End User Development in AmI: a user centered design overview of issues and concepts

Irene Mavrommati, John Darzentas
University of the Aegean
Product and Systems Design Engineering, Syros, Hellas
Computer Technology Institute
Kazantzaki str, University Campus, 26500 Patras, Hellas
mavrommati AT cti.gr, idarz AT aegean.gr

Abstract  A new generation of User Interface Design Tools is emerging, that facilitate users to reason about and to manipulate the behavior of an Ambient Intelligence Environment. The development of such tools has to face not only the various technological issues that ubiquitous computing poses, but also the challenges that occur due to the shifts in Human Computer Interaction within the inherently distributed AmI environments. HCI challenges that affect the design of such end user tools as well as the high level characteristics of them are discussed. We aim to provide AmI researchers with a comprehensive overview of issues that relate to end user development in AmI environments, thus promoting the user orientation of such systems.

1 Introduction

In an Ambient Intelligence Environment (AmI) the application and the interface tend to merge into one entity, as the augmented artifacts are the access points to applications but also carry the application functions. As ubiquitous computing develops, prototyping tools for ubiquitous computing applications will be in demand both for developers and end users. Such tools can 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 [19], [23].

End User tools aimed at inhabitants of AmI 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 [27], [32], [23]; 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 [10]. End User tools can also act as an observation window to the AmI system, helping users to reason about the workings...
of an AmI environment. Development of such tools 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 AmI and End Users

The core of the Ami vision is that information processing will be everywhere, in objects of various sizes, from keys to cars, to buildings. Computing will be invisibly integrated into everyday life and will be supporting people in their activities, as diverse as these may be. The main components of this vision are [33]:

- Reliable robust hardware with long lasting power supplies (possibly self managing or energy harvesting), or distributed systems of redundant low cost devices.
- Wireless and wired communications between computers, with collaborating different networks and an emphasis on add-hoc network.
- Intuitive interfaces easily used by everybody (i.e. multimodal interfaces, various sensors, biometrics)
- Embedded “intelligence”, capable for reasoning about people unobtrusively, so as to provide them with services when needed, assist in controlling interfaces, and from the system’s perspective automatically managing communications and maintenance (i.e. self-repairing).

This vision implies less direct and less conscious user input than the current systems. As Crutzen states [5], “Physical invisibility or perceptual invisibility mean that one cannot sense the AmI devices anymore; one cannot sense their presence nor sense their full (inter-)action, but only that part of interaction output that was intended to change the environment of the individual user”. A popular approach to cope with the perceptual invisibility is that the technology should reveal the system in order to motivate users to relate the capabilities of the technology to their actual needs, dreams and wishes [5]. In other words ”domestic technologies should be remarkable rather than unremarkable” [31].

There are two approaches regarding the visibility of AMI systems to end users. They can be seen not necessarily as opposite, but rather as complementary:

- People should not care about what’s going on inside the system. Interaction within AmI should be seamless. This is generally based on the assumption that AmI systems will be robust enough and error free, and that intelligent agents will make appropriate judgments
reacting to various situations based on the appropriate data they gather from the AmI environment.

- People should be given a degree of transparency into the workings of the system. Transparency could be varied, according to the user and the context of use [21]. A recombinant constructivist approach is often proposed, aimed at end users. These recombinant models are promoted with the claim that emerging niche applications can be achieved as a result of people’s inherent creativity and understanding of the application’s building blocks. From being able to see and handle applications comes the building of trust and adoption of AmI systems. Users feel safer when they can be in control of the system, even if they may not need to exercise this control. One can reason about and self-assist regarding system failures, servicing, or safeguarding privacy.

The human factor is a crucial element in the construction of an ambient intelligence world and needs to be taken into account early in the research and development process of AmI systems. The success of ambient intelligence will depend on how individuals perceive AmI environments, how secure the AmI world is made, how their individual rights (including privacy) are protected. If people get to trust the system and the intelligent decisions made at the background they are more likely to adopt AmI and appropriate it via suitable interfaces. Tools for End Users providing transparency and reasoning into the workings of AmI could provide one of the means towards that goal.

