Designing Proxemic-aware Cross-Device Applications: A Feasibility Study

Pedro Albuquerque Santos
NOVA LINCS, NOVA School of Science & Technology, NOVA University Lisbon
Portugal
Sustain.RD, Setúbal School of Technology, Polytechnic Institute of Setúbal
Portugal
pe.santos@campus.fct.unl.pt

Rui Neves Madeira
NOVA LINCS, NOVA School of Science & Technology, NOVA University Lisbon
Portugal
Sustain.RD, Setúbal School of Technology, Polytechnic Institute of Setúbal
Portugal
rui.madeira@estsetubal.ips.pt

Nuno Correia
NOVA LINCS, NOVA School of Science & Technology, NOVA University Lisbon
Portugal
nmc@fct.unl.pt

ABSTRACT

We live surrounded by the most varied computing devices, which may give us the opportunity to combine them to form a unified and richer user experience. Considering this opportunity, we created the *UnaxY Framework* to support the development of applications with UI components distributed by co-located devices. This paper is focused on a feasibility study based on two prototype applications created using *UnaxY*. We performed user studies to evaluate concepts associated to this type of applications and the framework they were based on. We had a special interest in assessing how managing the application state and collaborating across devices would be perceived and received by users. The results are positive and clearly indicate that we should continue developing solutions that support a generalized implementation of applications with the user interaction spanning multiple devices.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **Ubiquitous and mobile computing**.

KEYWORDS

cross-device interaction, application framework, proxemics, user study, UX, HCI

ACM Reference Format:
Pedro Albuquerque Santos, Rui Neves Madeira, and Nuno Correia. 2021. Designing Proxemic-aware Cross-Device Applications: A Feasibility Study. In *20th International Conference on Mobile and Ubiquitous Multimedia (MUM 2021), December 5-8, 2021, Leuven, Belgium.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3490632.3490658

This work is licensed under a Creative Commons Attribution-Share Alike International 4.0 License.

MUM 2021, December 5-8, 2021, Leuven, Belgium
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8643-2/21/05.
https://doi.org/10.1145/3490632.3490658

1 INTRODUCTION AND BACKGROUND

We are "living in a ubiquitous world" as we face widespread ubiquitous computing (Ubicomp) due to the billions of devices shaping the very fabric of an active world [18]. Nowadays, computing devices can be found everywhere around people, who are increasingly using a large number of them in their daily lives. Nevertheless, applications are typically confined to a single device.

We embrace the vision behind *Liquid Software* [8, 20]. We consider that it is important and beneficial to take better advantage of the devices around people by combining and integrating them. Therefore, we decided to develop a framework that allows the creation of cross-device applications that have their user interface (UI) spanning multiple co-located devices by matching their capabilities with the requirements of each UI component. The initial challenge for building the framework came from research about combining mobile devices with situated displays [14, 19]. Moreover, people also started to spontaneously use multiple screens at the same time by using mobile devices while watching TV to access trivia and engage with other viewers [5, 12].

Paternò and Santoro present a logical framework for understanding, analyzing, and comparing features of existing research on cross-device UIs [17]. Particularly, Paternò published an in-depth survey of research in this area [16]. Besides that we have found some previous efforts related with our work. For instance, *Liquid.js* enables the development of cross-device applications that operate on a shared decentralized state using Web-based technologies [4]. *Panelrama* and *AdaM* allow the specification of additional semantics to components which are used to automatically distribute them [15, 21]. Meanwhile, *XDBrowser* segments Web pages and distributes the fragments across devices, while *Vistribute* is capable of adapting to a multitude of devices depending on the available views and data [9, 13].

It is important to understand how people position themselves when multiple users interact with co-located devices. This lead us to the concept of proxemics which deals with how people use interpersonal distance to mediate their interactions [7]. Likewise, users should expect increasing connectivity and interaction possibilities as they bring their devices closer together [6]. Saul Greenberg and his team developed the *Proximity Toolkit* to test proxemic-aware applications [10]. They also developed *GroupTogether* to enable co-located collaboration across devices by using sensors in the devices and in the environment to capture the relationship between users and devices [11]. After studying existing solutions, we also developed an indoor positioning system (IPS) based on *Wi-Fi* and *Bluetooth Low Energy* in order to capture the proxemic relationships of the devices to enable association and interaction across them [3].

