this section, there is a strong correlation between the age interval and occupation groups. This entails homogeneous mean values for the respective categories and, consequently, the mean comparison results follow a similar pattern. Nevertheless, the limits of the age intervals (that divide the dataset into three parts with a similar number of samples) do not coincide with the partitions between occupation groups, affecting the outcome of the ANOVA procedure. More concretely, the variations in SUS scores are still considered significant enough, but the differences in time-on-task mean values –between the different classes– are too small to sustain a proper correlation. On the other hand, this new reclassification also serves to identify a significant relationship between the age of the participants and their subjective evaluation of the physical navigation techniques (with a difference of 10.1% between the second and third age intervals).

4. **Gender:** The gender of the participants does not have a significant impact on the results except for the second use case, where the female participants have a lower success rate (77.8%, as against 95.9%) and required, on average, 34 seconds more than their male counterparts to complete the different tasks. Although this can be attributed to the cognitive differences in spatial visualization abilities detailed in numerous studies [144, 145] (which is consistent with the increased rate in which the evaluators have to provide guidance about spatial positioning), it is important to mention that female participants declared to have less experience with 3D CAD software (a 14.8% of them, in clear contrast with the 31.5% rate of male users).

5. **Laterality:** Often contemplated in usability studies due to the occlusion problems that left-handed users experience with tactile screens, this variable has proved to have a relatively small effect on the experimental results. Specifically, data shows that left-handed participants have a higher success rate in the first use case (a 100%, as against the 95.1%) and a higher error rate on the second use case. However, the relatively small number of left-handed persons that took part in this study (9 out of 100), makes it difficult to guarantee that the same correlations can be replicated with a larger sample.

4.5.3. Results Discussion

As explained at the beginning of this chapter, there is no such thing as a perfect method to evaluate –and, thus, validate– a HCI model. Even if its theoretical foundation supports all existing interaction techniques, it is not feasible to test them in all possible scenarios (and, in fact, it would be counterproductive because it could limit their adaptation possibilities). Furthermore, the extreme diversity of human factors that have to be taken into consideration for each scenario makes such an attempt nearly impossible. Is for these reasons that, for the validation of the HCI model proposed in this document, a more straightforward approach had to be adopted.

As the main hypothesis establishes, the main goal of this research work is to demonstrate that *"it is possible, by means of Augmented Reality systems, a comprehensive Human-Computer Interaction model based on spatial user interfaces integrated within the physical environment"*. Consequently, the most reliable way to validate such hypothesis is to actually implement a system based on the proposed model and develop a spatial user interface that puts its different components to the test. An evaluation methodology that, in the context of user-centric interaction design, is referred to as a "usability testing".

The culmination of this research effort was an exhaustive usability study that showcased the different types of interaction techniques in specialized use cases. As indicated throughout this chapter, this usability study was conducted as a lab test, with more than satisfactory results. Even in a novel interaction space and under relatively rigorous conditions in terms of alloted time and task complexity, a large majority (a 80%) of the study participants successfully completed all the tasks with a rather low error rate. Furthermore, the usability metrics provided by the participants and the evaluators are really positive (considering the technological limitations and inherent complexity of the experimental platform), and the global results are relatively good compared with those of other studies based on current HCI models.

These satisfactory experimental results, in conjunction with the large sample size of the lab test (with one hundred participants), allow the validation of the main hypothesis of this research with more than enough confidence. An assertion that, nevertheless, will be further solidified by the conduction of additional –online– tests in the near future (as the advancements in the field allow the overcoming of the present technological limitations).

"But the beauty is in the walking. We are betrayed by destinations."

Gwyn Thomas

5

Conclusions

This last chapter of the dissertation reviews the research work and provides the key reasonings to understand its relevance within the scientific field. Beyond merely proving the validity of the stating hypothesis (and that of the research methodology employed), the following pages give a proper closure to the dissertation and define a possible course of action for its application in the current society (while also taking into consideration different real world problems that might endanger its general acceptance).

To achieve these goals, the present chapter reviews the main contributions generated during research project in section 5.1 and discusses its general results in section 5.2. Later, section 5.3 analyzes the current limitations of the proposed interaction model; a comprehensive list of present issues that also serves as the base for the determination of the venues of future work in section 5.4. Finally, to provide a satisfactory conclussion to the document, section 5.5 offers a global perspective of the research work and analyzes several factors that might affect its proper application in today's world.

5.1. Main Contributions

As explained in section 1.4, the research project presented in this dissertation employed from its beginning an iterative methodology to adequately explore with the large number of interrelated disciplines that conform HCI. An increasingly complex process that, in each iteration, has crystallized into a series of contributions that have been presented into different scientific and industry-oriented venues. In turn, this work resulted in a large amount of –invaluable– expert feedback, that allowed the author to keep the research efforts in the right track.