3 End User Development Tools for AmI

3.1 Related work

A number of existing infrastructures - such as Jini, UPnP, and others - address the configuration of Ubiquitous environments and applications. Nevertheless they are addressed to the developer rather than the ordinary person living in a ubiquitous environment. Development Environments for experts include the iQL programming model [3], a non-procedural language for specifying the behaviour of components in pervasive environments, and Papier-Mache [17] which provides tools for programming tangible user interfaces. Since 2002 there is a growing number of efforts towards the creation of tools facilitating End User Development of Ubiquitous applications (among others it is worth noting the efforts reported in [27], [32], [23], [12], [10]). The example of CollaborationBus [12] uses the concepts of a Pipeline and collaboration sharing; Memodules and Accord projects [26], [32] propose a visual editor based on the puzzle metaphor. Many of these approaches [23], [32], [27] create an intermediate component model allowing for the recombination of AmI elements,
but also interfacing to the user with appropriate constructs. The Phidgets toolkit [11] facilitates the development of physical user interfaces by providing a range of sensor and actuator elements. iCAP [36] allows users to rapidly prototype Ubiquitous Computing environments. ACAPpella [8] supports context aware programming by example. An interface for creating sensor-based environments and configuring tables are provided by eBlocks [4]. e-Gadgets [23] provides an editing tool for creating device associations in a home environment. The jigsaw [32] is an editor for getting control over the technological home environment through assembling pieces of a jigsaw puzzle. Some systems that are based on mobile devices to control configurations are developed for PDAs and TabletPCs [14] while other approaches [10], [23] try to be device independent by separating the interface layer from the function mechanisms.

3.2 Mental models, ontologies, software mechanisms

We propose to consider as tools that facilitate the development of AmI applications the following:

- Mental models

- Ontologies

- Application/Software mechanisms

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 as people’s conceptual models. One such example is the adoption and appropriate adaptation of component models that allows for the recombination of functions [10], [27]. To enable the recombination of elements into new functions, the basic concepts and elements of a component model need to be 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. An example of a high level programming model that provides a conceptual abstraction that allows end users to describe Ubiquitous scenarios is described in [10]. In fact such a model acts as a high level interface for the user within a ubiquitous computing environment. It acts 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 goes together with the creation of middleware, that acts as a bridge between core technology layers, devices, and people. To support such mental models, metaphors that stem from already existing (non-ubiquitous) widely recognizable paradigms that imply interconnectivity are often used. Interconnectivity can be implied by appropriate familiar
End User Development in AmI: a user centered design overview of issues and concepts

terms (like the verbal term ’Plugs’ used in [10] or familiar images like for example a ’Puzzle’ [32], [14], [26]).

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.

Finally, what is generally understood with the term ’Editing tools’ are software application mechanisms that support 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 [23], so that various point-application editors can be implemented with a variety of multimodal interfaces, and in a variety of devices. Some concepts that can be used in such software mechanisms are discussed in the next section.

4 Mechanisms and resources

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?
- Ambiguity of input
- Diversity and distribution of computing platforms

Available as resources from the environment to the Tools are [30]

- 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 [19]. 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 [10] reports, a tool providing a graphical interface 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 [13]. 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 [33] 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 [25].

Collaborative sharing of applications can provide the supporting means for community participation, for adopting, encouraging and supporting the creation of AmI applications. In the CollaborationBus example [12], Three types of sharing of compositions are possible: event sharing allows users to either share events of their own sensors or processed event data of their filters. Actuator sharing allows users to share the control of a personal actuator with other users, so that other users can send commands and control the system behavior. Pipeline sharing allows sharing of complete compositions with others. Efforts towards collaborative devel-
opment and sharing of applications are also seen in the ASTRA example [1]. Clearly, not all people will be inclined or able to create or configure an AmI application. Enthusiasts or motivated individuals should take the lead to address niche needs as they emerge and as they are recognized bottom up. We can expect more diversity and business opportunities to rise in the long term by such a shift in perspective (including for example increased usage of network services for network operators).

5 Concepts for Tools Interfaces

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

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

Several concepts can be borrowed from the report of the Future of User Interface Design Tools workshop [30] at CHI2005, which, although more generally aimed, can provide food for thought in this area.

5.1 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 [29]. 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. [29] 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.
5.2 Programming by example

Programming by example techniques and intelligent agents can help users with routine complex tasks. Programming by demonstration approach requires an initial time for extended observation of relevant sensor values. In the latter learning phase, the users specify relevant sensor events, so that appropriate artificial intelligence algorithms can detect patterns in the observed sensor values and automatically execute desired actuators [7]. Programming by demonstration tools hide most specific details of the underlying mechanisms from the users but the one hand this reduces the barrier for nontechnical users to configure Ubiquitous Computing environments. Nevertheless programming by example techniques cannot remedy all cases of application configuration. There are cases that there cannot 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 cannot be generalized as a stand-alone interaction form in End User tools; nevertheless, it can prove a useful complement to the tools functions.

