Proceedings of the Federated Conference on Computer Science and Information Systems pp. 653–656
DOI: 10.15439/2020F29
ISSN 2300-5963 ACSIS, Vol. 21

A Framework for Developing Proxemic Mobile Applications

Paulo Pérez
E2S / University of Pau (LIUPPA)
ESTIA Institute of Technology
Bidart, France
ppdaza@iutbayonne.univ-pau.fr

Philippe Roose
E2S / University of Pau (LIUPPA)
IUT de Bayonne
Anglet, France
philippe.roose@iutbayonne.univ-pau.fr

Yudith Cardinale
Universidad Simón Bolívar (USB)
Caracas, Venezuela
Universidad Católica San Pablo
Arequipa, Perú
ycardinale@usb.ve

Marc Dalmau
E2S / University of Pau (LIUPPA)
IUT de Bayonne
Anglet, France
dalmau@iutbayonne.univ-pau.fr

Dominique Masson
Dev1-0
Technopole Izarbel
Bidart, France
d.masson@dev1-0.com

Nadine Couture
University of Bordeaux
ESTIA Institute of Technology
Bidart, France
n.couture@estia.fr

***Abstract*—The widespread diffusion of smart and mobile devices continuously connected to the Internet has facilitated the users' contact and interaction with other people, devices, and with their physical surroundings. Proxemic interaction, derived from proxemics theory, focuses on how Human-Computer Interaction (HCI) works with smart devices, using the five proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO). The current tools for developing proxemic applications require fixed devices that make it difficult to build proxemic mobile apps. In this work, we propose a framework to manage all components in a proxemic environment (i.e., interaction objects and DILMO dimensions that govern the HCI). We demonstrate and evaluate the effectiveness and suitability of our framework, through the development of two proxemic mobile applications, as proof-of-concept.**

***Index Terms*—Proxemic interaction, proxemic zone, mobile devices, wearable technologies**

I. Introduction

The use of mobile technologies in our daily life is increasing in a way without precedent. People can interact with different contexts through electronic devices (e.g., personal mobile phones, tablets, wearable technologies, and smart-watches) to accomplish their daily tasks. Many of these tasks require a specific Human-Computer Interaction (HCI). Researchers are therefore seeking to develop new useful and enjoyable interfaces. Proxemic interaction arises as a novel concept to improve HCI [1], [2]. Proxemic interaction describes how people use interpersonal distances to interact with digital devices [3]–[5], using the five physical proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO).

Proxemic interaction is derived from the social Proxemics theory proposed in 1966 by the anthropologist Edward T. Hall [6]. Hall describes how individuals perceive their personal space relative to the distance between themselves and others. According to Hall's proxemics theory, interaction zones have been classified into four zones: (i) intimate zone, comprised between 0 and 50 cm of distance; (ii) personal zone, defined by a distance of 0.5 cm to 1 m; (iii) social zone, when the distance is between 1 m and 4 m ; and (iv) public zone, if distance is more than 4 m. He underlines the role of proxemic relationships as a method of communication based on the distance between people.

In this context, solutions such as Toolkit [7] and ProximiThings [8] have been proposed to support the development of proxemic interactions. However, existing solutions present limitations for implementing proxemic interaction in mobile technologies, because they require special hardware devices connected to the system (e.g., a Kinect Depth sensor, which must be installed on a PC for sensing proxemic information).

This work aims to propose a concrete solution, a framework, for developing proxemic environments comprised by entities, whose interactions are defined according to DILMO dimensions in mobile applications. Our proposed framework represents a threefold contribution: (i) it offers functionalities to define and manage all components in a proxemic environment: the interaction objects, the DILMO dimensions that govern the HCI, and the proxemic mobile applications; (ii) an API integrated into the framework, that allows developers to simplify the process of proxemic information sensing (i.e., measure of DILMO dimensions) by mobile phones and wearable sensors; and (iii) the proof-of-concept to demonstrate and evaluate the effectiveness and suitability of our framework by describing the implementation of two proxemic mobile applications; these two mobile apps are based on HCI defined as a function of different DILMO combinations that specify different context-based infrastructures for proxemic environments based on mobile devices.