The first complete iteration of this research project was presented at the Doctoral Consortium session during the 11th International Symposium on Mixed and Augmented Reality (ISMAR) in November, 2012 [137]. This opportunity allowed the author to pitch the original idea behind this dissertation to a panel of leading researchers and receive general directions on how to conduct the next phases of the research(as well as indications about the pitfalls to avoid).

While maintaining the presence in the research community (by taking an active part in panels disscussion the matter [146]), the author has also participated in several industry-oriented meetings [138] to gain a full understanding of the actual necessities of developers and manufacturers. Thanks to the additional feedback obtained from the industry experts that attended these meetings, the resulting interaction model (specially the proposed network submodel) has achieved a more solid structure and a wider relevance. Moreover, as a result of these contributions, the author of this dissertation has been awarded with a membership to the Khronos group (one of the most important standardization organizations in the industry).

Furthermore, at the time of writing this document, several contributions to the scientific community have been submitted: i) an article containing a detailed description of the last iteration of the proposed model has been submitted to the *Human-Computer Interaction* journal, ii) a publication detailing the overall concept of the approach to the ISMAR 2015 conference and iii) a demo shocasing the usability study to the SUI 2015 conference.

Meanwhile, in preparation for the online test (that further validate the proposed model), the author of this dissertation has published a modified version of the experimental software on the main distribution system of Android platforms.

5.2. Results Discussion

From their definition in section 1.3.2, the present document has provided the necessary arguments to accomplish the *objectives* of this research work:

- ☑ **Examination of the state of the art:** Chapter 2 provided a detailed analysis of the evolution and current situation of HCI. Additionally, it presented a comprehensive set of interaction techniques both from the hardware and software perspectives.
- ☑ **Formulation of an interaction model:** With the necessary conceptual foundations firmly established, chapter 3 undertook the the construction of a new interaction model in an attempt to overcome the current limitations of the scientific field. To that effect, the chapter offered three different –but complementary– perspectives of the proposed model: a purely theoretical overview, a logical representation (appropriate for the implementation in computer systems) and, finally, a network model to facilitate the communication through computer networks
- ☑ **Validation of the proposed model:** To adequately evaluate the interaction model, it is necessary to define a usability study that features the different interaction techniques. Chapter 4, offers a detailed account of this validation process, including the methodology employed, the design of the experimental platform and the analysis of the results.

Having successfully completed the intended research endeavor, the last step is to check its starting hypothesis:

> ***It is possible to construct a comprehensive Human-Computer Interaction model for Augmented Reality Systems, based on Spatial User Interfaces.***

As proven throughout the present document, it is not only possible to construct a viable HCI model for AR systems, but the creation of spatially-based interaction techniques can also facilitate the development of new and more intuitive user experiences. Therefore, the initial hypothesis can be considered as valid, paving the way for new research venues.

5.3. Current Limitations

Before examining the possible venues for future work, it is important to analyze the limitations of the proposed interaction model. In this way, any person (with either an academic or industry background) that wishes to expand this research work will be able to easily identify the best starting point for their own projects.

The main limitations of the proposed interaction model are the following:

- **Limited Definition:** The most obvious limitation of the solution presented in chapter 3 is the absence of a precise specification of the minor object nodes. Although this is intentional (to provide a better understanding of the proposed model and ensure its platform and language independence), this approach can make the implementation of the model a cumbersome process and might even lead to incompatibilities in the final software products.

- **Limited Widget Behaviors:** As aforementioned, in the present model, the functionality of the user interface is defined primarily by sequences of action-reactions contained within the widget behavior nodes. While this solution offers significant benefits in terms of simplicity and accessibility, ultimately, this solution can only perform –relatively– simple operations over the elements contained inside the very model; more advanced functionalities inevitably require the employment of additional script languages or mechanisms to interact with external programs. Nevertheless, this limitation is understandable from the point of view of systems architecture (that recommends to establish a division between the business logic and that of user interfaces) and, in the case of processes that require large data (i.e., remote services), it is often preferable.

- **Technological Limitations:** At the time of writing this dissertation, the use of Augmented Reality technologies is mostly restricted to individual applications and, although many hardware platforms –particularly, HMDs– provide more mechanisms to analyze the real world and present the augmentations to the end users, they are not yet enough to provide a comfortable user experience. Despite the best efforts of the author (including the creation of multiple hardware prototypes for the usability study), currently, there is not a computer system that can take full advantage of the proposed interaction model.

