End User Tools for Ambient Intelligence Environments: An Overview

Mavrommati Irene[1,2] and John Darzentas[1]

[1] University of the Aegean, Product & Systems Design Engineering, Syros, Hellas
[2] Computer Technology Institute, Kazantzaki str, University Campus, 26500 Patras, Hellas
mavrommati@cti.gr, idarz@aegean.gr

Abstract. New elements that are introduced by the nature of living and interacting within an Ambient Intelligence (AmI) environment lead to new HCI paradigms. While AmI User Interfaces are moving off the desktop and the GUI paradigm, and become augmented and diffused within the ubiquitous environments, a new generation of User Interface Design Tools to facilitate the design and realization of AmI applications, is emerging. Issues and specific shifts related to Human Computer Interaction in AmI environments, which affect the design of these tools, is outlined in this paper. The high level characteristics of End User Tools that facilitate users to reason as well as manipulate the behavior of the AmI environment are outlined.

Keywords: Human Computer Interaction, Ubiquitous Computing, End User Tools, Ambient Intelligence Environments.

1 Introduction

End User tools aimed at end users within Ubicomp environments stem from the perspective that it does not seem possible in ubiquitous computing environments to cater for all the potential needs of all categories of users [15, 17, 13]; it seems much more reasonable to enable people to cater for some of these needs themselves, and empower them (via provision of appropriate tools) to the creation of ubiquitous applications that fit their own idiosyncratic needs. The development of such tools for people living in Ambient Intelligent Environments (AmI) has to face not only the technological issues that Ubiquitous computing poses, but also the challenges that occur due to the shifts in human computer interaction that owe to the very nature of interaction within AmI environments.

2 Living in AmI Environment and New HCI Paradigms

New elements that are introduced by the nature of living and interacting within an ambient intelligent environment lead to new HCI paradigms. Some of the issues that have an impact in AmI HCI research practice [12] are:

© Springer-Verlag Berlin Heidelberg 2007

1. A shift in the nature of interaction

Interaction in Ambient Intelligence Spaces can range from explicit to implicit interaction. Unintended user actions on the environment may result in unintended control and manipulation of the applications. The implicit nature of interaction (achieved in AmI by both sensing and physical action) assumes a seamless human computer relationship where there may often be no conscious interaction. On the other hand, since no system is completely error free, issues of the appropriate level of visibility and transparency to the workings of the system are raised. The possibility of facilitating a degree of transparency (providing upon request some visibility into the workings of the ubiquitous system) is an element that could be considered for End User Tools. [3, 13].

2. Organizational concerns of users

AmI systems may be used by single users, but who may also operate in larger groups [2]. There can be more than one user for an AmI application, working on it simultaneously, from the same or remote locations. The same may also hold as a requirement for Tools aimed for the creation and editing of AmI applications. It may be in the interest of more than one user to co-edit an application that involves them, synchronously or asynchronously, but collaboratively and from different ends.

3. Different interaction channels

In AmI there is a shift in the nature of input and output devices. Interaction becomes multimodal and ubiquitous: many appliances and artifacts within an environment can be used and in many different operations as well as sensing capabilities of the environment; the same applies for the nature of the output devices of the application: speech, gesture, tangible interfaces, biometrics, are a few of the elements into play. A value to this approach could be the replacement of complex command languages with actions from manipulating directly the objects, and making use of multimodal interface combinations to interact with the system. Among the issues to be addressed [2] are the following: how to identify and select a possible interaction object, how to select one action and bind it to the object while voiding unintended selection, how to handle more abstract functions, and how to embody appropriate feedback and direct it to users attention.

4. The role of intelligence

Actors coming into play can be human, or agent software. In the latter case there are risks of unexpected behavior –resulting agent intervention- that may surprise the user. Such kind of surprise must be avoided; visibility on the workings of the agent and the intelligent application and its rationale should be available upon request, as well as an overwrite function for the applications or the agents within them (the overall off switch). Appropriate feedback so that users can be aware of the systems attention has to be considered [13].