II. Related work

Proxemic concepts are based on physical, social, and cultural factors that influence and regulate interpersonal interac-

tions [7]. In order to know how the factors should be applied to proxemic interactions for ubiquitous computing applications, Greenberg [4] identified five dimensions: Distance, Identity, Location, Movement, and Orientation (we call them DILMO as an abbreviation), which are associated with people, digital devices, and non digital things.

Proxemic interaction is a remarkable interaction technique that allows the user to control the digital devices in a flexible way [3], [7], [9]. Some previous works have proposed tools to support the development of proxemic applications considering proxemic interactions.

The work presented in [8], illustrates how the proxemic dimensions can support interaction among entities (people and objects), with a proposed context-aware framework. This framework provides capabilities that help developers build a front-end application. However, the framework requires a cloud computing architecture and an active connection to the server for processing proxemic information. In [7], a framework, called Proximity Toolkit, used to discover novel proxemic-aware interaction techniques is proposed. The framework is a guide on how to apply proxemic interaction design for domestic ubiquitous computing environments. This framework allows the rapid building of proxemic-aware systems and it offers a flexible architecture for sensing proxemic data from different types of sensors. However, the implementation of this framework requires a hardware architecture based on fixed devices (e.g., a Kinect Depth sensor and a client-server architecture) for allowing the server to process the proxemic information from appliances.

These studies demonstrate the current interest for researchers to develop tools that support the design and implementation of proxemic applications. However, proxemic interaction on mobile devices has not been implemented in full. In the next section, we describe the framework for developing proxemic mobile apps.

III. Framework to Develop Proxemic Mobile Apps

We propose a framework for supporting the design and development of proxemic mobile apps to control and manage a proxemic environment (denoted as P_E), based on mobile technology and smart wearable technology. In order to P_E, we propose a methodological process comprised by three single steps:

1) Define the distances that delimit the proxemic zones (denoted as P_Z), according to which entities will interact: intimate zone (denoted as $P\_Z_{intimate}$), personal zone (denoted as $P\_Z_{personal}$), social zone (denoted as P_Z_{social}), and public zone (denoted as P_Z_{public}).
2) Define the appropriate DILMO combination to define the HCI for the target mobile app to be developed.
3) Implement the mobile app, considering the technology supported by the entities in the P_E.

To support this development process, the framework is composed by mainly three components aligned with each step (see Figure 1): (i) Proxemic Zones module; (ii) DILMO module; and (iii) an API that supports the instantiation of the two previous mentioned components.

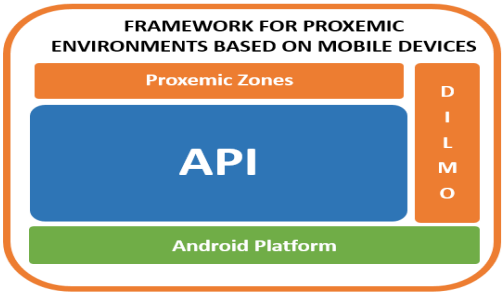

Fig. 1. Framework architecture.

1) The **Proxemic Zones module** allows the definition of the four proxemic zones (i.e., $P\_Z_{intimate}$, $P\_Z_{personal}$, P_Z_{social}, and P_Z_{public}), according to user needs. The interaction between two entities changes in accordance of the P_Z in which they are located. DILMO measurements are considered, based on the P_Z of the entities.
2) The **DILMO module** allows defining different scenarios of HCI on a P_E, based on different combinations of DILMO dimensions. This module provides a nomenclature to describe each combination of such proxemic dimensions (see Figure 2). For each DILMO combination, it is important to identify or create the interaction objects (i.e., entities) that will interact in the defined P_E. In proxemic environments, an entity (denoted as E) is an interaction object that can be detected by another E; if these interaction objects have a unique identification or specific role, they are designated as identities (denoted as I). Figure 2 is a guide that allows the developer to know which methods must be implemented on the API or which object must be created from the API according to DILMO for processing proxemic information. For example, a DIL proxemic environment means that Distance (D) and Location (L) are considered for Identities (I). Combination of proxemic dimensions are also valid, although the entities have not unique identification; e.g., *a person*, *a device beacon*, instead of *the smartphone's owner*, *my device beacon*. Hence, proxemic environments denoted in rows 10 to 20 in Figure 2, can become as `DL`, `DM`, `DO`, `LM`, `LO`, `MO`, `DLM`, `DLO`, `DMO`, `LMO`, and `DLMO`, respectively, when all interaction objects are not identifiable entities.
3) The **API** facilitates developers processing proxemic information and values. The API provides classes and methods to define the P_Z, as well as to manage the different combinations of DILMO dimensions. For example, for a DIL proxemic environment, methods to identify entities (I) and to process D and L are available in DILMO class. Thus, the API behaves as a bridge between the Proxemic Zones and DILMO modules.

