

Louvain School of Management

A hybrid notation for communicating system gestures to stakeholders

Author: Jean Germeau

Supervisor: Jean Vanderdonckt

Academic year 2021-2022

Master's thesis in order to obtain
a master's degree in business engineering

Daytime schedule

Abstract:

In the field of human-computer interaction, the development of applications allowing users to interact with any part of the body requires heterogeneous development teams to cover all the areas of expertise necessary for the success of such applications. Therefore, a common language to define and document the interaction gestures is necessary so that all stakeholders with different backgrounds can understand each other. This is particularly the case between designers and engineers who use different tools and working methods to perform their tasks while their collaboration is of crucial importance for the success of the application. In this work, we propose a new method for representing spatial gestures to facilitate communication around them. To do so, we have reviewed the state of the art of existing taxonomies and practices for representing gestures. Based on this, our representation allows to present dynamic gestures with GIFs that show the gesture modeled with a humanoid avatar. Moreover, movement properties that cannot be included in the graphical representation can be specified in textual sections.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management

Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique
www.uclouvain.be/lsm

Acknowledgement

First, I would like to thank my thesis supervisor for his involvement and his help throughout my work. Indeed, Mr. Vanderdonckt always made sure to quickly answer my questions, to advise me on certain references and to give me more general advice that helped me to complete this thesis.

Then, I would like to thank my relatives who always supported me in the writing of my thesis which was not always easy for me to achieve. Their love and support have been invaluable in helping me to finish this work. Among them, I think especially of my girlfriend, my mother, my brother, and my sisters. And finally, I would like to thank my close friends who have also given me their moral support throughout this experience.

Jean Germeau

Louvain-la-Neuve, August 2022

Table of contents

Chapter 1 - Introduction.....	1
1.1 Introduction to key concepts.....	1
1.1.1 Human-computer Interaction (HCI).....	1
1.1.2 User Experience (UX).....	2
1.1.3 Information Systems (IS).....	4
1.1.4 Gestural interaction.....	6
1.2 Identification of the problem.....	9
1.2.1 Background.....	9
1.2.2 Paradigm shift for user interfaces.....	10
1.2.3 Software engineering cycle: issues around communication.....	12
1.2.4 Collaboration between designers and engineers.....	13
1.2.5 Gesture description notations.....	14
1.3 Objectives & Research question.....	16
1.4 Research methodology.....	17
Chapter 2 - State of the art.....	19
2.1 Existing gesture interaction devices.....	19
2.1.1 Wearable sensor-based devices.....	19
2.1.2 Touch devices.....	21
2.1.3 Computer-based vision interaction devices.....	21
2.2 Taxonomy for gesture representation.....	23
2.2.1 McAweeney's taxonomy.....	23
2.2.2 Bernsen's taxonomy.....	28
2.2.3 Xia et al.'s factors for iteratively designing gesture vocabularies.....	30
2.3 Existing representations for gesture definition.....	37
2.3.1 Template-based gesture description.....	37
2.3.2 Textual gesture description.....	38
2.3.3 Graphical gesture description.....	44
2.3.4 Hybrid representation.....	51
2.3.5 GestureCards.....	52
2.3.6 Discussion.....	57
Chapter 3 - Development of new representation for spatial gestures.....	60
3.1 Requirements and objectives.....	60
3.2 Conciseness and readability.....	61
3.3 GestureCards extension for spatial gestures (3D).....	62
3.3.1 Representation for static poses by Hesenius.....	62

3.3.2	Representation for dynamic gestures by Hesenius.....	64
3.4	Our proposition for spatial gesture description.....	65
3.4.1	Choice for the modeling software	65
3.4.2	Representing gestures with Magic Poser within GestureCards	67
3.4.3	Integrating Xia et al.'s factors in GestureCards	70
3.5	Cost to create a card with our representation	71
3.5.1	Step 1: Modeling in Magic Poser.....	71
3.5.2	Step 2: Remove the background of the image	73
3.5.3	Step 3: Creating the GIF	73
3.5.4	Step 4: Creating the gesturecard	74
3.5.5	Total cost.....	76
Chapter 4	- Studies to test the representation.....	77
4.1	Study 1: Evaluation of readability	77
4.1.1	Study design.....	77
4.1.2	Results.....	79
4.1.3	Discussion	81
4.2	Study 2: Self-evaluating the representation with the Cognitive Dimensions framework	82
4.2.1	The 6 dimensions definition.....	82
4.2.2	Evaluation of the 6 dimensions for our representation	83
Chapter 5	- Conclusion	86
5.1	Reminder of the research question.....	86
5.2	Summary of the contributions.....	86
5.3	Benefits and shortcomings	87
5.3.1	Benefits	87
5.3.2	Shortcomings	89
5.4	Future Work	89
5.4.1	Short-term	89
5.4.2	Mid-term	90
References		91

Chapter 1 - Introduction

1.1 Introduction to key concepts

1.1.1 Human-computer Interaction (HCI)

The concept of *Human-Computer Interaction* quickly appeared with the emergence of computers or more generally of machines. Karray et al. (2008) explain that the need to develop technologies in this area is clear: "*most sophisticated machines are worthless unless they can be used properly by men*". Two main themes emerge from this need that must be considered in the design of HCI: *Usability* and *functionality* (Te'eni et al., 2007). These two principles are closely related. Indeed, functionality is defined by the actions and/or services provided by the application or machine to its users. However, it is necessary that the functionalities are usable by its users. Usability is therefore the degree to which users can exploit the functionality provided in order to reach a certain goal (Shneiderman, 1997). To be effective, a system must respect a balance between functionality and usability (J. Nielsen, 1994). The HCI is therefore the domain that allows to design the way users will interact with the computer in order to be able to exploit all the functionalities of the system as efficiently as possible.

The best-known application of HCI is the use of computers where users can interact with the user interface (UI) through a mouse and keyboard. This way of interacting is very suitable for UIs that are based on the WIMP paradigm, short for Windows, Icons, Menus and Pointers, where both point and click action are sufficient to navigate in the UI (Suarez Fernandez et al., 2016). This method of interaction is very common and almost everyone can navigate on a computer, but the new technologies allow to switch in the new paradigm of gestural control which allows us to free ourselves from our keyboards and mice to interact more naturally with the information systems that surround us (Jacob et al., 2008). The advent of this field does not aim to completely remove UIs based on WIMP but will allow to better respond to certain user needs and to offer solutions more adapted to certain situations in order to improve the user experience (UX), whose concept is defined in section 1.1.2 (van Dam, 1997).

Saffer (2009) explains that there are still applications for which the classic interaction with a mouse and a keyboard is best suited than gestural command. This is the case, for example, for applications that require the input of large amounts of data in the field of classic desktop applications as we know them today. While for other applications, especially on mobile

devices, gesture control can have advantages compared to WIMP interfaces. Bragdon et al. (2011) point out for example that this is the case for situations where users are distracted and would be performing multiple tasks at the same time. However, like any innovation, we can assume that the adoption of gestural control is also subject to the *Diffusion of Innovation* theory, established by Everett Rogers that segments potential users into five groups based on the time it takes for each of them to adopt a new technological innovation. Illustrated in Figure 1, this theory suggests that the majority of people take a long time to accept a new technology and adopt it (Kaminski, 2011).

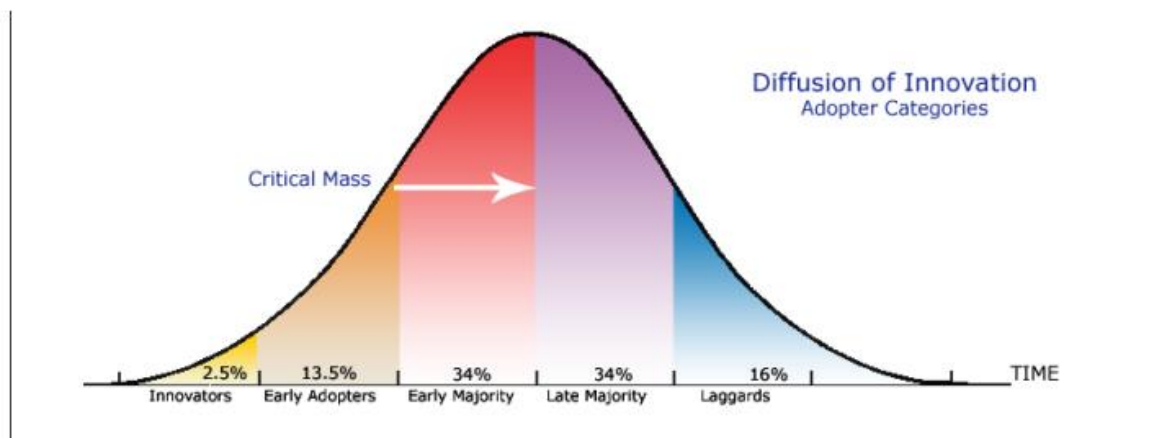


Figure 1: Rogers' diffusion of innovation theory (Kaminski, 2011).

To summarize the important points, it should be noted that the field of HCI, and more particularly based on gesture control, is in full development with the new technologies emerging. The goal is to offer users the best possible experience by integrating computer interactions in a natural way in their daily life by offering the best interaction method according to the situation and the purpose of it (Wigdor & Wixon, 2011).

1.1.2 User Experience (UX)

We have talked about what it means to develop user interfaces, but now we need to talk about an important point that plays a crucial role in the success or failure of an application: the *user experience* (UX). This concept was introduced for the first time in 1995 by Norman et al. (1995). However, there is no general consensus to define exactly what UX is, but the idea that the experience of an application is directly influenced by the way the interaction takes place has largely taken hold (Law et al., 2008). While this concept is now embraced by research and industry, many find attempts at definition too vague and elusive (Hassenzahl & Tractinsky,

2006). However, it seems to be accepted that a good UX is of decisive importance for the success and acceptance of an application.

In their work, Berni & Borgianni (2021) have tried to define what UX is based on the literature that, until now, could not agree on a clear definition. They propose a framework that is divided into two sub-categories, each built on three key concepts. The two sub-categories are the fundamental elements of interaction and the typologies of experience. The fundamental elements of interaction are divided into three main pillars:

- The user: this point refers to everything that concerns the person involved and their inner state.
- The system: it refers to everything related to the system and the product used to interact. Authors include everything that is provided by the system such as technologies involved, interactive products, services, and interfaces.
- Context of use: this concept can be influenced by several variables that must be taken into account: space and time, the physical environment or space, physical instrumental context (e.g., the technology involved, method of interaction or visualization) and the social and cultural context.

The second sub-category, identified by Berni & Borgianni (2021), which covers the different types of experience that the user may have includes three categories:

- Ergonomic experience: It corresponds to what appeals to usability, that is, is it simple to use, to take in hand and are the interaction tools effective?
- Cognitive experience: this has to do with the user's perception of the exterior and aesthetics of the system.
- Emotional experience: this refers to all the emotional components, i.e., everything that can make the use of the system pleasant from a hedonic point of view.

Berni & Borgianni (2021) point out that these different aspects of improving the UX are not independent. In fact, they are rather interdependent because if we play on one of these variables, it could impact the user's experience on another aspect (Hassenzahl & Tractinsky, 2006). As said before, this framework still doesn't give a clear definition of what UX is, but it allows to understand the stakes behind this concept. Knowing the variables allows designers and engineers to take them all into account to provide the best experience to potential users. The

HCI must therefore provide more than just a functional user interface (Berni & Borgianni, 2021).

Berni & Borgianni (2021) emphasize that user-application interaction is an essential part of UX and this necessarily implies the importance of gestural control in system design. Control via gestures gives more possibilities and freedom in the way people can interact with the application. For example, it is possible to vary the experience by allowing multiple fingers to interact or to allow variations in speed or pressure in the movement. Sometimes it is necessary to leave several possibilities in order to make the application more flexible according to the gesture choices of the user (Hinrichs & Carpendale, 2011).

The importance of UX has led to the development of new working methods and the development processes work with an iterative procedure, allowing to test and modify the application as it is being developed. The involvement of potential users is also very important and allows to validate the iterations (Abelein et al., 2013). Given the importance of UX in the success of an application, the results of early stages are also relevant for the rest of the project. It is therefore necessary to specifically document the interaction methods for the later stages of the project. However, Hartson & Pyla (2012) indicate that the information exchanged between UX designers and engineers is limited to use cases and UI sketches without providing any details about the interaction.

It seems clear that the way users interact with application systems plays a key role in the success and acceptance of new applications, and more specifically we emphasize the importance of interaction gestures that greatly contribute to improving the UX. The importance of communication within project teams around the documentation of interaction methods will be highlighted in section 1.2 as well as the different tools that facilitate this communication between members of an interdisciplinary project team.

1.1.3 *Information Systems (IS)*

As this work focuses on human-machine interaction, it seems important to us to specify what machines are. In this case, we speak about *information systems* (IS) with which the user interacts. To define what an *information system* is, we take the definition made by Bourgeois et al. (2014). This definition takes place in their book which is intended for the business field, but we consider it a good basis to understand what IS are. According to the researchers, the

definitions given in the literature agree on the fact that there are two axes of definition: the components that make up an information system and the role they play in an organization.

Bourgeois et al. (2014) explain that the IS consists of five components which can be divided into two categories. The first one is technology, which includes hardware, software and data. Each of these components is defined as follows:

- Information systems **hardware** includes the tangible, physical components that make up the technology. For example, in computers we find the processor, the hard disks or the motherboard (Bourgeois et al., 2014). If we could compare the components to a human body, the hardware would be the organs that allow the body to be physically functional.
- The **software** is a program or a set of programs that give a set of instructions to the hardware. Without the software, the hardware would not be functional and vice versa. This component is itself divided into two categories: operating systems and application software. The first one creates the interface between the user and the hardware and the second one allows the users to execute specific tasks. Figure 2 shows the flow of information from the hardware to the user and the reverse path because information circulates in both directions with the notion of input, which the user sends to the IS, and output, which is the information that the IS sends back (Bourgeois et al., 2014). To continue the comparison with the human body, we could say that the software is the brain that controls the functionalities provided by our different organs.



Figure 2: Information flow from hardware to user (Bourgeois et al., 2014)

- **Data** is described by the authors as the fuel that powers the IS. Data in its raw state is not useful, it needs to be transformed, stored, organized and processed to become real usable information (Bourgeois et al., 2014). Compared to the human body, data are the

nerve influxes that allow the transfer of information between the brain and the different parts of the body.

The authors insist that these first three components are complementary and that they form a whole that allows the best use of the information provided by the data.

Bourgeois et al. (2014) describes the last two components which are people and process as what separate the idea of IS from more specific domains like computer science. They are defined as follows:

- **People** simply refers to all the actors involved in an IS. All the roles are important for an IS to be effective, from the programmer, to the system analyst, to the help-desk worker. Each actor has a role to play and none should be neglected.
- The **process** is defined as a series of steps followed to reach an outcome or a goal.

The second part of the definition given by Bourgeois et al. (2014), which focuses on the role of IS in organizations, seems to us to be outside the scope of this work, so we will not dwell on it. The important thing for this section is to understand how an information system works with the notion of input and output. As a reminder, the input is the information that the user will give to the system and the output is the information that the IS will provide in response to the input received. And if everything works correctly the output received matches the input given by the user.

1.1.4 *Gestural interaction*

In this section, we will define what is meant by *gestural interaction*. In section 1.1.1 we have already discussed this topic, but we would like to add some clarifications and examples of use cases.

The most common method of interaction with an information system is to use a keyboard and a mouse to navigate through user interfaces with simple inputs that are mostly point and click. For a few years now, with the advancement of new technologies, a new interaction method has been developed that allows interaction with UIs via gestural commands.

The most widespread and already well adopted by the general public is the interaction with our smartphones and touchscreens in general. Indeed, in addition to simple tapping gestures, which are considered as point and click actions, gesture control allows us to use other functionalities. In this category, GestureML registers more than 300 realizable touch gestures, several examples are shown in Figure 3: A) By placing two fingers on an object and separating them, it allows to enlarge the targeted object. B) By placing two fingers on an object and turning them at the same time, it allows to rotate the targeted object. C) When drawing a path on an object, if a corresponding function is recognized by the IS, the corresponding event is returned (*Touch Gesture Index* | *GestureML Wiki*, n.d.).

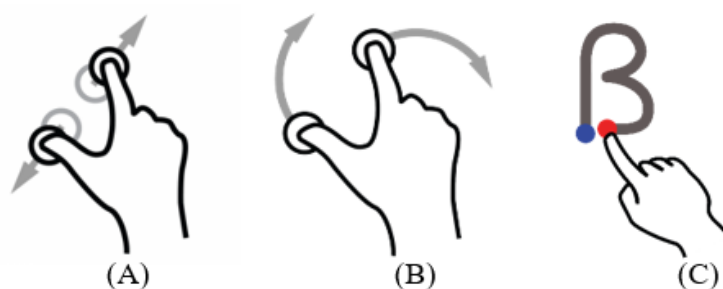


Figure 3: Examples of touch gestures. A) 2-Finger-Scale, B) 2-Finger-Rotate, C) 1-Finger-Stroke-Letter (*Touch Gesture Index* | *GestureML Wiki*, s. d.).

Another interaction method based on gestural command identified by GestureML are the 3D motion gestures. They are made possible by technologies such as motion tracking cameras or sensors positioned on different parts of the body (more details about devices in chapter 2.1). For example, in the study "*Designing Mid-Air TV Gestures for Blind People Using User- and Choice-Based Elicitation Approaches*" Dim et al. (2016) proposes the gestures in Figure 4 that would replace the use of a TV remote for blind people. The mid-air gestures involving the use of the hands are the ones that have been the most studied and the most elaborated, probably because the hands give more flexibility and possibilities in the gestures performed (Xia et al., 2022).

However, the researchers do not limit themselves to the hands and have looked, for example, at gestures made with the feet. This is the case for Velloso et al. (2015), who have identified different possible gestures with feet. Some examples are shown in Figure 5. The researchers explain that they are used as a primary modality for input when the user's hands are busy or dirty, or when the user has a disability or other accessibility issue that would prevent them from

using their hands. But most often the interaction gestures with the feet are used as a secondary modality to support tasks performed with the hands (Velloso et al., 2015).



Figure 4: Gesture set for blind people to control TV (Dim et al., 2016)

Gesture	Name	Description
	Toe tap	User raises and lowers the toes
	Heel tap	User raises and lowers the heel
	Toe rotation	User pivots the foot around the toes
	Heel rotation	User pivots the foot around the heel
	Toe click	User touches both toes together
	Heel click	User touches both heels together

Figure 5: Example of gestures with feet (Velloso et al., 2015).

There are other areas of application such as video games where the user experience is enhanced by virtual reality which allows the user to be virtually immersed in the user interface. Gesture control is then a very suitable tool to improve this feeling of immersion of the gamer. As

explained in point 1.1.1, the objective remains to improve the user experience by naturally including gesture control in our daily life in order to facilitate the interaction with information systems by adapting it to the situation and to the desired goal (Wigdor & Wixon, 2011).

1.2 Identification of the problem

In this section, we will present the problem that is at the basis of this work. This one has already been presented by Marc Hesenius in his doctoral thesis in 2019, but we want to bring more answers to this identified problem which seems to us to be still present during the development of software and/or application. In the following, we will therefore present the key points that will lead us to our research question addressed in this work.

1.2.1 Background

The starting point is that the importance of user experience (UX) in the acceptance of an application coupled with the arrival of new technology in the field of human-computer interaction (HCI) bring two major changes in the development of interactive applications. On the one hand, the processing of gestural commands, essential to interactive applications, requires a higher level of technical expertise (Chesta et al., 2004). On the other hand, new roles and domains of activity are involved in the development of these applications, leading to more heterogeneous teams in terms of expertise (Satyabrata, 2021).

Gestural control being more complicated to process than the classical WIMP user interface interaction, it requires different decisions from the developers in charge. The sensors allowing the user to interact with the application as well as the algorithms that interpret the data must be chosen (Hesenius, 2019). This implies that the input data must also be defined and if the sensors are already defined, it must also be checked whether the type of interaction defined by the project designers is compatible with the data provided and the given resources.

In addition, Hesenius & Gruhn (2019) explain that the specialization of certain tasks and activities continues throughout the development process. This is particularly true for designers who define, from the very first phases of development, the basic concept of the application and the way users will interact with it. These decisions, which are made even before the actual implementation of the application, will have an influence on the entire development process, making them of crucial importance. This requires qualified people in the field of HCI design, with a specific formation, in order to offer the best UX and guarantee the success of the

developed application (Leiva et al., 2019). Thus, the already interdisciplinary field of software engineering is seeing the emergence of new actors that need to be integrated. In the rest of this section, we will see the impact of these evolutions on the software engineering field.

Hesenius (2019) breaks down his analysis of the problem into several points. The first point will show the differences in the interactions between a classical WIMP interface and a gesture-controlled interface and the consequences for software development (1.2.2). Afterwards, an analysis of the software engineering cycle is performed. It shows the tasks and responsibilities of each person and identifies a major problem in this field: the communication between stakeholders who have different expertise (1.2.3). The following point shows that it is important that all members of the development team participate in the early development phases of the application. It is also shown that defining the interaction method is important but in practice not well documented and therefore lacks precision in communication around it. (1.2.4). The last point shows that there is a lack of human readability-oriented notations that would favor communication around the specification of interaction gestures between the members of the development team (1.2.5).

1.2.2 Paradigm shift for user interfaces

Classical interfaces controlled only with a mouse and a keyboard, following the WIMP paradigm, are limited to simple and concrete commands (e.g., clicking on an icon, moving the cursor, or checking boxes). On the other hand, interfaces based on gestural interaction have many relevant properties that allow to precisely distinguish the different gestures, which in some cases can count up to hundreds of possibilities (Jacob et al., 2008). These properties must therefore be considered and analyzed in order to restore the right gesture in the application and apply the output action desired by the user. Several parameters can be taken into account in the interpretation of the gesture; for example, the execution speed of a swipe gesture can influence the speed at which an object moves on the screen. In addition, defined gestures may have similarities or users may be imprecise in executing the gesture, making the interpretation of the gesture by the computer more complicated and giving it some margin of interpretation (Hinrichs & Carpendale, 2011).

From these observations, Hesenius (2019) emphasizes four main points of difference between interfaces with gestural interaction and classic WIMP interfaces:

- Performed gestures provide a constant stream of data that must be continuously interpreted. This requires the constant collection and analysis of the incoming data in order to precisely identify the input gesture and to call the corresponding function. On the WIMP side, each input element has a corresponding function, so there is a 1-to-1 relationship between an event and its corresponding function. Instead of considering the moment of interaction, as is the case for WIMP user interfaces, gestural control requires considering a period of time.
- Gestural control also requires taking into account the spatial components of the interaction, i.e., the order, the distance or the overlapping of the different gestures and the possible relations between them.
- In a WIMP user interface, simple and predefined actions can be easily specified and implemented. In application with gestural control, other characteristics such as speed or location must be considered as a gesture could have different meaning depending on those properties.
- In interfaces with gestural control, the input data can be blurred. This may be due to users not always executing gestures in the same way or with the same precision. A margin of error must therefore be taken into account by the computer that interprets the gestures.

To overcome these differences, new technical approaches are necessary, although the appropriate choice of components and a good implementation can limit the difficulties with these new technologies. As a result, developers have learned to work with a new methodology by introducing agile and iterative development processes that integrate the steps necessary to develop a good UX (Larman & Basili, 2003). Moreover, with this new paradigm, it can be important to integrate the target user group in the early stages of the application development (Abelein et al., 2013). However, software engineering is an activity essentially carried out in team because it requires very different and very specialized knowledge. Therefore, the distribution of tasks must be coordinated, and information must flow as clearly as possible between the parties involved (Hesenius, 2019).

1.2.3 Software engineering cycle: issues around communication.

In Figure 6, Hesenius (2019) presents the different steps in the development of an application that uses gestural control. In the first step, the goal is to collect the requirements of the application. To do so, they conduct interviews with expert users and document them mainly in natural language. The integration of these users, who are experts but not technicians in application development, brings a main problem which is to find a common language. Indeed, they must be able to understand the different scenarios proposed by the designers, who must therefore present the scenarios in a way that is understandable for all participants. The second step is the design of a user interface in order to define the look of the application but also the interaction concept which defines the way the user will interact with the application. With the requirements gathered and the user interface designs imagined, the software engineers can implement the application in the third step, and finally, the quality assurance managers can test the application in the final development step.

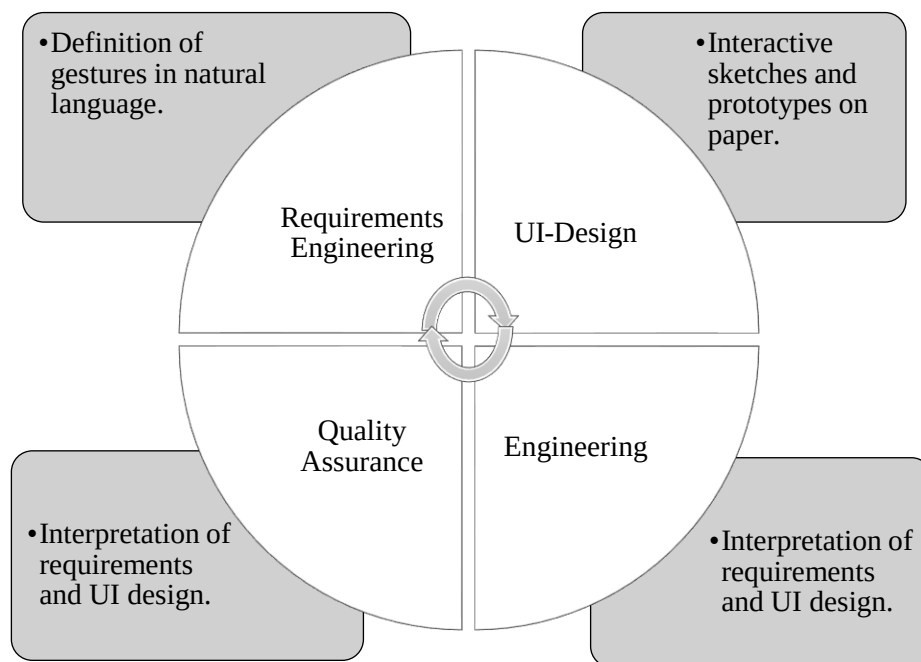


Figure 6: Gestures in Software Engineering cycle (Hesenius, 2019).

