Autonomous Adaptation of User Interfaces to Support Mobility in Ambient Intelligence Systems

Gervasio Varela
Integrated Group for Engineering Research, University of A Coruña
C/ Mendizábal S/N, 15403, Ferrol, A Coruña, Spain
gervasio.varela@udc.es

ABSTRACT
The work presented in this paper is focused on building Ambient Intelligence (AmI) applications capable of moving from one environment to another, while their user interface keeps adapting itself, autonomously, to the variable environment conditions and the available interaction resources.

AmI applications are expected to interact with users naturally and transparently, therefore, most of their interaction relies on embedded devices that obtain information from the user and environment. This work implements a framework for AmI systems that elevates those embedded devices to the class of interaction resources. It does so by providing a new level of abstraction that decouples applications, conceptually and physically, from the different specific interaction resources available and their underlying heterogeneous technologies.

In order to drive the adaptation process to environment changes, the system makes use of a set of models that describe the user, environment conditions and devices, and algorithms for context-aware selection of the interaction devices.

Author Keywords
User Interfaces; Ambient Intelligence; Distributed User Interfaces; Context Adaptation.

ACM Classification Keywords
D2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces;* H5. [**Information interfaces and presentation**]: User Interfaces – *Interaction styles, Input devices and strategies, user interface management system (UIMS).*

General Terms
Human Factors; Design; Algorithms.

INTRODUCTION
The operation of an Ambient Intelligence (AmI) system is quite different from a classic software system. An AmI system is expected to behave proactively and transparently [1], interacting with the users and their environment in the most natural way available. To achieve that, AmI systems must rely on their capacity to use any interaction resource available in a smart environment.

These interaction resources are also quite different from traditional interaction devices. AmI applications use mainly sensing/actuation devices and appliances to interact with the user and the environment [1]. Because of that, they are exposed to a complex world populated by a wide range of different technologies and devices that can be used to implement their user interaction subsystems.

As the user moves from one place to another, the smart environment in which the system is operating is changing, and with it, the available devices, the users and even the environment conditions. Predicting this variability at design time is quite difficult, and because of this, the majority of AmI systems are designed for a specific environment with a specific set of users, devices and conditions.

The main objective of this work is to provide AmI developers with a set of abstractions that isolate them from the user interaction capabilities of the system, which, combined with a distributed UI management system capable of connecting those abstractions to end devices, at runtime, facilitates the development of AmI systems adaptable to different devices/environments/users [2, 3].

This work tries to fulfill this objective by designing and implementing a UI management system integrated into an existing AmI application platform [4]. It uses a model-driven approach [5] to build user interfaces by specifying a series of high level declarative models which, at runtime, are transformed into a set of distributed interaction devices. Applications are decoupled from the specific characteristics and technologies of these devices by using a distributed agent communication protocol called General Interaction Protocol (GIP), which is implemented by the devices and abstracts them as interaction resources.

This paper is organized as follows. Section 2 describes related work; Section 3 explains the contributions of this research and Section 4 presents the current state of development and conclusions.

RELATED WORK

The problem of UI adaptation and reusability in AmI applications has been previously identified by different authors. In [2] Blumendorf et al. introduce the problematic associated with user interaction in Ambient Assisted Living (AAL) environments, which are a subset of AmI. The paper presents a framework for UIs development for those environments that use a model-driven approach and context information to drive the adaptation at runtime. In [3] Abascal et al. identify the necessity of adaptation to the users, because their capabilities and disabilities can greatly impact the performance of the UI of an AmI system. They propose the use of three models, User model, Task model and Environment model, in order to support the autonomous adaptation of the application GUI.

The difficulties associated with user and application mobility in AmI systems have also been studied by many authors. Ranganathan et al. [6] and Satoh et al. [7] provide a similar approach to the migration of applications in Ubiquitous Computing (UC) environments. Even if not specifically related to UI migration, they show that one of the main difficulties when moving a highly distributed system is how to deploy it in a new set of computing devices. They state that the deployment must be achieved bearing in mind the requirements of the system components, as well as the characteristics of the computing resources available. In [8] the authors provide a good overview of the specific problematic of supporting mobility in AmI, while [9] provides a more insightful view of the problems associated with the migration of user interfaces in AmI environments. The two main problems identified are: detecting and integrating the different devices available in each environment; and the need to provide users with an adequate UI able to use the available devices.

