A design-and-play approach to accessible user interface development in Ambient Intelligence environments

Sokratis Kartakis a, Constantine Stephanidis a,b,*

a Foundation for Research and Technology – Hellas (FORTH), Institute of Computer Science, N. Plastira 100, Vassilika Vouton, GR-70013 Heraklion, Crete, Greece
b Department of Computer Science, University of Crete, Greece

1. Introduction

The increased importance of user interface development in the context of the evolution of the Information Society has been widely recognized in recent years in the light of the profound impact that interactive technologies are progressively acquiring on everybody’s life and activities, and of the difficulty in developing usable and attractive interactive products and services. As the Information Society further develops, the issue of efficiently developing high quality user interfaces becomes even more prominent in the context of the next anticipated technological generation, that of Ambient Intelligence (AmI) environments.

AmI as a research and development field is rapidly gaining wide attention worldwide, and the notion of AmI is becoming a de facto key dimension of the evolving Information Society, since many of the new generation industrial digital products and services are shifted towards an overall intelligent computing environment. AmI will have profound consequences on the type, content and functionality of the emerging products and services, as well as on the way people will interact with them, bringing about multiple new requirements for the development of the Information Society (e.g., [3,2]).

One of the main intrinsic benefits of AmI is identified in the emergence of new forms of interactive applications supporting everyday activities in a variety of environments and application domains. A key application of AmI is smart home environments supporting home automation [5].

Although the cost and availability of AmI technologies will be important factors for their wider spread and market success, the ultimate criterion for their public acceptance and use will be the extent to which the complex environment will be able to communicate with its users, and developers will be facilitated...

* Corresponding author at: Foundation for Research and Technology – Hellas (FORTH), Institute of Computer Science, N. Plastira 100, Vassilika Vouton, GR-70013 Heraklion, Crete, Greece.
E-mail address: cs@ics.forth.gr (C. Stephanidis).

ARTICLE INFO

Article history:
Available online 13 January 2010

Keywords:
Ambient Intelligence (AmI)
Smart home
User interface development
User interface generation
Accessibility

ABSTRACT

User interface development in Ambient Intelligence (AmI) environments is anticipated to be a particularly complex and programming intensive endeavor. Additionally, AmI environments should ensure accessibility and usability of interactive technologies by users with different characteristics and requirements in a mainstream fashion. Therefore, appropriate user interface development methods and tools are required, capable of both reducing development efforts and 'injecting' accessibility issues into AmI applications from the early design stages. This paper introduces two tools, named AmIDesigner and AmIPlayer, which have been specifically developed to address the above challenges through automatic generation of accessible Graphical User Interfaces in AmI environments. The combination of these two tools offers a simple and rapid design-and-play approach, and the running user interfaces produced integrate non-visual feedback and a scanning mechanism to support accessibility. AmIDesigner and AmIPlayer have been evaluated to assess their usability by designers, and have been put to practice in the redevelopment of a light control application in a smart environment as a case study demonstrating the viability of the design-and-play approach. The results confirm the usefulness and usability of the tools themselves. Overall, the proposed approach has the potential to contribute significantly to the development, up-take and user acceptance of AmI technologies in the home environment.

© 2009 Elsevier B.V. All rights reserved.
in providing appropriate interaction means in an effective and efficient way.

Therefore, the issue of supporting user interface development becomes critical. In this respect, the following main requirements can be identified:

- User interface development in AmI will be a particularly complex endeavor, due to the complexity of the environment itself. Therefore, tool support becomes crucial. A graphical design environment combined with automatic generation of AmI applications can provide an appropriate solution for facilitating rapid prototyping and development of Graphical User Interfaces (GUIs).

- AmI is perceived as a technological development that should ensure accessibility and usability of interactive technologies by users with different profile characteristics and requirements in a mainstream fashion [3,10]. This is due to the fact that, in such an in fieri technological environment, it will not be possible to introduce ad hoc solutions that are implemented once the main building components of the new environment are in place. Therefore, appropriate user interface development methods and tools will be required, capable of ‘injecting’ accessibility into AmI applications from the early design stages and throughout the entire user experience lifecycle.