Decisions made in the early phases of development have a central role as there are so many possibilities for new interactions with new technologies. To summarize, the requirements captured with the potential users are used to sketch the different potential UIs which in turn serve as the basis for the decision on the components used during the implementation. The right sensors must be chosen to provide the data in a reliable way and the right algorithms must be chosen to analyze data and interpret the right gestures. Subsequent changes to these decisions

can have important consequences that would generate more work and costs due to the necessary adaptation (Hesenius & Gruhn, 2019).

We therefore see that at each stage of application development, the stakeholders use different tools and therefore different languages to communicate with each other. The software engineers must understand the requirements of the application and translate the definitions from the natural language to the implementation language, but they must also establish the link with the user interface projects and the prototypes developed (Leiva et al., 2019). And once they understand all this information, they can deduce the technical basis and implement the application trying to meet the expectations. Thus, we can say that different interests clash because of the different perspectives and working methods of the stakeholders in each phase of the development (Ferreira et al., 2011). Hesenius (2019) mainly identifies difficulties in the communication between the participants in the project, due to the translation of the different working tools from one step to another. At each transition from one development stage to the next ones, there is a margin of interpretation of the different interaction concepts defined, which can lead to misunderstandings and ultimately lead to application defects.

1.2.4 Collaboration between designers and engineers

In this point, we address a conflict between the working methods of designers, who work in the first two phases of application development, and engineers whose job is to implement the application. Of course, modern software engineering does not have a strict separation between the different phases but rather uses an agile method (Blandford, 2013). This method uses an iterative approach where the involvement of all team members is essential at each step. This allows each specialist to give his point of view and explain the repercussions that the decisions will have on his work at each stage. At the same time, the specialists remain responsible for the work related to their area of expertise. Buxton (2011) emphasizes the importance of all departments being involved in the early stages of development, i.e., the design process.

With the agile method, several prototypes are designed and only the best performing and the one that best meets expectations will be retained. It is this winning prototype that will define what the engineering team will have to implement (Greenberg et al., 2012). An essential aspect of the usability of an application is the interaction and it is precisely at this prototyping stage that the interaction method is defined. It is therefore on this basis that the engineers will have to select the sensors and algorithms allowing the chosen interaction method. The participation

of the engineers in these prototype development phases is therefore very important in view of the impact that these decisions will have on their future work.

We have shown that the collaboration between designer and engineer is very important, but we will now see that they have different methods and strategies of resolution. Wigdor & Wixon (2011) explain that software engineers prefer to work in a bottom-up way, i.e., they start from the possibilities given by each type of hardware and the users' capabilities as a basis for the application. Designers, on the other hand, are more likely to work top-down and consider the possible usage scenarios of the users and the context in which they will use the application. So, on one side we have engineers who rely on what they know and already exist, and on the other side designers generate new ideas by thinking about the best possible UX. There is no need to choose one method or the other, but you must know how to use both methods in order to produce new ideas and to stay within the limits of what is possible at the same time (Wigdor & Wixon, 2011).

As mentioned earlier, the chosen interaction method has an impact on the choice of hardware and software components. However, the experimental prototypes are mostly very unclear and imprecise regarding the interaction methods. The necessary information is defined implicitly and is not explicitly documented. Altakrouri & Schrader (2014) explain that the need to document interaction gestures is generally recognized by designers but is largely ignored in practice and perhaps due to lack of standards in the field. These imprecisions on the definitions of the interaction gestures can lead to a poorly performing final application.

1.2.5 Gesture description notations

There are several methods that can be used to describe gestures: textual descriptions, graphical representations and hybrid forms which is a mix of both former methods that allow more flexibility to meet individual needs. However, the notations used to describe interaction gestures are often too formalized because they are rather engineering oriented, i.e., we privilege the readability for the machines rather than for the humans. Shipman & Marshall (1999) show that this has a negative impact on the creative process and can hinder the design phases.

Low formalization documentation with simple rules is mostly used for communication between participants with very different technical knowledge. This is the case, for example, with User Stories, which allow to describe the different functionalities from the users' point of view without going into technical details (Cohn, 2004). The Gherkin Domain Specific Language

(DSL) also allows to create test cases with a pseudo-natural syntax following a defined structure with appropriate keywords. Frameworks based on Gherkin allow all project participants without specific knowledge to understand and formulate the created scenarios (Wynne & Hellesøy, 2012). However, this framework generally focuses on WIMP user interfaces and offers few possibilities to describe gestures.

Gesture description languages for interactive applications can serve different purposes. Dumas et al. (2010) classify these languages along the continuum shown in Figure 7. The more they are located on the right, the more they lose in formality to facilitate the communication between participants in the development. Towards the left, they allow to minimize the effort in the implementation by allowing to easily configure appropriate gestures recognition to the detriment of communication between stakeholders. The languages tending towards the left are more appropriate for engineers while those on the right of the continuum promote communication and are therefore readable by all. The current approaches tend strongly towards engineering and favor machine readability rather than human readability (Dumas et al., 2010). This again does not favor ease of communication around interaction methods and gesture descriptions.

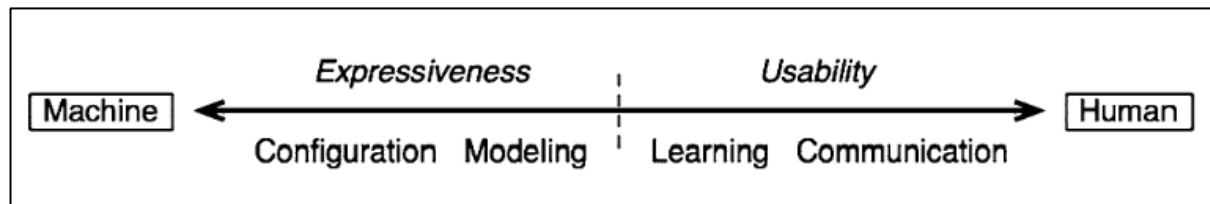


Figure 7: Different purposes for gesture description language (Dumas et al., 2010).

Hesenius (2019) shows that communication around gestural control is multiple and that different properties are relevant to keep in mind depending on the application implemented. And it is precisely these properties that need to be specified and defined as they are necessary for the subsequent discussion and processing of gestures. The more properties a gesture has, the more precisely it is necessary to describe it.

Existing approaches to gesture description generally focus on either purely graphical or textual methods (see section 2.3). However, choosing one or the other method means that one is either on the right or on the left side of the continuum presented in Figure 7, which is not preferable to facilitate both communication and implementation of the application.

1.3 Objectives & Research question

Our identified problem can be summarized as follows: in the development of applications based on gestural control, the choice of the interaction method, and thus of the gestures themselves, plays a crucial role in the subsequent decisions. Therefore, it is important that these gestures are precisely documented by designers on their execution and properties so that software engineers can correctly implement them in the application. This communication is not facilitated by the heterogeneous nature of the project participants. Indeed, they have different priorities and interests that are reflected in their working methods and tools. In addition, current solutions to better document the interaction methods are often strongly oriented towards engineering, which does not favor communication on gestures. These communication problems can result in ambiguities that lead to misinterpretations. As a result, misunderstandings arise that can turn into implementation errors or different expectations among team members, leading to frustration within the teams. In the worst case, this can lead to additional costs and work for the whole project.

In their research work, Hesenius & Gruhn (2019) developed a new hybrid notation, GestureCards, which documents gestures in a way that facilitates communication. We present it in detail in Chapter 2.3.5. However, this notation has been thought exclusively for multi-touch gestures, i.e., those that we most often perform with our fingers on a touch screen, like smartphones. Therefore, in a world where new technologies allow more and more interaction with our environment via gestural control, new applications are emerging whose development could need a gesture specification tool. We therefore want to extend this type of gesture notations to a larger set of gesture possibilities. We focus especially on spatial gestures that are performed with the whole body, or gestures that are limited to arms and hands. Based on that observation we formulate our central research question as follows:

What is the most appropriate representation to define and document spatial gestures controlling information systems in order to improve communication between application designer and development engineer?

We think that given the multiple ways of communicating with a computer via gestures, each one of them have different needs in terms of specification. For some, a simple textual description is enough while for others a good graphical representation can do the job. However, we have seen that a hybrid representation can take into account a larger number of gestures because it allows to combine the advantages of textual and graphical notations at the same time.

Therefore, we believe that this is the most suitable type of notation to take into account a maximum of interaction gestures. Our goal is to propose a new way of describing and documenting gestures in order to facilitate communication around them.

1.4 Research methodology

To conduct our research, we applied what we call a Narrative Literature Review. It is defined by Rumrill & Fitzgerald (2001) as a comprehensive, critical and objective analysis of existing knowledge on a specific topic. More precisely, we applied the methodology that Onwuegbuzie & Frels (2016) calls general literature review which aims to retain the most meaningful, critical and important aspects of the current knowledge of the research subject.

For our research, we browsed online libraries using databases such as Google Scholar, ResearchGate, IEEE Xplore, ACM Digital Library and RERO ILS. Our criteria for selecting the most relevant articles included the following: authors and their background, research objective, research methodology and publication date. The date of publication did not mean that we excluded any article published before a certain year, but it could in some cases direct us towards more recent research as our field of study is still young and knowledge is evolving rapidly.

This methodology has been applied for all the theoretical contributions that will be developed in this work. This is the case for the **first chapter** in which key concepts like Human-Computer Interaction (HCI), User Experience (UX), Information Systems (IS) and Gestural interaction are introduced. It has also been used to develop the problematic which is at the base of our work and has led to the elaboration of our research question.

Chapter 2 aims to review the state of the art on research topics that will serve as the basis for establishing a new gesture representation presented in **Chapter 3** of this work. The purpose of **section 2.1** is to present the technologies allowing gestural interaction. We considered this important in order to understand all possible interactions between humans and information systems. In **section 2.2**, we present the results of Usui et al. (2020)'s research work which analyzed Gesture Elicitation Studies through two taxonomies that classify gesture representations based on elements included in them. This allowed the researchers to identify the most used elements in these representations. In the same section we also presented Xia et al. (2022)'s factors that helps designers to take every aspect of gestures into account. These factors can be useful in describing gestures and will be used in our proposed new

representation. **Section 2.3** presents the already existing gesture representations. This is not an exhaustive list, but we have selected representations that share the same objectives as ours. This allowed us to identify the important aspects to include in our new representation.

In **chapter 3**, we develop our new representation of spatial gestures. To do so, we used the methodology of Design Science Research. Baskerville et al. (2009) explain that the basic principle of this research method is to formulate a hypothesis (called "a design"), perform an experiment (instantiating an artifact), and see if the results correspond to the expectations (evaluating).

The authors also explain that it is an iterative process, as shown in Figure 8. In our work, the "Search" is all the research where we performed the general literature review. The "Design" corresponds to the part where we imagine the representation and where we set the requirements, the "Ex Ante Evaluation" and the "Construction" are the phases in which we select the elements of our representation so that they meet the established requirements. All these processes lead to the "Artifact" which is our representation proposal. The last step "Ex Post Evaluation" corresponds to **chapter 4** in which we carry out a study to test the effectiveness of our representation.

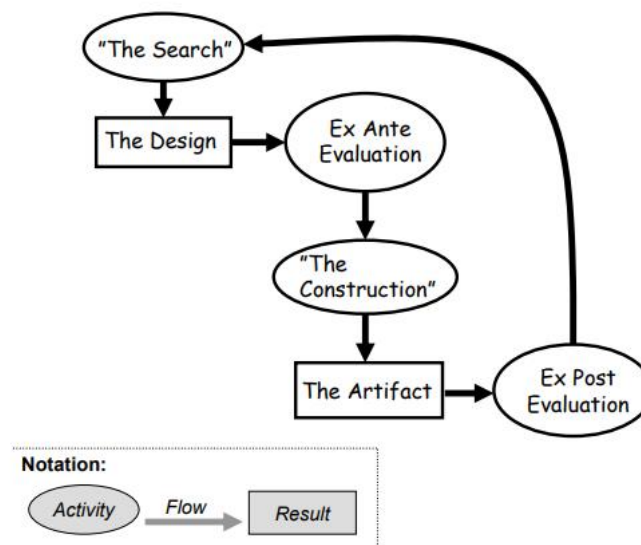


Figure 8: Iterative Design Science Research Method (Baskerville et al., 2009).

Chapter 5 concludes our work by highlighting the contributions made by this work, its benefits and shortcomings, and by making suggestions for future research in short and mid-terms.

Chapter 2 - State of the art

2.1 Existing gesture interaction devices

In this part, we will make a non-exhaustive inventory of the existing technologies allowing the human-computer interaction. Nowadays, everybody is familiar with the classic user interface, called WIMP for Window, Icon, Menu and Pointer, in which the user can interact with the interface through a mouse and a keyboard. However, in recent years, many technologies have seen a great development and new ones have been developed to offer new possibilities to the user. Li et al. (2019) have identified three different types of devices: wearable sensor-based devices, touch-based devices, and computer vision-based devices. Each of these categories includes several technologies that we will detail in this section.

2.1.1 Wearable sensor-based devices

In this category we find first the data gloves which can identify the posture and the movements of the hand in real time. To do this, the glove device is equipped with sensor networks that detect signals in each joint of the hand. It uses the coupling relationship between the data vector of each part of the hand, calculated through sensors, and the data vector of each bending angle of the hand to obtain the degree of inclination and extension of each part of the hand (J.-H. Kim et al., 2009). These data gloves are mainly used in the field of virtual reality. They can have two functions: the first one as an input device because it allows to translate in real time the real movement of the user into a virtual movement in the user interface. The second one is to give feedback to the user by making him feel physically a reaction, such as vibrations, to his actions in the user interface. For example, this last feature allows to increase the realism in video games (Li et al., 2019). The CyberTouch II data glove is shown in Figure 9.



Figure 9: CyberTouch II data glove. Retrieved from <http://www.cyberglovesystems.com/cybertouch>

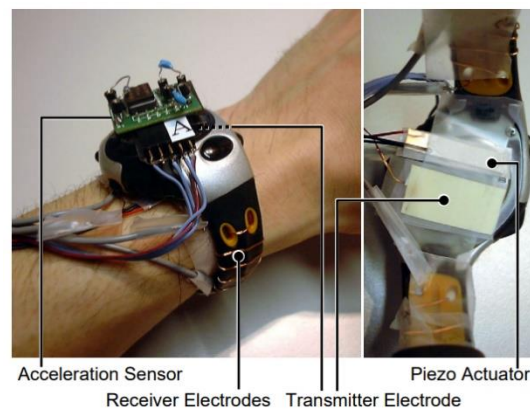


Figure 10: GestureWrist (Rekimoto, 2001).

The next technology is based on inertial sensor, which works with micro-electro-mechanical systems (MEMS). This intelligent system consists of three main parts: sensor, actuator, and micro energy, all three of which are used to capture information on gestures made by the user. Common MEMS systems include an accelerometer, a gyroscope sensor and a pressure sensor (Li et al., 2019). The GestureWrist, shown in Figure 10, invented by Rekimoto (2001), is a wrist-worn device that identifies hand movements through an acceleration sensor. Nintendo has also used this technology for its Nintendo Wii game console via its controller to translate the player's movements directly into the video game (Maenaka, 2008).



Figure 11: Myo Armband. Retrieved from <https://time.com/4173507/myo-armband-review/>

The third technology presented for wearable sensor-based devices uses electromyography. This technology identifies a user's movement via electrodes that calculate the bipotential changes in the myoelectric signal that is generated during the movement of the muscles (Li et al., 2019). To put it simply, it tracks electrical activity within the muscles and a change in the amplitude of these signals can detect movement (Mendez et al., 2017). For example, Thalmic Labs launched the Myo Armband in 2013, shown in Figure 11. It is a wristband with eight electromyography (EMG) sensors that collect muscle activity from the forearm. It translates this muscle activity into hand and arm movement. It is also equipped with a 9-axis inertial

measurement unit that calculates the movement, orientation and rotation of the forearm (Rechy-Ramirez et al., 2018).

2.1.2 *Touch devices*

Touch devices are very common nowadays, especially in the smartphone industry. Everybody is familiar with this technology and particularly with touch screens whose effectiveness is no longer to be proven as well as their ease of use. There are different types of touch screens: Resistive, capacitive, surface acoustic and infrared. We will not detail each of these technologies but the most widespread, especially for our smartphones is the capacitive touch screen as it is the most durable (Li et al., 2019). It consists of a layer of insulator coated with a transparent conductor. Since the human body is also a conductor, by touching these layers the electrostatic field of the screen changes which is measurable as a capacitance change. Thanks to these calculations, the device can locate the position of the contact on the screen (Baxter & Baxter, 1997). It is also possible to interact with touch screens using a pen. This technology is very used for jobs like graphic designers or draftsmen for which the precision is very important.

2.1.3 *Computer-based vision interaction devices*

This category of devices works on the tracking of different parts of the body with cameras. The principle is simple: The camera(s) can identify explicit human gestures as input and process these gesture representations for device control through mapping of commands as output (Rautaray & Agrawal, 2015). There are several types of devices capable of tracking different body parts like: Hand and fingers, eyes, head, full body or 3D objects (*3D Motion Gesture Index | GestureML Wiki*, n.d.).

As with sensor-based wearable devices, cameras allow to track hands and fingers in very precise ways with devices like Leap Motion or RealSense. Each of these methods has its advantages and disadvantages but we note that wearable devices have an advantage in precision for gesture recognition and are less complex to implement. On their side, vision based recognition systems are more user friendly (Rautaray & Agrawal, 2015). Hand movements are recognized through a hand model using six points of reference (palm point and finger tips) for the simplest models and there can be more than 22 points of reference for the most complex ones (*3D Motion Gesture Index | GestureML Wiki*, n.d.). In Figure 13, you can see the mapping

of the hands done with the Gemini algorithms used by Leap Motion and Figure 12 shows the controller used to capture the hand movements (*Gemini*, n.d.).



Figure 12: Leap Motion controller
retrieved from <https://www.ultraleap.com/>

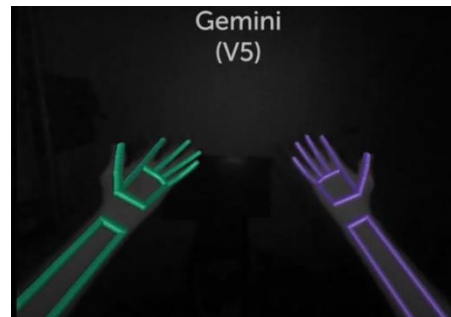


Figure 13: Hands mapping with Leap Motion's algorithm Gemini. Retrieved from <https://youtu.be/jKRZgI0T2vI>

For the tracking of head movements, the device, such as RealSense, uses depth-map based 3D motion tracking methods. For most cases, this consists of a central point between the eyes with a directional vector normal to the facial plane. This method identifies a set of simple and relatively robust head gestures (*3D Motion Gesture Index | GestureML Wiki*, n.d.). New technologies make it possible to track eye movements more and more accurately. This is especially used on screens in order, for example, to know which part of the screen the user is looking at. This data can be interesting in areas such as marketing to know what attracts the consumer's gaze (Wedel & Pieters, 2008). Software like Tobii or Eye Tribe can also distinguish blinks and winks (Li et al., 2019).

There are also devices that track the entire body such as the Microsoft Kinect controller. They allow to identify the whole body and to track it in a 3D space. To do this, it generates a skeleton of the body whose position, movements and poses of different parts of the body are tracked (Wang et al., 2020). Finally, devices like Intel's RealSense camera can isolate a medium-sized object in a 3D space. Once the object is known, its features can be recognized and tracked (*3D Motion Gesture Index | GestureML Wiki*, n.d.).

We have seen in this section that new technologies allow users to interact with computers in many ways with almost any part of the body. This underlines the need for gesture representation that can support all these new gestures. The next chapter provides a state of the art of existing interaction gesture notations.

2.2 Taxonomy for gesture representation

In this section we present a study that was conducted during the realization of a master thesis by Usui et al. (2020). The objective of their work was to implement a gesture recognition technology in a hospital environment. The study we are interested in is about how to represent gestures and what are the most common design elements to represent an interaction gesture. To do this, Usui et al. (2020) used a systematic review of Gesture Elicitation Studies (GES) conducted by Villarreal-Narvaez et al. (2020). GES could be defined as a way to work with end-users to better understand their preferences in order to know how to design gesture representations (Usui et al., 2020). This systematic review summarizes the characteristics and findings of 216 gesture elicitation studies that include a total of 148,340 elicited gestures. Then, these gestures were analyzed through two taxonomies that classify gesture representations according to several criteria. The first one, created by McAweeney et al. (2018), analyses the characteristics of gesture representation and the second one, made by Bernsen & Dybkjær (2010), focuses on the elements allowing the transfer of information. Based on these two taxonomies and with the 148,340 elicited gestures, Usui et al. (2020) analyzed which characteristics and elements included in the gesture representations are most often used.

2.2.1 McAweeney's taxonomy

This taxonomy created by McAweeney et al. (2018) is composed of 6 dimensions each containing categories (in total: 26 categories). The idea is that each gesture description belongs to a category for each dimension. Figure 14 presents each dimension and its corresponding categories.

The authors provide more specific definitions for each of the six dimensions:

- **The Perspective dimension** refers to the point of view used to represent the gesture. There are 5 categories for this one: a first-person viewpoint, a third person viewpoint, a side angle, a bird's eye, or a mirror representation.
- **The Frame dimension** simply explains if the representation of the gesture is made via one frame (single-framed) or several frames (multi-framed). Multiple frames could be used to describe different aspects of a gesture such as motion and time.
- **The Body Context dimension** specifies which part of the body is represented.
- **The Environmental Context dimension** expresses whether one or more objects are represented. These can be real or virtual and serve to put the gesture in its context of

use. For example, a touchscreen can be added in the representation of a multi-touch gesture.

- **The Color dimension** explains if colors are used to represent particular aspects of the gesture, this may include the use of hue and shade to detail certain aspects of it.
- **The Gesture Element dimension** precises what kind of graphical elements are added to the gesture representation in order to make it more understandable.

Dimension	Category	Dimension	Category
Perspective	1 st Person	Color	Yes
	3 rd Person		No
	Mirror	Gesture Elements	1-Sided Arrow
	Bird's eye		2-Sided Arrow
	Side Angle		Dotted Lines
Frame	Single		Ghost
	Multi		Finger Trail
Body Context	Full-Body		Other Motion Lines
	Lower-Body		Touchpoints
	Upper-Body		Numbers
	Other Body Parts		Text
Environmental Context	Physical Objects		Bending Joints
	Virtual Objects		Axis

Figure 14: McAweeney et al's Taxonomy (McAweeney et al., 2018).

Usuiel et al. (2020) point out that this taxonomy has advantages and disadvantages. The advantage is that it allows to quickly analyze a gesture and to highlight its main characteristics. The disadvantage is that it is limited to its 6 dimensions whereas other aspects of gestures could be taken into consideration to analyze a gesture representation. They propose for example a dimension that would describe the form of the gesture: dynamic or static. Moreover, this taxonomy does not allow to clearly differentiate between 2D and 3D gestures, or to distinguish between the possible inputs for gestures, i.e. touch, motion or sensor inputs (Usuiel et al., 2020).

Figure 15 shows examples, used by McAweeney & al. (2018), of gestures analyzed through the presented taxonomy:

- In gesture representation A, there is a first-person perspective, it is multi-framed, it uses physical objects, it uses color changes to differentiate the faces of the object, there are gesture elements like ghosts and touchpoints.
- In gesture representation B, it is a third-person perspective, it is single-framed, there are physical objects added, there is color to represent the touch surface, it uses gesture elements like 1-sided arrows, dotted lines, and finger trails.
- In gesture representation C, it is a third person perspective, it is single-framed, it focuses on the upper body, there is a physical object, it uses colors to differentiate the elements, it uses gesture elements like motion lines and numbers.
- In gesture representation D, it uses a side angle, it is single-framed, there is a virtual object, it uses different tints, there are gesture elements like 2-sided arrows, dotted lines, ghost and axis.