UI adaptation to users and devices is also a highly studied topic by the Human-Computer Interaction (HCI) community. In 1999 [10] Thevenin and Coutaz introduced the term 'plasticity of user interfaces' as the capacity of a UI to support changes in the system's physical characteristics and in the environment while preserving usability. They also proposed the use of model-driven engineering techniques in order to support UI adaptation. This proposal has been very successful within the HCI community and many different authors have used it as the basis for their own approach to UI adaptation [5, 11]. Also in the field of UC and AmI, these kinds of model-driven approaches have been applied with proven results [2, 3].

Another topic of great importance for this thesis is the field of distributed multimodal user interfaces. Due to the intrinsic distributed nature of AmI systems, their UI will operate using devices that are physically distributed throughout the smart environment [1]. Furthermore, these devices will use different modalities to interact with the user. Two prominent approaches to distributed multimodal UIs are: the Cameleon-rt [12] reference model for distributed, migratable and adaptive user interfaces; and the W3C conceptual framework to support multimodal UIs [13]. These two frameworks share some similarities in their way of abstracting distributed UI resources, but the W3C approach ignores the adaptation problem.

INNOVATIVE APPROACH AND RELEVANCE OF TOPIC

As shown by exploring the state of the art, some work has been carried out in the topic of UI adaptation and mobility in UC and AmI systems [2, 3]. The approach followed by these projects is mainly focused on graphical user interfaces and their adaptation to the user characteristics and the displays available. In contrast, the novel approach presented by this work proposes the utilization of sensing/actuation devices and appliances as the interaction resources of an AmI system.

The main contribution of this work to its research field consists in providing a new level of abstraction that isolates applications from devices at the user interaction level. In the proposed framework devices are seen as generic user interaction resources, thus developers can achieve a great level of decoupling between their AmI applications and the hardware devices used to interact with the user and the environment. This decoupling makes applications more easily adaptable to new scenarios.

PROPOSED SOLUTION

The proposed solution, called Dandelion [14], is being developed integrated in the HI^3 general purpose AmI development platform [4]. Dandelion aims to facilitate the migration of HI^3 applications by decoupling them from the interaction devices. An overview block diagram of the solution can be seen in Figure 1. It uses a model-driven approach in which a series of models and device selection algorithms are used to build, at runtime, as the user moves

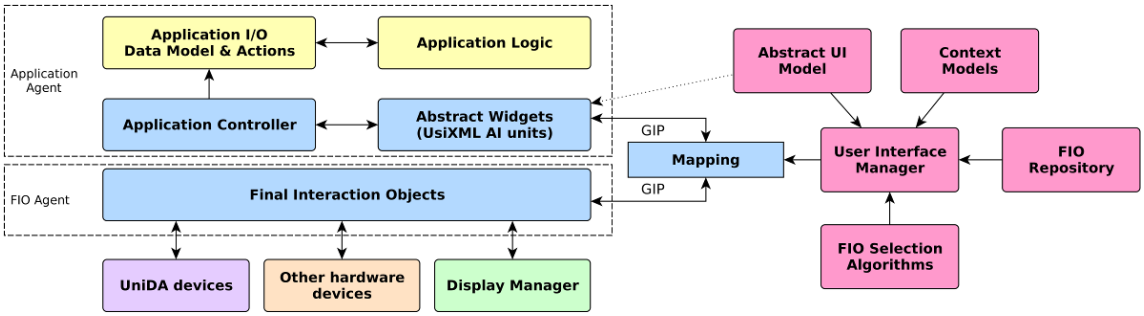

Figure 1. Block diagram of the proposed solution for UI adaptation in AmI

from one place to another, a UI that is appropriate for the user characteristics and preferences, the environment conditions and the devices available in each location.

The HI^3 [4] platform conceptual architecture follows a layer-based design that enables the division of system elements into levels. From bottom to top, the uniform device access layer, UniDA [15], provides homogeneous and distributed access to the physical devices. The sensing and actuation services layer provides virtual representations of sensors and actuators in the physical environment. The service layer is populated by components that provide shared functionalities to other services or applications. Finally, the applications layer is at the highest-level, and hosts the elements implementing applications that provide particular functionalities a user expects from the system.

Inside the HI^3 platform, Dandelion is integrated in the application and device abstraction layers by implementing a distributed user interface system inspired by the principles of the Cameleon-rt [12] reference model.