This paper introduces two tools, named AmIDesigner and AmIPlayer, which have been specifically developed to address the above challenges through automatic generation of accessible GUIs in AmI environments, and have been applied and tested in the context of smart home applications. AmIDesigner is as a graphical design environment, whereas AmIPlayer is a support tool for GUI generation. The combination of these two tools is intended to significantly reduce the complexity of developing GUIs in AmI environments through a design-and-play approach, while at the same time offering built-in accessibility of the generated user interfaces, by integrating non-visual feedback and a scanning mechanism. The tools can be easily extended with additional interaction styles and services, as well as interaction platforms. AmIDesigner and AmIPlayer have been evaluated to assess their usability by designers, and have been put to practice in the redevelopment of a light control application in a smart environment as a case study demonstrating the viability of the design-and-play approach. The results of the conducted evaluation and of the case study confirm the usefulness and usability of the tools themselves. An additional benefit is that the developed tools, besides from the fact that they provide built-in accessibility, support very rapid prototyping-assessment-redesign cycles to efficiently address usability problems.

This paper is structured as follows. Section 2 discusses smart home environments and their requirements regarding user interaction and user interface development. Section 3 overviews current approaches to automated user interface generation. Section 4 provides a rationale for the design-and-play approach, outlines the underlying process and its implication in a smart home environment. Section 5 presents the main characteristics of the user interfaces produced by the developed tools, focusing on built-in accessibility. Section 6 presents the design of AmIDesigner, and Section 7 outlines the main features of AmIPlayer. Section 8 reports on the usability assessment of AmIDesigner, as well as the conducted case study.

2. Smart home AmI applications

Currently, large companies worldwide are investing in research and development of smart home environments in anticipation of market expansion. For example, Philips has a facility in The Netherlands called HomeLab, which is specifically designed as a development and test site for in-home AmI technology, and where technologies like Easy Access are emerging. Easy Access recognizes a hook line from somebody humming, automatically compares it with a song database and plays the song on the room’s stereo equipment. The underlying goal is to create a “perceptive home environment”, which is able to identify and locate humans in it, as well as to determine their intentions, with the ultimate aim to provide them with the best service possible.

Similar developments are taking place in the academic and research community. The Georgia Institute of Technology has developed a 5040-square-foot “Aware home” used as a laboratory for AmI development. Various products have been tested, providing valuable insight into the scope of applications that are available. Such an initiative is the Gesture Pendant, which exploits simple hand gestures to control multiple devices, thus helping reduce the number of remote controls necessary in most homes today and minimizing technological clutter. Other related initiatives are the PlaceLab of MIT and the inHaus Innovation Center of Fraunhofer-Gesellschaft.

A common objective of many current projects is to enable, in the near future, the control of a large number of standard electrical and electronic devices residing in the home environment through computing technologies. In [8], an experiment is described with the main target to analyze different kinds of ambient assistance in living room environments. The participants had to interact with a networked and partly autonomously acting home entertainment system, which presented integrated functions like TV, radio, audio and video playing, telephone, and light controls. Based on the obtained results, seven kinds of assistance were identified, ranging from situations where the user is fully informed about all details of the environment changes to situations where the environment changes without any direct access of the user.

The vision of Ambient Intelligence promises to deliver more natural ways for controlling devices (e.g., by using hand gestures and speech) [1] or even offer proactive, non-user-initiated, controls (e.g., a “smart” room knows when and how to change the state of each device in it) [4]. However, due to the intrinsic characteristics of the new technological environment, it is likely that interaction will pose different perceptual and cognitive demands on humans compared to currently available technology [6], and the challenge emerges to identify and avoid forms of interaction, which may lead to negative consequences such as confusion, cognitive overload, frustration, etc. This is particularly important given the pervasive impact of the new environment on all types of everyday activities and on the way of living.