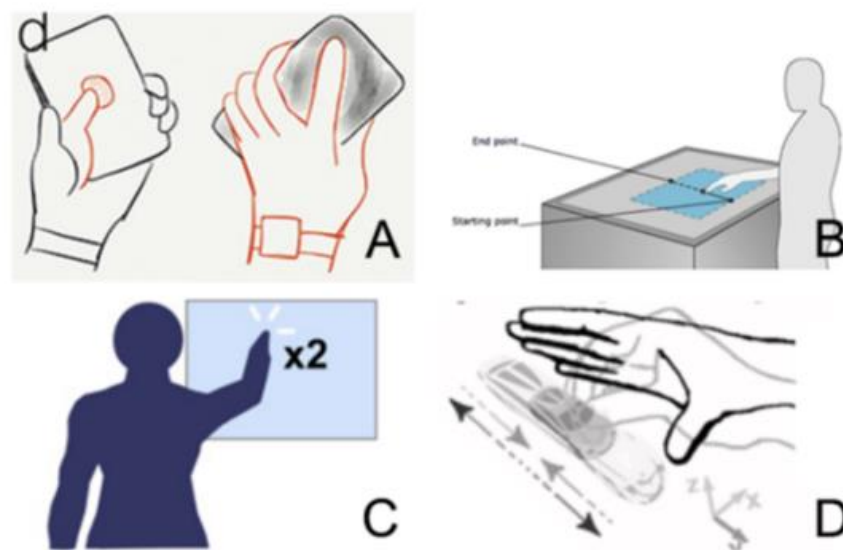


Figure 15: Examples of gesture representations (McAweeney et al., 2018).

To determine which elements were most often used in gesture representations, Usui et al. (2020) applied that taxonomy to the 216 GES identified by McAweeney & al. (2018) and to 45 additional studies for a total of 261 studies. Their results are shown in Figure 16. Concerning the perspectives used, the graph (a) shows that the first-person perspective is used in 32% of the representations and the third person in 23% of the cases. A majority (65%) of the representations are done in a single frame (graph (b)). For the Body Context dimension, most of the representations are made with an illustration of the hands, 57% of the cases to be precise. In 69% of the representations, physical objects were included to, most often, specify the

medium of interaction (e.g., smartphones, smartwatch, etc.). In 65% of the representations, they use color changes to differentiate the elements that are in them. For gesture elements, the ones that come up the most are: Arrows (17%), Finger Trail (16%) and Text (14%).

To summarize, if we were to create a perfect representation according to the results of this study, it would have the following characteristics for each McAweeney & al. (2018) dimension:

- It would use a first-person or third person perspective.
- The representation would be done in one frame.
- The body context would focus on hands depiction.
- For a better contextualization of the representation, physical objects should be added.
- The representation should use colors to emphasis elements and differentiate them.
- Arrows, finger trails and texts should be added to clarify the representation.

These results show which graphic elements were most often used to represent interaction gestures. Although very interesting, care must be taken in interpreting these results. Indeed, this study only analyses the frequency of appearance of graphic elements in the 261 papers analyzed (Usui et al., 2020). For example, these 261 GES could have been divided into groups based on the type of gestures and interaction involved in their study. The underlying question is whether, for example, the representation of mid-air gestures performed in front of a screen requires the same degree of precision as for gestures performed on a smartphone screen with our fingers? Our intuition is that the results could be different from one group to another.

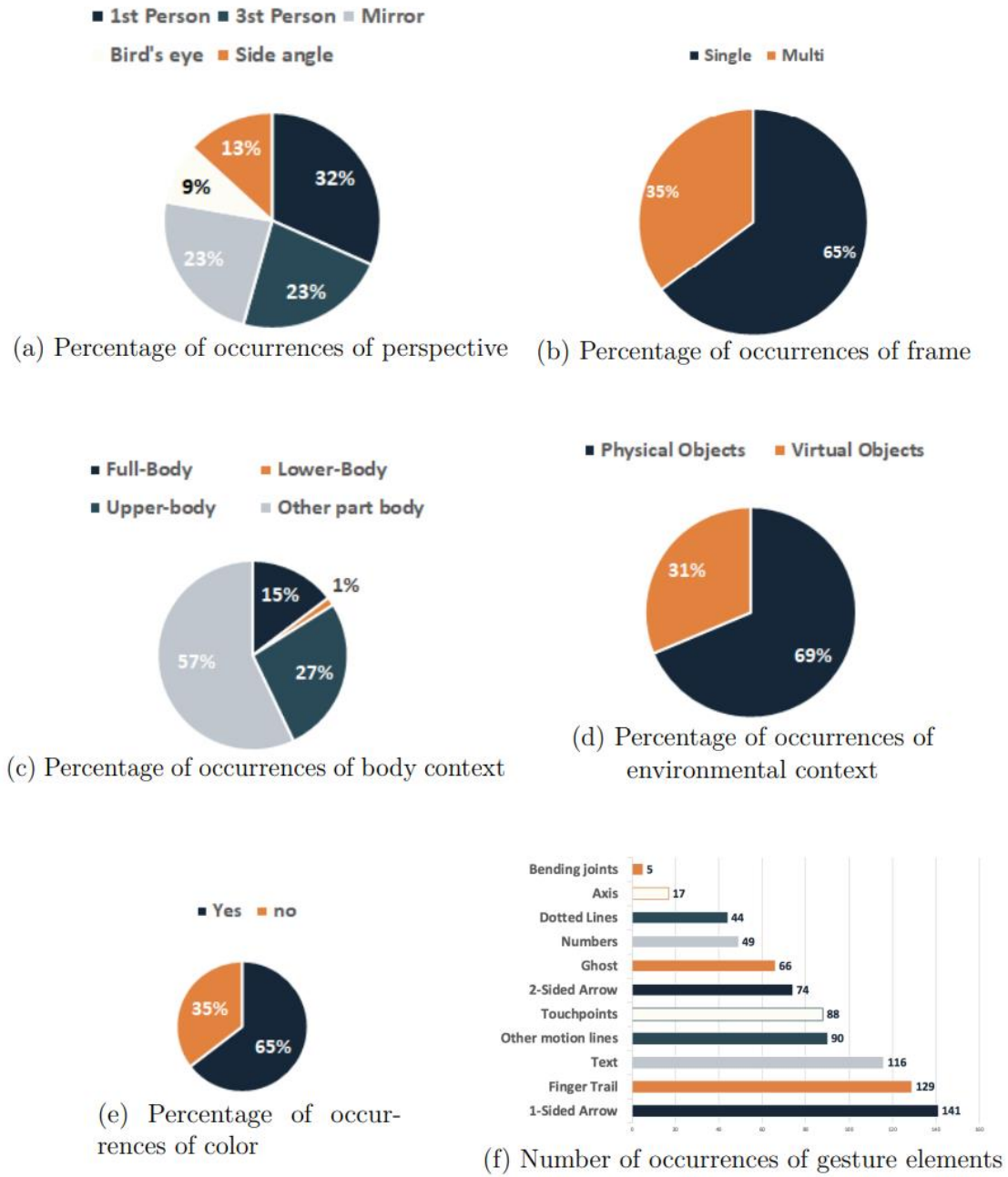


Figure 16: Frequency of occurrence of McAweeney's taxonomy elements in the 261 GES analyzed (Usui et al., 2020)

2.2.2 Bernsen's taxonomy

To have a second angle of analysis and complete the analysis done on McAweeney & al. (2018) taxonomy, Usuwiel et al., (2020) conducted the same analysis with Bernsen's taxonomy (Bernsen & Dybkjær, 2010). Bernsen's taxonomy is built from the modalities that can be used to interact with an information system (see Figure 17). It is composed of four different levels: the super level, the generic level, the atomic level, and the sub-atomic level (Usuwiel et al., 2020).

Super level	Generic level	Atomic level	Sub-atomic level
Linguistic modalities →	1. Sta. an. graphic ele. 2. Sta.-dyn. an. acoustic ele. 3. Sta.-dyn. an. haptic ele. 4. Dyn. an. graphic →	4a. Sta.-dyn. gest. disc. 4b. Sta.-dyn. gest. lab. 4c. Sta.-dyn. gest. not. 5a. Written text →	5a1. Typed text 5a2. Hand-writ text 5b1. Typed lab. 5b2. Hand-writ lab. 5c1. Typed not. 5c2. Hand-writ not.
	5. Sta. non-an. graphic →	5b. Written lab. → 5c. Written not. →	
	6. Sta.-dyn. non-an. acoustic →	6a. Spoken disc. 6b. Spoken lab. 6c. Spoken not.	<i>Legend</i> <i>an.</i> = analogue <i>compos.</i> = compositional <i>disc.</i> = discourse <i>dyn.</i> = dynamic <i>ele.</i> = elements <i>gest.</i> = gesture <i>lab.</i> = labels /keywords <i>non-an.</i> = non-analogue <i>not.</i> = notation <i>sta.</i> = static <i>writ.</i> = written
	7. Sta.-dyn. non-an. haptic →	7a. Haptic text 7b. Haptic lab. 7c. Haptic not.	
	8. Dyn. non-an. graphic →	8a. Dyn. written text 8b. Dyn. written lab. 8c. Dyn. written not. 8d. Sta.-dyn. spoken disc. 8e. Sta.-dyn. spoken lab. 8 f. Sta.-dyn. spoken not.	
Analogue modalities →	9. Static graphic →	9a. Images 9b. Maps 9c. Compos. diagrams 9d. Graphs 9e. Conceptual diagrams	
	10. Sta.-dyn. acoustic →	10a. Images 10b. Maps 10c. Compos. diagrams 10d. Graphs 10e. Conceptual diagrams	
	11. Sta.-dyn. haptic →	11a. Images → 11b. Maps 11c. Compos. diagrams 11d. Graphs 11e. Conceptual diagrams	11a1. Gesture 11a2. Body action
	12. Dyn. graphic →	12a. Images → 12b. Maps 12c. Compos. diagrams 12d. Graphs 12e. Conceptual diagrams	12a1. Facial expression 12a2. Gesture 12a3. Body action
Arbitrary modalities →	13. Sta. graphic 14. Sta.-dyn. acoustic 15. Sta.-dyn. haptic 16. Dyn. graphic		
Explicit structure modalities →	17. Sta. graphic 18. Sta.-dyn. acoustic 19. Sta.-dyn. haptic 20. Dyn. graphic		

Figure 17: Bernsen's taxonomy (Bernsen & Dybkjær, 2010)

The super level contains 4 categories related to the modalities:

- The **linguistic modalities** are ways to communicate information through language. For example, one way that is often used is written text to add precision to what is described in the representation.
- The **analogue modalities** are the means to represent information using similarities between the representation and what is represented. For example, a drawing representing the hands of a user performing the represented gesture.
- The **arbitrary modalities** are selected by designers and others to represent something without relying on an already existing system of meaning. For example, in a representation it may be established that a certain graphical element, like a star, means that a gesture must be performed at a certain speed. A person who does not know the meaning of this element will not be able to interpret it naturally.
- The **explicit structure modalities** are used to explicitly mark distinctions and separations among representations. This can be for example columns or tabular structures to define the spatial layout of the representation.

There is a notion of inheritance, i.e., each sub-level is created on the basis of its predecessor and each modality inherits the properties of its previous levels but also has its own characteristics. For example, the analogue modalities are divided into graphic, acoustic and haptic modalities at the generic level (Usui et al., 2020).

The research of Usui et al. (2020) allowed to put forward which modalities were the most used in the elaboration of the representations of the interaction gestures. Applying this taxonomy to the 261 GES, they obtained the results displayed in Figure 18: linguistic modalities are the most used with 70% of occurrence, followed by analogue modalities with 30% of use. Arbitrary modalities appeared only once and explicit structure modalities were never used on the 265 papers analyzed. The authors explain that this result makes sense because language can be used in written text to explain the gesture that is to be performed and that analogous modalities are based on the similarity between the representation and the represented gesture. Moreover, at the atomic level, three modalities appeared most often: images (35%), written lab. (35%) and written text (28%). These results make sense since images give information in a blink of an eye by the reader, who can quickly get a global idea of the gesture

to be performed and written information allows to bring more precision to the image (Usui et al., 2020).

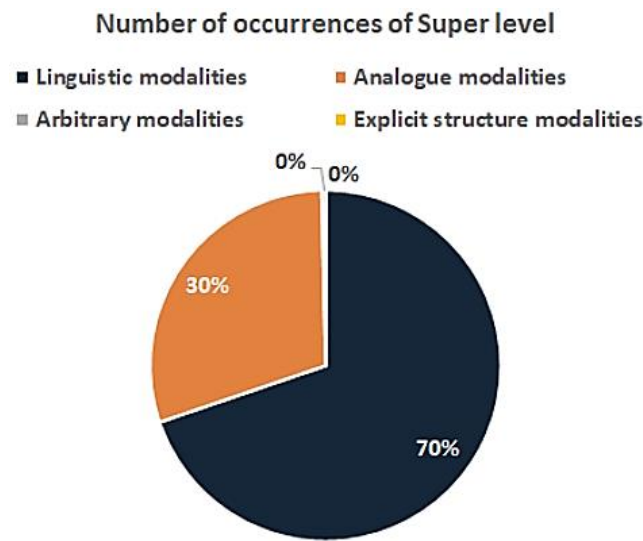


Figure 18: Bernsen's taxonomy: percentage of occurrences of super level gesture (Usui et al., 2020).

2.2.3 Xia et al.'s factors for iteratively designing gesture vocabularies

To the two taxonomies previously presented, we think it is interesting to consider the work of Xia et al. (2022) which provides designers and developers with a tool to facilitate the creation of what they call gesture vocabularies. This corresponds to all the gestures created and thought out by the designers and developers for the users to communicate with the developed application. The problem identified by the researchers is that the methodologies used to create gesture vocabularies rely on the expertise, experience and capabilities of experts, users, and systems to fulfill different requirements. However, in practice, these methodologies do not take into account all the factors that can influence gesture design choices. Their work has therefore consisted in establishing a framework that allows all these factors to be taken into consideration. They identified a total of 13 factors crucial to establishing a gesture vocabulary. You can find them in Figure 19, divided in 4 different categories: situational, cognitive, physical and system factors (Xia et al., 2022).

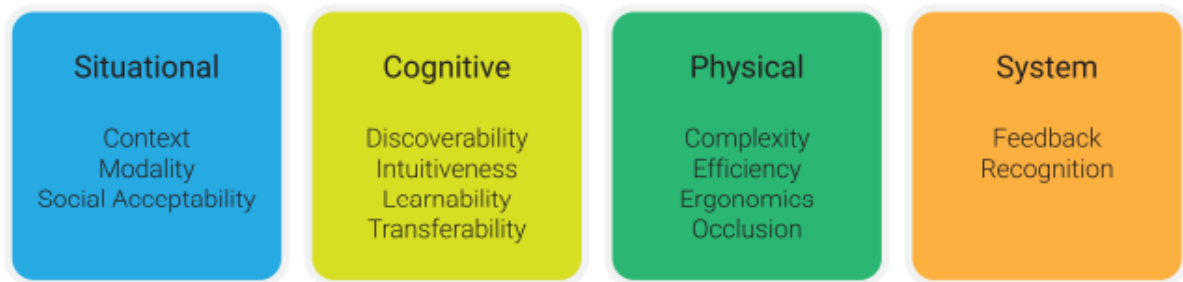


Figure 19: The 13 factors essential for gesture design identified by Xia et al. (2022).

2.2.3.1 Situational factors

This category includes three factors to be considered for gesture design (see Figure 20): the context that describes where the gesture is performed, the modalities used to perform the gesture, and social factors that may influence the way the gesture is performed. These factors are further described as follows.



Figure 20: Situational factors from Xia et al. (2022).

2.2.3.1.1 Context

The *context* refers to the users, their environment and the task(s) to be performed. Dey (2001) defines the context as any information that characterizes the situation in which the gesture will be performed. For example, if a gesture is intended to be performed in a vehicle to communicate with the dashboard, the limitations of sitting behind the wheel must be taken into account as well as the risks involved. Gestures performed while driving should distract the driver as little as possible from driving, so it is necessary to think about gestures that do not take the hands too far from the wheel or the eyes from the road (Xia et al., 2022).

2.2.3.1.2 Modality

The authors use the definition of Coutaz et al. (1995) which describes the *modalities* as the sensory input channel used to perform a gesture as well as the output channel that an information system uses to return the information corresponding to the input.

Inputs refer to all possible ways to communicate with an IS such as fingers, hands, feet, the whole body or even objects such as a stylus or a remote control. Each modality has its advantages depending on the situation and the choice of input channel must therefore be adapted to the context of use (Xia et al., 2022).

Outputs are also important to consider and can influence how gestures are performed and how users respond to them. The diversity of devices that can be used offers users multitudes of ways to interact with IS, as shown in section 2.1.. For example, if a large screen is used, people tend to make larger gestures that will take longer to perform (Hu et al., 2015).

2.2.3.1.3 Social acceptability

The researchers take the definition of Williamson (2012) to define this factor. According to her, it refers to the social-cultural factors that influence the user experience when he performs a gesture. This can include the place where the gesture is performed as well as the presence and composition of the audience present at that place, the age of the user, the body parts needed to perform the gesture but also the size and duration of the gesture, etc. This factor also includes how the user feels while making the gesture and how nearby spectators will perceive the gestures. The perceptions and reactions of others are data to be taken into account when designing gestures (Xia et al., 2022).

This factor implies that certain gestures are more appropriate to perform in certain places (in an elevator versus on the street), or in front of certain groups of people (strangers versus family) (Rico & Brewster, 2010). Cultural norms also play a role, as certain gestures are perceived as inappropriate in some cultures while they have no negative connotation for others. As a result, a user will not dare to perform a culturally inappropriate gesture or will feel embarrassed to do so (Ackad et al., 2015).

2.2.3.2 Cognitive factors

Cognitive factors, in Figure 21, focus on the importance of the discoverability of gesture vocabulary, whether it is intuitive to use, whether it is easily learnable and whether it is easily transferable to other domains or contexts. All these factors are influenced by the user's

cognitive understanding of the system and the movements they must perform (Xia et al., 2022). These four factors are defined as follows.

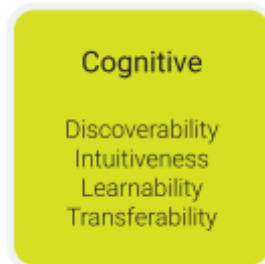


Figure 21: Cognitive factors from Xia et al. (2022).

2.2.3.2.1 Discoverability

Discoverability is also called guessability or approachability and is defined as the degree to which a user can guess the gesture he must execute to reach a certain goal by interacting with an IS (Wobbrock et al., 2005). For example, if a user encounters an interactive IS for the first time, it is important that the gesture vocabulary is discoverable so that he can quickly access the information he is looking for. If this is not the case, it would create frustration and he could pass by.

2.2.3.2.2 Intuitiveness

To define *intuitiveness*, the researchers rely first on Raskin's definition (1994), which states that it is the degree to which a gesture makes use of transferable and existing knowledge. Later on, Jacob et al. (2008) added that “*newly proposed interaction techniques should draw strength from users’ pre-existing knowledge of the everyday, non-digital world.*”. For example, using swipe to turn a virtual page is an intuitive gesture because it copies the action of turning a page of a physical book in real life (Wachs et al., 2008). The gesture that must be executed must correspond to what the user expects to do. If the user is not able to make a parallel between reality and the gesture to be executed, he will make the wrong gesture or none. The researchers point out that a gesture can be intuitive without being discoverable. They take the example of the time when touchscreens were not yet as widespread as they are now (before 2007). Indeed, at that time, novice users of touchscreens did rarely guess the gesture “pinch to zoom” but once the experts showed them, they remembered it without difficulty because of the intuitiveness (Wigdor & Wixon, 2011).

2.2.3.2.3 *Learnability*

Learnability refers to the ease with which a new user can pick up a new interaction gesture and achieve maximum performance with it. This is a complementary factor to intuitiveness because an intuitive gesture is easier to learn. However, the difference is that intuitiveness refers rather to the matching between a movement and the expected outcome while learnability focuses on how this matching, and other factors, allow users to remember the gestures (Xia et al., 2022).

2.2.3.2.4 *Transferability*

To define *transferability*, they use the definition of Perkins & Salomon (1999) which say that it is the degree to which the learning of a gesture can enhance or undermine its performance in another context. The literature also calls it generalizability or adaptability. We talk about transferability, for example when we want to reuse a known gesture with another hand, other modalities, with other applications or devices, for another user population, etc. (Xia et al., 2022).

2.2.3.3 *Physical factors*

The third category *physical factors*, in Figure 22, includes factors that are influenced by the physical movement that the user makes during the execution of the movement. Xia et al. (2022) address the efficiency with which a gesture can be made, their complexity, their ergonomics, the naturalness of the movement and the degree of occlusion of the user interface by the executed gesture.



Figure 22: Physical factors from Xia et al. (2022).

2.2.3.3.1 *Efficiency*

According to the International Organization for Standardization (2007) within the ISO 9241-400, *efficiency* refers to the precision and completeness with which a user performs an action or gesture. Efficiency is an important factor because it allows users to increase their throughput,

and thus accelerate their "speed of interaction", and to decrease the mental and physical efforts to execute the right gestures. For example, to select an object in a user interface, it is easier to point to it with a simple tap rather than a lasso gesture pointing to the object which will be less precise and will increase the chances of error (Xia et al., 2022).

2.2.3.3.2 *Complexity*

The *complexity* of a gesture refers to the degree of difficulty with which the gesture is executed. The designers and developers must adapt the complexity of the gestures according to the task to be performed. For example, unlocking a smartphone is an action that can be done dozens of times over a day, so the gesture to unlock must be simple and adapted to several situations in which the user may be (Xia et al., 2022).

2.2.3.3.3 *Ergonomics*

Ergonomics for gestures indicates that users should feel physically comfortable when performing a gesture. That is, they should not feel pain, tiredness, or discomfort. Designers must therefore take into account the potential risks that users face when performing a gesture error (Xia et al., 2022). An example of a gesture that is considered not ergonomic is asking a driver to make a complex gesture to communicate with his dashboard, taking his attention away from the road. Or for example certain gestures that are regularly repeated can lead to repetitive stress injuries like carpal tunnel syndrome (M. Nielsen et al., 2004).

2.2.3.3.4 *Occlusion*

The researchers use Vogel & Balakrishnan's (2010) definition that *occlusion* is the degree to which a system's output is covered or blocked when a gesture is performed. This occlusion problem is called the fat finger problem. A simple example is that of a user who would like to place a pin on a card in his smartphone. When placing his finger on it, the user does not know exactly where the pin will be placed on the map because his finger obstructs the portion of the map that interests him (Yatani et al., 2008).

2.2.3.4 System factors

The last category of factors, in Figure 23, relates to all the features that include information systems and devices. It refers to techniques and technologies that allow to recognize the gestures as input and the feedback of the system as output given to the user (Xia et al., 2022).

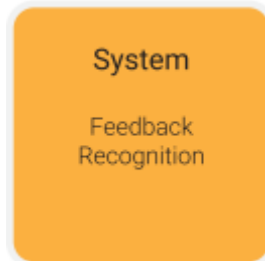


Figure 23: System factors from Xia et al. (2022).

2.2.3.4.1 Feedback

Shneiderman (1997) defines *feedback* as the communication that a system has with a user resulting directly from the user's actions. Simply put, feedback is the system's response to the user's gesture. This response can be transmitted in several forms: auditory, visual, haptic, olfactory, or tactile. Designers must select the most appropriate option for the situation the user is in to make the experience as comfortable as possible (Xia et al., 2022).

2.2.3.4.2 Recognition

Rautaray & Agrawal (2015) define *recognition* as the process of tracking a gesture from its initial representation through to its later conversion into a semantically meaningful command.

2.2.3.5 Discussion

This framework composed of the 13 factors presented is a tool for designers and developers when establishing their gesture vocabulary. In particular, it allows not to neglect any aspect of the interactions between the potential users and the information systems (Xia et al., 2022). Our work focuses on the communication around the interaction gestures between the different stakeholders of a development team, and more precisely between designers and engineers. Therefore, we think that it can be interesting to integrate the possibility of specifying some of these factors in the description of gestures. This will be discussed in 3.4.3 dedicated to the development of a new gesture description method.

2.3 Existing representations for gesture definition

In recent years, various notations have been developed to support the development of gesture-based applications, most often focusing on multi-touch gestures. Most of them are linked to a specific gesture recognizer which requires the introduction of additional elements to the notation to provide machine-readability. Hesenius & Gruhn (2019) distinguishes between four different notation types: template-based, textual, graphical and hybrid notations.

2.3.1 *Template-based gesture description*

This method is often closely related to the configuration of appropriate gesture recognition algorithms. Rubine (1991) describes the general principle of this method as follows: a gesture (or several variations) is recorded and used as the basis for recognition by comparing all performed gestures with the previously recorded one. To simplify, there is an example gesture that is used as a specification. Thus, template-based approaches are a simple way to specify and document gestures.

Researchers consider Rubine (1991) to be the pioneer of this template-based description method, but more efficient variations of this method, allowing the recognition of more gestures, have been developed in recent years. The most notable are those belonging to the \$-family with \$1 (Wobbrock et al., 2007), \$N (Anthony & Wobbrock, 2010) and \$P (Vatavu et al., 2012). And later, other methods appeared optimizing these first mentioned with the arrival of 1¢ (Herold & Stahovich, 2012), \$N-Protractor (Anthony & Wobbrock, 2012) and PennyPincher (Taranta II & Laviola Jr., 2015). Still recently a lot of work in the field of gesture recognition is based on the \$-family.

Despite the simplicity of this method, it has some drawbacks that make it not very useful for our research work on gesture specification. For example, the \$-family notations are not appropriate to describe multi-touch gestures. These types of notations are mainly based on the shape of the gesture. That is, they are able to recognize different versions of a gesture having the same shape, such as two same gestures having a different orientation. However, it is possible that the shape of a gesture has not been defined at the time of design, and therefore cannot be defined with the template-based approaches. Thus, they are able to recognize different versions of a gesture having the same shape, such as two same gestures having a different orientation but a lot of important information, due to the multitude of possible gestures

nowadays, is lost (e.g., speed, pressure, etc.)(Hesenius & Gruhn, 2019) (Schmidt & Weber, 2013).