Our work intends to generalize the development of cross-device applications, which incorporate the proxemic relationships of co-located devices into their interaction experience. This paper presents a feasibility study for this vision based on two prototype applications that was conducted to evaluate the concept behind these applications and their development using our framework.

2 FRAMEWORK FOR CROSS-DEVICE APPLICATIONS

This section is only intended to provide a brief overview of the *YanuX Framework* just to make the study more understandable. The *YanuX Framework* [1] was created to enable the development of applications that span their UI across multiple co-located devices. Detailed documentation and source code for the framework components and application prototypes can be found at: https://yanux-framework.github.io/.

Figure 1 shows the framework's architecture. The components at the bottom are server-side components and include: authentication and authorization services (*YanuX Auth*); a broker to sync data across multiple application instances (*YanuX Broker*); an indoor positioning system (*IPS Server*) and an integration module that connects it to the rest of the framework (*YanuX IPS Bridge*). The devices running the applications appear on top. Each one hosts multiple applications (*Apps*) and access the framework's services through a library that can be used in web applications (*YanuX Coordinator*). The devices run a service to gather their capabilities (*YanuX Orchestrator*) and a client that is part of the positioning system (*IPS Client*).

The framework allows users to store multiple application states (*resources*) per application that can be shared to allow collaboration. Therefore, we have developed a component to assist with the management of multiple *resources* (Figure 2). The automatic distribution of components takes into account the restrictions defined by the developers along with the proxemic relationships and capabilities of the devices. However, we created a component that allows users to change the distribution of UI according to their preferences (Figure 3). The last column indicates if the distribution for a particular instance was determined automatically (pressed green state) or modified by a user (unpressed red state).

3 FEASIBILITY STUDY

Our study was composed of two phases. The first one was mainly focused on experimenting with the reusable components that were introduced into our solution (see Figures 2 and 3). The *Calculator* prototype was purposely built to be as simple as possible to keep the focus on those components. The second phase was intended to study a more realistic scenario to better explore the collaboration capabilities and the concept of cross-device applications. Therefore, we developed a prototype called *JuxtBoard*, which is a pin board application with support for taking multimedia notes.

3.1 Calculator and JuxtBoard Prototypes

The *Calculator* is split into two components: a *screen* with a line for the current arithmetic expression and another line for the current result; and a *keypad* with the numerical digits and the basic operations of a calculator. They can be shown in the same device (Figure 4) or separately (Figures 5 and 6). If users scroll down, they will find the *resource management* and *components distribution* elements (Figures 2 and 3, respectively). Moreover, we envisioned a scenario where the *screen* component is displayed on larger screens that primarily serve as output devices. The *keypad* component should be placed on personal devices to perform the role of input devices (e.g., a smartphone).

The *JuxtBoard* application allows users to manage collections which contain notes of different types. Each note can be either a text, image or video note. The UI has three components that can be distributed by multiple devices: the *List* shows previews of each note (Figure 7); the *Note* displays a note in full-screen (Figure 8); and the *Edit* component allows to edit a note (Figure 9). The *resource management* element allows users to manage *resources* representing collections of notes that can be shared to enable collaboration. Meanwhile, the *components distribution* element can be used to distribute UI components. The scenario designed for *JuxtBoard* was to let users manage notes using the *List* and *Edit* components on personal devices. If a large display is available, the *Note* component would be displayed there for a better viewing experience and to foster collaboration between multiple users.

3.2 Participants and Design

Regarding the user study with the *Calculator*, we had a total of 13 participants and for the *JuxtBoard* based study we had 27 participants. The user studies took place in a University campus, so most participants were young adults. We acknowledge these samples are not representative of the general population. Nevertheless, the results should generalize well for this population group and they can be indicative of a current general trend and a strong perspective of the future.

In both cases, participants were told that they had to collaborate with another user to perform some tasks. The role of the other user was performed by a researcher. The test setup consisted of a PC connected to a large display and two smartphones. Each participant sat in front of the display showing the application and received a smartphone logged in with the same user account. The researcher would take the other smartphone, which was logged in with a separate account, and sit besides the participant.