5.4. Future Work

Due to the inherent complexity and constant evolution of the HCI field, the research work presented in this document offers numerous opportunities for future work (so many, in fact, that it can be said that this project *has no real end*). Nevertheless, in an attempt to overcome the aforementioned limitations, there are three main courses of study that the author of this dissertation will continue to pursue:

- **Standardization Effort:** To facilitate the adoption of the proposed interaction model in both academic and industrial environments, it is necessary to provide a specification document that enumerates the different encoding formats and provides implementation details for different development environments. Apart from making this document freely available, it is essential to present it in scientific conferences and standardization meetings to collect the feedback of leading experts and employ this knowledge to constantly refine and update the technical standard.

- **Reference Implementation:** To allow this standard to properly mature, the next step is to construct a reference implementation from which third party developers can create their own implementations. Rather than being a proper development project, this open source implementation should serve as a blueprint for other middleware solutions that extending the functionality of the proposed model (while also increasing the number of interaction mechanisms supported by the widget behaviors). Furthermore, with the contributions of external developers, this project can grow to include versions in different programing languages and tools to facilitate the transition from previous interaction model.

- **Creation of a complete platform:** Once the reference implementation is robust enough, the goal becomes the construction of an open hardware platform specially designed to take advantage of the all its features. A lofty goal that will require the participation of manufacturers and retailers (although the Do-It-Yourself approach should also be encouraged), to reach a wider audience and allow the spontaneous generation of self-sustaining software ecosystems.

5.5. Final Remarks

To properly conclude this dissertation, it is necessary to review the entire research work and analyze several factors (both related with the research itself and external) that might have an impact on its realization in the present world.

- **Technological Limitations:** First and foremost, it is of great importance to note that the proposed interaction model has not only been designed to address the limitations on previous HCI paradigms, but also provides a solution to several problems that both the scientific community and industrial experts have identified as possible *technological bottlenecks* for the development of Augmented Reality-based hardware systems [5]. Most notably, even through current mobile platforms are not –yet– capable of acquiring a proper definition of the physical surroundings and the psychological situation of the users, the theoretical model presented in this document provides a relatively simple way to construct them (or retrieve it from other sources). In this way, it is possible to create advanced interaction techniques that take into account the situation of the users and, even, provide mechanisms to protect them against physical threats. A forward-looking approach that greatly facilitates the creation, sharing and discovery of meaningful user experiences, while also being flexible enough to be adapted to different hardware platforms, current or future.

- **Privacy Concerns:** The present state of technology is far from being the only factor that has been taken into consideration during the design of the proposed interaction model. As can be seen in chapter 3, the definition of the main interaction components and the security mechanisms of the network model reflect current concerns about privacy and misuse of personal data (in large part, resulted from recent revelations about governmental surveillance programs [146]). A critical issue not just because any implementation of the model would greatly benefit from having a good definition of the user context but, more importantly, because not providing a good protection mechanism puts the security of the end users at risk. Something that can very easily thwart any attempt to build a large enough user base.

- **Unrealistic Expectations:** Many potential end users are conditioned by the same sources of inspiration mentioned back in section 1.1, and often have unrealistic expectations about what can be achieved with present AR platforms. A dangerous scenario that, coupled with the limitations of current see-through HMDs, might predictably result in a *tech bubble* and, later, force the industry to recover from its inevitable burst.

- **Societal Issues:** Even if the previous pessimistic scenario does not materialize, academic researchers and industry experts alike would still have to find solutions to other issues related with the use of this technology in the present society: from the *Digital Divide* effect (presented in chapter 2), to the health and environmental risks associated to the products themselves. Furthermore, it will also be important to monitor the interactions of the user in real-life environments to identify and prevent (or, at least, minimize) the new issues that might arise, such as technological dependence, cyberbullying or restrictions to inter-human communication (i.e., the employment of HMDs might partially occlude the head of the user, thus, limiting the range of facial expressions that the later can employ to convey his/her emotions).

- **Business Models:** While the network model presented in section 3.3 do not expressly establish any mechanism to perform economic transactions, such possibility has also been taken into account during its design phase and can be easily implemented using the existing protocols. Moreover, the proposed security mechanisms facilitate the monetization of the transmission of specific interaction elements (especially, *content* nodes, because their metadata can detail the licensing terms they are subjected to), without having to maintain a strict control over them. In this way, the users can freely discover and share new interaction experiences, while providing the appropriate retribution to their authors in a direct way (the use of intermediaries is a viable option, if not recommended). Furthermore, by employing a DHT system to uniquely identify each interaction element, it is possible to automatically detect, trace and deactivate scam schemes. However, it is important to remark that, ultimately, the control –and responsibility– of the creation of a sustainable business model lies on the end users that conform the network.