Moreover, the properties of a gesture cannot be made explicit via this kind of notation. Indeed, information such as the speed of execution or the pressure exerted for a tactile gesture are not specified, making these notations not very readable by humans and not very suitable for the communication of gestures. For these reasons, we will not dwell on this type of notation.

2.3.2 *Textual gesture description*

The textual notations define the gestures via several keywords and operators that allow to describe the partial gestures, their relationships to each other and their features. Multiple textual approaches have been developed in the last years and we will focus on the most interesting ones for our work in this section. While many of these notations allow for both machine and human readability, they more often focus on machine readability, which can make the notation rather difficult to digest for humans. Indeed, for machine readability, the introduction of keywords, structuring elements or additional rules is required. All these additions, which (Kammer et al. (2015) call “syntactic overheads”, do not relate to the description of the gesture itself and thus obtrude the human readability of the notation (Hesenius & Gruhn, 2019). In our work, we focus mainly on the descriptive aspects of notations that allow human readability. Hoste & Signer (2014) use readability of notation languages as one of the criteria for comparing them. According to them, a language with a large number of syntactic rules, such as parentheses, brackets, colons or semicolons, are perceived as less readable and therefore more complex.

There is a lot of textual notations but few of them interest us for our work which seeks to go towards the essential in order to provide a human readable notation giving the most important information on the described gesture. In fact, a lot of notations have a heavy language for a person who does not have the required formation to read it. This being the case, we exclude these notations from our scope because they are not in agreement with our objective of human readability and easy communication between people with different backgrounds. For example, in the GISpL notation, which is a further development of GDL, a swipe with two fingers is defined in Figure 24 (Echtler & Butz, 2012). As you can see, at first glance, this notation presents a complex and complicated language to understand if you do not have the necessary background. To give an idea, line 8 of Figure 24 defines the use of the fingers and line 9 defines the field constraints, i.e., the minimum and maximum number of touches, two in this case.

Also, the problem that meets many textual notations in terms of readability is the non-use of graphic elements. Here, for example, the shape is described, in lines 14 to 17, as a set of vectors that constrain movement (Echtler & Butz, 2012).

```

1 {
2   "name": "two finger swipe",
3   "flags": "sticky, oneshot",
4   "custom": "userdata",
5   "features": [
6     {
7       "type": "Count",
8       "filters": 2046,
9       "constraints": [2, 2],
10      "result": []
11    }, {
12      "type": "Motion",
13      "filters": 2046,
14      "constraints": [
15        [0.01, -0.01, -0.01],
16        [0.1, 0.01, 0.01]
17      ],
18      "result": []
19    }
20  ]
21 }

```

Figure 24: Notation in GISpL for a 2-finger swipe (Echtler & Butz, 2012).

Other notations that are also based on Echtler et al. (2010)'s GDL have emerged such as the one of Hachaj & Ogiela (2014) that focuses on describing gestures and poses in space by not limiting it to hands. Figure 25 shows an example of this notation for hand reconciliation. Khandkar & Maurer (2010) have also developed their own notation based on GDL. Compared to other textual notations, they use few structuring elements to focus on the properties of gestures. Figure 26 shows an example of a gesture described by this notation for a gesture using any closed form.

```

1 RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) < 100 THEN HandsTogether
2 RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) >= 100 THEN HandsSeparate
3 RULE sequenceexists("[HandsSeparate,0.5][HandsTogether,0.5]
4   [HandsSeparate,0.5]") THEN Clapping

```

Figure 25: Description of a hand reconciliation with the GDL notation of Hachaj & Ogiela (2014).

```

1 name: Lasso
2
3 validate
4   Touch state: TouchUp
5   Touch limit: 1
6   Closed loop and
7   Touch path bounding box: 200x200..1000x1000 and
8   Touch path length: 600..100000 and
9   Enclosed area: 5000..1000000
10
11 return
12   Touch points

```

Figure 26: Description of a gesture using any closed form in Khandkar & Maurer (2010) GDL notation.

There are other notations in the same genre as the three presented above but as mentioned earlier, we will not retain them because they seem to us too unreadable for the object of our research. However, some textual notations hold our attention because of their simplicity of use and the relative ease of reading, although it still requires a minimum knowledge of the language. We are going to present what we believe to be the most efficient notations in terms of simplicity and conciseness.

2.3.2.1 *GeForMT*

GeForMT created by Kammer et al. (2010) is a Domain-Specific Language (DSL) made to describe multitouch gestures. One of the goals of this notation is to make it concise and clear for humans to read (Kammer et al., 2015), which makes it particularly interesting for our work. GeForMT uses a basic mathematical syntax that is composed of five key elements described by the authors as follow:

- *Number and shape of the touch contact:* It is a prefix that defines which part of the body is in contact with the surface. As this notation is most used for touch screen applications, it is most often the finger that is indicated by an F (1F, 2F or more “F” depending on the number of fingers in contact) or H to indicate that it is a hand contact necessary for the gesture.
- *The performed movement:* It describes the movement made by the finger or hand on the touch surface. The syntax uses only very simple shapes, called atomic gestures, like points, lines, semicircles or circles but these can be combined with operators to describe more complex gestures. Other directional constraints can be added to the shape description with cardinal points. For example, it is possible to specify that a line is drawn to the north or that a circle is drawn in the clockwise direction. 1F(LINE_E) defines a line drawn to the east (Figure 27 a)) or 1F(SEMICIRCLE_S_CW) describes a semicircle drawn in the clockwise direction stopping at the south (Figure 27 b)).
- *Composition Operators:* These operators, mostly used when we want to describe more complex gestures, allow to specify in which order the atomic gestures are performed. For example, whether it is necessary to perform two atomic gestures simultaneously or whether they should be performed one after the other. They are placed between the gestures that are related and there are three of them: an asterisk indicates that the gestures are synchronous. For instance, if you want to define a pinch gesture where both fingers move together, the notation indicates: 1F(LINE)*1F(LINE) (Figure 27 c)).

A plus-sign indicates asynchronous movements. This is the case for defining a hold and move gesture, where one finger performs the hold position and the other finger the move gesture: $1F(HOLD)+1F(MOVE)$. And the comma is used when several atomic gestures are performed successively. For example, to describe a "U" we define the gesture as $1F(LINE_S, SEMICIRCLE_E_CCW, LINE_N)$.

- *Spatial relations*: These are key words that allow you to better situate the gestures in relation to each other. For example, the prefix JOIN describes convergent atomic gestures, or SPREAD means that two atomic gestures depart from each other. If two lines have to cross each other (Figure 27 d)), we use the prefix CROSS to indicate it, and this will give the definition $CROSS[LINE_S;_LINE_E]$ (Figure 27 d)).

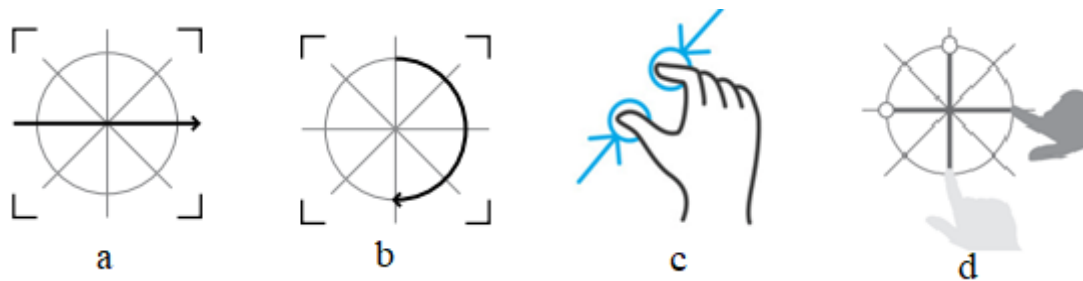


Figure 27: a) $1F(LINE_E)$; b) $1F(SEMICIRCLE_S_CW)$; c) $1F(LINE)*1F(LINE)$; d) $CROSS[LINE_S;_LINE_E]$.

Sources: a), b) from Kammer et al. (2010). c) from <https://stock.adobe.com>. d) from Kammer et al. (2015).

- *Focus objects*: They are added to link a gesture to a concrete element of the user interface. For example, to indicate that you have to hold your finger on a specific button of the interface, you can define a gesture as $HOLD(myButton)$.

With this simple syntax that is quickly learned, GeForMT allows to define very simple gestures but also more complex ones by combining atomic gestures. In their experiment, Kammer et al. (2015) show that GeForMT performs better than other textual notations in terms of conciseness and readability. Moreover, they also showed that this notation is easily embedded in text, e.g., in source code, without having to be defined as separate blocks, which is an advantage during application development. However, Hesenius & Gruhn (2019) explain that some complex gesture expressions can lead to very long notations and therefore more complicated to interpret. Furthermore, the syntax is limited to a few terms to describe the characteristics of the gesture

and leads to imprecision. This lack of precision can also lead to different interpretations of gestures for the same description.

2.3.2.2 Proton/Proton++

Proton is a notation developed by Kin et al. (2012b) that also allows the description of multi-touch gestures using regular expressions (Regex). Like GeForMT, Proton is useful for describing simple gestures, such as a single surface touch, or more complex gestures that are then a combination of atomic gestures. There are only three touch actions with this notation: D for "Down" when the finger touches the surface, M for "Move" when the finger moves on the surface and U for "Up" when the finger leaves the surface. With this notation, a gesture always starts with a D and always ends with a U. In an expression, each touch event, usually equal to the number of fingers needed, has a touch ID, which can take the values 1 to X, with X touch actions. Later, Kin et al. (2012a) created Proton++, which is an extension of Proton that allows developers to add touch attribute values such as a direction or a hit-target to specify a target object in the interface. The generic form of a touch event, following Proton++' structure, is the following:

$$E_{TID}^{A_1:A_2:A_3...}$$

Figure 28: Basic structure of a touch event with Proton++ (Kin et al., 2012a).

In Figure 28, E corresponds to the touch action and can be D, M or U, as described earlier. The T_{ID} corresponds to the touch ID, allowing to differentiate between several touch events, and $A_1:A_2:A_3:...$ are the attribute values where A_1 corresponds to the first attribute, A_2 to the second attribute, etc. Proton gives developers the freedom to customize touch events with attributes they can add. This requires writing attribute generators that map hardware touch information (e.g., position, pressure, etc.) to discrete attribute values. Two operators are used in Proton: the asterisk "*" to indicate a repetition and a vertical bar "|", placed between two touch events, indicates that they can be performed in any order (Kin et al., 2012a).

In Figure 29, you can see the description of a swipe with two fingers to the right. At the beginning, the first finger lands on the surface (D_1) and starts to move to the right ($M_1^E *$), where "E" stands for east. Then the second finger touches the surface (D_2) and both fingers continue to move to the right ($M_1^E * | M_2^E *$). The last part indicates that either the first finger is

lifted ($U_1 M_2^E * U_2$) and then the second, or vice versa ($U_2 M_1^E * U_1$). As the user could choose either alternative, both possibilities must be specified with the vertical bar.

$$D_1 M_1^E * D_2 (M_1^E * | M_2^E *) (U_1 M_2^E * U_2 | U_2 M_1^E * U_1)$$

Figure 29: A swipe with two fingers to the right described in Proton++ (Hesenius & Gruhn, 2019).

Proton++ certainly achieves the goal of conciseness, but in terms of readability, we quickly end up with very dense and complicated expressions to read. Moreover, because expressions are based only on three touch actions, there is a risk that different gestures lead to identical expressions (Hesenius, 2019). However, the authors of this language have developed in parallel a graphical editor that helps to define gestures, but it is only used to structure the expression in a clearer way, this one does not help to reach a better readability of the expressions (Kin et al., 2012a).

2.3.2.3 GestIT

GestIT, developed by Spano et al. (2013), is another textual notation that, like Proton, has a declarative and compositional approach. It allows modeling multi-touch gestures on screen but also allows modeling gestures for full body tracking devices. As for GeForMT and Proton, GestIT defines gestures with expressions that are composed of ground terms, which correspond to atomic gestures in GeForMT notation and touch action in Proton. These ground terms are linked together with operators. It is also possible to add predicates to each ground terms in order to put a constraint (trajectory, speed, etc.) on movement it refers to. The ground terms are *Start*, *Move* and *End*, and the operators are described as follow: the "*" is the iterative operator indicating that a gesture is repeated an indefinite number of times, the ">>" indicates a sequence of gestures with an order from left to right, the "||" defines gestures performed simultaneously, the "[]" gives the choice to recognize only one gesture among the connected sub-gestures, the "[>" stops the recognition of another gesture, typically used for stopping an iteration loop and the "|=" means that linked gestures can be performed in any order.

For example, using GestIT, a pinch movement is described in Figure 30. The expression starts with two fingers touching the screen ($Start1 \mid = \mid Start2$). Then, the two fingers move simultaneously ($Move * 1 \parallel Move * 2$). Finally, both fingers leave the screen at the same time ($End1 \mid = \mid End2$). The information that both fingers are moving towards each other is missing

but authors explain that the corresponding event handler makes the distinctions for that gesture (Hesenius & Gruhn, 2019).

$$(Start_1| = |Start_2) >> (Move_1^*||Move_2^*)[> (End_1| = |End_2)$$

Figure 30: Pinch gesture with GestIT (Spano et al., 2013).

Like Proton, GestIT allows a concise definition of gestures but also encounters the same problems. Indeed, this notation can quickly lead to complex expressions that are not very readable for people who are not familiar with it. Moreover, like the previous textual notations, the expressions can encounter two problems: a lack of precision to describe the characteristics of the gestures, and it presents a margin of interpretation, which can lead to different interpretations for the same gesture or the same expressions for two different gestures.

2.3.3 Graphical gesture description

Graphical notations are intended to reduce the complexity of textual expressions by using visual elements, like arrows or symbols, to define a gesture. However, the complexity of some gestures forces to add visual elements to describe with precision. As for textual notations, this leads to gesture definitions that are not very readable. Although effective for describing complex geometric shapes, graphical notation must translate words, that would be used in textual notation, into icons or shapes that do not always guarantee readability. We thus encounter the same problem as for textual notations, team members would need to learn those icons and symbols to understand the language (Hesenius & Gruhn, 2019).

2.3.3.1 EventHurdle

EventHurdle (J.-W. Kim & Nam, 2013) is a visual gesture authoring tool to support designers' explorative prototyping. It was created to allow application designers who do not have extensive knowledge in the use of sensors to prototype gestural interaction. The promise is that they can do it without technically experienced engineers in order to be more autonomous in the early stages of application development. EventHurdle can model gestures with different types of sensors such as a camera, a mobile device or a multitouch screen. It is limited to single direction gestures, but supports surface and spatial gestures (Hesenius, 2019). Indeed, for gestures performed in more than 2 dimensions and to keep their notation simple to use, the authors give the possibility to link hurdles from different workspaces. As a result, gestures using 3 or more sensors can be decomposed by separating the sensors into two dimensions.

Gestures are defined under EventHurdle in what is called the interaction workspace. This is a 2D space in which designers can place hurdles that define the important points through which the gesture must pass. The movement itself is then represented by an arrow. Figure 31 shows an example of a simple gesture described with EventHurdle (J.-W. Kim & Nam, 2013). Figure 31 (a) shows the layout of the basic elements: The starting point in the center, represented by a star, and two hurdles annotated 1 and 2 on the figure. In (b), the first part of the movement is described, it goes from the starting point to hurdle 1. In (c), the gesture turns around, crosses hurdle 1 again and continues its path until it meets hurdle 2. Users can also define the time required to go from one hurdle to another from 0 to infinite in milliseconds. This makes it possible to differentiate between fast and slow movements as well as stopping points.

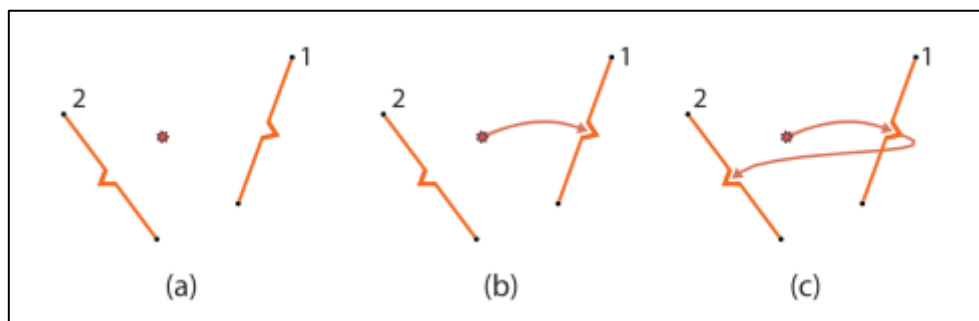


Figure 31: Description of a simple gesture with EventHurdle (J.-W. Kim & Nam, 2013).

2.3.3.2 Monox

Monox (Ganhör & Spreicer, 2014) is an extensible sketching notation for mobile gestures. This graphical notation was designed to facilitate communication among interdisciplinary groups of researchers, designers, and developers. To do this, this notation is designed to be easy to use and understand by anyone. Monox is limited to mobile gestures and describes the basic gestures that can be performed on a smartphone screen.

The simple contacts, called tapping gestures, are represented by circles. If a gesture includes several taps meaning that several fingers are used, each finger is described by a circle. The notation can differentiate between multiple types of taps. Figure 32 describes: (a) a short single tap, (b) a double tap, (c) a long tap, where the finger is held longer on the screen. However, it is not possible to determine how long the tap is held or to define a pause time between taps (Hesenius, 2019). Multiple taps are built from single taps and the reading of their representation is from outside to inside: Figure 32 (b) is the combination of 2 short taps, (d) describes a short tap followed by a long one while (e) starts with a long tap followed by a short one.

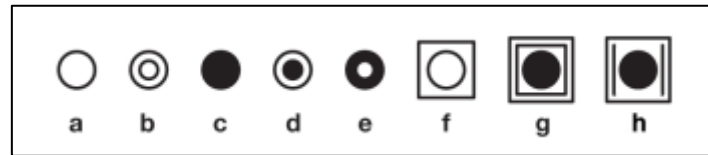


Figure 32: Tapping gestures with Monox (Ganhör & Spreicer, 2014).

Monox adds to its descriptions of simple tapping gestures a launching area. This area defines the boundaries within which the tapping should be performed. If a user touches the area outside the launching area, the system either ignores the gesture or accepts it as a gesture tapping another item. When the launching area is represented by a simple line as in Figure 32 (f), it means that a tap outside this area will have no consequence. Whereas a double line as in (g) indicates that a tap outside the launching area will have consequences. The size of the area is arbitrary and can be very small as in (f) or larger. In (h), the combination of single and double lines indicates that there are consequences for taps to the left and right of the area, but none if it is above or below (Ganhör & Spreicer, 2014).

Movements with the fingers, such as swipes, are represented by arrows as in figure 11. The authors make the distinction between slow and fast movements: Figure 33 (a) describes a slow movement gesture, characterized by an arrow while a fast movement is indicated by a double arrow as in (b). The movement is in the direction of the arrow but if the direction of movement is not to be specified, arrowheads are on both sides of the line (c). When a movement must be done with several fingers, the number of fingers is indicated by the number of bars crossing the arrow. The Figure 33 (d) describes a slow movement to the right with two fingers (Ganhör & Spreicer, 2014).

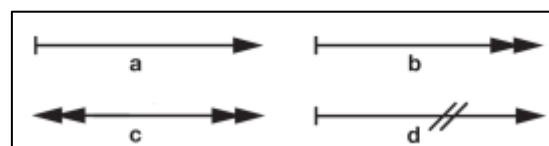


Figure 33: Moving gestures with Monox fingers (Ganhör & Spreicer, 2014).

2.3.3.3 Hamburg Notation System (HamNoSys)

HamNoSys, first developed in 1984 and improved in 1989 by Prillwitz et al. (1989), is a notation system that is used to transcribe signs from sign language. There are many different sign languages in different countries, but HamNoSys has the advantage of being universal and can describe signs in all languages. This consists of the description of the shape of the hands, their orientation, location, and movement. It also transcribes non-manual properties (Kaur &

Kumar, 2016). We found it interesting to present this gesture representation because it also aims to describe gestures performed in a three-dimensional space.

To describe the shape of the hands, HamNoSys uses a series of symbols that correspond to a specific configuration of the hand according to the position of the fingers in relation to each other. Figure 34 shows some examples of hand configurations and their representation with HamNoSys (Kaur & Kumar, 2016).

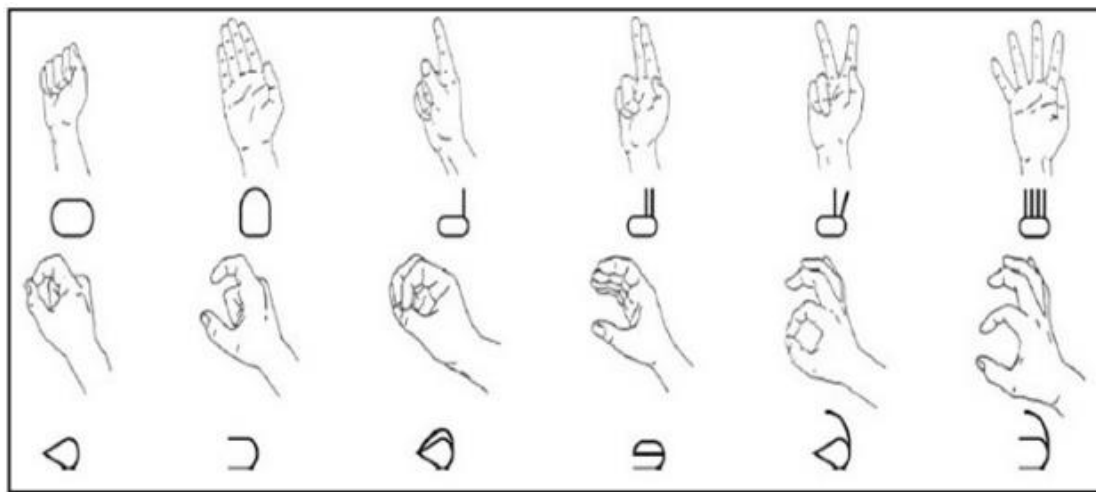


Figure 34: Hand-shapes description examples with HamNoSys (Kaur & Kumar, 2016).

The orientation of the hand is represented by combining 2 components: the direction of the extended finger and the orientation of the palm. With these two components, the notation gives a fairly accurate description of the orientation of the hand. For the direction of the extended finger, 3 perspectives are used: sign's view, birds' view, and a view from the right. The symbols representing the possible directions are shown in Figure 35 where you will notice that a bar is added at the bottom if it is described from a birds' view, on the left if it is a view from the right and no bar if it is a view from the signer. For palm orientation, the notation uses 8 symbols as shown in Figure 36. Note that the orientation of the palm is given according to the direction of the extended finger (Hanke, 2004).

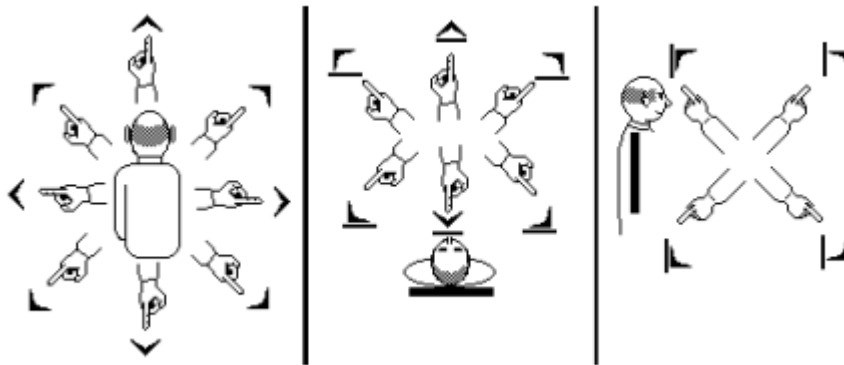


Figure 35: Extended finger's orientation in HamNoSys (Hanke, 2004).

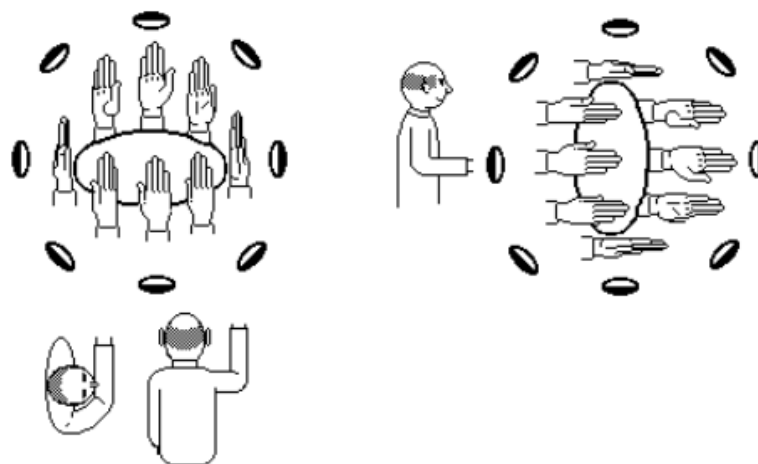


Figure 36: Palm orientation in HamNoSys (Hanke, 2004).