In the current version of our framework, the API considers the extraction of DILMO values from smartphones or mobile

devices based on the Android operating system by using motion sensors and cameras that the majority of smart devices have in their hardware configuration [10]. For example, through the BLE mechanisms [11], it is possible to know the distance (D) between two mobile devices. Another way to estimate the distance between two entities is to use computer vision. In the next section, we describe the proof-of concept.

	D	I	L	M	O	mix	Proxemic Environment
1	■					D	Physical length (D) between entities.
2		■				I	Identifiable (I) entities in a specific role in the interaction space.
3			■			L	Position (L) of an interaction object (entity).
4				■		M	Motion (M) capture of an interaction object (entity).
5					■	O	Face orientation (O) between two entities.
6	■	■				DI	Interaction based on proximity (D) between identifiable (I) entities.
7		■	■			IL	Interaction based on the physical location (L) of identifiable (I) entities.
8		■		■		IM	Interaction based on movement tracking (M) of identifiable (I) entities.
9		■			■	IO	Interaction based on face to face orientation (O) of identifiable (I) entities.
10	■	■	■			DIL	Interaction based on proximity (D) and positions (L) of identifiable (I) entities.
11	■	■		■		DIM	Interaction based on proximity (D) according to movement tracking (M) of identifiable (I) entities.
12	■	■			■	DIO	Interaction based on proximity (D) and face to face orientation (O) of identifiable (I) entities.
13		■	■	■		ILM	Interaction based on physical location (L) and movement tracking (M) of identifiable (I) entities.
14		■	■		■	ILO	Interaction based on face to face orientation (O) and position (L) of identifiable (I) entities.
15		■		■	■	IMO	Interaction based on movement tracking (M) and face to face orientation (O) of identifiable (I) entities.
16	■	■	■	■		DILM	Interaction based on proximity (D) and physical location (L) with movement tracking (M) of identifiable (I) entities.
17	■	■	■		■	DILO	Interaction based on proximity (D), location (L) and face to face orientation (O) of identifiable (I) entities.
18	■	■		■	■	DIMO	Interaction based on proximity (D) and face to face orientation (O) according to movement (M) of identifiable (I) entities.
19		■	■	■	■	ILMO	Interaction based on location (L) and face to face orientation (O) according to movement (M) of identifiable (I) entities.
20	■	■	■	■	■	DILMO	Interaction based on all proxemic interaction dimensions.

Fig. 2. DILMO proxemic dimensions and nomenclature to describe each combination that is available through the API methods for processing proxemic information.

IV. Proof-Of-concept Of our Framework

Our goal is to create proxemic environments based on mobile devices or wearable technology and demonstrate that our framework allows developers to build proxemic mobile applications effectively. For this purpose, we show the implementation of two mobile applications, called IntelliPlayer and Tonic, based on proxemic interactions. These apps were implemented using Android Studio version 3.3.

Both apps have been developed by undergraduate students, as part of their final project in computer science. The developing team was integrated by four students whose average age was 21 years-old. Two training sessions of two hours each were organised for the students. The training process allows students to understand the development process for building proxemic mobile applications with our framework. They learned: (i) how to define each P_Z; (ii) how to select each proxemic dimension for recreating a P_E; and (iii) how to use methods and classes in the API. The development time of both applications was 64 hours by two developers over a period of 4 weeks. IntelliPlayer took 44 hours of work, while Tonic was finished in 20 hours, in the same four weeks.