- **General Acceptance:** The final factor that determine the applicability of the proposed interaction model is its adoption by manufacturers, third party developers and end users. While this statement may seem obvious, achieving this goal is a difficult endeavor that implies convincing the different participants of the benefits of using a common language to describe the interaction between. As mentioned in the previous section, this process requires the negotiation of a standard with the main industry experts and provide developers with a reference implementation that allows the easy creation of new user experiences. However, this process might not mean anything if the end users do not have a proper computer platform that enables them to access them and, therefore, it will be necessary an additional effort to begin a "snowball effect" that allow the overcoming this initial hurdle.

Bibliography

[1] I. Pedersen and L. Simcoe, "The Iron Man Phenomenon, Participatory Culture, & Future Augmented Reality Technologies," in *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, (New York, NY, USA), pp. 291–300, ACM, 2012.

[2] D. Van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations," *International Journal of Virtual Reality*, vol. 9, no. 2, p. 1, 2010.

[3] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented reality technologies, systems and applications," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 341–377, 2011.

[4] H.-L. Chi, S.-C. Kang, and X. Wang, "Research trends and opportunities of augmented reality applications in architecture, engineering, and construction," *Automation in Construction*, vol. 33, pp. 116–122, 2013.

[5] S. Feiner, D. Schmalstieg, S. Izadi, and B. H. Thomas, "Have We Solved the User Interaction Problem for Augmented Reality?." `http://ismar.vgtc.org/ismar/2014/panel/sct/have-we-solved-user-interaction-problem-augmented-reality`.

[6] T. Langlotz, J. Grubert, and R. Grasset, "Augmented reality browsers: essential products or only gadgets?," *Communications of the ACM*, vol. 56, no. 11, pp. 34–36, 2013.

[7] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical Control of a Prosthetic arm for Self-feeding," *Nature*, vol. 453, no. 7198, pp. 1098–1101, 2008.

[8] F. Noz and J. An, "Cat Cat Revolution: An Interspecies Gaming Experience," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2661–2664, ACM, 2011.

[9] R. T. Azuma *et al.*, "A survey of augmented reality," *Presence*, vol. 6, no. 4, pp. 355–385, 1997.

[10] G. Kipper and J. Rampolla, *Augmented Reality: an emerging technologies guide to AR*. Elsevier, 2012.

[11] B. Butchart, M. Salazar, and T. Liao, "AR Glossary of Terms." Ninth AR Standards Community Meeting. `http://www.perey.com/ARStandards/ninth-meeting-agenda-2/#Session13`.

[12] A. C. Traub and J. Orbach, "Psychophysical studies of body-image: I. the adjustable body-distorting mirror," *Archives of General Psychiatry*, vol. 11, no. 1, pp. 53–66, 1964.

[13] B. J. Steinhoff, N. Freudenthaler, and W. Paulus, "The influence of established and new antiepileptic drugs on visual perception," *Epilepsy research*, vol. 29, no. 1, pp. 49–58, 1997.

[14] A. Roch, "Fire-Control and Human-Computer Interaction. Towards a History of the Computer Mouse (1940-1965)," *Vectorial Elevation Relational Architecture*, vol. 4, pp. 115–128, 2000.

[15] T. P. Caudell and D. W. Mizell, "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes," in *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, vol. 2, pp. 659–669, IEEE, 1992.

[16] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum," in *Photonics for industrial applications*, pp. 282–292, International Society for Optics and Photonics, 1995.

[17] S. Mann, "Mediated reality with implementations for everyday life," *Presence Connect, August*, vol. 6, 2002.

[18] M. C. Juan, M. Alcaniz, C. Monserrat, C. Botella, R. M. Baños, and B. Guerrero, "Using Augmented Reality to Treat Phobias," *Computer Graphics and Applications, IEEE*, vol. 25, no. 6, pp. 31–37, 2005.

[19] C. Botella, M. Juan, R. M. Baños, M. Alcaiz, V. Guillén, and B. Rey, "Mixing Realities? An Application of Augmented Reality for the Treatment of Cockroach Phobia," *Cyberpsychology & Behavior*, vol. 8, no. 2, pp. 162–171, 2005.

[20] T. N. Arvanitis, A. Petrou, J. F. Knight, S. Savas, S. Sotiriou, M. Gargalakos, and E. Gialouri, "Human factors and qualitative pedagogical evaluation of a mobile augmented reality system for science education

used by learners with physical disabilities," *Personal and ubiquitous computing*, vol. 13, no. 3, pp. 243–250, 2009.

[21] E. Richard, V. Billaudeau, P. Richard, and G. Gaudin, "Augmented Reality for Rehabilitation of Cognitive Disabled Children: A Preliminary Study," in *Virtual Rehabilitation, 2007*, pp. 102–108, IEEE, 2007.

[22] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

[23] A. Van Dam, "Post-wimp user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–67, 1997.

[24] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Physical review letters*, vol. 71, no. 24, p. 4083, 1993.