For the location of the hand, two components are needed: one to determine the position in the frontal plane (x and y coordinates) and the other to determine the z coordinate. If there is no information for the z-coordinate, the hand is considered at a "natural" distance from the body. And if both components are missing, we consider the hand in the "neutral signing space", i.e. at a natural distance from the upper torso. The first component uses symbols corresponding to the different parts of the body as shown in Figure 37 (Hanke, 2004).

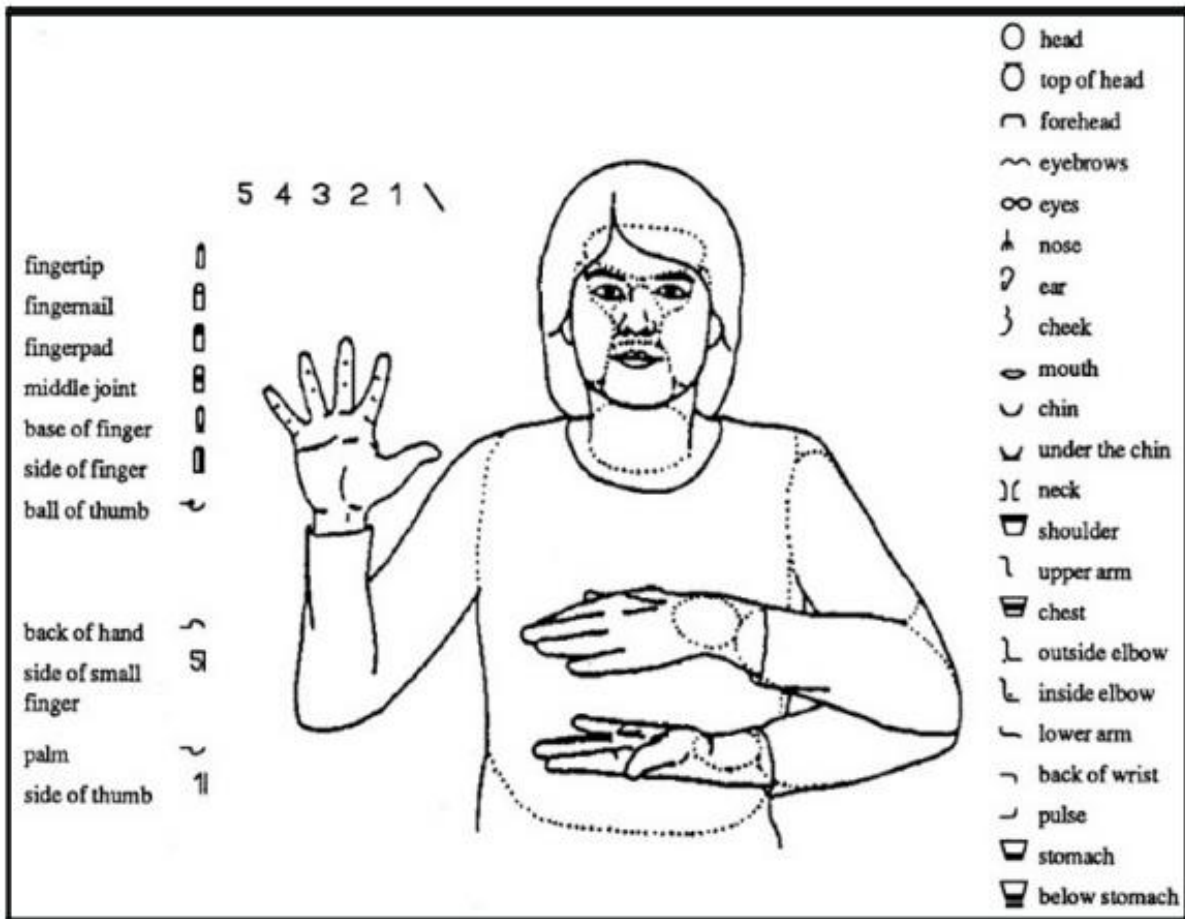


Figure 37: Hand location relative to body parts in HamNoSys (Kaur & Kumar, 2016).

The movements of the hand are distinguished in several categories: straight, curved, wavy, zigzag, circular and spiral movements. Examples of symbols are shown in Figure 38. The notation also gives the possibility to describe non-manual components. For example, it may be necessary to describe movements of the mouth, cheeks or more generally facial expressions. Although these symbols are not yet developed, some examples are shown in Figure 39 for mouth movements (Kaur & Kumar, 2016).

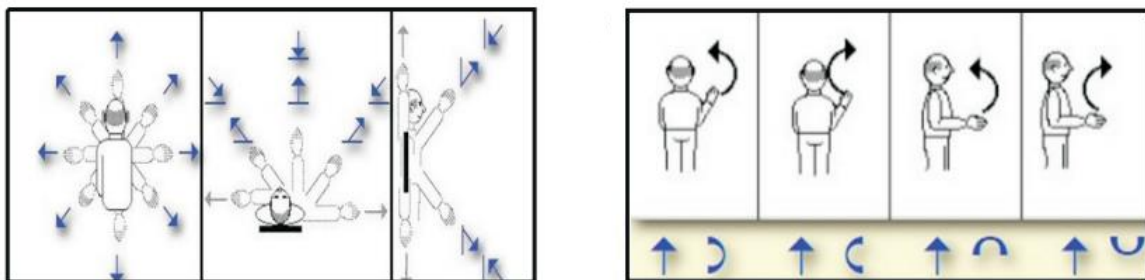


Figure 38: Example of symbols describing hand movements in HamNoSys (Kaur & Kumar, 2016).

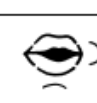
	C01	cheeks puffed	(static)
	C02	cheeks and upper and lower lip areas puffed	(static)
	C03	cheeks puffed gradually	(dynamic)
	C04(C)	one cheek puffed	(static)
	C05(C)	one cheek puffed; blow out air briefly at corner of one's mouth	(dynamic)
	C06(C)	one cheek puffed; blow out air briefly at corner of one's mouth when touching cheek with index finger	(dynamic)
	C07	cheeks sucked in, without sucking in air	(static)
	C08	cheeks sucked in, sucking in air through slightly open lips	(dynamic)
	C09(C)	tongue pushed into cheek (visible from outside)	(static)
	C10(C)	tongue pushed into cheek several times (visible from outside)	(dynamic)

Figure 39: Symbols representing mouth movements in HamNoSys (Hanke, 2004).

In Figure 40, you can see a full description using HamNoSys for the word "Today" from the Pakistani sign language (Farooq et al., 2021). As the image in Figure 40 is not very clear, the position of the hand is better shown in Figure 41.

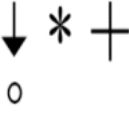
Sign in PSL	HamNoSys				
	Hand Shape	Hand Position	Hand Location	Hand Movement	Complete Vector
 Today					

Figure 40: "Today" gesture in Pakistani sign language described with HamNoSys (Farooq et al., 2021)



Figure 41: "Today" in Pakistani sign language from <https://www.psl.org.pk/play/dictionary/category/22/1193>.

HamNoSys has the advantage of being able to describe all the gestures of sign language using an exclusively graphic notation thanks to its symbols. However, it is not very readable for people who are not familiar with these symbols and this shows the complexity of describing spatial gestures on a two-dimensional support using only graphical elements.

2.3.4 Hybrid representation

The final type of notation presented by Hesenius (2019) is the hybrid one. It does not bring new concepts but mixes those presented in the above notations to describe gestures. The goal is to take advantage of the properties of each notation to create a new one that is as clear and efficient as possible. Some gestures are easier to describe textually while others are better understood graphically. This type of notation allows you to switch between different methods and to choose the most appropriate one for the gesture being described. These hybrid notations should therefore help gesture specification and communication between stakeholders during application development. However, they are rarely used and the reason for this under-use by researchers is unclear (Hesenius & Gruhn, 2019). There are only a few hybrid notations. For the purpose of this work, we will retain only GestureCards because the others, like Gesture Script (Lü et al., 2014), present limitations. The next section is entirely dedicated to GestureCards because this notation is the most complete for describing gestures and has been designed to facilitate communication between the different stakeholders of an application development team.

2.3.5 *GestureCards*

2.3.5.1 *Basic structure for simple gestures*

GestureCards (Hesenius, 2019) was created for clarity and conciseness so that it can be used in multi-functional teams. Like other notations presented previously, GestureCards defines simple or atomic gestures that can be combined to describe more complex gestures. These are composed of 4 sections: a unique name, a graphical representation, the traits that define the gesture with more precision and a list of traits, called return values, that will be interesting for further processing in the application. The traits and return values are optional as long as the graphical representation provides the needed information to identify a gesture (Hesenius & Gruhn, 2019).

Section 1: Name	Section 2: Graphical Representation	Section 3: Traits
		Section 4: Return Values

Figure 42: Basic building blocks to describe gestures with GestureCards (Hesenius & Gruhn, 2019).

Figure 42 shows the basic structure of GestureCards. The first section contains the name given to the gesture. This name is arbitrary and is chosen by the developer, but it must be unique within the application. This name can be reused in the description of other gestures to reference it. The second section, dedicated to the graphical representation, is mandatory. This one is very useful because it allows to give a lot of information in a glance. It is therefore important to make this graphic representation as good as possible, although for more complex gestures it is usually not enough. Note that it is possible that the form of a gesture does not matter, the word "FREE" in this section is then used to indicate it. The third section is reserved for traits and allows you to add information that cannot be described graphically. The last section "Return Values" also contains traits but these are the ones that will be used for further processing of the gestures e.g. the speed of execution or the pressure applied on a touch screen (Hesenius & Gruhn, 2019).

To clarify, the difference between the 3rd and 4th sections is that the traits defined in the 3rd section are conditions that must be met for the gesture to be recognized by the system. On the other hand, section 4 contains properties that correspond to the return values of the gesture. This means that this information, placed in section 4, is not necessary for the recognition of the gesture but provides information that the developed application can use. But if no result of the

gesture execution is needed for the application and only the recognition itself is relevant, this last section can be empty (Hesenius, 2019).

SwipeLeft		Speed: Fast
		Speed, Location

Figure 43: A swipe to the left with GestureCards (Hesenius & Gruhn, 2019).

Figure 43 shows an example of a simple gesture description, a swipe to the left, with GestureCards. The section on the left defines the name: SwipeLeft. The second section is the graphical representation of the gesture, here the gesture is simple so this one is very meaningful: a simple arrow pointing to the left is enough for understanding. The third section contains the traits necessary for recognition and indicates that the speed of execution must be fast. The term "fast" should be defined beforehand in the application by the development team. It is possible to give precise values, measurable with sensors, with appropriate units for some return values, e.g., for a duration it is possible to specify it in seconds. It is therefore up to the team to choose whether a specification of a trait can remain abstract or whether a precise value is required for the application. The last section indicates that the speed and location of the gesture are relevant to the application (Hesenius, 2019).

The nuance between the features in section 3 and those in section 4 is clearly shown in Figure 44. For the LineLeft movement, the speed is not a necessary condition for the recognition of the gesture whereas for a swipe, which has the characteristic of being a fast movement, the speed is a condition and is therefore specified in the features in section 3 (Figure 43). However, the speed of execution can be interesting in the application for any movement to the left, so it is taken into account in section 4 for both SwipeLeft and LineLeft (see Figure 43 and Figure 44) (Hesenius, 2019).

LineLeft		
		Speed

Figure 44: A simple movement to the left with GestureCards (Hesenius, 2019).

2.3.5.2 The traits

Hesenius & Gruhn (2019) have established a list of available traits that you can see in Table 1. Most of the traits defined in GestureCards are presented with their name and an associated value, called name-value pairs. The names are unique and predefined while the values can either be a single value or a range of possible values. As said before, these values can also be abstract, e.g., we can define a duration trait either as "Duration: Long" or "Duration: 2s".

Tool Count	Contact Area	User
Orientation	Movement Direction	Duration
Distance	Pressure	Ending Location
Shape	Speed	Multiplicity
Location of Execution	Reaction	Rotation
Starting Location	Tool	Size
Spatial Relation	Temporal Relation	

Table 1: List of traits in GestureCards (Hesenius & Gruhn, 2019).

They also gave the possibility to use operators to define the values of these traits. The three basic ones are: The smaller (<), the greater (>) and the interval (..) operator. The basic arithmetic operators are also available (+, -, *, /) to incorporate additional calculations. The last operator (->) allows to refer to the specific value of the stroke of another gesture. Examples are shown in Table 2.

Operator	Example
<	Duration: <2s
>	Duration: >2s
..	Duration: 1..2s
+	Duration: x + y
-	Duration: x - y
*	Duration: x * y
/	Duration: x / y
->	Duration: ->S1

Table 2: Available operators with GestureCards (Hesenius, 2019).

2.3.5.3 Definition for compound gestures

Compound gestures are made up of several partial gestures that are individually specified and form a whole. It is possible that the partial gestures and the compound gesture they form have different traits. Figure 45 shows the basic structure of a compound gesture. The partial gestures forming it are defined as presented previously. The traits of the compound are placed at the top right of the gesture card and its return values at the bottom right.

Figure 46 shows an example of a compound gesture definition named "MultiStrokeV". It contains two partial gestures performed one after the other without pause. The first partial gesture S1 is a downward right movement and the second S2 is upward right. The traits of S2 specify that the duration and size of the gesture are the same as S1. Also, the "Start: S1.End" trait indicates that S2 should start where S1 ended. And finally, the compound gesture has its own trait: the gesture must last in total less than 2 seconds.

Compound's Name		Compound's Traits	
Name Partial Gesture No 1	Graphical Representation PG1	Traits PG1	
		Return Values PG1	
Name Partial Gesture No 2	Graphical Representation PG2	Traits PG2	
		Return Values PG2	
Compound's Return Values			

Figure 45: Structure for compound gestures in GestureCards (Hesenius & Gruhn, 2019).

MultiStrokeV		Duration: <2s
S1		
S2		Duration: S1 Size: S1 Start: S1.End

Figure 46: Example of a V-Shaped multi-stroke gesture in GestureCards (Hesenius, 2019).

GestureCards gives the possibility to reference already defined gestures in new cards. When it is within the same gesture compound, the name can simply be reused. But when you want to reference a partial gesture coming from outside the compound, you have to add a prefix referencing the compound gesture where it comes from. For example, in Figure 46, the partial gesture S1 must be referenced as MultiStrokeV.S1 if it is to be used in other cards while it is simply referenced "S1" in the traits of S2 because they are part of the same compound. When referencing a gesture for one of its traits, the gesture must be specified as well as its trait, e.g., "Start: S1.End" in Figure 46. However, if the same trait is targeted, you should not specify it but only the gesture from which it comes for the sake of brevity, e.g. "Duration: S1" corresponds to "Duration: S1.Duration" in Figure 46.

It is important to specify the temporal relation of the partial gestures between them, in order to know in which order they are performed. To do this, the authors have established a spatial arrangement of the partial gestures within the compound gesture cards depending on that temporality. Figure 47 shows the idea. Single-stroke gestures are connected to each other (Figure 47 (a)), multi-stroke gestures are separated by a wiggly line (Figure 47 (b)). These lines indicate a pause between two strokes and a time indication can be added. Multi-touch gestures are represented by placing the gestures performed simultaneously next to each other. (Figure

47 (c) and (d)). Note that wiggly lines can also be added for multi-touch gestures to indicate a delay between two partial gestures.

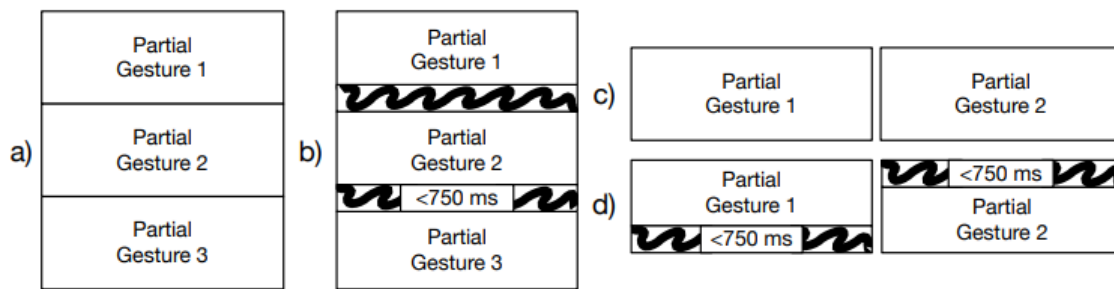


Figure 47: Partial gestures position based on their temporal relation. a) Single-stroke gestures, b) Multi-stroke gestures, c) Multi-touch gestures, d) Multi-touch gestures with delays (Hesenius & Gruhn, 2019).

It is possible to link gestures to virtual objects present in the application. In this case the target object must be specified as a value for a trait via the keyword "Object". To distinguish between different objects, an identifier in brackets is added after the "Object" keyword, e.g., "Object (Star)". These identifiers are unique in the context of the current GestureCard which means that a reuse of the same identifier in this GestureCard refers to the same object. It is therefore possible to refer to several objects in the same GestureCard, as shown in Figure 48 where The gesture starts on the first object (Start: Object(Source)) and ends on the second (End: Object(Sink)). They use the keyword "FREE" as a graphic representation to indicate that the shape of the gesture to go from one object to another does not matter.

Connect	FREE	Start: Object(Source) End: Object(Sink)
		Object(Source), Object(Sink)

Figure 48: Gesture linked to two objects (Hesenius & Gruhn, 2019).

2.3.5.4 GestureCards' Limitations

Gesturecards is a very complete notation that allows to describe a large set of gestures. However, Hesenius & Gruhn (2019) identified two potential drawbacks: overly large gesturecards and difficulties in describing spatial gestures.

The spatial arrangement allowing to specify the temporal relations between the partial gestures implies that one can quickly end up with compound gestures that take a lot of space. This is especially the case for multi-touch gestures where the partial gestures are placed next to each other, whereas single- and multi-stroke gestures are placed one above the other, avoiding this

problem. To overcome this, it is possible for parallel gesture strokes to be combined in the same graphical representation if they share the same traits.

The second problem comes from the fact that the notation as it is hardly allows to describe spatial gestures. The first reason identified is that representing a three-dimensional movement on a two-dimensional support is a difficult task. The second reason is that spatial gestures include static poses that cannot be described with arrows, as is the case for gestures on a touch screen for example. As explained in the presentation of the problem, this is a limitation that we want to solve in order to take into account an even larger set of gestures.

2.3.6 *Discussion*

We will now summarize what we have learned by detailing each of these notations. The goal is to see what these methods allow to do and not to do to represent gestures and what we can keep in order to imagine a new notation allowing to describe a larger number of gestures.

We will present the results of two studies, conducted by Hesenius & Gruhn (2019). The first one aims to find out if their GestureCards notation is more readable and concise than other textual notations, in this case, they compared it to GeForMT. The second study seeks to compare GestureCards, GeForMT, and Monox for conciseness, accuracy, and participants' personal preferences. Other studies have been conducted by other researchers, but we decided to keep the results of those presented as they are the most recent. For example, Kammer et al. (2015) show in their research that GeForMT is more readable and consistent than other existing textual notations at the time of the study.

The first study consisted of having subjects perform gestures by showing them gestures described in the tested notations. One group was tested with GestureCards, the other with GeForMT, both groups had same gestures to identify. Subjects had access to a cheatsheet if needed and had to rate their degree of certainty on a Likert scale after each gesture performed.

The results of the study are presented in Table 3. They distinguish time per gesture and viewing time per gesture because subjects using GeForMT wasted a lot of time looking at the cheatsheet. This time was therefore removed from the Viewing Time column to avoid skewing the analysis. It shows that GestureCards is more readable and easier to understand than GeForMT. It was statistically shown that the group using GestureCards identified the gestures better than the other group and that they used the cheatsheet less. In addition, the GestureCards group showed a little more confidence in their answers and in terms of duration, they also

performed better than with GeForMT. From these results and taking into account the participants' comments, the authors conclude that the GeForMT group had difficulty translating the keywords and operators into a movement (Hesenius & Gruhn, 2019).

Participant	Cheatsheet Calls	Median of Confidence	Correctly identified Gestures	Avg. Time per Gesture (s)	Avg. Viewing Time per Gesture (s)
GeForMT1	18	1	3	101	62
GeForMT2	8	2	6	71	41
GeForMT3	14	2	5	87	54
GeForMT4	27	2,5	5	117	70
GeForMT5	20	1,5	5	99	67
GeForMT6	21	2	1	88	41
GC1	0	2	8	26	24
GC2	0	1,5	8	36	33
GC3	2	1	6	44	35
GC4	1	1	6	30	24
GC5	2	2	5	36	27
GC6	0	1,5	8	30	28

Table 3: Aggregated results per participant. Confidences values range from 1 (confident) to 4 (uncertain) (Hesenius & Gruhn, 2019).

For the second study, the selected participants are students in mobile software engineering who are therefore already experienced in the field. During this one, a video in which the selected gestures were performed was shown to the participants. The participants were given a questionnaire on which the expressions of the gestures presented in the video were noted in the three notations compared, i.e., GestureCards, GeForMT and Monox. Subjects were asked to rate each gesture for each notation on a Likert scale for conciseness and accuracy. Participants were also asked to rank the three notations in order of preference.

Results are shown in Figure 49 where the x-axis shows the four gestures tested and the y-axis the Likert scale where "+++" is the best positive score and "---" the worst negative score. Their first observation is that GeForMT and Monox have a lot of fluctuation in their results depending on the gesture described. This comes from a fact already evoked in our work: purely textual or graphical notation is very suitable to describe some gestures but are unable to define other gestures with precision. Secondly, GestureCards received more stable scores on the globality of the gestures, with a high average score although Monox or GeForMT were better scored on some gestures, but never both. This suggests that GestureCards shows more consistent results, for both conciseness and exactness, than the other notations and that it is better suited to describe a large number of gestures. And finally, GestureCards received the highest ranking for participants' preferences (Hesenius & Gruhn, 2019).

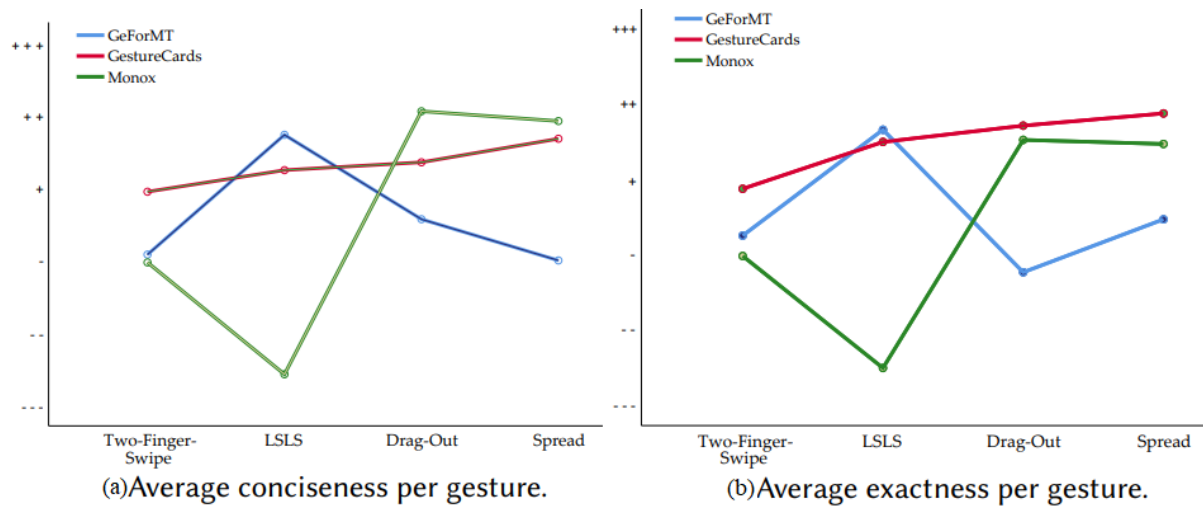


Figure 49: Results of the 2nd study (Hesenius & Gruhn, 2019).

From these two studies we can see that a hybrid notation is more efficient to describe a large set of gestures, either for conciseness or accuracy. Moreover, the results showed that GestureCards was more consistent than purely textual or graphical notations. All this confirms the fact that a hybrid notation allows to take advantage of the benefits of the two other types of notations and has a better adaptation to take into account a larger number of gestures.

In the next chapter, we develop our ideas for creating a new notation that should be applicable to a larger set of gestures. The goal is to meet the conditions that make it a human-readable notation that should facilitate communication in heterogeneous project groups.

Chapter 3 - Development of new representation for spatial gestures

3.1 Requirements and objectives

As we have seen in 2.3, there are many notations, more or less efficient, allowing to describe interaction gestures. However, few of them meet the need that we have identified in this work: facilitate communication between stakeholders with different backgrounds who do not usually use a common language for their work. To meet this need, the notation must find a balance between conciseness and readability. On the one hand, some notations are considered concise but achieve this by adding symbols or abbreviations that make the syntax not readable for a person who is not familiar with it. On the other hand, some notations use simple and easily understandable ways to describe the interaction gestures, but this leads to long expressions and therefore does not meet the goal of conciseness, leading to ambiguity in the interpretation of it. We also observed that for the sake of conciseness, some notations did not allow for the description of all the gestures due to a lack of detail in their expressions. This problem leads to identical expressions for two different gestures, and conversely, the same gesture can be described in several different ways, which can lead to confusion. Hesenius (2019) explains for example that this is the case for Proton++ because it only distinguishes three different touch actions.