5. Visibility. Reversibility of action(s). Error tolerance

Feedback should be provided for actions upon the physical environment that involve the AmI application. Syntactic correctness of sequences of actions has to be checked at all times, in order to appropriately inform users, so as to avoid errors before they are made. To be able to undo actions (reversibility of actions) is also very important in the case of error: a requirement stemming from these is the ability to recover previous state. Visibility, reversibility, and syntactic correctness of actions, can also be dependant on issues of context awareness and compatibility of platforms. In an AmI environment many interoperating platforms may come into play, while the locus and nature of both feedback and of stored actions is distributed. Therefore error prevention, tolerance and reversibility all pose grave but interesting system design challenges [2, 3, 12].

3 End User Tools

In AmI the application and the interface tend to merge into one entity, as it is the augmented artifacts/spaces that are the access points to applications, but may also carry the application functions. As ubiquitous computing develops, prototyping tools for ubiquitous computing applications will be in demand, for developers, but also for end users. Such tools will initially be aimed to application designers so that they can participate in the development of applications that currently requires a high-level of technical expertise [10, 13]. End User Tools can also be appropriated to facilitate users to reason as well as manipulate the behavior of the AmI environment, so that they can supervise and eventually create or modify AmI applications to fit their own idiosyncratic wishes and needs. Emergent functionality in AmI can be the result of niche, but previously unforeseen implementations, created by end users themselves [4, 13]. This new generation of prototyping tools can help shape the future of ubiquitous computing and eventually accelerate the development of next generation ubiquitous applications [10, 13, 1].

What we can consider as tools that facilitate the development of AmI applications are:

- Mental models
- Ontologies
- Application/Software mechanisms

Mental Models

People are an intrinsic part of a Disappearing Computer environment, as it is their actions and behavior, as well as their needs that define the environment. The human element can be catalytic for the whole system: one of things that can create ‘emerging’ and previously unforeseen functionality is the inherent human creativity and the human capability for problem solving and expression. Nevertheless, this relies on people’s adoption of ubiquitous computing, and in turn the technology’s understandability and openness for adaptation. Mental models can be considered as End User tools, since they facilitate end users to gain an understanding of the workings of the AmI system, so that they can reason about the AmI applications. Such models need to be suitable to act both as high level technology models as well people’s conceptual models.

One such example is the adoption and appropriate adaptation of component models that allows for the recombination of functions [4, 15]. To enable the recombination of elements into new functions, the basic concepts and elements of a component model need to designed in a way that they are capable to be easily communicated to people, so that there is a degree of transparency into the –otherwise invisible- workings of a ubiquitous environment. This can be done by an appropriately designed mental model –that carries along the basic technology concepts that allow for inter-associations of artefacts. An example of a high level programming model that provides a conceptual abstraction that allows end users to describe Ubiquitous scenarios is described in [4]. In fact such model acts as a high level interface for the user within a ubiquitous computing environment; they act as a communication layer, which people can understand, and by having access to it they can manipulate the 'disappearing computers' within their environment. The creation of such models as interfaces, for the broader interaction with ubiquitous computing environments, goes hand in hand with the creation of middleware, that acts as a bridge between core technology layers (such as protocols, communication etc), devices, and people.

To support such mental models, metaphors may be used, that stem from already existing (non-ubiquitous) widely recognizable paradigms that imply interconnectivity. Examples that imply interconnectivity can be appropriate familiar terms (like the verbal term 'Plugs' used in [4] or familiar images like for example a 'Puzzle' [6]) that may be used and transferred into the realm of Ubiquitous Computing.

Ontologies
An ontology can provide a common basis for communication and collaboration between heterogeneous artefacts and AmI environments. The ontology can describe the basic conceptual terms, the semantics of these terms, and define the relationships among them. It is therefore fundamental for the creation of ubiquitous applications, and can be considered as a tool, in the broad sense of the term.