[25] T. Freeth, Y. Bitsakis, X. Moussas, J. Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey, *et al.*, "Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism," *Nature*, vol. 444, no. 7119, pp. 587–591, 2006.

[26] L. T. White, *Medical astrologers and late medieval technology*. Brepols Publishers, 1975.

[27] L. Taub, "On scientific instruments," *Studies in History and Philosophy of Science Part A*, vol. 40, no. 4, pp. 337–343, 2009.

[28] B. Randell, "From analytical engine to electronic digital computer: The contributions of ludgate, torres, and bush," *Annals of the History of Computing*, vol. 4, no. 4, pp. 327–341, 1982.

[29] "2008 computer pioneer recipient: Betty jean jennings bartik." IEEE Society, Retrieved on May 20, 2015 `http://www.computer.org/web/awards/pioneer-betty-jean-bartik`.

[30] R. Rojas, "Konrad zuse's legacy: the architecture of the z1 and z3," *Annals of the History of Computing, IEEE*, vol. 19, no. 2, pp. 5–16, 1997.

[31] P. Russo and S. Boor, "How fluent is your interface?: designing for international users," in *Proceedings of the INTERACT'93 and CHI'93 conference on human factors in computing systems*, pp. 342–347, ACM, 1993.

[32] T. A. Wadlow, "The Xerox Alto computer," *BYTE Mag*, vol. 6, no. 9, pp. 58–68, 1981.

[33] J. Johnson, T. L. Roberts, W. Verplank, D. C. Smith, C. H. Irby, M. Beard, and K. Mackey, "The Xerox Star: A Retrospective," *Computer*, vol. 22, no. 9, pp. 11–26, 1989.

[34] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a Wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pp. 11–14, ACM, 2008.

[35] S. N. Pisutha-Arnond, "Intuitive gesture-based graphical user interface," Apr. 28 1998. US Patent 5,745,116.

[36] B. Buxton *et al.*, "Multi-touch systems that i have known and loved," *Microsoft Research*, vol. 56, pp. 1–11, 2007.

[37] R. Francese, I. Passero, and G. Tortora, "Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 116–123, ACM, 2012.

[38] D. Wigdor and D. Wixon, *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[39] D. Kammer, J. Wojdziak, M. Keck, R. Groh, and S. Taranko, "Towards a formalization of multi-touch gestures," in *ACM International Conference on Interactive Tabletops and Surfaces*, pp. 49–58, ACM, 2010.

[40] D. A. Norman, "Natural User Interfaces are not natural," *interactions*, vol. 17, no. 3, pp. 6–10, 2010.

[41] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the Accuracy and Robustness of the Leap Motion Controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[42] Y. Xi, S. Cho, Y.-S. Jeong, K. Cho, and K. Um, "Surface Touch Interaction Method Using Inverted Leap Motion Device," in *Future Information Technology*, pp. 469–474, Springer, 2014.

[43] "App Stores Growth Accelerates in 2014." AppFigures, Retrieved on May 20, 2015 `http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/`.

[44] "Windows 8: Design over Usability." MIT Technology Review 2013, Retrieved on May 20, 2015 `http://www.technologyreview.com/review/511116/windows-8-design-over-usability/`.

[45] B. Schwartz, "The Paradox of Choice: Why More is Less," Ecco New York, 2004.

[46] "Total number of Websites." Internet LiveStats, Retrieved on May 20, 2015 http://www.internetlivestats.com/total-number-of-websites/.

[47] D. Mac Síthigh, "More than words: the introduction of internationalised domain names and the reform of generic top-level domains at ICANN," *International Journal of Law and Information Technology*, vol. 18, no. 3, pp. 274–300, 2010.

[48] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2004.

[49] T. Grossman and R. Balakrishnan, "The design and evaluation of selection techniques for 3d volumetric displays," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pp. 3–12, ACM, 2006.

[50] J. Liang and M. Green, "JDCAD: A highly interactive 3D modeling system," *Computers & graphics*, vol. 18, no. 4, pp. 499–506, 1994.

[51] A. O. S. Feiner, "The flexible pointer: An interaction technique for selection in augmented and virtual reality," in *Conference supplement of ACM symposium on user interface software and technology*, pp. 81–82, 2003.

[52] R. Stoakley, M. J. Conway, and R. Pausch, "Virtual reality on a WIM: interactive worlds in miniature," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 265–272, ACM Press/Addison-Wesley Publishing Co., 1995.

[53] J. S. Pierce, B. C. Stearns, and R. Pausch, "Voodoo dolls: seamless interaction at multiple scales in virtual environments," in *Proceedings of the 1999 symposium on Interactive 3D graphics*, pp. 141–145, ACM, 1999.

[54] C. Hand, "A survey of 3-d input devices," *DMU CS TR94/2*, 1994.