In the case of the *Calculator*, the test started with the PC monitor showing the *screen* component like in Figure 5 and the smartphone of the participants showing the *keypad* like in Figure 6. The participants went through scenarios in which they had to use the application to collaborate with the researcher. For*JuxtBoard*, the tasks were performed twice to compare the application in single device mode, i.e., using a smartphone to display all components, and in cross-device mode, i.e., using the smartphones to display the *List* and *Edit* components and a large monitor to display the *Notes* component. Half of the participants tried the single device mode first and the other half the cross-device mode.

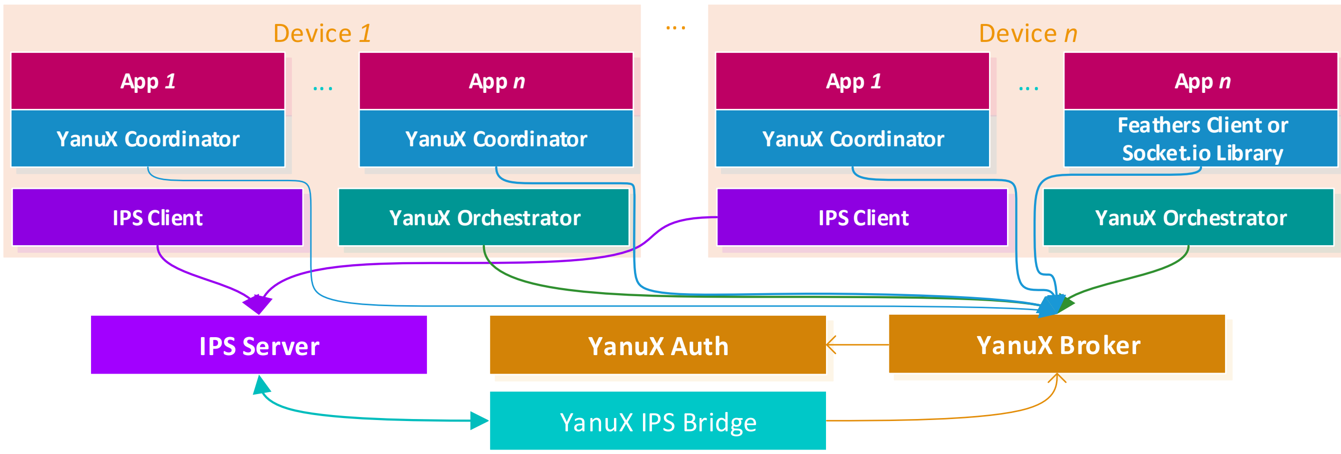

Figure 1: *YanuX Framework* Simplified Architecture

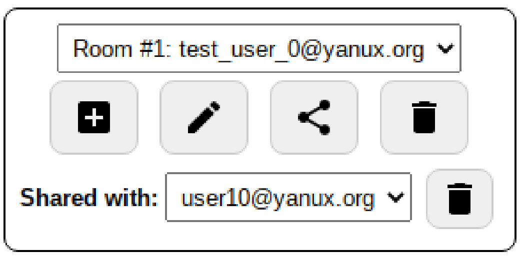

Figure 2: *YanuX Resource Management Element*

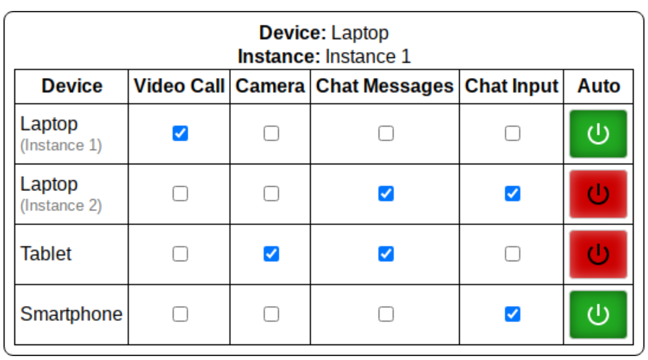

Figure 3: *YanuX Components Distribution Element*

After using the prototypes in cross-device mode, the participants were asked to move away from the large display and the researcher accompanied them. In the case of the *Calculator*, the smartphones started to show both components, like in Figure 4, because there was no longer a large screen in the vicinity. Therefore, the smartphones fell back to showing the *screen* on a sub-optimal display. In *JuxtBoard*, the *Note* component started to be shown on the participant's and researcher's smartphones because they were no longer facing the monitor or they were too far away from it. In both studies, participants were told to move to the initial position and to take notice that the initial distribution was automatically reestablished. They could then experiment with the manual distribution of UI components and with resetting the distribution to automatic. Finally, they were asked to fill a questionnaire about their experience.

3.3 Results and Discussion