Application/Software Mechanisms
What is generally understood with the term 'Editing tools' in the most commonly used form, are application mechanisms that supports the establishment and management of applications. With a range of external devices, the 'Editors', people can supervise available sources (artifacts, services, etc) and create associations between them, thus making AmI applications. These mechanisms' core structure can be independent of particular modalities [13], so that various point-application editors can be implemented with a variety of multimodal interfaces, and in a variety of devices.

4 Resources and High Level Mechanisms

There are three basic questions that we can ask regarding the software mechanisms for End User Tools for creating AmI applications:

- What is available as resources from the environment to the Tools?
- What should the Tools do upon the environment?
- Which elements that may come into play, that can be considered for these Tools?

Available as resources from the environment to the Tools are [8]:

- Multiple sensor approach
- Ambiguity of input
- Diversity and distribution of computing platforms
- Diversity of contexts of use
- Diversity of users
- Amount of Data
- Interactions available in the environment

The handling of these resources can lead to a number of challenges, the most prominent of which is *context awareness*. Such resources can also provide the input needed for the concept of *computing ahead* the next likely editing stages, so that guidance in editing actions is provided by the software tools.

Context-aware applications are one of the most important forms of next generation interactive systems. As ubiquitous computing develops, prototyping tools for context-aware applications will be in demand. Such tools can also help produce more usable context-aware applications in an efficient way [10]. Context awareness therefore has to be used by editing tools, has to be specified or configured, so that it can be used as a necessary element of the AmI applications that are being created.

Decision making is a related aspect according to which the developer or advanced user may want to dynamically define or change the rules determining an individual artifact or even an application's behavior, in a high level manner. Dynamically defining the parameters for an application is another aspect that can be defined or altered using tools aimed at developers. As [4] reports, a tool providing a GUI for creating or changing rules, can provide the advantage that rules can be dynamically altered in a high level manner, without disturbing the operation of the rest of the system.

Assuming substantially more computing power in the application context of distributed, multimodal sensing and recognition techniques we might want to consider constantly *"computing ahead"* – modeling the user's actions and pre-computing the results along perhaps the five most likely next inputs - as Scott Hudson states in [5]. This can be *used* to provide new kinds of feedback as well as shortcuts for the user, but can also enable proactive pre-fetching from other media. According to Hudson, related to the concept of "computing ahead" is the notion that we should move from the idea of a single state of a system or object of interest, to maintaining multiple alternative states simultaneously. This will be useful both in interesting new interaction techniques which allow "what if" explorations to happen naturally [18] and as basic support for dealing with ambiguity and asynchrony.

Several challenges may arise from this proposed approach that can lead to sophisticated models of probability –which in turn could be based on machine learning approaches for adaptation to users and tasks, and models of ambiguity of inputs so that it is made easier to deal with [14].

5 Tools Interfaces and Metaphors: Some Concepts

Elements that can be considered for the interfaces of AmI editing tools are:

- Automatic interface generation
- Programming by example techniques
- High level abstractions

Several concepts can be borrowed from the report of the Future of User Interface Design Tools workshop [8] at CHI2005, which, although more generally aimed, can provide food for thought in this area.

Automatic Interface Generation
Automatic interface generation requires specific information about the user and the current situation to be incorporated into the design of the user interface. A user interface could be displayed on whatever device the user has available. Another possibility is that the interface could use familiar elements that the user has seen recently, and personalize according to the user's profile. Model-based systems attempt to formally describe the tasks, data, and users that an application will have, and then use these formal models to guide the generation of the user interface [16].

Systems can use this input to automatically design the user interface, or to design assistance to a human. When a user requests an interface to control an appliance, the user's device downloads a functional model from the appliance and uses that model to automatically generate an interface. Although there have been successful developments in limited domains (namely dialog box design and remote controls), it is noted that model-based user interface tools have not become common. [16] Nevertheless, assuming a manageable scope of foreseen interfaces for tools, we should note that those model-based techniques may hold potential for the interface instantiations of AmI tools.