[55] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell, "A survey of design issues in spatial input," in *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pp. 213–222, ACM, 1994.

[56] "USB Specification." USB Implementers Forum, Inc., Retrieved on May 20, 2015 http://www.usb.org/developers/docs/.

[57] "Bluetooth Core Specification." Bluetooth SIG, Inc. , Retrieved on May 20, 2015 https://www.bluetooth.org/en-us/specification/adopted-specifications.

[58] B. A. Myers, "A brief history of human-computer interaction technology," *interactions*, vol. 5, no. 2, pp. 44–54, 1998.

[59] H. Song, H. Benko, F. Guimbretiere, S. Izadi, X. Cao, and K. Hinckley, "Grips and gestures on a multi-touch pen," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1323–1332, ACM, 2011.

[60] E. Hoggan, J. Williamson, A. Oulasvirta, M. Nacenta, P. O. Kristensson, and A. Lehtiö, "Multi-touch rotation gestures: Performance and ergonomics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3047–3050, ACM, 2013.

[61] C. Preetham, G. Ramakrishnan, S. Kumar, A. Tamse, and N. Krishnapura, "Hand talk-implementation of a gesture recognizing glove," in *India Educators' Conference (TIIEC), 2013 Texas Instruments*, pp. 328–331, IEEE, 2013.

[62] M. Brown, A. Kehoe, J. Kirakowski, and I. Pitt, "Beyond the gamepad: HCI and game controller design and evaluation," in *Evaluating User Experience in Games*, pp. 209–219, Springer, 2010.

[63] T. P. Pham and Y.-L. Theng, "Game controllers for older adults: experimental study on gameplay experiences and preferences," in *Proceedings of the International Conference on the Foundations of Digital Games*, pp. 284–285, ACM, 2012.

[64] "Designing the xbox one controller." Gamesradar, August 12, 2013, Retrieved on May 20, 2015 `http://www.gamesradar.com/video-designing-xbox-one-controller/`.

[65] T. A. Mazzuchi, S. Sarkani, D. F. Rico, *et al.*, "Minimizing human factors mishaps in unmanned aircraft systems," *Ergonomics in Design: The Quarterly of Human Factors Applications*, vol. 21, no. 1, pp. 25–32, 2013.

[66] J. N. Sanders-Reed and P. L. Koon, "Vision systems for manned and robotic ground vehicles," in *SPIE Defense, Security, and Sensing*, pp. 76920I–76920I, International Society for Optics and Photonics, 2010.

[67] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, p. 93, IEEE Computer Society, 2003.

[68] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[69] J.-C. Junqua and J.-P. Haton, *Robustness in automatic speech recognition: Fundamentals and applications*, vol. 341. Springer Science & Business Media, 2012.

[70] A. Clark, C. Fox, and S. Lappin, *The handbook of computational linguistics and natural language processing*. John Wiley & Sons, 2013.

[71] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic n-grams as machine learning features for natural language processing," *Expert Systems with Applications*, vol. 41, no. 3, pp. 853–860, 2014.

[72] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3D human body tracking with an articulated 3D body model," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1686–1691, IEEE, 2006.

[73] "Remotte glass." Remotte Labs Inc., Retrieved on May 20, 2015 `http://remotte.com/remotte-glass/`.

[74] G. Kramer, *Auditory display: Sonification, audification, and auditory interfaces*. Perseus Publishing, 1993.

[75] M. Noisternig, A. Sontacchi, T. Musil, and R. Hóldrich, "A 3d ambisonic based binaural sound reproduction system," in *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*, Audio Engineering Society, 2003.

[76] S. Boyer, "A virtual failure: evaluating the success of nintendo's virtual boy," *The Velvet Light Trap*, no. 64, pp. 23–33, 2009.

[77] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: the design and implementation of the cave," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 135–142, ACM, 1993.

[78] P. Mistry and P. Maes, "Sixthsense: a wearable gestural interface," in *ACM SIGGRAPH ASIA 2009 Sketches*, p. 11, ACM, 2009.

[79] H. L. Pryor, T. A. Furness, and E. Viirre, "The virtual retinal display: A new display technology using scanned laser light," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 42, pp. 1570–1574, SAGE Publications, 1998.

[80] "Definition of user." Oxford University Press, Retrieved on May 20, 2015 `http://oxforddictionaries.com/definition/english/user`.

[81] J. Pruitt and J. Grudin, "Personas: practice and theory," in *Proceedings of the 2003 conference on Designing for user experiences*, pp. 1–15, ACM, 2003.

[82] B. Rind, "The power of the persona."

[83] A. Cooper and P. Saffo, *The inmates are running the asylum*, vol. 1. Sams, 2004.