Moreover, as explained above, the representations that have been presented have problems in describing spatial gestures. Some of them try to do so, like EventHurdle (J.-W. Kim & Nam, 2013), in which it is possible to connect two hurdles coming from different workspaces thanks to a time-wise relation, but it is necessary to have a good knowledge of the representation method to use and understand it. GestureCards also allows users to define gestures in three dimensions, but as explained in chapter 2.3.5.4, it is difficult to graphically represent a 3D gesture on a 2D support and a static pose is not describable with arrows due to its static nature.

Based on these observations, our notation should meet the following objectives:

- The description of gestures must be **concise**.
- The gesture description must be **readable**. By this we mean that a person with no specific training in this area can quickly learn to understand the expressions of interaction gestures.

- The description of gestures must be **adapted to spatial movements** performed with any part of the body for both static and dynamic gestures.

It should also be remembered that the starting point of this work is the lack of communication tools within heterogeneous application development teams. The origin of this communication problem is identified between the UI-Design and Engineering phases. We could therefore say that our new representation solution is addressed to designers and engineers and that the goal is to provide them with a common language in order to avoid any information transmission and interpretation problems, as explained in section 1.2.3.

However, as shown in chapter 1.2, as development processes have evolved towards agile work methods and teams are becoming more diverse in their skills, the involvement of all stakeholders in application development is necessary at all stages. Therefore, our solution is not only intended for engineers and designers but for all these stakeholders who must also understand the chosen interaction methods and more precisely the interaction gestures that will compose the application.

3.2 Conciseness and readability

In order to fulfill these effectiveness criteria, we looked at what had already been created before. Therefore, based on the discussion in section 2.3.6 and in view of the studies conducted by Hesenius & Gruhn (2019), we chose to keep the main structure of GestureCards. Indeed, their experiments show that hybrid notations, combining a textual and a graphical description, are more efficient for gesture description in terms of readability and conciseness than the other notations tested. Moreover, GestureCards proved to be more consistent, in terms of conciseness and precision, on all the gestures presented, whereas the other notations were only efficient for some gestures.

As a reminder, the structure is represented in Figure 42 and consists of 4 parts: The name of the gesture, the graphical representation, the traits necessary to identify the gesture and the return values which are values that can be useful for further processing in the application. Although Hesenius & Gruhn (2019) specify that it is difficult to represent spatial gestures with GestureCards, they point out that the list of traits they established for surface gestures has already been designed to accommodate 3D gestures. Even though not all the features thought for gestures on a surface will be useful for describing spatial gestures, some may prove to be required. At the graphic level, we thought about several possibilities to represent the gestures

as well as possible while respecting the conciseness required for good communication. Our choices are presented in the following sections.

3.3 GestureCards extension for spatial gestures (3D)

Hesenius (2019) explains that describing 3D gestures is challenging because it requires a much higher degree of precision than for surface gestures. Indeed, besides the description of the movement itself, other parts of the body can also play a role in the identification of a movement. It is therefore also important to describe them and the relationships between them. For example, to define a hand gesture, it is necessary to describe the fingers that are stretched out as well as those that are bent. In addition, the problem of 2D support still needs to be solved.

In this section, we present the method developed by Hesenius (2019) to represent spatial gestures.

3.3.1 *Representation for static poses by Hesenius*

In his thesis, Hesenius (2019) brings his thinking to find a solution to the representation of spatial gestures. This one is realized for gestures with hands only. First of all, he talks about static poses for which he proposes a representation with a photo of the hand, which has advantages and disadvantages. The main advantage is that this solution makes the gesture very recognizable because of its resemblance to reality. Another advantage is that taking a picture is quickly done and almost everyone has a smartphone with satisfactory quality. However, the integration of these photos into GestureCards can be problematic. First of all, the perspective of the photo must be appropriate, requiring some skill. Secondly, the image has to be processed, i.e. the background has to be removed because any other information than the hand is superfluous. Finally, the image of the hand must be scaled appropriately for the GestureCard. An example for this alternative is shown in Figure 50. Note that the third part on GestureCards is empty because no additional properties need to be specified in these cases. But as mentioned before, it is totally possible to add traits to the described gestures. In the case of static poses, it might be relevant to add the trait "Duration: Long" to mean that the pose must be held for more than 2s to be recognized.



Figure 50: Representation of hand pose in a GestureCard (Hesenius, 2019).

Another alternative would be to represent only a silhouette of the hand, by blackening it as in Figure 51. For Hesenius (2019), it has the advantage that it can be manipulated more easily without it being immediately noticeable. Moreover, the hand of the collaborator is not identifiable following the blackening of this one. The disadvantage of this method is that it only shows the outline of the hand and hides the details inside the silhouette. It is also difficult to tell if it is the right or left hand, front or back.



Figure 51: Representation of the hand silhouette in a GestureCard (Hesenius, 2019).

A third option is to use a hand model that mimics the human hand. The researcher proposed to represent it via the use of Leap Motion, which is a camera that recognizes gestures made with hands and fingers, presented in section 2.1.3. Its Software Development Kit (SDK) provides different predefined hand models, with varying levels of detail. In Figure 52, Hesenius (2019) used the Capsule model from Leap Motion. The blue balls represent the joints and the links between them are the bones. This technique allows to keep only the essential details for the recognition of the gesture and the anonymity of the employee who provided the hand is respected. Other Leap Motion models can be used depending on the preferences and the details needed in the representation of the hand.

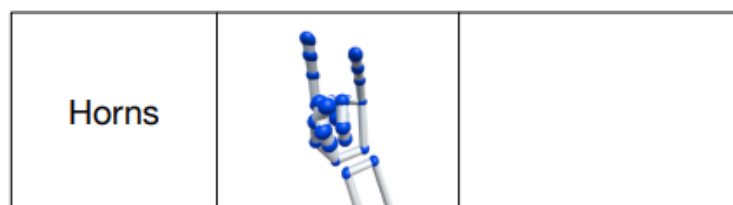


Figure 52: representation of the hand with the Capsule model of Leap Motion (Hesenius, 2019).

A last possibility explored by Hesenius (2019) is to reproduce the pose in a video. In terms of accuracy of the description, this is probably the most efficient method, but it would require more technical means than the solutions proposed above. Moreover, in the case of static poses,

the question of whether the path to the pose is an integral part of the gesture will arise, which may lead to ambiguity.

3.3.2 Representation for dynamic gestures by Hesenius

For the representation of dynamic gestures, Hesenius (2019) also focuses on gestures made with the hands. He illustrates his method with the example of a drag and drop gesture on an object. Figure 53 illustrates this in a GestureCard. Unfortunately, we can't see it in this report, but its idea is to incorporate a graphic animation of the hand, always represented with Leap Motion. In the GestureCard, the first partial gesture "Grab" is the action of closing the fist and the line "Location: Object" indicates that it must be performed on an object. The second partial gesture is the action of moving the object previously grabbed, the graphic representation is "FREE" because the path taken is not important in this case, it is only necessary to move the object. And finally, "Drop" is the action of opening the hand again, as if you were releasing the object. The return values of the compound gesture specify which object is grabbed and where it is dropped. Since the animations cannot be presented in this document, the two partial gestures have been decomposed into several successive images in Figure 54.

DragNDrop		
Grab		Location: Object
Drag	FREE	
Drop		
Grab.Object, Drop.Location		

Figure 53: Drag and drop gesture in GestureCards (Hesenius 2019).

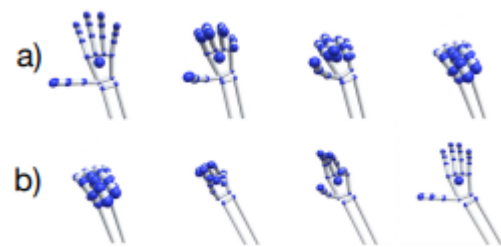


Figure 54: Decomposition of the movements: a) "Grab and b) "Drop" for the compound gesture "DragNDrop" (Hesenius 2019).

The advantages and disadvantages of this method are almost the same as for the proposal to represent static poses through a video. On the one hand, it gives a very precise description of the performed gestures but on the other hand it requires high technical means. Indeed, these representations are made via the Leap Motion camera, so it requires to make the movement in front of the camera so that it tracks and records it. Then, it is necessary to recut the recording to retain only the necessary part of the movement. And finally, to share the GestureCard, it

requires to present the graphic part as a video file, which is larger than a simple image to share. To this, although marginal for some large projects, we must add the cost of acquisition of the device which is between \$110 and \$130 (*Tracking | Leap Motion Controller | Ultraleap*, n.d.).

3.4 Our proposition for spatial gesture description

As mentioned in section 3.2, we will take the basic structure of GestureCards. For the graphical representation, following the alternatives imagined by Hesenius (2019), we choose to model the human body via a 3D modeling software. For the choice of the software, we explored two possibilities.

3.4.1 Choice for the modeling software

The first was Blender which is an open-source 3D software developed by a non for profit organization, the Blender Foundation, with the single goal of providing usable 3D software to anyone in the world. Companies like Amazon, Microsoft, Nvidia, Ubisoft and other big names are investing in the application development fund (bailyldesign, n.d.). The advantage of Blender is that its efficiency is recognized by professionals in the modeling field and that it allows a very high level of detail and customization to represent a gesture. Moreover, it is possible to record animated sequences directly in the software. However, the counterpart of its efficiency is that it is not easy to use and not everyone can master the skills necessary to properly use its features in a few dozen minutes. Figure 55 shows the basic interface when you open the application. The first impression when we access it is the feeling of being lost in a sea of tabs and drop-down menus. A training is therefore necessary to be able to handle it as desired.

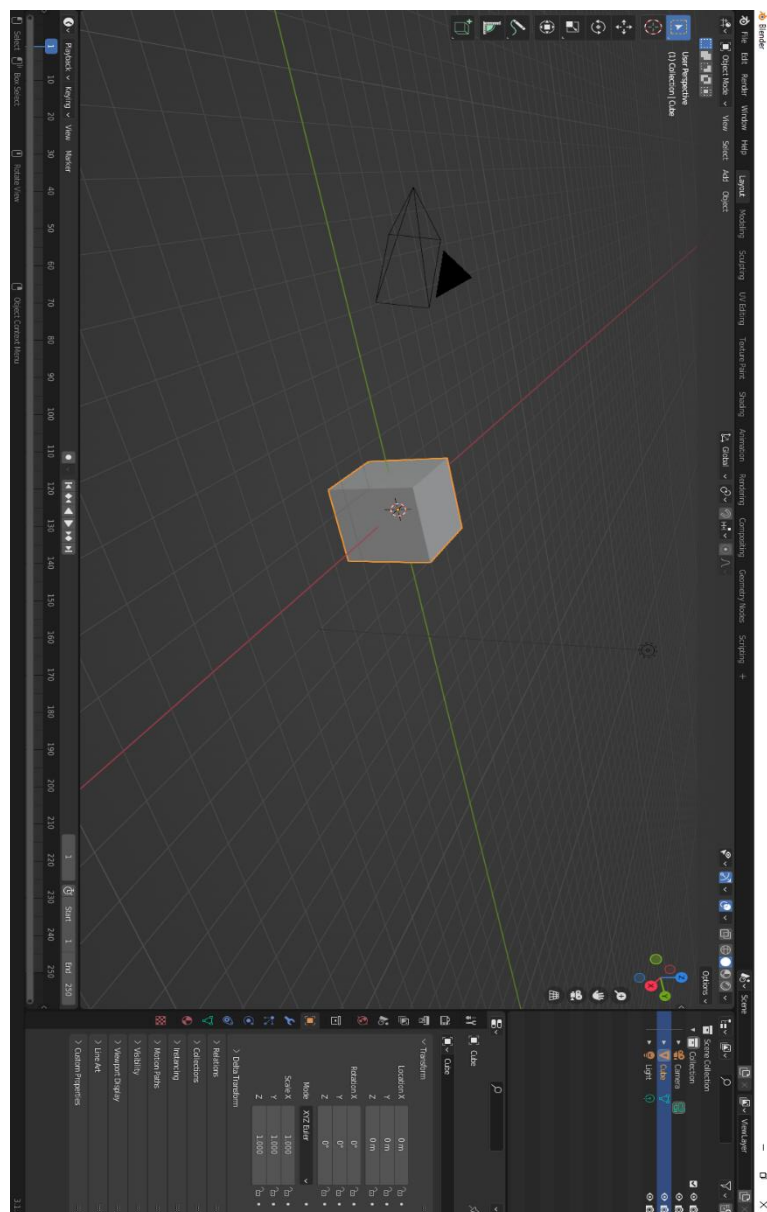


Figure 55: Blender's interface

The second alternative is to use Magic Poser which is a web application and also available as a smartphone app. As shown in Figure 56, which shows the interface when arriving on the application, it allows to model an avatar, of human form, in a 3D space. Compared to Blender, Magic Poser allows less detail in the image, but its features are sufficient to model interaction gestures. The interface is much easier to use than Blender's. From our experience, it took us 15 to 20 minutes to get the hang of its features. However, this application does not allow you to record image sequences, it is only limited to modeling human body poses.

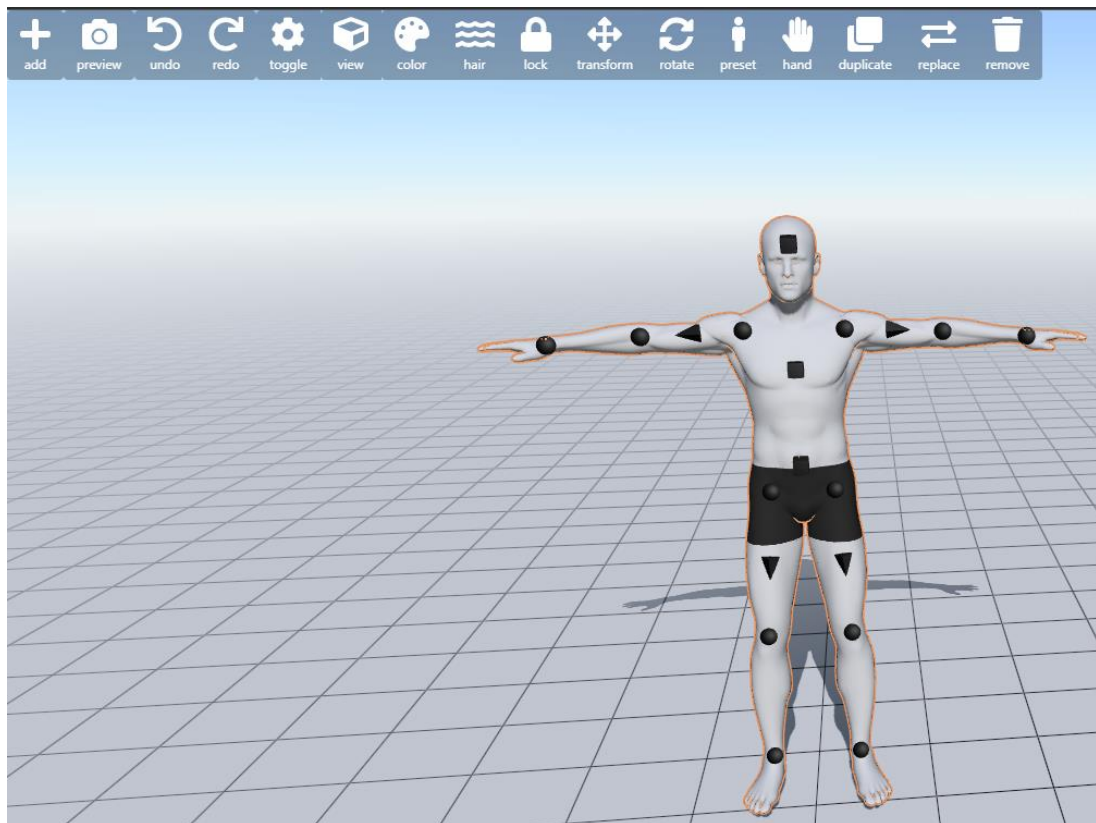


Figure 56: Magic Poser's interface.

The choice is between privileging the detail and the multiple possibilities offered by Blender or the simplicity and the fast handling of Magic Poser. We have chosen to use Magic Poser to represent gestures. We think that the skills required to use Blender are a too big barrier in the approach of our thesis which wants to facilitate the communication between the different stakeholders of a project group. The ease of use is therefore a major asset and with Magic Poser, any member of the team, regardless of his professional background, should be able to handle it.

3.4.2 Representing gestures with Magic Poser within GestureCards

With our solution, it is important to distinguish between static poses and dynamic gestures. As you will see below, the representation methods are somewhat different for these two types of "gestures" (we make a shortcut by calling static poses "gestures" which should by nature involve movement). Indeed, static poses are well enough represented with a single image made with Magic Poser, while dynamic gestures need to be decomposed to be correctly represented. For the latter, we will use GIFs which allow to sequence several images and thus to restore an illusion of dynamic gesture (see the end of this section).

With this choice, static poses are easy to represent and integrate in a GestureCard. Figure 57 shows a static pose with Magic Poser as a graphic representation. For this example, we show the representation of a movement asking to lean the chest to the left. In this case, it is the holding of the pose that counts for the application, not the path taken to reach it.

One of the disadvantages of this software is the representation of the hands. The level of detail for the position of the fingers is quite low but there is a basic set of hand positions that can be used as for example the thumbs up in Figure 58. We believe that these predefined hand poses may be adequate to describe a sufficiently large set of gestures. The possibility of modifying hand's shape is available but is not very developed. Both descriptions have the "Duration: >2s" trait because we assume that a pose must be held for at least 2s to be recognized as such. And we can specify the use of the right hand with the line "Tool: RightHand" in Figure 58.

LeanChestLeft		Duration: >2s
		Location

Figure 57: Static pose LeanChestLeft

Thumbs up		Duration : >2s
		Tool : RightHand

Figure 58: Thumbs up

As mentioned before, the question of how to include dynamic gestures in GestureCards still arises. To meet this need, we first thought of breaking down a dynamic gesture into intermediate poses and simply aligning them in the GestureCard. However, this could make them overly large, and we could end up with gesture descriptions taking up an entire A4 page. For example, Figure 59 shows a dynamic gesture indicating "no" with both hands crossing at belly level for which we specify the feature "Multiplicity: 3" to indicate that the gesture sequence must be repeated 3 times to be recognized. This solution is viable for some gestures, which are considered simple, as is the case here where it can be broken down into 3 parts.

CrossHandsNo		Multiplicity: 3

Figure 59: "No" gesture by crossing hands.

For more complete gestures, we propose the use of GIF's which allow to animate a sequence of images, and thus in a certain way to reproduce the movement. This is similar to the solution proposed in section 3.3.2, which was to use a video for the graphic representation. But the GIF has the advantage of being very easy to use, several websites allow you to simply upload your images and assemble them to form the animated sequence. Moreover, a GIF file is less voluminous in terms of bytes, and therefore easier to share within a team than a file with a video of the gesture performed. An example, using the "CrossHandsNo" gesture from Figure 59, is shown in Figure 60 with the use of a GIF. MS Word allows you to embed GIFs in reports, making them easily shareable. This report being in PDF format, we attach an MS Word file named as an appendix containing the GestureCards with animated GIFs.

CrossHandsNo		Multiplicity: 3

Figure 60: Gesture "CrossHandsNo" using a GIF.

Another example has been made in Figure 61. This one is a more complex gesture, where we imagine that a user starts from a standing position, stoops down to take an impulse and ends up jumping with arms in the air. It's a movement where the whole body comes into action and we've broken it down into five images, assembled in a GIF rendering the whole gesture. In this case, the graphic representation is enough, and we don't have to add any line for the identification of the gesture.

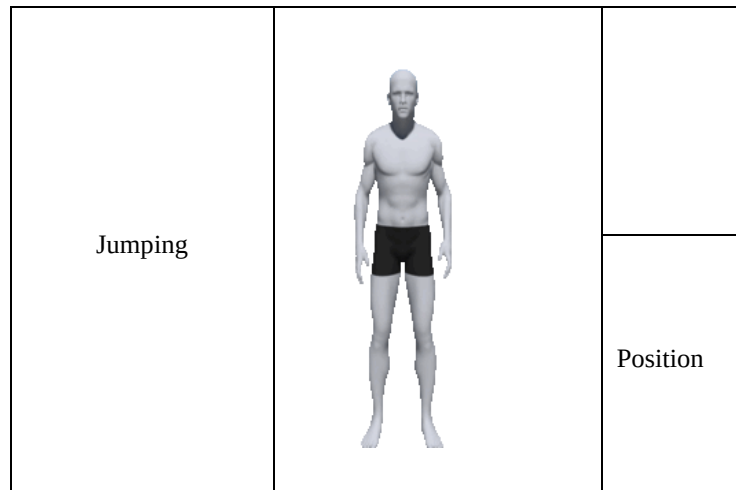


Figure 61 : Gesture “Jump” with a GIF

3.4.3 Integrating Xia et al.’s factors in GestureCards

As mentioned in section 2.2.3, we think it may be interesting to give designers and engineers the possibility to specify some factors of the Xia et al.’s (2022) framework in gesturecards. As explained, the latter is intended to support the creation of a gesture vocabulary in order to keep all aspects of gestures in mind. However, the communication on interaction gestures could be improved by having the possibility to integrate these factors in gesturecards.

A reminder of these different factors is shown in Figure 62. Probably not all of them are relevant, but in particular we think that situational factors can make some gesturecards clearer, especially when certain gestures are restricted to certain situations.

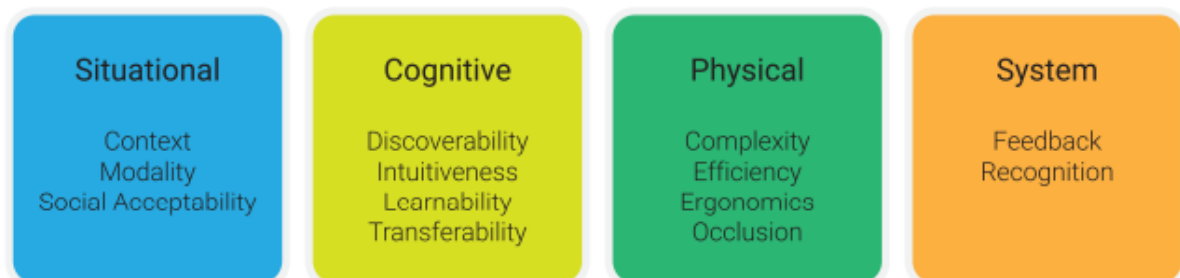


Figure 62 : The 13 factors essential for gesture design identified by Xia et al. (2022).

For example, when describing interaction gestures that are intended to be performed while driving a car, it may be interesting to directly specify the context in which the user is located. Just like the section dedicated to features in GestureCards, we could add a space dedicated to these factors. Figure 63 shows an example of a gesturecard for a gesture where it is important to specify some factors. We take the example of a mid-air swipe with the hand to the right that should be usable by a driver of a car. It is therefore interesting to specify the context by adding

the line "Context: Driving", because the GIF alone presents a standing person performing a swipe to the right that could fit in any situation. It is also interesting to specify that the gesture is performed in front of a car screen with the line "Modality: Car Screen". Indeed, Hu et al. (2015) have shown that users adapt the size of their gestures to the size of the display they are facing. Engineers can therefore use this information to adapt the recognition of this gesture.

SwipeRight		Speed : Fast
Context: Driving Modality: Car screen		Speed

Figure 63: SwipeRight while driving a car

We give designers and engineers the freedom to add one or more factors to their gesturecards. Note that it works the same way as with traits, i.e. to specify a factor you need a "factor: value" pair. Since each factor has so many possibilities, we don't think it is necessary to make an exhaustive list of "values" for each factor. The idea is that the creator of a gesturecard chooses the term that seems most appropriate to communicate a factor with his colleagues or, the team members agree to establish a list related to their project and use it throughout the whole project.

3.5 Cost to create a card with our representation

In this section, we will evaluate the cost in time and complexity necessary to describe a gesture with our solution. The idea is to describe the necessary steps in order to evaluate the costs.

3.5.1 Step 1: Modeling in Magic Poser

The first step in the creation process is to perform the modeling via Magic Poser. This is probably the most complex step, especially if this is the first time the user has tried it. Indeed, the tool is very easy to use, but as mentioned earlier, there is a short learning period of about 15 to 20 minutes. Once the tool is mastered and depending on the complexity of the gesture to be modeled, the time to reach the desired pose can take 30 seconds if you just need a pose where the avatar has his arms in the air as in Figure 64 a) , and it can take up to 2 minutes for a more complex pose as for a crouching position in Figure 64 b).



Figure 64: Example of gestures made with Magic Poser. a) Right arm in the air; b) Crouching position.

Once the desired pose is well represented, a screenshot of the avatar should be taken so that it can be processed in the next steps. However, a single image is sufficient to represent a static pose, but for dynamic gestures, it will be necessary to break down the movement into several steps and therefore to have a screenshot of each intermediate step. For example, for a gesture where the user describes a circle with his palm as in Figure 65 , it is necessary to decompose the movement in 6 steps minimum to reach a sufficient degree of precision. It should be noted that each pose does not take 2 minutes to achieve. Indeed, once the first pose is done, it is quite quick to move the model to the second intermediate pose and so on for the following ones. However, one difficulty we encountered with GIFs is that it requires screenshots of the same format, in size and in centering of the model, in order to have a more readable result. For centering, Magic Poser allows you to zoom in and out as well as rotate the view 360 degrees around the model.



Figure 65: Steps for a gesture describing a circle with the palm in Magic Poser.

In terms of total time, for this first step, we estimate that it takes up to one minute for a simple static pose, up to 3 minutes for a complex static pose and to realize the screen captures for dynamic gesture as in Figure 65, we can count up to 6 minutes.