We present a summary of the responses to some domain specific questions (Q1-Q5 were asked in both user studies, while Q6-Q8 were asked only in the *Calculator* study). Table 1 presents the relative frequency of the level of agreement (Likert scale from *1 - Strongly Disagree* to *7 - Strongly Agree*), median ($\tilde{x}$), mean ($\bar{x}$) and standard deviation (s) of the answers. The results are generally good, with a high level of agreement with the statements.

The results of *Q1* and *Q3* are indicative that our system reacts to input according to user expectations and that our indoor positioning solution can keep up with the changes in the proxemic relationship of co-located devices. Moreover, *Q2* confirms that distance between devices can be used to determine if devices are related. The automatic distribution of UI components was also within expectations according to *Q4*. Meanwhile, *Q5* and *Q6* indicate that it was easy to change the components distribution and to reset the distribution to automatic. Nevertheless, we received feedback about making the *Auto* buttons more predictable and to improve their visual cues. *Q7* and *Q8*, which focused on managing and sharing application states (*resources*) got very good results. However, in the *Calculator*, some users expected that after creating a *resource* the application would automatically switch on to it. This was fixed for the *JuxtBoard* study.

The *Calculator* obtained a *SUS* (System Usability Scale) score of 87.69 [2]. This value is well above the threshold of 70, which is generally considered to be the minimum for an acceptable level of usability. It is within the range of *Excellent* usability corresponding to a solid *B* grade. Even if we consider the lower bound of the 95% confidence interval ([80.43, 94.95]) it can be considered a *Good* rating. *JuxtBoard* got a *SUS* score of 85.68 for the single-device mode and 89.87 for the cross-device mode [2]. These are also *Excellent* scores corresponding to a *B* grade. These ratings remain unchanged even if we consider the 95% confidence intervals of [83.31, 88.04]