[84] A. Maslow and A. Herzeberg, "Hierarchy of needs," *AH Maslow. ea., Motivation and Persona/ity. Harper, New York*, 1954.

[85] P. Milgram and H. Colquhoun, "A taxonomy of real and virtual world display integration," *Mixed reality: Merging real and virtual worlds*, pp. 5–30, 1999.

[86] J.-M. Normand, M. Servières, and G. Moreau, "A new typology of augmented reality applications," in *Proceedings of the 3rd Augmented Human International Conference*, p. 18, ACM, 2012.

[87] R. W. Lindeman and H. Noma, "A classification scheme for multi-sensory augmented reality," in *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pp. 175–178, ACM, 2007.

[88] W. E. Mackay, "Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers," in *Proceedings of the working conference on Advanced visual interfaces*, pp. 13–21, ACM, 1998.

[89] "Google Glass website." Retrieved on May 20, 2015 `http://www.google.com/glass/start/`.

[90] "Microsoft HoloLens Glass website." Retrieved on May 20, 2015 `https://www.microsoft.com/microsoft-hololens/`.

[91] M. Weiser, "Ubiquitous computing," *Computer*, vol. 26, no. 10, pp. 71–72, 1993.

[92] K. Ashton, "That 'Internet of Things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[93] E. Zelkha, B. Epstein, S. Birrell, and C. Dodsworth, "From devices to ambient intelligence," in *Digital living room conference*, vol. 6, 1998.

[94] M. R. Mine, J. van Baar, A. Grundhöfer, D. Rose, and B. Yang, "Projection-Based Augmented Reality in Disney Theme Parks.," *IEEE Computer*, vol. 45, no. 7, pp. 32–40, 2012.

[95] M. Inami, N. Kawakami, D. Sekiguchi, Y. Yanagida, T. Maeda, and S. Tachi, "Visuo-haptic display using head-mounted projector," in *Virtual Reality, 2000. Proceedings. IEEE*, pp. 233–240, IEEE, 2000.

[96] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, vol. 57. Elsevier, 2004.

[97] A. LaMarca and E. De Lara, "Location systems: An introduction to the technology behind location awareness," *Synthesis Lectures on Mobile and Pervasive Computing*, vol. 3, no. 1, pp. 1–122, 2008.

[98] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal, "GPS: location-tracking technology," *Computer*, vol. 35, no. 4, pp. 92–94, 2002.

[99] "Date and time format - ISO 8601." ISO Webpage, Retrieved on May 20, 2015
`http://www.iso.org/iso/home/standards/iso8601.htm`.

[100] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen, "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks," *Computer networks*, vol. 52, no. 11, pp. 2097–2128, 2008.

[101] "Uniform Resource Identifier (URI): Generic Syntax." IETF Webpage, Retrieved on May 20, 2015
`http://www.ietf.org/rfc/rfc3986.txt`.

[102] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," *Communications of the ACM*, vol. 46, no. 2, pp. 43–48, 2003.

[103] "Domain Names: Implementation and Specification." IETF Webpage, Retrieved on May 20, 2015
`http://tools.ietf.org/html/rfc1035`.

[104] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[105] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 299–314, 2002.

[106] Z. Antoniou and D. N. Kalofonos, "User-centered design of a secure p2p personal and social networking platform," in *Proceedings of the 3rd IASTED International Conference on Human Computer Interaction, HCI*, vol. 8, pp. 186–191, 2008.

[107] D. Otway and O. Rees, "Efficient and timely mutual authentication," *ACM SIGOPS Operating Systems Review*, vol. 21, no. 1, pp. 8–10, 1987.

[108] J. A. Clark and J. L. Jacob, "A survey of authentication protocol literature: Version 1.0," 1997.

[109] R. Khare and A. Rifkin, "Weaving a web of trust," *World Wide Web Journal*, vol. 2, no. 3, pp. 77–112, 1997.

[110] P. Dourish, "The parting of the ways: Divergence, data management and collaborative work," in *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work ECSCW'95*, pp. 215–230, Springer, 1995.

[111] C. Sun and C. Ellis, "Operational transformation in real-time group editors: issues, algorithms, and achievements," in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pp. 59–68, ACM, 1998.

[112] "RTP: A Transport Protocol for Real-Time Applications." IETF Webpage, Retrieved on May 20, 2015 `http://tools.ietf.org/html/rfc3550`.

[113] J. A. Halderman and V. Teague, "The new south wales ivote system: Security failures and verification flaws in a live online election," *arXiv preprint arXiv:1504.05646*, 2015.

[114] F. Lai, D. Li, and C.-T. Hsieh, "Fighting identity theft: The coping perspective," *Decision Support Systems*, vol. 52, no. 2, pp. 353–363, 2012.