IntelliPlayer is a mobile application that plays a video in a smartphone and reacts according to four proxemic zones and DILO proxemic dimensions. In the first step of the methodological approach the four P_Z were created: P_Z_{intime} (0 mts to 0.25mts), $P\_Z_{personal}$ (0.26 mts to 0.45 mts), P_Z_{social} (0.46mts to 1 mts), and P_Z_{public} (1.1 mts to 2 mts). Then, in the second step, the HCI was designed according to D, I, L, and O, thus a DILO P_E was defined. Figure 5 shows the proxemic zones that have been defined by developers through the API, as shown in Figure 3.

```
//Définition des zones proxémiques utilisée
proxzone = new ProxZone(0.25D, 0.45D, 1.0D, 2.0D)
```

Fig. 3. Example of ProxZone Class Constructor invocation.

With this application, we illustrate a proxemic environment using a mobile player app that reacts to the distance (D) and location (L) of a person (E_1) and his face orientation (O), with respect to the smartphone (I_1) displaying a video. The computer vision technique has been used for this purpose, based on the properties of an Android camera and through the API methods `setFaceHeight(float faceHeight)` and `getDistance()`. Figure 4 shows a block code of this case.

```
Distance d= new Distance();
d.setfaceHeight(faces.valueAt(i).getHeight());
String id =String.valueOf(faces.valueAt(i).getId());
dilmo.setProxemicDI(id, d.getDistance());
dilmo.getProxemicDI(id);
changerVolume(dilmo.getProxemicDI(id));
```

Fig. 4. Block code of IntelliPlayer.

With the distance (D) between the user (E_1) and the smartphone (I_1), IntelliPlayer determines the proxemic zone (P_Z) of E_1 (a user), with respect to the smartphone (I_1). To do so, it invokes the method `getProxemicZoneByDistance()`.

IntelliPlayer automatically adjusts the volume of the video according to the P_Z in which E_1 (the user) is with respect to the smartphone (I_1): when E_1 is in P_Z_{intime}, it decreases to 25% volume of speaker; for $P\_Z_{personal}$, it increases to 50% volume; for P_Z_{social}, it increases to 75% volume; and for P_Z_{public}, it increases to 100% volume) (see Figure 5).

Fig. 5. IntelliPlayer proxemic zones.

Another useful function of IntelliPlayer is to provide a video description that users can read on the screen according to user location (L). When a user (E_1 or E_2) is in the

$P\_Z_{personal}$ and her/his orientation (O) is in front of the screen, the application can obtain the face location (L) (by using `setRelativeLocationScreen(float L)` and `getRelativeLocationOnScreen()` methods from the API) to split the screen, with the video (running) and information about the video on the right or on the left, according the detected L.

Tonic is an educational mobile app for learning musical notes, developed for illustrative purposes. In the first step of the methodological approach, the students have defined four proxemic zones: P_Z_{intime} (0 mts to 0.5mts), $P\_Z_{personal}$ (0.51 mts to 1 mts), P_Z_{social} (1.1 mts to 2 mts), and P_Z_{public} (2.1 mts to 4 mts). In the second step, a DIMO combination was stated for the proxemic environment, P_E.

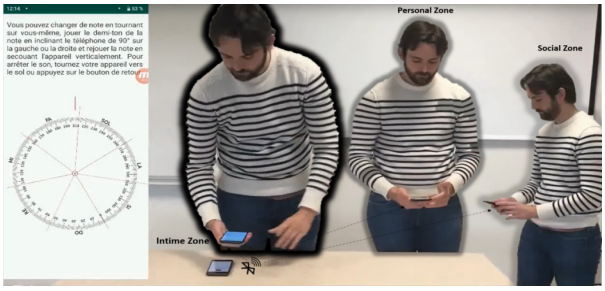

Fig. 6. Tonic Proxemic Zones based on Bluetooth Low Energy.