In [19], an experiment is described, that targeted the matching of user interface intelligence in smart home environments with the type of task to be carried out and the age of the user. Following [17], tasks are classified as skill-based tasks, rule-based tasks and knowledge-based tasks. In skill-based tasks, performance is governed by patterns of preprogrammed behaviors in human memory, without conscious analysis and deliberation. In rule-based tasks, performance is governed by conditional rules. Finally, at the highest level of knowledge-based tasks, performance is governed by a thorough analysis of the situation and a systematic comparison of alternative means for action. Interfaces are also

---

distinguished according to three levels of intelligence (low, medium and high). The results of the study, involving two user groups of young and older people respectively, show that the level of intelligence of the user interface significantly affects performance for different types of tasks. In particular, the study arrives at the conclusion that, when completing skill-based tasks, users have better performance (less time and fewer errors) on low-intelligence-level interface, whereas when completing rule-based tasks, users perform better on high intelligence user interfaces. However, due to the decline of the cognitive abilities, the performance of older users does not differentiate as clearly as that of young people while using different user interfaces of smart home, especially for high cognitive demanded tasks.

Therefore, the need arises for highly usable GUIs allowing direct and accessible user control, avoiding the need to use a variety of different applications to control different devices, and providing an intuitive overview of the overall current state of the smart environment. The development of appropriate user interfaces also needs to be facilitated through the availability of suitable tools to reduce development efforts and guarantee built-in accessibility.

3. User interface development tools

User interface development is a complex task that is widely recognized as requiring appropriate tool support. In current development practice for graphical environments, tool support is mainly based on:

- the provision of graphical environments for user interface prototyping supporting a WYSIWYG ("What You See Is What You Get") design paradigm;
- the provision of markup languages for the description of GUIs; and
- the provision of mechanisms for the automatic generation of user interfaces, intended to fill the so called design-implementation gap, thus considerably reducing development time.

A schematic classification of existing tools is provided in Fig. 1. Available GUI WYSIWYG editors offer graphical editing facilities that allow designers to perform rapid prototyping visually. Such editors may be standalone or embedded in integrated environments, and may differ with respect to: (i) the output of the GUI design process: this may be a GUI description in some markup language or GUI code that needs to be integrated in the overall application by programming; and (ii) the facilities made available to designers.

Various user interface markup languages are available, whose main role is to help keeping the GUI separated from the logic of an application, but also provide a common language between GUI design and automatic GUI generation. Based on the markup description produced by GUI editors, GUI generators can produce a running user interface automatically, thus saving much programming efforts.

Alternative more research-oriented efforts towards support for user interface development focus on model-based tools. A variety of user interface design tools and environments are based on modeling relevant user interface related knowledge (e.g., [18]). Model-based design environments vary according to many factors, including overall objectives, number and types of models, available design support and design outcomes [15]. Almost all design environments include graphical editors for task hierarchies and other models. Design environments also differ with respect to the number and type of design activities that are automated. In some tools, the entire user interface is automatically produced [16]. The emergence of a variety of interaction platforms has led to various model-based approaches for supporting the design of multiplatform user interfaces (e.g., [14]). These approaches usually address issues related to handling the constraints posed by different platforms and screen sizes, selecting and generating appropriate presentation structures, and managing contexts of use. Some model-based tools for user interface development in AmI have also been proposed (e.g., [12]). Overall, however, design automation from models appears to be a difficult task, due to the large dimensions of the design space, and the difficulty in automatically

Fig. 1. Tools for Graphical User Interface prototyping and generation.
selecting appropriate designs. The impact of many of the above tools on design practices has been limited so far [18,15]. Many tools have been criticized for lack of usability due to their complexity. Additionally, automatic generators are usually bound to generate specific or limited types of interfaces, and do not take into account accessibility requirements for the generated user interfaces.

Therefore, although automatic user interface generators are often criticized for the quality of the code they produce, the combination of a GUI WYSIWYG Editor with a markup description and an automatic GUI generator currently appears to be a more promising approach towards rapid and efficient development of user interfaces, more familiar to designers and in-line with current practice. However, no tool currently available offers the possibility of creating GUIs for AmI services provided through various devices in a smart home environment, as no tool is appropriately equipped for communicating with such an environment. Additionally, another very important limitation of available tools is that they do not produce accessible user interfaces, and, therefore, they are not suitable for addressing the mainstream accessibility requirement posed by AmI environments, and smart home applications in particular. Finally, many of the described tools are bound to specific development platforms or interaction object toolkits. In AmI, however, it is particularly important to ensure that new interaction metaphors, appropriate to the new types of applications that the AmI environment integrates, can be elaborated, evaluated and used. Therefore, the extensibility of the basic toolkit is particularly important when designing for AmI.