Dandelion follows a model-driven approach based on the recognized approaches proposed by UsiXML [15] and MASP [2, 11]. It reuses some of the models proposed by UsiXML, but instead of relying on model transformations at design time, it takes the MASP approach, using the models at runtime, along with real-time information of the environment and user, in order to build the user interface. The main difference with MASP is that Dandelion is focused on distributed UIs using not only displays, but every device available in a smart environment, like home automation devices, appliances or sensors. Thus, Dandelion is especially tailored for AmI applications requiring multi-modal interaction using everyday objects, but it can also accommodate GUIs, voice or gesture recognition.

The system is designed to support an arbitrary number of models describing the application domain, its interaction requirements, the user, the environment and the resources available. The information in those models is used to select the interaction elements (mainly automation devices and appliances), available in the environment, that better suit the requirements of the application, the user and the environment. Once selected, those interaction elements are transparently and remotely connected to the application logic, so that it does not need to know anything about the interaction technologies and devices it is using.

The diagram in Figure 1 shows an overview of this process. Along with the application logic, implemented as a multi-agent system in the application layer of HI^3, the developers provide:

1. a description of the UI (AUI), using the UsiXML abstract UI model, that generically describes the user interaction requirements of the application;
2. a set of associations between AUI elements and some data and action objects that will be used as the model to manage data input/output with the user.

Those associations are managed by an application controller using the observer pattern to monitor the data objects. When the application modifies a data object, the controller redirects the change to the UI, and vice versa. The application controller is a physical component of the application agent, so that the Final Interaction Objects (FIO) agents and the application agents interchange messages directly.

As can be seen in Figure 1, the final implementation of the UI is provided by a set of mappings between the AUI elements and the FIOs. This mapping is 1..N, so that the same AUI can be associated with many FIOs. It is stored by the application controller, but it is generated, managed and updated, at runtime, by the User Interface Manager (UIM). It uses the information available in the AUI model, the user model and the environment model, to select among the different interaction resources available in the environment.

FIOs are abstractions of devices and appliances capable of input/output to the user, like switches, lamps, presence sensors, alarm systems, or even higher level interaction resources, like gesture or voice recognition software. They are a key element of Dandelion, because they implement the concrete logic required to interact with a device, and because they provide a generic view of the device as an interaction resource. Their goal is to decouple the rest of the system from the underlying interaction technologies, and this is achieved by hiding the device behavior behind the General Interaction Protocol (GIP) interface

The GIP is another key aspect of this work. It is an event based multi-agent communication protocol which provides a common interface for interaction resources. The set of events defined is inspired by the I/O actions supported by the AUI model of UsiXML. Figure 2 shows an overview of the protocol. Its operation can be summarized with a simple example. An application changes a data object that represents the output state of an alarm. The controller detects the change, and sends an output event to the associated FIO/s. They provide the output to the user depending on the concrete implementation of each FIO, for example powering on a light or playing a sound.

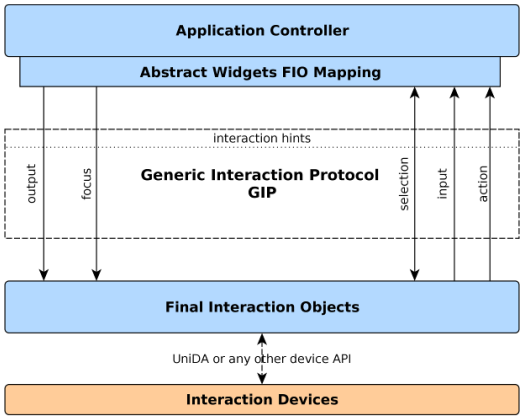

Figure 2. Overview of the GIP for abstracting devices as interaction resources

FIOs are implemented as distributed agents in the sensing/actuation layer of HI^3. They implement the GIP interface, but it is not mandatory for a FIO to support all the GIP events, only a subset of them. There can be devices that support only user input, only user output, etc. The supported events are specified in the FIO description, which is used by the AUI during the FIO selection process.

GIP events can have attached a set of variables that are used by developers to provide interaction hits to the FIOs. One example could be an urgent property, or a color property, so that a FIO can adapt (if possible) its response to more specific requirements of an application. Those interaction hints are specified by applications in the association between AUI elements and the I/O data model.

A FIO can encapsulate any kind of interaction resource: a home automation device, a hand-gesture recognition engine or even a GUI. Developers of FIOs are only required to export the interaction capabilities of the resource as GIP events. Thus, for example, a home automatic switch will only provide an action event, while a GUI form for entering data will provide input/output events with a string property for each data field, and maybe an action event for a button.