Tonic allows a user to play a note and modify it from his smartphome on another smart mobile. The user's smartphone is identified (I_1) by the mobile device which plays the sound (I_2). Thus, I_2 plays and modifies a sound, by using proxemic interactions based on the P_Z, and on D, I, M, and O dimensions (i.e., a DIMO proxemic environment). In Tonic, the distance (D) between the two devices is obtained by using BLE technology. I_1 broadcasts its identifier to nearby portable electronic devices, thus it is caught by I_2. The volume of the sound is adjusted according to the P_Z in which I_2 is, with respect to I_1 (see Figure 6). The musical notes are changed according to the movement (M) and orientation (O) of I_2, with respect to I_1. According to M a tone is increased/decreased, while according to O a semi-tone is increased/decreased. M and O are calculated based on the capabilities of the smartphone, such as accelerometer, gyroscope, compass, and magnetometer. These sensors provide proxemic information that is mainly used in the API. Movement M was determined by using methods `setAzimuthWithRange(float MAX, float MIN, float value)` and `isAzimuthInRange()`; while O was managed by methods based on the smartphone's technical capacities. To manage P_Z and D, the methods used from the API, in the third step of the approach: `getProxemicDI(String I)`, `setProxemicDIL(String I,double D,float L)`, and `setBluetoothDistance(double rssi, double txtPower)`.

The development of these applications show how the framework provides a development process that allowed students to rapidly build proxemic apps and creating proxemic environments based on the exclusive use of mobile devices. These apps offer a portable implementation that facilitates using the proxemic interaction in comparison to previous works that have used fixed platforms for similar purposes.

V. Conclusion

Proxemic interaction provides different options for HCI and mobile interaction while offering several advantages and better experiences for users. Through this work, we have explored the use of proxemic interaction based on the combination of proxemic dimensions – i.e., Distance, Identity, Location, Movement, and Orientation (DILMO) – to offer a new context-oriented interaction for mobile applications. We have presented a methodological process to guide developers on creating realistic proxemic environments supported by an API and a framework, in which the end-users only need to use mobile or wearable devices. The API [1] and Apps[2] are available for free downloading.

References

[1] F. Brudy, C. Holz, R. Rädle, C.-J. Wu, S. Houben, C. N. Klokmose, and N. Marquardt, "Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 562, https://doi.org/10.1145/3290605.3300792.

[2] J. E. Grønbæk, C. Linding, A. Kromann, T. F. H. Jensen, and M. G. Petersen, "Proxemics play: Exploring the interplay between mobile devices and interiors," in *Proceedings of Companion Publication of the Conference on Designing Interactive Systems*, ser. DIS '19. ACM, 2019, pp. 177–181, https://doi.org/10.1145/3301019.3323886.

[3] T. Ballendat, N. Marquardt, and G. Saul, "Proxemic interaction: designing for a proximity and orientation-aware environment," in *Proceedings of International Conference on Interactive Tabletops and Surfaces*, ser. ITS' 10. ACM, 2010, pp. 121–130, http://doi.org/10.1145/1936652.1936676.

[4] S. Greenberg, N. Marquardt, T. Ballendat, R. Diaz-Marino, and M. Wang, "Proxemic interactions: the new ubicomp?" *Interactions*, vol. 18, no. 1, pp. 42–50, 2011, https://doi.org/10.1145/1897239.1897250.

[5] J. E. Grønbæk, M. S. Knudsen, K. O'Hara, P. G. Krogh, J. Vermeulen, and M. G. Petersen, "Proxemics beyond proximity: Designing for flexible social interaction through cross-device interaction," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–14, https://doi.org/10.1145/3313831.3376379.

[6] E. T. Hall, *The Hidden Dimension: An anthropologist examines man's use of space in private and public*. New York: Anchor Books; Doubleday & Company, Inc, 1966.

[7] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg, "The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ser. UIST '11. ACM, 2011, pp. 315–326, https://doi.org/10.1145/2047196.2047238.

[8] C. Cardenas and J. A. Garcia-Macias, "Proximithings: Implementing proxemic interactions in the internet of things," *Procedia Computer Science*, vol. 113, pp. 49–56, 2017, https://doi.org/10.1016/j.procs.2017.08.286.

[9] D. Ledo, S. Greenberg, N. Marquardt, and S. Boring, "Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies," in *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI'2015. ACM, 2015, pp. 187–198, https://doi.org/10.1145/2785830.2785871.

[10] G. Developers, "Sensors overview, developer guides," 2019, https://developer.android.com/guide/topics/sensors.

[11] AltBeacon, "The open and interoperable proximity beacon specification," 2018, https://altbeacon.org/.

[1]The API is available in https://www.iutbayonne.univ-pau.fr/ ppdaza/

[2]The APPS are available in https://https://github.com/llagar910e/