AmIDesigner and AmIPlayer aim to address the above gaps through a design-and-play approach inspired by current tools for prototyping and automatic generation of GUIs.

4. Design-and-play user interface development in smart home environments

AmIDesigner and AmIPlayer have been specifically developed to achieve the following development support objectives:

- easy and quick development of user interfaces, along the lines of the GUI generators discussed in the previous section;
- automatic integration of the developed interfaces in smart home environments without the need of further programming; and
- extensibility of the tools themselves with additional interaction styles and additional services, as well as, potentially, additional interaction platforms.
AmIPlayer uses the export XML description produced by AmIDesigner to generate GUIs ready for run time. These GUIs can send and receive signals from the middleware and call services, thus allowing control of the room devices. In the current implementation, AmIDesigner is limited to the MS Windows platform due to the usage of the .NET platform. Nevertheless, AmIPlayer can potentially run on various operating systems with minor changes. Future extensions of the tools will address multiplatform development.

In summary, a very simple design-and-play process is carried out for building applications with AmIDesigner and AmIPlayer, which is illustrated in Fig. 3.

The design of the GUI is performed using AmIDesigner, which exports the GUI description to an XML file. Then, AmIPlayer uses the XML file in order to generate an accessible GUI that controls the devices of the AmI environment, and the user can directly interact with it.

5. Generated user interfaces

The user interfaces generated by AmIDesigner and run by AmIPlayer are rich Flash GUIs that can include animations. Since interface widgets can be freely created by designers, alternative interaction metaphors beyond the traditional desktop and more appropriate for the home environment can easily be designed and implemented, and attractive graphical effects can be included. However, the GUIs generated by the proposed tools also integrate accessibility features, which make them usable for a large population of target end users, including hand-motor and visual impaired users, children, elders, and inexperienced users. Accessibility is integrated in a multimodal fashion, i.e., alternative rendering of widgets and alternative interaction techniques are supported. In particular, such interfaces embed directly from design:

- speech and audio feedback associated to interface widgets;
- hierarchical scanning of interface elements; and
- compatibility with various input/output devices (e.g., touch screen, remote control, binary switches, etc.).

Speech and audio feedback, in combination with keyboard input or input through other appropriate devices, addresses the needs of blind users in the context of smart room devices control. Additionally, non-visual feedback (in association with visual feedback) can also be useful for sighted users, especially when they are busy with other tasks. In conventional GUIs, non-visual feedback is usually limited to few sounds and does not include speech. Non-visual rendering of GUIs is usually achieved through screen readers, i.e., assistive technologies for blind users that ‘read aloud’ user interface items to the user. However, besides necessitating extra software that the user needs to possess and install, the use of screen readers is problematic with Flash GUIs. These problems can be completely overcome in the proposed approach through built-in non-visual feedback.

Scanning is an interaction technique that enables motor-impaired users to have access to interactive applications and services [13]. The basic idea is that a special “marker” (e.g., a colored frame) indicates the interaction item that has the input focus. The user can shift the focus marker to the next/previous interaction object using any kind of switches (e.g., keyboard keys, special switch hardware, remote control). When the focus is over

![Fig. 4. AmIDesigner Graphical User Interface.](image-url)
an object the user wants to interact with, another switch is used to
perform “selection”. Additionally, in cases where the user can use
just a single switch, focus movement can be automatically
generated by the system at constant time intervals. This variation
of the technique is referred to as “automatic scanning”, while any
other case is generically called “manual scanning” (even if hands
are not actually used at all). Scanning is difficult to add to a user
interface after development, as the scanning mechanism needs
information on the user interface structure implementation.
AmIDesigner embeds a scanning mechanism directly in the
produced user interfaces, and the designer only has to set the
order in which interface widgets need to be scanned. As a
particularly innovative feature, scanning can also be associated
with non-visual feedback in a multimodal fashion, thus making
switch-based interaction accessible to blind users.

6. AmIDesigner

AmIDesigner is targeted not only to experienced, but also to
beginner designers and developers of smart home applications.
The user interface of the tool has been designed according to the
following requirements:

- Organization of GUI design based on the home environment
  rooms, so that the designed application is able to control specific
  rooms.
- Minimum possible number of steps to complete each design sub-
  task (usually two steps) through edit-in-place facilities.
- Main editing area inspired by Microsoft Visual Studio .Net,7 as
  the latter is the design environment currently most familiar to
  many designers.