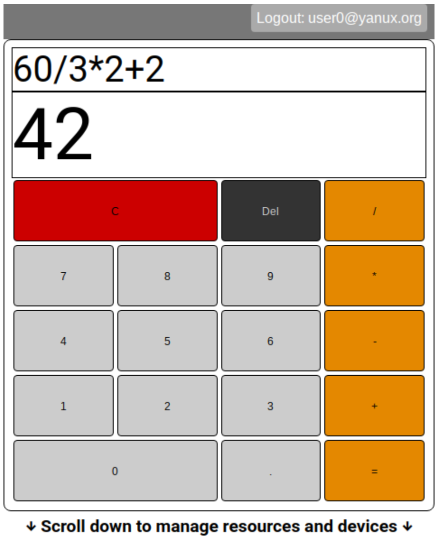

Figure 4: *YanuX Calculator* – All components

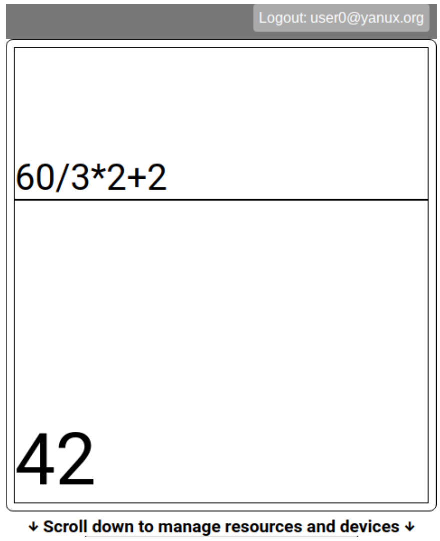

Figure 5: *YanuX Calculator* – Only the *screen* component

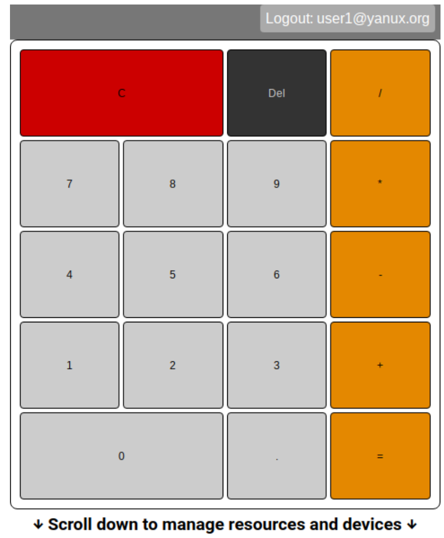

Figure 6: *YanuX Calculator* – Only the *keypad* component

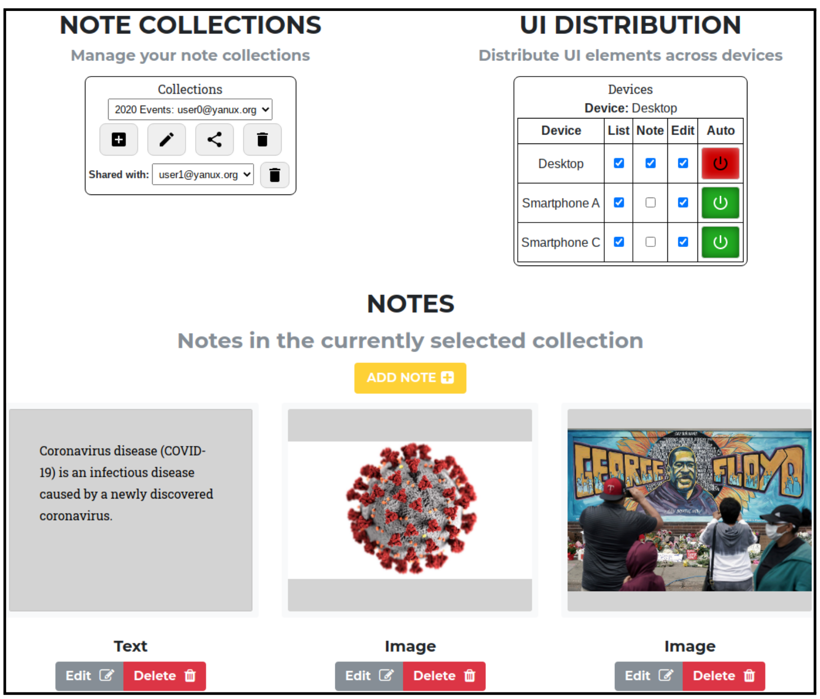

Figure 7: *JuxtBoard* running on PC with all components manually assigned to it. The *List* component is shown in the bottom half.

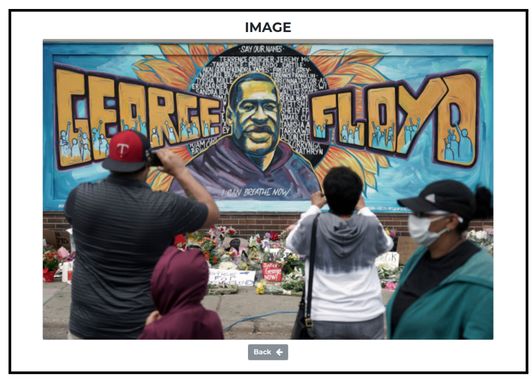

Figure 8: *JuxtBoard* – The *Note* component.