Programming by Example
Programming by example techniques and intelligent agents can help users with routine complex tasks. Nevertheless they can not remedy all cases of application configuration. There are cases that there can not be a task example performed –i.e. because the application splits between different locations, different time periods or in situations that cannot be replicated (for example the application pertaining many people present, or a specific point in time, while the application creator does not need to require the availability for these in order to configure an application for them). Although programming by example provides ease of use for end user programming of AmI applications in specific cases, it can not be generalizable as an interaction form in broader aimed End User tools. Nevertheless, it can prove a useful complement to the tools functions.

High Level Abstractions
In prototyping an AmI application designers and users acting as designers need to explore the large and ubiquitous input space and specify the contexts of use. A design tool can facilitate this task by providing high level abstractions. In Topiary [11] a map abstraction was used to represent spatial relations of entities and thus allowed designers to capture location contexts of interest by demonstrating scenarios. In e-Gadgets on the other hand, a high level conceptual model was used to explain the workings of the system to the users and enable them to make connections.

Metaphors
Metaphors are a commonly used way to facilitate the use of a tools based on existing paradigms that are familiar to the user experience. Various forms of connectible

‘puzzles’ are metaphors often used in editors user interfaces. A fridge magnet metaphor is used in [9] while a browser approach is used in the Speakeasy system [15] -where components are connected using a visual editor based on file-system browsers. [6] uses a “jigsaw puzzle” metaphor in the Graphical User Interface (GUI). Sensors as well as devices are represented by puzzle piece-shaped icons that the user “snaps” together to build an application. Nevertheless in all the above cases the interactions are simplified to sequential execution of actions and reactions depending on local properties (e.g. sensor events), which limits the potential to express many of the user’s ideas [3]. As Drossos et al state in [3] the non existence of emergent properties, and the absence of rule based logic, can result in very simple application behavior.

A high-level mechanism to abstract context, that also allows the rapid construction of ambient computing applications, is presented by [7]; this is complemented by a clear conceptual model for AmI. A well-defined vocabulary is used in this model that tries to map the physical and virtual world to elementary component objects that can be interconnected in order to create AmI applications. The architecture seems to limit the real world’s representation to sets of sensors; that in turn restricts the model’s scope, as well as the autonomy of the components [19].

The e-Gadgets UbiComp Application Editor is another example of a prototype that realizes a conceptual model. The experiences reported after expert and user trials suggest that an architectural approach where users can compose predefined components seem to be worthwhile [13], however, it is pointed out that further improvement is necessary in the design and interaction of editing tools.

6 Conclusions

Editing tools are required to manage the ubiquity and understand the logic of the AmI environments. Several of these tools are addressed more to the developer or the advanced user rather to an everyday end-user. The purpose of these tools is to configure certain aspects of the system’s behavior, and implement certain applications within AmI environments.

Such tools need to allow for utilizing context awareness, but also provide adaptive interfaces to cater for a variety of user profiles. The consideration of modalities (augmented reality, gesture, or speech interfaces for example) poses a number of challenges on the design of editing tools. Robustness and adaptation to changes -different environments and infrastructure-, being able to dynamically define the parameters of an application, providing many perspectives for accessing services, are but a few of the challenges posed.

Further challenges involve developing techniques for providing support for debugging and testing the applications build, providing feed forward, as well as feedback, allow for transparency into the workings of the system, and design and develop the system and the tools so as to prevent errors, minimize them, tolerate them, and recover from them -within the ubiquitous computing infrastructure.

Tools aimed at end users need to be researched in their own merit, as they constituting a very important part of the Ambient Intellience vision. The research community needs to work on defining a roadmap towards appropriate, efficient and effective editing tools for creating of AmI applications.

Acknowledgments. We would like to thank the project ASTRA (FET IST-29266), and the project extrover-Gadgets (FET IST-2000-25240), for providing ground towards the development of end user tools for pervasive environments.