Fig. 4 depicts the main areas of the GUI of AmIDesigner. These
are: the main menu and toolbar, the widgets toolbox, the stage, the
widget properties editor, the signals editor and the scanning order
editor.

Accordingly, the steps that need to be carried out, so that a user
interface is produced, are:

(1) Creating a project, selecting the smart home room which will
be controlled by the application, and setting the dimensions of
the GUI.

(2) Selecting the widgets to be included in the interface.
(3) Setting the widget properties, including layout issues, dialogue
flow and calls to services.
(4) Associating user interface elements and Middleware signals.
(5) Establishing a scanning order for the widgets in the user
interface.
(6) Generating an XML description of the designed GUI, viewing it
through AmIPlayer and testing it.

This process is described in more details in the following
subsections.

6.1. Widgets

The widget toolbox displays the widgets that can be used to
design a GUI. Three different presentations of the widgets are
offered within the toolbox: large icons, list and small icons (see
Fig. 5).

To insert a widget in the GUI being created, the corresponding
item is dragged from the toolbox and dropped on the stage. The
only actions that can be performed within the stage are the
creation of new items and moving and resizing existing items.

6.2. Widget properties

The widget properties editor on the right side of the
AmIDesigner interface displays the following information: (a) the
properties of the currently selected item on the stage; (b) a list of
all the items within the stage with the currently selected item
marked; and (c) a list of all available services and functions, as well
as the arguments of the currently selected function.

Using this editor, the designer can set the properties of the
currently selected item. Properties vary according to each item
category, are detected dynamically, and include name, size,
dimension, etc. An important property is the voice text associated
to an item, which is used at run time to “speak aloud” the interface,
so as to provide non-visual interaction (for example, for blind
users, or for users busy with other tasks in the environment).

As previously mentioned, action widgets can be used to define
the dialogue flow in the designed GUI. More specifically, action
widgets can be used to alter the visibility state of container
widgets. This is achieved by binding an action widget to a container
widget, implying that the action widget will change the visibility
status of the widget it binds.

Fig. 5. AmIDesigner Widget toolbox.

---

Additionally, widgets can be grouped to produce complex artifacts. This is achieved by attaching widgets to containers. When an item is attached to a container, by moving the container on the stage, the attached item is also moved.

An action widget can also be bound to a function of a service. On the bottom-right part of the interface, a list containing the services and functions of the selected room is displayed. A function can be bound to an action widget in a way similar to binding interface items (see Fig. 6).

6.3. Signals

The smart environment should be able to interact with the generated interface to inform it about changes occurring in the environment, thus affecting the status of the GUI. To this purpose, two types of signals are implemented: Middleware Signals and GUI Signals. Middleware signals are transmitted from the middleware with specific arguments to produce GUI Signals. Middleware signals are automatically detected by AmIDesigner. When GUI signals are produced, they modify properties of items within the current stage. The items that are affected by each signal, as well as the new properties resulting from changes in the environment, are presented at the bottom of the AmIDesigner interface. In the signals editor, the designer can select Middleware signals and associate them to GUI signals, so that, when a middleware signal is received with specific arguments, the respective GUI signals are produced in the interface (see Fig. 7).

6.4. Scanning

A scanning mechanism is built-in in the user interfaces produced by AmIDesigner. The designer has no other task than setting the order according to which elements in the interface will be scanned at run time. This is achieved by defining (and modifying) a scanning hierarchy in the scanning order editor in the left bottom part of the interface (see Fig. 8). Only container widgets are allowed to have children in the scanning hierarchy.

7. AmIPlayer

AmIPlayer has no specific interface. It is an application that parses the XML file generated by AmIDesigner and communicates...
with the Middleware so as to activate devices and receive signals from them. AmIPlayer consists of two parts: a C# application, which communicates directly to the Middleware, and a Flash stage, which is responsible for the GUI generation. Robust communication between the C# application and the flash stage is essential, and is achieved through calls of Middleware functions with specific arguments.