Even if GUI or other interaction technologies are supported, we are especially interested in supporting user interaction through everyday objects. Therefore, in order to facilitate the development of FIOs that implement the concrete logic required to interact with those kinds of devices. We have developed UniDA [15] as the hardware abstraction layer of HI^3, but also as a solution for hardware access within Dandelion. While FIOs provide a common interface for any kind of interaction resource, UniDA provides a common interface to any kind of hardware device. Every kind of device is accessed using the same paradigm and concepts, and each class of devices is reduced to a set of common operations, so it is possible to use entire classes of devices from different manufacturers or technologies using the same exact API.

CONCLUSION

The application of model-driven engineering techniques combined with interaction resource selection algorithms seems a very promising approach to alleviate the problems of developing user interfaces that require the integration and utilization of many different technologies and devices.

The current implementation of Dandelion includes: a device abstraction technology to decouple applications from the hardware technology of their interaction devices; a component migration system to physically move HI^3 components from one platform to another and a distributed UI system, allowing applications to operate distributed and decoupled from their interaction resources.

The next step in this work is the development of FIO selection algorithms to adapt the application to changes in the environment by changing the FIOs mapping at runtime.

REFERENCES

1. Dadlani, P, Peregrin Emparanza, J, & Markopoulos, P. Distributed User Interfaces in Ambient Intelligent Environments: A Tale of Three Studies. *Proc. 1st DUI*, University of Castilla-La Mancha (2011), 101-104.
2. Blumendorf, M., & Albayrak, S. Towards a Framework for the Development of Adaptive Multimodal User Interfaces for Ambient Assisted Living Environments. *Proc. 5th UAHCI*, Springer (2009), 150–159.
3. Abascal, J., & Castro, I. F. de. Adaptive interfaces for supportive ambient intelligence environments. *Proc. 11th ICCHP*, Springer (2009), 30–37.
4. Paz-Lopez, A., Varela, G., Becerra, J.A., Vazquez-Rodriguez, S., & Duro, R. J. Towards ubiquity in ambient intelligence: User-guided component mobility in the HI3 architecture. Science of Computer Programming, 11/2012, Elsevier (2012).
5. Collignon, B., Vanderdonckt, J., & Calvary, G. Model-Driven Engineering of Multi-target Plastic User Interfaces. *Proc. 4th ICAS*, IEEE (2008), 7–14.
6. Ranganathan, a., Chetan, S., & Campbell, R. Mobile polymorphic applications in ubiquitous computing environments. *Proc. 1st MOBIQ.*, 2004, 402–411.
7. Satoh, I. Mobile applications in ubiquitous computing environments. IEICE TRANS. COMMUN. VOL.E88-B, NO. 3, 2005, 1026–1033.
8. Aizpurua, A., Cearreta, I., & Gamecho, B. Extending in-home user and context models to provide ubiquitous adaptive support outside the home. User Modeling and Adaptation for Daily Routines, Springer (2013), 25–59.
9. Miñón, R., & Abascal, J. Supportive adaptive user interfaces inside and outside the home. Advances in User Modeling, Springer (2012), 320–334.
10. Thevenin, D., & Coutaz, J. Plasticity of user interfaces: Framework and research agenda. *Proc. INTERACT'99*, IOS Press (1999), 110-117.
11. Blumendorf, M., Lehmann, G., & Albayrak, S. Bridging models and systems at runtime to build adaptive user interfaces. *Proc. 2nd EICS 2010*, ACM (2010), 9-18.
12. Balme, L., Demeure, A., Barralon, N., Coutaz, J. & Calvary, G. Cameleon-rt: A software ar-chitecture reference model for distributed, migratable, and plastic user interfaces. *Proc. EUSAI 2004*, Springer-Verlang (2004), 291-302.
13. W3C Multimodal Interaction Framework. W3C (2011).
14. Varela, G., et al. Decoupled Distributed User Interfaces in the HI3 Ambient Intelligence Platform. *Proc. 6th UCAmI 2012*, Springer (2012), 161-164.
15. Limbourg, Q., Vanderdonckt, J. UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence.Engineering Advanced Web Applications, Rinton Press, Paramus, 2004, pp. 325-338.
16. Varela, G, Paz-Lopez A, Becerra, J. A., Vazquez-Rodriguez, S., & Duro, R. J. UniDA: Uniform Device Access Framework for Human Interaction Environments. Sensors 11 (10), MDPI (2011), 9361–9392