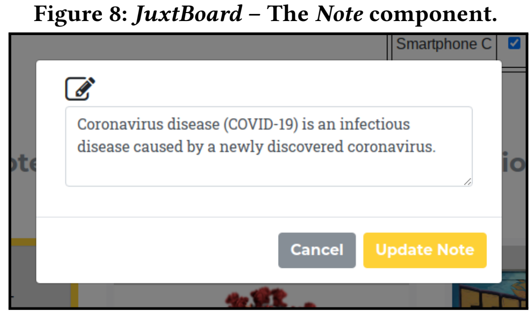

Figure 9: *JuxtBoard* – The *Edit* component.

and [86.53, 93.22], respectively. Although relatively close, the results show that users have a preference towards the cross-device mode. A paired sample t-test has a p-value of 0.003 which is below the common threshold of 0.05 required to reject the null hypothesis.

Participants were also asked to rate their preference between the two interaction modes they experienced. For this question, a positive integer between 1 and 3 meant an increasing preference towards the cross-device mode; a negative integer between -1 and -3 meant an increasing preference towards single device mode; and a 0 means no preference between the two modes. Table 2 summarizes the responses to these questions.

Table 1: Summary of Domain Specific Questions (*Calc* for the *YanuX Calculator* and *JB* for the *JuxtBoard* application)

App	1	2	3	4	5	6	7	$\tilde{x}$	$\bar{x}$	s
Q1: The time it takes for the application to react to my input is adequate.										
Calc	0.0%	0.0%	0.0%	7.7%	7.7%	38.5%	46.2%	6.0	6.23	0.93
JB	0.0%	0.0%	0.0%	0.0%	7.4%	40.7%	51.9%	7.0	6.44	0.64
Q2: Distance is a good measure of how closely related two devices are.										
Calc	0.0%	0.0%	0.0%	15.4%	7.7%	23.1%	53.8%	7.0	6.15	1.14
JB	0.0%	0.0%	0.0%	0.0%	7.4%	40.7%	51.9%	7.0	6.44	0.64
Q3: The shared display reacted to my presence is quickly enough.										
Calc	0.0%	0.0%	0.0%	0.0%	23.1%	38.5%	38.5%	6.0	6.15	0.80
JB	0.0%	0.0%	0.0%	3.7%	3.7%	44.4%	48.1%	6.0	6.37	0.74
Q4: The automatic distribution of UI elements across devices is adequate.										
Calc	0.0%	0.0%	0.0%	0.0%	15.4%	38.5%	46.2%	6.0	6.31	0.75
JB	0.0%	0.0%	0.0%	7.4%	7.4%	37.0%	48.1%	6.0	6.26	0.90
Q5: Changing the distribution of UI elements to suit my needs is easy.										
Calc	0.0%	0.0%	0.0%	0.0%	23.1%	23.1%	53.8%	7.0	6.31	0.85
JB	0.0%	0.0%	3.7%	3.7%	14.8%	37.0%	40.7%	6.0	6.07	1.04
Q6: I could easily figure out how to reset the UI distribution to automatic.										
Calc	0.0%	0.0%	7.7%	7.7%	15.4%	0.0%	69.2%	7.0	6.15	1.41
Q7: It was easy to add, rename or remove resources (application states).										
Calc	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100%	7.0	7.0	0.0
Q8: I could easily share resources (application states) with other users.										
Calc	0.0%	0.0%	0.0%	0.0%	0.0%	23.1%	76.9%	7.0	6.77	0.44

Table 2: *JuxtBoard*: Summary of Single Device vs. Cross-Device Mode

-3	-2	-1	0	1	2	3	$\tilde{x}$	$\bar{x}$	s
Q1: In which mode was it easier to manage the collections of notes in the application?									
0.0%	3.7%	0.0%	59.3%	18.5%	14.8%	3.7%	0.0	0.52	1.01
Q2: In which mode was it easier to manage the notes in the application?									
0.0%	3.7%	0.0%	59.3%	11.1%	18.5%	7.4%	0.0	0.63	1.15
Q3: In which mode was it easier to collaborate and show notes to other users?									
0.0%	0.0%	3.7%	3.7%	3.7%	22.2%	66.7%	3.0	2.44	1.01
Q4: What is your degree of preference between the two modes when using the application to perform tasks alone?									
22.2%	33.3%	3.7%	14.8%	7.4%	0.0%	18.5%	-2.0	-0.74	2.18
Q5: What is your degree of preference between the two modes when using the application to perform tasks in collaboration with other users?									
0.0%	0.0%	3.7%	3.7%	3.7%	14.8%	74.1%	3.0	2.52	1.01