The steps that are carried out, so that a GUI is generated and the user is able to use the devices, are the following (see Fig. 9). Firstly, the configuration file of AmIPlayer is modified to define the XML file that the AmIPlayer will use. The next step is to start the AmIPlayer application. This is a simple flash object that is embedded in the C# application. Finally, the GUI is generated and the user can interact with it and control the corresponding smart room.

8. Evaluation and validation

8.1. Evaluation of AmIDesigner

AmIDesigner targets a particularly expert user population, namely user interface designers. The evaluation process of the tool has, therefore, followed a hybrid approach involving a mixture of user satisfaction and expert user interface evaluation in order to obtain detailed designers’ comments and suggestions on the interface design of AmIDesigner.

The evaluation was conducted in an in-house AmI laboratory space where, among other technologies, several computer-controlled lights have been installed. Seven expert users were involved, five males and two females. All users had at least a University degree in Computer Science or a related subject. All of them had at least a few years experience in the field of Human–Computer Interaction and in the design of accessible user interfaces, but no previous experience in the design of AmI applications.

The following evaluation procedure was adopted. The group of users was briefly introduced to the functions and goals of the room, as well as to the objectives of the evaluation experiment, i.e., the evaluation of user satisfaction and the collection of experts’ comments on the system, and was provided with the related material. After that, AmIDesigner was demonstrated to each user.

All participants were then asked to perform the following series of design tasks for a light control application:

- Task 1: Design a Simple GUI.
- Task 2: Change Widgets’ Properties.
- Task 3: Bind Items and Services and attach Widgets.
- Task 4: Signals’ Manipulation.
- Task 5: Create a Scanning Order.

Upon completion of the design tasks, the participants exported the XML description and “played around” with their own user interface (changing the lighting in the room). During the experiment, the participants were requested to freely express their opinion about the system and mention any problems they faced. The designers were then requested to fill-in the IBM user satisfaction questionnaire, as well as a brief evaluation report including their expert evaluation comments on AmIDesigner and their suggestions for further improvement.

8.2. Evaluation results

The results of the user satisfaction questionnaire are reported in Table 1 (ASQ), Table 2 (CSUQ) and Fig. 10. The mean user satisfaction, presented in the diagram of Fig. 10, was 87.61, that is considered very high.
the design-and-play approach. Most of them also provided useful comments towards improving the user interface of the tool. Those comments are briefly summarized below:

- The visibility of the binding process for services and items can be improved. The same holds for the process of attaching widgets to each other.
- The design stage can be enriched with additional functionality. Apart from drag and drop, widgets could be inserted in the stage by simply clicking on their image in the widget toolbox.
- The property grids could be improved by offering tools for easier selection of graphical images, colors and other features.

Overall, this evaluation experiment, although small-scale, provided a clear indication of the usefulness and usability of the developed tools, and most importantly, of the potential role of the design-and-play approach in AmI environments.

### 8.3. A case study

In order to test them in practice, AmiDesigner and AmiPlayer have been employed in the redevelopment of an accessible light control application for smart home environments, named CAMILE. The original development and evaluation of CAMILE is reported in [7]. Target users are home inhabitants of any age, including people with visual or motor disabilities. In order to accommodate such a wide range of users, CAMILE provides four different modes of interaction: (a) touch screen-based for sighted users with no motor impairments; (b) remote controlled operation in combination with speech feedback for visually impaired users or tele-operation by sighted users; (c) scanning (both manual and automatic) for...
motor-impaired users using binary switches; and (d) speech enabled scanning for visually impaired users. The GUI of CAMILE is illustrated in Fig. 11.

For the redevelopment of CAMILE, the following steps were followed. AmIDesigner was used to recreate the user interface by means of simple widgets such as labels, panels and image buttons. The light services were then bound to specific action widgets, followed by the definition of the widgets’ scanning order. Finally, GUI signals were created to influence the GUI depending on the incoming Middleware Signals. After completing the redesign and producing the XML description, AmIPlayer was ran on the touch screen PC, and the application that controls the lights in the AmI laboratory was ready for use.