References

1. ASTRA project website: http://www.astra-project.net/
2. Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A., Lopes, C.: Making Sense of Sensing Systems: Five Questions for Designers and Researchers. In: Proceedings of the Conference on Human Factors and Computing Systems (2002) pp. 415–422 (2002)
3. Dey Anind, K.: End-User Programming: Empowering Individuals to Take Control of their Environments. In: Proceedings of the CHI 2005, workshop on the Future of User Interface Design Tools (2005)
4. Drossos, N., Goumopoulos, C., Kameas, A.: A Conceptual Model and the Supporting Middleware for Composing Ubiquitous Computing Applications. Journal of Ubiquitous Computing and Intelligence (JUCI), special issue on Ubiquitous Intelligence in Real Worlds, vol. 1, pp. 1–13, American Scientific Publishers (2007)
5. Hudson, S.: Leveraging 1,000 and 10,000-Fold Increases: Considering the Implications of Moore’s law on Future UI Tools Research. In: Proceedings of the CHI 2005, workshop on the Future of User Interface Design Tools (2005)
6. Humble, J., et al.: Playing with the Bits, User-Configuration of Ubiquitous Domestic Environments. In: UbiComp 2003, pp. 256–263. Springer, Heidelberg (2003)
7. Jacquet, C., Bourda, Y., Bellik, Y.: An Architecture for Ambient Computing. In: The IEE International Workshop on Intelligent Environments (June 2005)
8. Olsen, D., Klemmer, S.: In: proceedings of the CHI 2005, workshop on the Future of User Interface Design Tools (2005)
9. Truong, K.N., Huang, E.M., Abowd, G.D.: CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In: Proceedings of Ubicomp, pp. 143–160 (2004)
10. Yang, Li.,Landay, J.A.: Rapid prototyping tools for context aware applications. In: Proceedings of the CHI 2005, workshop on the Future of User Interface Design Tools (2005)
11. Li, Y., Hong, J.I., Landay, J.A., Topiary: A Tool for Prototyping Location-Enhanced Applications. UIST 2004, CHI Letters 6(2), 217–226 (2004)
12. Mavrommati, I., Darzentas, J.: An overview of AmI from a User Centered Design perspective. In: IET Proceedings of IE06, Athens (2006)
13. Mavrommati, I., Kameas, A., Markopoulos, P.: An editing tool that manages device associations in an in-home environment. In: Personal and Ubiquitous Computing. ACM. 8(3-4), pp. 255–263. Springer, Heidelberg (2004)
14. Mankoff, J., Hudson, S., Abowd, G.: Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 368–375 (2000)

15. Newman, M., Sedivy, J., Neuwirth, C.M., Edwards, W.K., Hong, J.I., Izadi, S., Marcelo, K., Smith, T.: Designing for serendipity. In: Serious Reflection on Designing Interactive Systems (ACM SIGCHI DIS2002), June 25-28, 2002 London, England. NY, pp. 147–156. ACM, New York (2002)
16. Nichols, J., Faulring, A.: Automatic Interface Generation and Future User Interface Tools. In: Proceedings of the CHI 2005, workshop on the Future of User Interface Design Tools (2005)
17. Rodden, T., Benford, S.: The evolution of buildings and implications for the design of ubiquitous domestic environments. In: Proceedings of the CHI 2003 conference on human factors in computing, Florida, USA, April 5-10, 2003 (2003)
18. Terry, M., Mynatt, E., Nakakoji, K., Yamamoto, Y.: Variation in element and action: supporting simultaneous development of alternative solutions. In: Proceedings of the 2004 conference on Human factors in computing systems, pp. 711–718 (2004)
19. Drossos, N., Mavrommati, I., Kameas, A.: Towards ubiquitous computing applications composed from functionally autonomous hybrid artifact. In: Roddick, J.F., Hornsby, K. (eds.) To appear in the Disappearing Computer book, Springer, Heidelberg (2007)