3.5.2 Step 2: Removing the background of the image

This second step consists in removing the background of the model so that only the "avatar" remains on the image. The background on Magic Poser is not that disturbing but we think that for the sake of detail it is recommended to remove it. To do this, we used the web application remove.bg that allows you to upload an image and re-download it in its new version with the background removed. With a "normal" internet connection, it takes no more than 20 seconds per image. You can see the result of the images in Figure 65 without background in Figure 66.



Figure 66: Steps for gesture describing a circle with the palm without backgrounds.

If it is a static pose, the modeling stops at this step and we can already go to step 4 where we have to integrate the image in the gesturecard. For dynamic movements, step 3 consists in creating the GIF to assemble the images of the decomposed gesture.

3.5.3 Step 3: Creating the GIF

To create the GIF, we again use a web application to select the images to be assembled to form the animation. We used ezgif.com which allows to upload the images of the gesture, for example the 6 images in figure 66. Figure 67 shows the interface where you choose the images to be assembled. Once selected, we arrive at the interface of the settings to customize the GIF, which you can see in Figure 68. Basically, the images are sorted in alphabetical order

(according to their file name), but you can change the order by dragging and dropping the images. Some changes are necessary to have a better readability like increasing the delay between each image which is set automatically at 0.2s. After several tests, we chose to set 0.4s as default for our GIFs. During the study made in Chapter 4, we also notified that it was judicious to put a longer delay on the last image for movements which are not repetitive, in order to mark the fact that they are realized only once. To do this, we simply select a delay of 0.8s on the last image, for example under image 6 in Figure 68. Note that the values are displayed in hundredths of a second in the interface and that $0.2s = 20$ hundredths of a second. It is also important to check the "don't stack frames" box so that the images will appear one by one.

Animated GIF Maker

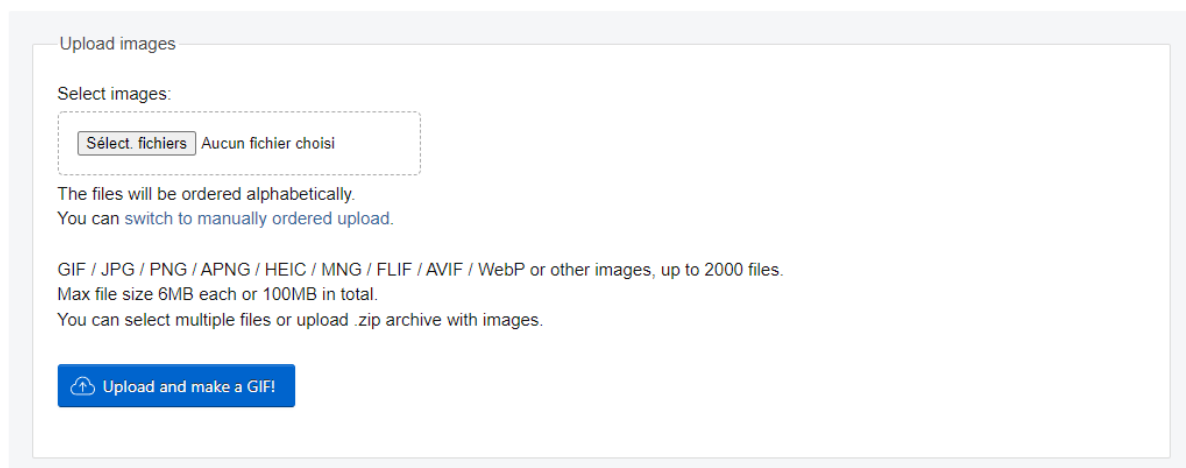


Figure 67: ezgif uploading interface from ezgif.com.

Once the parameters are selected, click on "Make a GIF!" to create the GIF and have a preview. If it fits your expectations, you can download it and add it to the gesturecard in the next step. If the images have been well processed and have a similar size and the same centering, this step is very simple and fast to achieve. We counted 3 minutes maximum to realize this step.

3.5.4 Step 4: Creating the gesturecard

This last step consists in assembling the different parts of the gesturecard of the represented gesture. For this purpose, we have a template that takes the basic structure of gesturecards that was explained earlier. This one is visible in Figure 69.

Animated GIF Maker

(drag and drop frames to change order)

1 2 3 4 5 6

Delay: 40 Delay: 40 Delay: 40 Delay: 40 Delay: 40 Delay: 80

skip copy skip copy skip copy skip copy skip copy skip copy

Toggle a range of frames:

From: 1 To: 5 Skip Enable

GIF options:

Delay time: 40 (in 1/100 of second, changing this value will reset delay for all frames)

Loop count: (Empty - loop forever)

☐ use global colormap (Use the same set of colors for all frames to reduce file size)

Effects:

☐ crossfade frames

☒ don't stack frames (Remove the frame when it's time to display next one, use for images with transparent background)

☐ use the first frame as background

Make a GIF!

Figure 68: ezgif settings interface from ezgif.com.

This step is very simple because it is only needed copy/paste your image or GIF made in the previous steps and fill in the other sections with the necessary data to describe the gesture. We counted that this one should take less than 3 minutes to properly layout the gesturecard.

Name of the gesture	Image or GIF of the gesture	Traits
Factors to be specified		Return Values

Figure 69: Structure of the gesturecard

3.5.5 *Total cost*

We were able to evaluate the cost necessary to create these gesturecards. In terms of time, which is the easiest variable to calculate, for a complex movement like making a circle with the palm, we counted 6 minutes for the first step. For the second step, we count 30 seconds for each image to be processed, so 3 minutes in our example. The creation of the GIF takes no more than 3 minutes. And the integration of all the elements in the gesturecard took us 3 minutes. This gives us a total of 15 minutes to create a gesturecard that we consider complex. This means that some may take less than half that time for simpler gestures like static poses. However, let's remind that there is a small learning phase to use Magic Poser, which we have estimated at between 15 and 20 minutes in order to be able to use its functionalities correctly.

In terms of difficulty, the most complicated part is to create the template and make screenshots that will fit the GIF format, i.e., that it is of a similar format and that the centering of the "avatar" is similar too. If these two steps are done correctly, the rest should be quick and easy.

Chapter 4 - Studies to test the representation

4.1 Study 1: Evaluation of readability

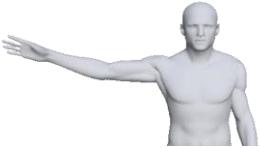
This study was designed to test the readability of our representation. To do this, we will present descriptions of gestures in our representation to participants by asking to match them with their literal description from a proposed list.

4.1.1 Study design

To do this study, we formulated a questionnaire on Google Form that was sent to people without restriction on their profile. We were able to administer our test to N=89 people. We did not pre-sort the participants because we believe that if our solution allows the general public to recognize gestures, this should also be the case for all stakeholders of an application development team. Moreover, we were confident in the fact that our solution is easily readable.

Before starting the test, we briefly explained how the gestures were described with the gesturecards. Then, the test consisted in showing a gesturecard whose name was removed so as not to give them a clue. Based on it, among a predefined list of literal gesture description names, they had to select the one that seemed to correspond to the shown gesturecard. We also added names to the list that did not correspond to any gesturecard in order to make the exercise more complicated and confusing. There were exactly seven gesturecards for twelve possible names. We also gave the choice "I can't find a matching description" in case they felt that no literal description matched. You can find the gestures shown in Table 4 associated with their corresponding literal description.

The only variable we study in this test is correctness, meaning that our results will be presented as a percentage of correct answers. When the participants are wrong, we will be able to observe what their choices were other than the corresponding one.

	Gesturecard					Literal description
1				Duration : >2s		Crouched position
2				Tool: RightArm	Location	Arm elevation
3				Tool: HandPalm	Location, Speed	Draw a circle with the palm
4					Position	Jumping
5				Repetition: 2		Saying no with two hands

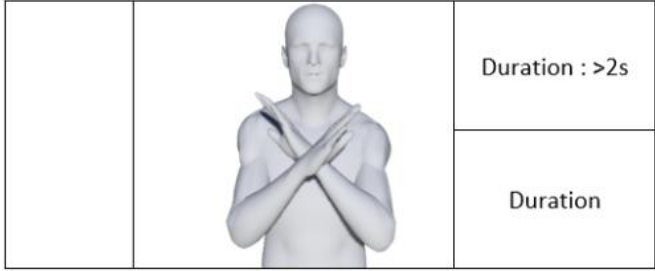
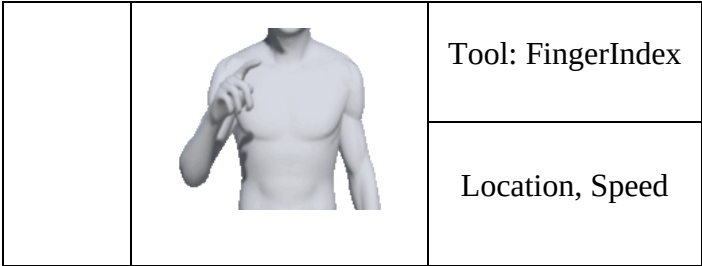
6	 Duration : >2s Duration	Crossed arms in X
7	 Tool: FingerIndex Location, Speed	Draw an X with a finger
8	No corresponding card	Bent over position
9	No corresponding card	Draw a square with a finger
10	No corresponding card	Hand sweep to the right
11	No corresponding card	Standing up
12	No corresponding card	Crossed arms

Table 4: Gestures shows and their corresponding literal description for the study.

4.1.2 Results

Our results are displayed as pie charts in which the percentage of each literal name chosen by gesturecard is shown. You can find these results in Figure 70. The gesture number 1 - Crouched position was correctly recognized at 98.9%. The N°2 gesture - Arm elevation was recognized at 66.3%, and 31.5% of the participants matched it with the hand sweep to the right. The gesture N°3 - Draw a circle with the palm was recognized by 82%, 12.4% of the participants matched it with the hand sweep to the right and 4.5% did not recognize it in the list. 93.3% of the participants correctly recognized the gesture N°4 - Jumping. The gesture N°5 - Saying no with 2 hands was correctly matched by 77.5% but 16.9% did not find a corresponding description

for it. The gesture N°6 - Crossed arms in X was found at 94.4% and the gesture N°7 - Draw an X with a finger at 92.1%.

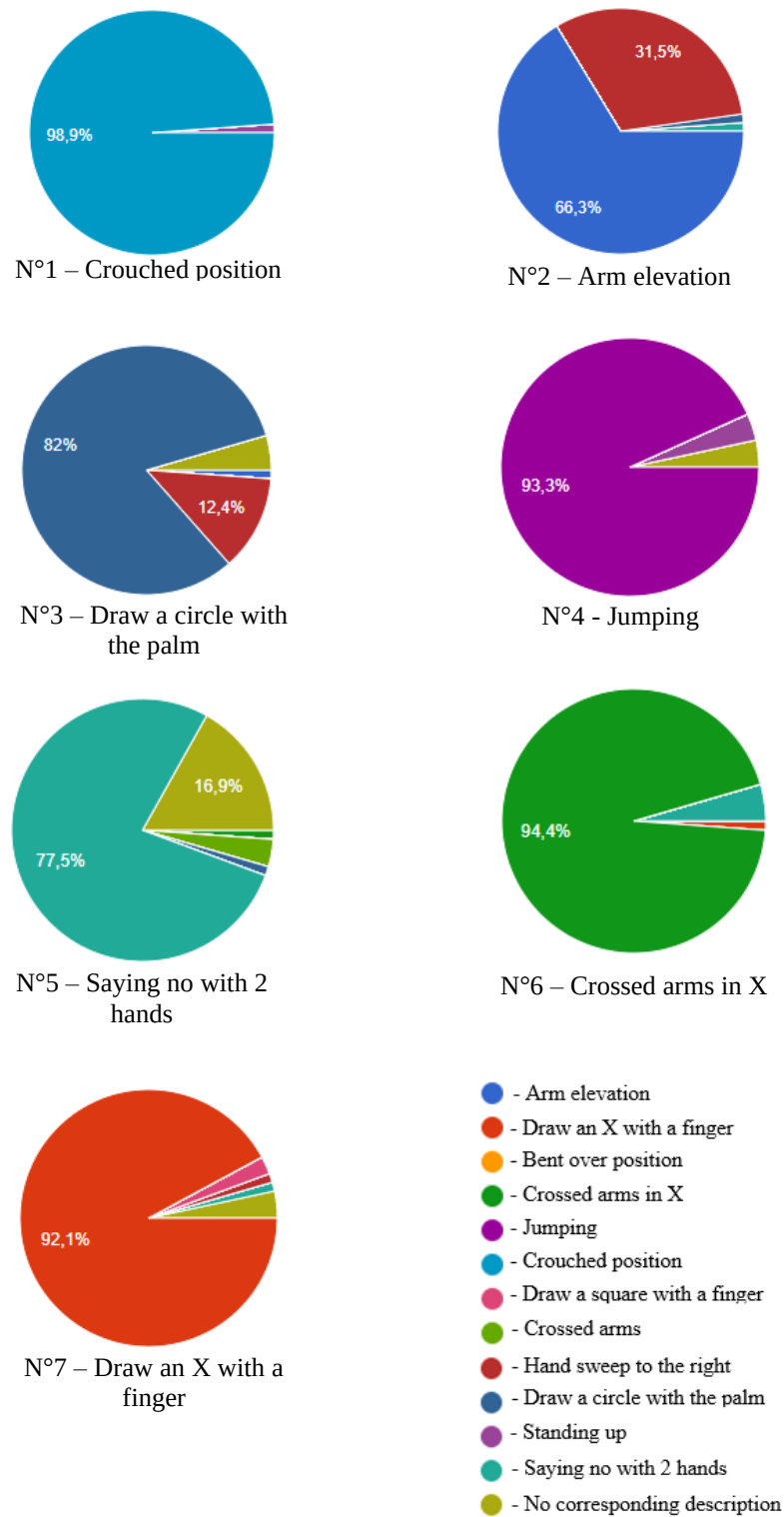


Figure 70: Results for the study per gesture.

4.1.3 Discussion

Our first observation is that the results of our test are satisfactory because we obtain an average recognition rate of 86.35%. We note that the best recognized gestures are those that are static poses (N°1 and N°6), probably because they are simple images and not GIFs which seems to be more complicated to interpret.

The gesture that caused the most difficulty was N°2 - Arm elevation where 31.5% of the participants thought it was a sweep of the hand to the right. The fact is that they are not wrong, and the problem comes more from the list of choices we made. Indeed, when we added the choice "Hand sweep to the right", we were thinking more of a hand movement from the left to the right in front of the character, as if he was pushing away an object in front of him. So, we should have been more precise in the title of the choices. Another solution would have been to make the GIF associated with the arm elevation better and marking the stop of the movement between the last and the first image. To do this, the GIF maker allows us to choose the time interval between each image and we can establish that the last image, when the character has the arm at the highest, remains displayed longer. Initially, we set a default delay of 0.4s between each image which we thought was comfortable to understand the gesture. In Figure 71, we doubled the time of appearance (0.8s) of the last image to mark the end of the gesture. We think this is a good solution for movements that do not require repetition and are done in one go.

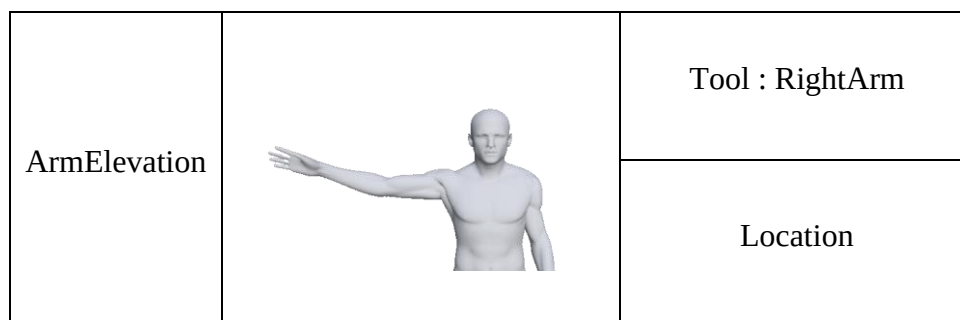


Figure 71: ArmElevation's gesturecard with a longer delay on the last frame.

The gesture N°5 - Saying no with 2 hands is the second lowest result with 77.5% of correct answer, which is still a good score. The second most chosen answer was "No corresponding description" and we suppose that it comes from the wrong choice of name corresponding to the gesture. Indeed, the name chosen "Saying no with 2 hands" is not a description of the gesture but rather an interpretation of its meaning, which is therefore subjective. We also note that 3 participants chose "crossed arms" and 1 chose "arms crossed in X", which is not wrong because the arms do cross at a certain moment in the execution of the gesture. We believe that they

made this choice out of spite as they did not find a description that corresponded 100% to their eyes.

For the conciseness criterion, we did not use tools to calculate the time taken by participants to identify the gesture shown. However, we had the opportunity to discuss with some of them who told us that they had completed the test in less than 5 minutes. We cannot draw any statistical conclusion from this, but it supports us in the idea that our representation allows to quickly recognize spatial gestures.

One limitation is that it is possible that the format of the test greatly facilitated the recognition of the gestures. Indeed, the participants had the possibility of leaving a comment at the end of the questionnaire and one of them emphasized that the fact of giving an exhaustive list and having the possibility of seeing the following gestures greatly helped in making his choices. An idea for another format would be to conduct interviews where participants are shown the gesturecards and asked to describe the gesture as best they can, without any proposition. Their responses should then be analyzed to see how well their descriptions match the gesturecards shown. This would require more time and resources and would reduce the number of participants.

4.2 Study 2: Self-evaluating the representation with the Cognitive Dimensions framework

For this second evaluation of our representation, we did not conduct a study with participants, but we will evaluate the effectiveness of it according to Sousa et al. (2012)'s method. These researchers propose a framework allowing the evaluation of graphical notations via 6 of the 12 dimensions of the Cognitive Dimensions Framework established by Blackwell & Green (2003).

4.2.1 The 6 dimensions definition

We will present and define the 6 dimensions included by Sousa et al. (2012) but some will be easier to evaluate than others depending on the elements they take into account. For each of them, the level is evaluated on a scale going from "low" to "high". The dimensions are as follows:

- *Viscosity*: A low-level means that it is easy to make changes and a high-level means that it is difficult to make changes (because of interdependence between the different elements).
- *Hidden dependencies*: a low-level means that the links between the different elements of the notation are explicit while a high-level shows that these links are implicit. When implicit, the types of relationships between the elements are unknown.
- *Visibility*: A low-level is for representations in which it is difficult to distinguish the elements (because of colors, shapes, etc.) and a high-level when the elements are well differentiated from each other.
- *Premature commitment*: A low-level indicates that there is no set order for creating the representation and that the user is free to perform the actions in any order. A high-level shows that the process for creating gesture representations is strict and constrained allowing few freedoms.
- *Abstraction*: a low-level indicates that the notation does not allow adding new ones to the rating. A high-level shows that the notation allows to easily add elements to the representation such as new symbols, new types of relations, etc.
- *Secondary notation*: a low level indicates that the notation is very formalized and does not allow the use of informal syntax. A high level indicates the presence of informal syntax, i.e., that it allows the addition of informal elements to give additional information.

4.2.2 Evaluation of the 6 dimensions for our representation

We will now evaluate our representation through these 6 dimensions presented. This analysis does not allow us to measure the efficiency of our representation but simply to show what it allows and does not allow. Note that this framework is designed to evaluate graphical representations only. Ours being hybrid, the analysis of textual elements is not as appropriate as for graphical elements, but they will still be taken into account. To give a more precise evaluation we have established a Likert scale from 1 to 5 where 1 corresponds to the low level and 5 to the high level and the intermediate values are gradual between these 2 extremes.

- *Viscosity*: for the graphic part, we would give a 5 because if the user wants to modify it, he must restart the steps presented in section 3.5 from the beginning. Even for small changes, since the creations in Magic Poser in the web version cannot be saved, it is necessary to restart the modeling from scratch. Therefore, if changes are to be made,

the costs of creation, especially in time, are doubled. For textual elements, we give them an evaluation of 1 out of 5 because they are by nature very simple to modify. Since the graphic part represents a large part of the description of the gesture, the total evaluation of the viscosity remains on the *score of 4*.

- *Hidden dependencies*: This dimension is difficult to identify because the different elements of our representation are not directly linked. For the graphical representation, only the avatar is represented, and no other element is added, thus not giving any relation between elements. Therefore, we could give a 1 for the graphical part.

We could evaluate the links between the textual and graphic parts. For example, we believe that the links between the graphical representation and the section dedicated to the traits are explicit because the latter always define properties, of the gesture represented in the graphical part, such as the speed of the execution. And the same reasoning is applicable to the other sections of the representation. On these facts, we give the *score of 2* for this dimension.
- *Visibility*: this dimension is very simple to evaluate since the elements constituting our representation are separated into sections. For the graphical part, at first sight, being constituted by only one element (the avatar), we would give the note of 1. However, we must mitigate this one because the spatial gestures being made in three dimensions and the representation being on a two-dimensional surface, it is inevitable that we lose some details of the gesture having only one perspective. For the graphic section, we would give a score of 4. For the textual part, we rather evaluate the understanding of what we want to represent for each of them. In our study it appeared that some people did not understand the distinction between "Features" and "Return values". We can therefore believe that some ambiguity could arise from this. Nevertheless, we believe that the different elements are rather explicit and simple to understand. We would give the *score of 4* for the visibility.
- *Premature commitment*: for the graphic part it is obvious that the steps presented in section 3.5 are difficult to customize. The user cannot realize them in another order, so we give a score of 5. For the textual part, the freedom is total although in general the thinking process brings the elements one by one considering first the traits that are

necessary for the recognition. Based on that, we would give a score of 2 for the textual elements. To evaluate this dimension on the representation in total, the graphic representation weighing a more important part in the balance, we would give the *note of 4*.

- *Abstraction*: for the graphical part, our representation does not give the possibility to add elements to it, so we give the score of 1. For the textual part, considering the traits section although a list has been established, the possibility is given to the users to add new ones if required for the description of gestures. The factors (mentioned in section 3.4.3) which are presented in the form "factor: value" have a list of 13 factors established leaving no possibilities for other ones but the "value" part is left to their imagination. Therefore, we give a score of 3 to the text part. We give a *neutral score of 3* for this dimension to our representation.
- *Secondary notation*: as our representation wants to be as concise and readable as possible, it is very formal and must stick to its basic syntax in this sense. We have not left the possibility to add elements outside of those constituting it. Therefore, we give the *score of 1* for this dimension.

This self-evaluation allowed us to better present certain aspects of our representation. It highlights some points that can be considered as limitations. This is the case for the fact that our graphic part does not allow to modify a small element of the avatar without having to restart from the beginning. It could be interesting to consider this possibility. Moreover, although we are satisfied with the result given by our solution, the visibility dimension highlights once again the difficulty to represent at 100% a 3D spatial gesture on a 2D support. We thought of presenting the gestures from different angles, but we judged that the cost was not worth it. Indeed, this would increase the preparation time of the GIFs and the gesturecard would be too large, deteriorating the conciseness criteria.

Chapter 5 - Conclusion

5.1 Reminder of the research question

In this work, we have identified that there is a communication problem in documenting and defining interaction gestures. Indeed, due to their heterogeneous nature, application development team members, especially designers and engineers, use different working methods and tools. As a result, the transmission of information from one stage of the project to another can be distorted and this can lead to ambiguities that can lead to implementation errors and in some cases to additional costs and work. Gesture representation tools exist to overcome this communication problem, but we have seen that most of them are rather intended for engineering and do not allow sufficient readability for humans. Moreover, these representations are either textual or graphic, which limits the possibility of describing the gestures. Hybrid gesture representation like GestureCards have been developed combining the advantages of textual and graphic representations and allowing for better gesture description with a much better readability. However, such representation essentially focusses on touch gestures in contact with a screen. As gesture control is not limited to touching gestures, but to all parts of the body, it was necessary to develop a new representation that would be more appropriate to describe and define spatial gestures in order to facilitate the communication flow between designers and engineers.

5.2 Summary of the contributions

First, definitions of concepts and technical terms of the HCI domain can be found in our work at **section 1.1**. We bring broad definitions to understand these concepts without diving into details. We consider them sufficient to be used in the scope of works like ours, which focuses on managerial implications. We believe the definitions that we provided can be used in future research, even for works that require more specific definitions, those can be used as a basis to understand their conceptual dimensions.

Second, **section 1.2** identifies and highlights a problem that can be found in many project teams. Although in the context of our work we are talking specifically about communication within application development teams working with gestural interaction, it seems to us that this is a problem that could arise in other domains in which stakeholders with very heterogeneous

skills collaborate. In particular, we identified the lack of appropriate tools that would facilitate communication by providing a common language that everyone could understand.

Third, our general literature review has allowed us to establish the state of the art for several topics in **Chapter 2**. Although not exhaustive as we were selecting the relevant information in the scope of our work, we presented in **section 2.1** the devices and technologies allowing gesture interaction with information systems. We believe that this information can be used for further work in the field of HCI in order to get a quick overview of the possibilities of interaction and the technologies available.

Fourth, in **section 2.2** we introduced taxonomies allowing to classify gesture representations according to the elements used in them. In addition to these definitions, we reported a study conducted that identified which elements are most used in gesture representations. In this same section, we presented a framework that allows designers to take into account all the aspects of a gesture during its elaboration. In **section 2.3**, we listed already existing gesture representations. We have selected those that have been designed with the same goal of conciseness and readability as ours.