[115] I. Ng, R. Maull, G. Parry, J. Crowcroft, K. Scharf, T. Rodden, and C. Speed, "Making value creating context visible for new economic and business models: Home hub-of-all-things (hat) as platform for multi-sided market powered by internet-of-things," in *Hawaii International Conference on Systems Science (HICSS), Hawaii, USA*, 2013.

[116] H. M. El-Bakry and N. Mastorakis, "Studying the efficiency of xml web services for real-time applications," *Transport*, vol. 7, p. 8, 2009.

[117] M. Lechner, "ARML Augmented Reality Markup Language," *Ginzkeyplatz*, vol. 11, p. 5020, 2010.

[118] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: an introduction to soap, wsdl, and uddi," *IEEE Internet computing*, vol. 6, no. 2, pp. 86–93, 2002.

[119] C. Hand, "A survey of 3d interaction techniques," in *Computer graphics forum*, pp. 269–281, Wiley Online Library, 1997.

[120] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[121] F. Daiber, L. Li, and A. Krüger, "Designing gestures for mobile 3d gaming," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, p. 3, ACM, 2012.

[122] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 249–256, ACM, 1990.

[123] J. D. Gould and C. Lewis, "Designing for usability: key principles and what designers think," *Communications of the ACM*, vol. 28, no. 3, pp. 300–311, 1985.

[124] J. Nielsen, "Coordinating user interfaces for consistency," *ACM SIGCHI Bulletin*, vol. 20, no. 3, pp. 63–65, 1989.

[125] O. Demirbilek and B. Sener, "Product design, semantics and emotional response," *Ergonomics*, vol. 46, no. 13-14, pp. 1346–1360, 2003.

[126] G. de España, "Ley oficial de protección de datos (lopd)," *Ley orgánica*, vol. 15, 1999.

[127] W. Albert and T. Tullis, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[128] J. R. Lewis, "Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq," *ACM SIGCHI Bulletin*, vol. 23, no. 1, pp. 78–81, 1991.

[129] C. F. Manski, "Measuring expectations," *Econometrica*, vol. 72, no. 5, pp. 1329–1376, 2004.

[130] J. Brooke, "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[131] J. R. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human-Computer Interaction*, vol. 7, no. 1, pp. 57–78, 1995.

[132] K. Norman, B. Shneiderman, B. Harper, and L. Slaughter, "Questionnaire for User-Interface Satisfaction," 1998.

[133] N. Mantel, "Chi-square tests with one degree of freedom; extensions of the mantel-haenszel procedure," *Journal of the American Statistical Association*, vol. 58, no. 303, pp. 690–700, 1963.

[134] W. S. Cleveland *et al.*, *The elements of graphing data*. Wadsworth Advanced Books and Software Monterey, CA, 1985.

[135] S. Few, *Show me the numbers: Designing tables and graphs to enlighten*, vol. 1. Analytics Press Oakland, CA, 2004.

[136] "Android UI Style Guide." Google, Retrieved on May 20, 2015 `http://developer.android.com/design/style/index.html`.

[137] "ISMAR 2012, Doctoral Consortium." Retrieved on May 20, 2015 `http://ismar2013.vgtc.org/ismar/2012/info/overview/doctoral-consortium`.

[138] "New Opportunities and Challenges."
Eleventh AR Standards Community Meeting (Session 10).
http://www.perey.com/ARStandards/september-2014-ar-community-agenda/.

[139] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[140] O. Merhi, E. Faugloire, M. Flanagan, and T. A. Stoffregen, "Motion sickness, console video games, and head-mounted displays," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 49, no. 5, pp. 920–934, 2007.

[141] J. F. Golding, "Motion sickness susceptibility," *Autonomic Neuroscience*, vol. 129, no. 1, pp. 67–76, 2006.

[142] C. T. Tan and D. Soh, "Augmented reality games: A review," *Proceedings of Gameon-Arabia, Eurosis*, 2010.

[143] J. Sauro, "Measuring usability with the system usability scale (sus)."

[144] E. Fennema and J. Sherman, "Sex-related differences in mathematics achievement, spatial visualization and affective factors," *American educational research journal*, vol. 14, no. 1, pp. 51–71, 1977.

[145] L. F. Jacobs, S. Gaulin, D. F. Sherry, and G. E. Hoffman, "Evolution of spatial cognition: sex-specific patterns of spatial behavior predict hippocampal size.," *Proceedings of the National Academy of Sciences*, vol. 87, no. 16, pp. 6349–6352, 1990.

[146] Z. Bauman, D. Bigo, P. Esteves, E. Guild, V. Jabri, D. Lyon, and R. Walker, "After Snowden: Rethinking the Impact of Surveillance," *International Political Sociology*, vol. 8, no. 2, pp. 121–144, 2014.