We wanted to assess if adding a cross-device mode would make it more difficult for people to use the application. *Q1* and *Q2* do not give a clear advantage to any interaction mode, with most participants considering it to be easy to perform the basic tasks of the application in either of the modes. Therefore, enhancing existing applications with cross-device interaction should not significantly hinder their ease of use. *Q3* and *Q5* were used to assess if users found the addition of more devices to be advantageous in collaborative situations. The responses of the participants give a clear advantage to the cross-device scenario for this type of usage, hinting that these are the scenarios in which this type of experience is valued. Conversely, *Q4* show us that when users are alone they may tend to use a single device. However, there is a significant percentage of users choosing 0 (neutral) or 3 (maximum preference for cross-device).

4 CONCLUSIONS AND FUTURE WORK

We have briefly presented the *YanuX Framework* along with two application prototypes, which were used to perform the feasibility study in two phases. We wanted to evaluate how potential end users would receive the concepts promoted by our research. We allowed users to easily manage the distribution of UI components according to their preferences. We also enabled users to manage and share multiple application states (*resources*) to enable collaboration.

The results and feedback we received were largely positive with users demonstrating interest in our vision and in the possibilities opened by cross-device applications. Most of the participants showed enthusiasm while taking part of our user studies due to the novelty, usefulness and perceived advantages of a way to have their applications seamlessly integrate across all of the devices they use in their daily lives.

The framework along with its system components, the indoor positioning system and the application prototypes performed adequately according to responses to the questionnaire and to spoken feedback. For instance, the applications were responsive enough to the user input and to their movements. Users found it intuitive that the distribution of components across devices changes according to the distance between them and their relative orientation. Notwithstanding, the participants found it straightforward to manually change the distribution if they so desire. They were also able to manage multiple application states and to share them to collaborate with the researcher. Users preferred to use the cross-device interaction mode for collaboration, but many were comfortable with using it alone since the mode was not perceived as being needlessly complex.

Given the promising results, we find that it is feasible to keep developing this type of applications by using a framework that simplifies their development and an indoor positioning system that captures the proxemic relationship of the devices. We will also continue to refine the framework to support more scenarios. We intend to enable the distribution of UI components and integration between distinct applications on multiple devices. We plan to develop more complex applications and continue studying how people can benefit from these applications. Moreover, we will also conduct studies with developers to evaluate if they can understand how to build applications using it.

ACKNOWLEDGMENTS

This work is funded by Fundação para a Ciência e Tecnologia (FCT) through a Ph.D. Studentship grant (SFRH/BD/96899/2013), and supported by NOVA LINCS Research Center, which is partially funded by UID/CEC/04516/2020, granted by FCT.

REFERENCES

[1] Pedro Albuquerque Santos, Rui Pedro da Costa Porfírio, Rui Neves Madeira, and Nuno Correia. 2021. Computational Framework to Support Development of Applications Running on Multiple Co-Located Devices. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Virtual Event, Netherlands) *(EICS '21)*. Association for Computing Machinery, New York, NY, USA, 63–69. https://doi.org/10.1145/3459926.3464758

[2] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Stud.* 4 (2009), 114–123. http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS{_}Bangor{_}May2009.pdf

[3] Ramon F. Brena, Juan Pablo García-Vázquez, Carlos E. Galván-Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, and James Fangmeyer. 2017. Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors* 2017 (March 2017), 1–21. https://doi.org/10.1155/2017/2630413

[4] Andrea Gallidabino and Cesare Pautasso. 2016. The Liquid.js Framework for Migrating and Cloning Stateful Web Components across Multiple Devices. In *Proc. 25th Int. Conf. Companion World Wide Web - WWW '16 Companion*. ACM Press, New York, New York, USA, 183–186. https://doi.org/10.1145/2872518.2890538

[5] Fabio Giglietto and Donatella Selva. 2014. Second Screen and Participation: A Content Analysis on a Full Season Dataset of Tweets. *J. Commun.* 64, 2 (April 2014), 260–277. https://doi.org/10.1111/jcom.12085

[6] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. 2011. Proxemic interactions: the new ubicomp? *interactions* 18, 1 (Jan. 2011), 42. https://doi.org/10.1145/1897239.1897250

[7] Edward Twitchell Hall. 1990. *The Hidden Dimension* (27 ed.). Anchor Books. 217 pages.