Then, in **Chapter 3** we have developed the path that led us to our proposal of a new representation for spatial gestures. We first established the objectives and prerequisites in **section 3.1**. After that, the following sections allowed us to select the elements included in the final version of our representation, based on the research conducted previously in **Chapter 2**.

After that, in **Chapter 4**, we evaluated our representation. First with a study conducted with participants. The design of the study is presented in **section 4.1.1** and the results in **section 4.1.2**. The analysis of the results is discussed in **section 4.1.3** and proved to be satisfactory although the study allowed us to improve our representation. In **section 4.2**, to complete our analysis, we made a self-evaluation via a framework allowing to evaluate our representation through 6 dimensions.

5.3 Benefits and shortcomings

5.3.1 Benefits

Here we will evaluate whether the objectives of our research work have been met. As a reminder, the objective was to develop a method for representing spatial interaction gestures

in order to facilitate communication between designers and engineers. For this, the criteria established were the following:

- The representation had to be **concise** so that the time taken to recognize the gesture had to be minimal.
- The representation had to be **readable** for a person with no specific training in the field.
- The representation had to be **adapted to spatial gestures** performed with any part of the body for static poses and dynamic gestures.

To meet the first two criteria, we based ourselves on studies presented in section 2.3.6 which showed that GestureCards performed better than other existing ratings. That's why we decided to keep its basic structure.

For the "graphical representation" part, we had to find a new solution to easily document it within our representation. In order to model the gestures, we chose to use the web application Magic Poser, which allows to modify the position of a humanoid avatar and thus to represent it in different poses. For static poses it was sufficient to take a screenshot of the avatar modeled in the desired pose. For dynamic gestures, we chose to use GIF's which allow to sequence a series of images and give the illusion of movement. GIFs have the advantage of being smaller files than video files, which makes them easier to share within a team.

We were able to test the readability criterion in our study in Chapter 4, which gave us an average recognition percentage of 86.35%. Based on this satisfactory result, we consider that this criterion is fulfilled. To evaluate whether our representation can effectively describe spatial gestures, we estimate that the percentage of gesture recognition in the study also allows us to say that this criterion is met as we only assessed our representation for spatial gestures. For the conciseness criterion, this one was not directly evaluated, and we relied on the fact that previous studies on GestureCards showed that it was more concise than the other representations. Moreover, some of the participants we spoke with said that they completed the test in less than 5 minutes, for which they had to recognize 7 gestures. On this non-statistical basis, we can think that the recognition of gestures is done quickly with our representation, without being able to affirm it. In evaluating whether our criteria have been met, we believe that our proposed solution is a good solution to our research question.

From a managerial point of view, this solution allows to bring a better communication within application development teams and to avoid loss of time and money. In fact, as summarized in

section 1.3, the communication problems identified could result in ambiguities and misinterpretations between team members. These could then lead to implementation errors and flawed applications ultimately leading to wasted time and money to fix these errors.

5.3.2 *Shortcomings*

The first limitation identified concerns the study conducted to test our representation. Indeed, the design of the study could have evaluated more precisely the chosen criteria. Although satisfactory for the readability criterion, as one of the participants indicated, the test format would facilitate gesture recognition. Moreover, it could have been interesting to set up a tool allowing to calculate the time taken by the participants to recognize each gesture and thus to evaluate the conciseness criterion.

Still concerning the design of the study we conducted, it could have been interesting to categorize the gestures to be recognized and to evaluate the effectiveness of the representation for each of them. For example, we could have categorized the gestures made with the hands, the mid-air gestures made with the arms and the hands or the gestures with the whole body.

A second limitation is that we did not collaborate directly with designers and/or application development engineers. It could have been interesting to collect their experiences to see how they actually work and then to collect their opinion on the solution we brought.

A third limitation is that we could have pushed the research of modeling software further and considered other methods to represent gestures in three dimensions. We were satisfied with the result provided by Magic Poser and especially with its ease of use, but it is possible that other software could be adapted.

5.4 **Future Work**

5.4.1 *Short-term*

Based on the identified limitations, we can suggest further studies to evaluate the effectiveness of this gesture representation. As mentioned earlier, it could segment the gestures into several categories to refine the analysis of the results and to be able to tell which category of gestures is more or less appropriate with our representation method. Moreover, it could be interesting to add a tool to calculate the time taken by users to recognize the gestures. Such tests could be used to further highlight areas for improvement.

It could also be interesting to test our representation not for its recognition but for the elaboration of gesture descriptions. For example, we could ask participants to use our tools to elaborate gesture descriptions. This could also allow us to improve these tools.

As mentioned in the limitations, deeper work could be done in collaboration with designers and engineers. Being the main ones concerned by our solution, their experience should be taken into account to improve these communication tools.

5.4.2 *Mid-term*

For medium-term work, it might be interesting to further develop our representation for spatial gestures. In collaboration with software engineering teams, we could imagine the development of an application where all the tools to create gesture descriptions would be integrated. This would facilitate the elaboration of gesture descriptions and prevent having to juggle between different tools.

As technologies and the field of HCI evolve rapidly, communication tools for gestural interaction are quickly becoming outdated and no longer used. It is therefore necessary to follow with attention the evolutions in the field in order to adapt these tools as quickly as possible.

References

- 3D Motion Gesture Index* | *GestureML Wiki*. (n.d.). GestureML - Gesture Markup Language - Wiki. Retrieved May 10, 2022, from http://gestureml.org/doku.php/gestures/motion/gesture_index
- Abelein, U., Sharp, H., & Paech, B. (2013). Does Involving Users in Software Development Really Influence System Success? *IEEE Software*, 30(6), 17–23. <https://doi.org/10.1109/MS.2013.124>
- Ackad, C., Clayphan, A., Tomitsch, M., & Kay, J. (2015). An in-the-wild study of learning mid-air gestures to browse hierarchical information at a large interactive public display. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 1227–1238. <https://doi.org/10.1145/2750858.2807532>
- Altakrouri, B., & Schrader, A. (2014). Describing movements for motion gestures. In *CEUR Workshop Proceedings* (Vol. 1190).
- Anthony, L., & Wobbrock, J. O. (2010). *A Lightweight Multistroke Recognizer for User Interface Prototypes*. 8.
- Anthony, L., & Wobbrock, J. O. (2012). *\$N\$-protractor: A fast and accurate multistroke recognizer*. 4.
- baillydesign. (n.d.). *Which Global Companies Use Blender 3D?* Blender Base Camp. Retrieved July 6, 2022, from <https://www.blenderbasecamp.com/which-global-companies-use-blender/>
- Baskerville, R., Pries-Heje, J., & Venable, J. (2009). *Soft Design Science Methodology*. 12.
- Baxter, L., & Baxter, L. K. (1997). *Capacitive sensors: Design and applications* (1. print). IEEE Press.
- Berni, A., & Borgianni, Y. (2021). FROM THE DEFINITION OF USER EXPERIENCE TO A FRAMEWORK TO CLASSIFY ITS APPLICATIONS IN DESIGN. *Proceedings of the Design Society*, 1, 1627–1636. <https://doi.org/10.1017/pds.2021.424>
- Bernsen, N., & Dybkjær, L. (2010). *Multimodal Usability: Conclusions and Future Work*. https://doi.org/10.1007/978-1-84882-553-6_18
- Blackwell, A., & Green, T. (2003). *Notational Systems—The Cognitive Dimensions of Notations Framework*. <https://doi.org/10.1016/B978-155860808-5/50005-8>

- Blandford, A. (2013). Engineering works: What is (and is not) engineering for interactive computer systems? *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 285–286. <https://doi.org/10.1145/2494603.2480342>
- Bourgeois, D. T., Smith, J. L., Wang, S., & Mortati, J. (2014). *Information Systems for Business and Beyond*. <https://open.umn.edu/opentextbooks/textbooks/189>
- Bragdon, A., Nelson, E., Li, Y., & Hinckley, K. (2011). Experimental analysis of touch-screen gesture designs in mobile environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 403–412. <https://doi.org/10.1145/1978942.1979000>
- Buxton, B. (2011). *Sketching user experiences: Getting the design right and the right design* (Nachdr.). Morgan Kaufmann.
- Chesta, C., Paternò, F., & Santoro, C. (2004). *Methods and Tools for Designing and Developing Usable Multi-Platform Interactive Applications*. 17.
- Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.
- Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., & Young, R. M. (1995). Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The Care Properties. In K. Nordby, P. Helmersen, D. J. Gilmore, & S. A. Arnesen (Eds.), *Human—Computer Interaction: Interact '95* (pp. 115–120). Springer US. https://doi.org/10.1007/978-1-5041-2896-4_19
- Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1), 4–7. <https://doi.org/10.1007/s007790170019>
- Dim, N. K., Silpasuwanchai, C., Sarcas, S., & Ren, X. (2016). Designing Mid-Air TV Gestures for Blind People Using User- and Choice-Based Elicitation Approaches. *Conference on Designing Interactive Systems*. <https://doi.org/10.1145/2901790.2901834>
- Dumas, B., Lalanne, D., & Ingold, R. (2010). Description languages for multimodal interaction: A set of guidelines and its illustration with SMUIML. *Journal on Multimodal User Interfaces*, 3(3), 237–247. <https://doi.org/10.1007/s12193-010-0043-3>
- Echtler, F., & Butz, A. (2012). GISpL: Gestures made easy. *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, 233–240. <https://doi.org/10.1145/2148131.2148181>

- Echtler, F., Klinker, G., & Butz, A. (2010). Towards a unified gesture description language. *Proceedings of the 13th International Conference on Humans and Computers*, 177–182.
- Farooq, U., Mohd Rahim, M. S., Khan, N. S., Rasheed, S., & Abid, A. (2021). A Crowdsourcing-Based Framework for the Development and Validation of Machine Readable Parallel Corpus for Sign Languages. *IEEE Access*, 9, 91788–91806. <https://doi.org/10.1109/ACCESS.2021.3091433>
- Ferreira, J., Sharp, H., & Robinson, H. (2011). User experience design and agile development: Managing cooperation through articulation work. *Software: Practice and Experience*, 41(9), 963–974. <https://doi.org/10.1002/spe.1012>
- Ganhör, R., & Spreicer, W. (2014). Monox: Extensible gesture notation for mobile devices. *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services - MobileHCI '14*, 203–212. <https://doi.org/10.1145/2628363.2628394>
- Gemini: Fifth-generation hand tracking platform | Ultraleap. (n.d.). Retrieved July 2, 2022, from <https://www.ultraleap.com/tracking/gemini-hand-tracking-platform/>
- Greenberg, S., Carpendale, S., Marquardt, N., & Buxton, W. (2012). *Sketching user experiences: The Workbook*. Elsevier/Morgan Kaufmann.
- Hachaj, T., & Ogiela, M. R. (2014). Rule-based approach to recognizing human body poses and gestures in real time. *Multimedia Systems*, 20(1), 81–99. <https://doi.org/10.1007/s00530-013-0332-2>
- Hanke, T. (2004). *HamNoSys—Representing sign language data in language resources and language processing contexts*.
- Hartson, R., & Pyla, P. S. (2012). *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Elsevier.
- Hassenzahl, M., & Tractinsky, N. (2006). User experience—A research agenda. *Behaviour & Information Technology*, 25(2), 91–97. <https://doi.org/10.1080/01449290500330331>
- Herold, J., & Stahovich, T. F. (2012). The One Cent Recognizer: A Fast, Accurate, and Easy-to-Implement Handwritten Gesture Recognition Technique. *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 8 pages. <https://doi.org/10.2312/SBM/SBM12/039-046>
- Hesenius, M. (2019). *Spezifikation von Gestensteuerung—Ein hybrider Ansatz* [University of Duisburg-Essen]. <https://doi.org/10.17185/DUEPUBLICO/47889>

- Hesenius, M., Griebe, T., & Gruhn, V. (2014). Towards a behavior-oriented specification and testing language for multimodal applications. *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 117–122. <https://doi.org/10.1145/2607023.2610278>
- Hesenius, M., & Gruhn, V. (2019). GestureCards: A Hybrid Gesture Notation. *Proceedings of the ACM on Human-Computer Interaction*, 3(EICS), 1–35. <https://doi.org/10.1145/3331164>
- Hinrichs, U., & Carpendale, S. (2011). Gestures in the wild: Studying multi-touch gesture sequences on interactive tabletop exhibits. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3023–3032. <https://doi.org/10.1145/1978942.1979391>
- Hoste, L., & Signer, B. (2014). *Criteria, Challenges and Opportunities for Gesture Programming Languages*. 8.
- Hu, C.-Y., Lin, H.-Y., & Chen, L.-C. (2015). The Effects of Screen Size on Rotating 3D Contents Using Compound Gestures on a Mobile Device. *Displays*, 41. <https://doi.org/10.1016/j.displa.2015.11.002>
- International Organization for Standardization. (2007). *ISO 9241-400:2007*. ISO. <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/88/38896.html>
- Jacob, R. J. K., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., & Zigelbaum, J. (2008). Reality-based interaction: A framework for post-WIMP interfaces. *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08*, 201. <https://doi.org/10.1145/1357054.1357089>
- Kaminski, J. (2011). Diffusion of Innovation Theory. *Canadian Journal of Nursing Informatics*, 6(2). <https://cjni.net/journal/?p=1444>
- Kammer, D., Henkens, D., Henzen, C., & Groh, R. (2015). Gesture Formalization for Multitouch. *Software: Practice and Experience*, 45(4), 527–548. <https://doi.org/10.1002/spe.2247>
- Kammer, D., Wojdziak, J., Keck, M., Groh, R., & Taranko, S. (2010). Towards a formalization of multi-touch gestures. *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*, 49. <https://doi.org/10.1145/1936652.1936662>
- Karray, F., Alemzadeh, M., Saleh, J. A., & Arab, M. N. (2008). Human-Computer Interaction: Overview on State of the Art. *International Journal on Smart Sensing and Intelligent Systems*, 1(1), 137–159. <https://doi.org/10.21307/ijssis-2017-283>

- Kaur, K., & Kumar, P. (2016). HamNoSys to SiGML Conversion System for Sign Language Automation. *Procedia Computer Science*, 89, 794–803. <https://doi.org/10.1016/j.procs.2016.06.063>
- Khandkar, S. H., & Maurer, F. (2010). A domain specific language to define gestures for multi-touch applications. *Proceedings of the 10th Workshop on Domain-Specific Modeling*, 1–6. <https://doi.org/10.1145/2060329.2060339>
- Kim, J.-H., Thang, N. D., & Kim, T.-S. (2009). 3-D hand motion tracking and gesture recognition using a data glove. *2009 IEEE International Symposium on Industrial Electronics*, 1013–1018. <https://doi.org/10.1109/ISIE.2009.5221998>
- Kim, J.-W., & Nam, T.-J. (2013). EventHurdle: Supporting designers' exploratory interaction prototyping with gesture-based sensors. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 267–276. <https://doi.org/10.1145/2470654.2470691>
- Kin, K., Hartmann, B., DeRose, T., & Agrawala, M. (2012a). Proton++: A customizable declarative multitouch framework. *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology - UIST '12*, 477. <https://doi.org/10.1145/2380116.2380176>
- Kin, K., Hartmann, B., DeRose, T., & Agrawala, M. (2012b). Proton: Multitouch gestures as regular expressions. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2885–2894. <https://doi.org/10.1145/2207676.2208694>
- Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. A brief history. *Computer*, 36(6), 47–56. <https://doi.org/10.1109/MC.2003.1204375>
- Law, L.-C., Roto, V., Vermeeren, A., Kort, J., & Hassenzahl, M. (2008). *Towards a shared definition of user experience* (p. 2398). <https://doi.org/10.1145/1358628.1358693>
- Leiva, G., Maudet, N., Mackay, W., & Beaudouin-Lafon, M. (2019). Enact: Reducing Designer-Developer Breakdowns when Prototyping Custom Interactions. *ACM Transactions on Computer-Human Interaction*, 26(3), 1–48. <https://doi.org/10.1145/3310276>
- Li, Y., Huang, J., Tian, F., Wang, H.-A., & Dai, G.-Z. (2019). Gesture interaction in virtual reality. *Virtual Reality & Intelligent Hardware*, 1(1), 84–112. <https://doi.org/10.3724/SP.J.2096-5796.2018.0006>
- Lü, H., Fogarty, J. A., & Li, Y. (2014). Gesture script: Recognizing gestures and their structure using rendering scripts and interactively trained parts. *Proceedings of the SIGCHI*

- Conference on Human Factors in Computing Systems*, 1685–1694.
<https://doi.org/10.1145/2556288.2557263>
- Maenaka, K. (2008). MEMS inertial sensors and their applications. *2008 5th International Conference on Networked Sensing Systems*, 71–73.
<https://doi.org/10.1109/INSS.2008.4610859>
- McAweeney, E., Zhang, H., & Nebeling, M. (2018). User-Driven Design Principles for Gesture Representations. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3173574.3174121>
- Mendez, I., Hansen, B. W., Grabow, C. M., Smedegaard, E. J. L., Skogberg, N. B., Uth, X. J., Bruhn, A., Geng, B., & Kamavuako, E. N. (2017). Evaluation of the Myo armband for the classification of hand motions. *2017 International Conference on Rehabilitation Robotics (ICORR)*, 1211–1214. <https://doi.org/10.1109/ICORR.2017.8009414>
- Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann Publishers Inc.
- Nielsen, M., Störring, M., Moeslund, T. B., & Granum, E. (2004). A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI. In A. Camurri & G. Volpe (Eds.), *Gesture-Based Communication in Human-Computer Interaction* (pp. 409–420). Springer. https://doi.org/10.1007/978-3-540-24598-8_38
- Norman, D., Miller, J., & Henderson, A. (1995). *What You See, Some of What's in the Future, And How We Go About Doing It: HI at Apple Computer*.
<https://doi.org/10.1145/223355.223477>
- Onwuegbuzie, A. J., & Frels, R. (2016). *Seven Steps to a Comprehensive Literature Review: A Multimodal and Cultural Approach*. SAGE.
- Perkins, D., & Salomon, G. (1999). *Transfer Of Learning*. 11.
- Prillwitz, S., Leven, R., Zienert, H., Hanke, T., & Henning, J. (1989). *HamNoSys Version 2.0. Hamburger Notationssystem für Gebärdensprachen – eine Einführung*.
- Raskin, J. (1994). Viewpoint: Intuitive equals familiar. *Communications of the ACM*, 37(9), 17–18. <https://doi.org/10.1145/182987.584629>
- Rautaray, S. S., & Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: A survey. *Artificial Intelligence Review*, 43(1), 1–54.
<https://doi.org/10.1007/s10462-012-9356-9>
- Rechy-Ramirez, E. J., Marin-Hernandez, A., & Rios-Figueroa, H. V. (2018). Impact of commercial sensors in human computer interaction: A review. *Journal of Ambient Intelligence and Humanized Computing*, 9(5), 1479–1496.
<https://doi.org/10.1007/s12652-017-0568-3>

- Rekimoto, J. (2001). GestureWrist and GesturePad: Unobtrusive wearable interaction devices. *Proceedings Fifth International Symposium on Wearable Computers*, 21–27. <https://doi.org/10.1109/ISWC.2001.962092>
- Rico, J., & Brewster, S. (2010). Usable gestures for mobile interfaces: Evaluating social acceptability. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 887–896. <https://doi.org/10.1145/1753326.1753458>
- Rubine, D. (1991). Specifying gestures by example. *ACM SIGGRAPH Computer Graphics*, 25(4), 329–337. <https://doi.org/10.1145/127719.122753>
- Rumrill, J., & Fitzgerald, S. M. (2001). Using narrative literature reviews to build a scientific knowledge base. *Work*, 16(2), 165–170.
- Saffer, D. (2009). *Designing gestural interfaces* (1st ed). O'Reilly.
- Satyabrata, J. (2021, November 15). 10 Crucial Team Roles in a Software Development Team. *GeeksforGeeks*. <https://www.geeksforgeeks.org/ten-crucial-team-roles-in-a-software-development-team/>
- Schmidt, M., & Weber, G. (2013). Template based classification of multi-touch gestures. *Pattern Recognition*, 46(9), 2487–2496. <https://doi.org/10.1016/j.patcog.2013.02.001>
- Shipman, F. M., & Marshall, C. C. (1999). Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems. *Computer Supported Cooperative Work (CSCW)*, 8(4), 333–352. <https://doi.org/10.1023/A:1008716330212>
- Shneiderman, B. (1997). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (3rd ed.). Addison-Wesley Longman Publishing Co., Inc.
- Sousa, K., Vanderdonckt, J., Henderson-Sellers, B., & Gonzalez-Perez, C. (2012). Evaluating a graphical notation for modelling software development methodologies. *Journal of Visual Languages & Computing*, 23(4), 195–212. <https://doi.org/10.1016/j.jvlc.2012.04.001>
- Spano, L. D., Cisternino, A., Paternò, F., & Fenu, G. (2013). GestIT: A declarative and compositional framework for multiplatform gesture definition. *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '13*, 187. <https://doi.org/10.1145/2494603.2480307>
- Suarez Fernandez, R., Sanchez-Lopez, J., Sampedro Pérez, C., Bavle, H., Molina, M., & Campoy, P. (2016). *Natural user interfaces for human-drone multi-modal interaction* (p. 1022). <https://doi.org/10.1109/ICUAS.2016.7502665>

- Taranta II, E., & Laviola Jr., J. J. (2015). *Penny Pincher: A Blazing Fast, Highly Accurate \$-Family Recognizer*. 8.
- Te'eni, D., Carey, J., & Zhang, P. (2007). *Human-Computer Interaction: Developing Effective Organizational Information Systems*.
- Touch Gesture Index* | *GestureML Wiki*. (n.d.). GestureML - Gesture Markup Language - Wiki. Retrieved July 14, 2022, from http://gestureml.org/doku.php/gestures/touch/gesture_index
- Tracking* | *Leap Motion Controller* | *Ultraleap*. (n.d.). Retrieved July 5, 2022, from <https://www.ultraleap.com/product/leap-motion-controller/>
- Usuwiel, H., Wilberz, J., d'Aspremont Lynden, T., & Nothomb, B. (2020). *Opportunity study for the implementation of gesture recognition technology in the hospital environment Commercialization through a university spin-off*.
- van Dam, A. (1997). Post-WIMP user interfaces. *Communications of the ACM*, 40(2), 63–67. <https://doi.org/10.1145/253671.253708>
- Vatavu, R.-D., Anthony, L., & Wobbrock, J. O. (2012). Gestures as point clouds: A \$P recognizer for user interface prototypes. *Proceedings of the 14th ACM International Conference on Multimodal Interaction - ICMI '12*, 273. <https://doi.org/10.1145/2388676.2388732>
- Velloso, E., Schmidt, D., Alexander, J., Gellersen, H., & Bulling, A. (2015). The Feet in Human--Computer Interaction: A Survey of Foot-Based Interaction. *ACM Computing Surveys*, 48(2), 1–35. <https://doi.org/10.1145/2816455>
- Villarreal-Narvaez, S., Vanderdonckt, J., Vatavu, R.-D., & Wobbrock, J. O. (2020). A Systematic Review of Gesture Elicitation Studies: What Can We Learn from 216 Studies? In *Proceedings of the 2020 ACM Designing Interactive Systems Conference* (pp. 855–872). Association for Computing Machinery. <https://doi.org/10.1145/3357236.3395511>
- Vogel, D., & Balakrishnan, R. (2010). Occlusion-aware interfaces. *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10*, 263. <https://doi.org/10.1145/1753326.1753365>
- Wachs, J. P., Stern, H. I., Edan, Y., Gillam, M., Handler, J., Feied, C., & Smith, M. (2008). A Gesture-based Tool for Sterile Browsing of Radiology Images. *Journal of the American Medical Informatics Association*, 15(3), 321–323. <https://doi.org/10.1197/jamia.M2410>

- Wang, L., Huynh, D. Q., & Koniusz, P. (2020). A Comparative Review of Recent Kinect-Based Action Recognition Algorithms. *IEEE Transactions on Image Processing*, 29, 15–28. <https://doi.org/10.1109/TIP.2019.2925285>
- Wedel, M., & Pieters, R. (2008). *Eye tracking for visual marketing*. Now Publ.
- Wigdor, D., & Wixon, D. (2011). *Brave NUI world: Designing natural user interfaces for touch and gesture*. Elsevier, Morgan Kaufmann.
- Williamson, J. R. (2012). *User experience, performance, and social acceptability: Usable multimodal mobile interaction* [University of Glasgow]. <http://oatd.org/oatd/record?record=oai%5C%3Atheses.gla.ac.uk%5C%3A3260>
- Wobbrock, J. O., Aung, H. H., Rothrock, B., & Myers, B. A. (2005). Maximizing the guessability of symbolic input. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 1869–1872. <https://doi.org/10.1145/1056808.1057043>
- Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: A \$1 recognizer for user interface prototypes. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology - UIST '07*, 159. <https://doi.org/10.1145/1294211.1294238>
- Wynne, M., & Hellesøy, A. (2012). *The cucumber book: Behaviour-driven development for testers and developers*. Pragmatic Bookshelf.
- Xia, H., Glueck, M., Annett, M., Wang, M., & Wigdor, D. (2022). Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature. *ACM Transactions on Computer-Human Interaction*, 29(4), 1–54. <https://doi.org/10.1145/3503537>
- Yatani, K., Partridge, K., Bern, M., & Newman, M. (2008). *Escape: A target selection technique using visually-cued gestures* (p. 294). <https://doi.org/10.1145/1357054.1357104>