The original development of CAMILE required three weeks of extensive development, as well as specialized programming knowledge and experience. On the other hand, the redevelopment through AmIDesigner by an expert user interface designer took approximately two working days, without any programming, while obtaining the same functionality, user interface and communication with the AmI environment. Thus, the use of AmIDesigner can be claimed to greatly simply development and reduce the required time and experience resources. Additionally, AmIDesigner made the development easy because the produced results are visible at design time, and changes in the user interface need only a few “clicks”. This offers the opportunity of very fast prototyping cycles for effective usability evaluation, where adjustments to the user interface can be made directly during experiments when needed. The above provide strong evidence that the developed tools, AmIDesigner and AmIPlayer, have significant potential to bring positive impact on the development, adoption and acceptance of AmI technologies.

9. Conclusions

This paper has introduced a design-and-play approach to the development of accessible GUIs in AmI environments, based on automatic GUI generation. Such development process is supported through two tools, AmIDesigner and AmIPlayer.

AmIDesigner is a graphical editor that creates descriptions of a designed application in XML. It includes a variety of widgets which are detected dynamically. Developers can program their own widgets. AmIDesigner communicates with the Middleware of the AmI environment, is continuously informed about the available services of a room and detects Middleware signals. Therefore, it supports the definition of how the application manipulates the devices in the room and how the external environment interacts with the GUI.

AmIPlayer, on the other hand, uses the export XML description of AmIDesigner to generate running GUIs. The produced rich GUIs can be manipulated through several input devices, embed built-in non-visual feedback and scanning, and allow end users to directly control smart room environment services.

The tools have been undergone evaluation and obtained very positive usability scores. Additionally, they have been employed in a case study concerning the development of a light control application in a laboratory environment. This effort has demonstrated in practice the value of the tools towards simplifying and considerably shortening the user interface development process.

Overall, it is claimed that the proposed approach and its supporting tools achieve the following objectives:

- easy and quick development of user interfaces without the need of programming;
- automatic development of the developed interfaces in smart home environments without the need of further programming;
- accessibility of the produced user interfaces through a multimodal combination of interaction devices and techniques without reducing the use of rich graphics;
- support for efficient usability assessment lifecycle through the design-and-play approach; and
- potential for further development through extensibility.

The above can provide a significant contribution to the development and up-take of AmI technologies in the home environment. In particular, offering the possibility to decouple user interface provision from extensive programming, the reported work can facilitate the emergence of new professional roles targeted to effectively and efficiently set-up smart environments for their diverse inhabitants, including people of all ages, abilities and backgrounds, and test them in place.

Foreseen extensions of AmIPlayer include enhancement of the editing facilities, multilingual support, selection of input devices and association of devices to specific task, and integration of recognition techniques (e.g., speech and gesture). Foreseen extensions of AmIDesigner mainly concern multiplatform support, thus making the generated interfaces available through PDAs, mobile phones and other interactive devices in the AmI environments.

References

Sokratis Kartakis holds a B.Sc. and an M.Sc. in Computer Science from the Department of Computer Science of the University of Crete, Greece. From July 2003 to September 2006 Mr. Kartakis worked at the Human–Computer Interaction Laboratory of the Institute of Computer Science of the Foundation for Research and Technology – Hellas (ICS-FORTH) as an assistant with an undergraduate scholarship. From October 2006 to September 2008, Mr. Kartakis worked at the Human–Computer Interaction Laboratory as an assistant with a postgraduate scholarship.

Constantine Stephanidis, Professor at the Department of Computer Science of the University of Crete, is the Director of the Institute of Computer Science, and member of the Board of Directors of the Foundation for Research and Technology – Hellas. He is the Head of the Human–Computer Interaction Laboratory, and of the Centre for Universal Access and Assistive Technologies, and Head of the Ambient Intelligence Programme of ICS-FORTH. He is also the President of the Board of Directors of the Science and Technology Park of Crete. He is member of the National Council for Research and Technology (Hellenic Ministry of Development), National representative (General Secretariat for Research and Technology, Hellenic Ministry of Development) in the Specific Configuration Programme Committee of FP7, and member of the Management Board of ENISA (European Network and Information Security Agency). He has been engaged as the Scientific Responsible on behalf of ICS-FORTH in more than 50 National and European Commission funded projects, and he introduced the concept and principles of Design for All in Human–Machine Interaction and for Universal Access in the evolving Information Society. He has published two Handbooks and more than 300 technical papers, and he is Editor-in-Chief of the Springer international journal “Universal Access in the Information Society”. URL: http://www.ics.forth.gr/stephanidis/index.shtml.