[8] John Hartman, Udi Manber, and L Peterson. 1996. *Liquid software: A new paradigm for networked systems.* Technical Report. Tucson, AZ, USA.

[9] Tom Horak, Andreas Mathisen, Clemens N. Klokmose, Raimund Dachselt, and Niklas Elmqvist. 2019. Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups. In *Proc. 2019 CHI Conf. Hum. Factors Comput. Syst. - CHI '19*. ACM Press, New York, New York, USA, 1–13. https://doi.org/10.1145/3290605.3300846

[10] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In *Proc. 24th Annu. ACM Symp. User interface Softw. Technol. - UIST '11*. ACM Press, New York, New York, USA, 315. https://doi.org/10.1145/2047196.2047238

[11] Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. Cross-device interaction via micro-mobility and f-formations. In *Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST '12*. ACM Press, New York, New York, USA, 13. https://doi.org/10.1145/2380116.2380121

[12] Timothy Neate, Matt Jones, and Michael Evans. 2017. Cross-device media: a review of second screening and multi-device television. *Pers. Ubiquitous Comput.* 21, 2 (April 2017), 391–405. https://doi.org/10.1007/s00779-017-1016-2

[13] Michael Nebeling. 2017. XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 4574–4584. https://doi.org/10.1145/3025453.3025547

[14] Tim Paek, Maneesh Agrawala, Sumit Basu, Steve Drucker, Trausti Kristjansson, Ron Logan, Kentaro Toyama, and Andy Wilson. 2004. Toward universal mobile interaction for shared displays. In *Proc. 2004 ACM Conf. Comput. Support. Coop. Work - CSCW '04*. ACM Press, New York, New York, USA, 266. https://doi.org/10.1145/1031607.1031649

[15] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Feit, Hana Vrzakova, Niraj Dayama, Hui-Shyong Yeo, Clemens Klokmose, Aaron Quigley, Antti Oulasvirta, and Otmar Hilliges. 2018. AdaM: Adapting Multi-User Interfaces for Collaborative Environments in Real-Time. In *Proc. 2018 CHI Conf. Hum. Factors Comput. Syst. - CHI '18*. ACM Press, New York, New York, USA, 1–14. https://doi.org/10.1145/3173574.3173758 arXiv:1803.01166

[16] Fabio Paternò. 2019. Concepts and design space for a better understanding of multi-device user interfaces. *Univers. Access Inf. Soc.* 19, 2 (June 2019), 409–432. https://doi.org/10.1007/s10209-019-00650-5

[17] Fabio Paternò and Carmen Santoro. 2012. A logical framework for multi-device user interfaces. In *Proc. 4th ACM SIGCHI Symp. Eng. Interact. Comput. Syst. - EICS '12 (EICS '12)*. ACM Press, New York, New York, USA, 45. https://doi.org/10.1145/2305484.2305494

[18] Tom Rodden. 2008. Living in a ubiquitous world. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1881 (2008), 3837–3838. https://doi.org/10.1098/rsta.2008.0146

[19] Pedro Albuquerque Santos, Rui Neves Madeira, and Nuno Correia. 2013. FCT4U - When Private Mobile Displays Meet Public Situated Displays to Enhance the User Experience. In *2013 IEEE 10th Int. Conf. Ubiquitous Intell. Comput. 2013 IEEE 10th Int. Conf. Auton. Trust. Comput.* IEEE, 186–193. https://doi.org/10.1109/UIC-ATC.2013.112

[20] Antero Taivalsaari, Tommi Mikkonen, and Kari Systa. 2014. Liquid Software Manifesto: The Era of Multiple Device Ownership and Its Implications for Software Architecture. In *2014 IEEE 38th Annu. Comput. Softw. Appl. Conf.* IEEE, 338–343. https://doi.org/10.1109/COMPSAC.2014.56

[21] Jishuo Yang and Daniel Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 2783–2792. https://doi.org/10.1145/2556288.2557199