5.3 High level abstractions

In prototyping an AmI application designers and users acting as designers need to explore the ubiquitous input space and specify the contexts of its use. A design tool can facilitate this task by providing high level abstractions. In Topiary [20] 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. In this approach more complex artefact behaviour can emerge from interactions among more elementary artefacts. This approach can scale both 'upwards' (towards the assembly of more complex objects, i.e. from objects to rooms, up to buildings, cities and so on) and 'downwards' (towards the decomposition of given gadgets into smaller parts, i.e. towards the concept of 'smart dust') [23].

A high-level mechanism to abstract context, that also allows the rapid construction of ambient computing applications, is presented by [16]; 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. Nevertheless the supporting 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 [10].

5.4 Metaphors

Metaphors are a commonly used way to facilitate the use of tools based on existing paradigms that are familiar to the user experience. Various forms of connectable ‘puzzles’ are metaphors often used in editors user interfaces. A fridge magnet metaphor is used in [38] while a browser approach is used in the Speakeasy system [27] -where components are connected using a visual editor based on file-system browsers. The metaphor of specifying the applications’ behavior by assembling pieces of a jigsaw puzzle is often used, [32][14][26] since it is intuitive at first, providing a recognizable metaphor for connecting services. Sensors as well as devices are represented by puzzle piece-shaped icons that the user “snaps” together to build an application. Nevertheless this metaphor seems to restrict the potential for development and richness of programming expression [32][10]. The interactions are simplified to sequential execution of actions and reactions which limits the potential to express many of the user’s ideas while the non existence of emergent properties, and the absence of rule based logic, can result in very simple application behavior. Other techniques are needed to give users control over their application than what the metaphor of jigsaw pieces allows. The Pipeline metaphor is used by CollaborationBus [12], (allowing for nested and parallel pipelines allowing logical (AND, OR, NOT) conditions); in the pipeline metaphor sensors are the sources of any event in the pipeline, filters represent single conditions, and actuators (software or hardware) are at the end side of the pipeline.

6 Issues Introduced in Human Computer Interaction

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 impact AmI HCI research practice [22] but also End User Development are:

6.1 A shift in the nature of interaction

In AmI people do not act on the world, but act with the world. Thinking, (according to Distributed Cognition theory [18]), is not just within the head, but in the external relationships with things in the world and with other people. People are in constant dialog with the physical environment; they use this information stored in artefacts and their physical location to trigger and guide their actions [9]. In this context incidental interaction
that happens within an AmI environment is of considerable importance for the system design and may impact on End User Development.

Interaction in Ambient Intelligence Spaces can range from explicit to implicit interaction. The user may be unaware where the interaction is taking place (i.e. via gestures, sensors, movement detectors, secret cameras) [9]. 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. [7],[23]. Schmidt argues for an AmI interaction model in which users can always choose between implicit and explicit interfacing: ‘The human actor should know ... why the system has reacted as it reacted’ [34].

We are witnessing a radical shift in the field of HCI: the basic models of interaction that have proved universal across technologies are questionable for AmI [9], [2], [35]. The main changes from traditional HCI, as described in [9] are:

– The focus becomes that of activities, rather than tasks.
– An emphasis is given to the design of continuously available interaction
– There are no starting or ending points for interaction in AmI environments
– Interruptions and multiple actions can be in operation, that are loosely connected (no longer in terms of achieving certain goals).
– Multiple perspectives are in operation, that imply the reuse of information for different functions (i.e. associative models of information are needed).

6.2 Organizational concerns of users

In an AmI environment there is more than one inhabitants and therefore more than one user per application. The input in AmI can be distributed, while the user may not always be aware that their action is in fact an interaction within the AmI environment. 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 [1] and from different location than the application environment.

6.3 Different interaction channels

In AmI the direct engagement is with the world itself. [9] In the real world the input is physical, yet in the AmI world physical action gets converted to digital information too, and has consequences as an action of direct manipulation. Here the physical world becomes the interface of the AmI system, but is more than interface, as it cannot be separated (mentally or physically) from the physical environment.

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 affecting end user development [2] are:

- 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,
- how to embody appropriate feedback and direct it to users attention.

6.4 The role of intelligence

Actors coming into play in an AmI environment can be human, or agent software. Proactive agent behavior holds the promise of seamless interaction within the AmI environment. That, on the other hand, holds 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. Intelligent agents can also be used in programming by demonstration techniques, thus facilitating a set of actions for end user development within the environment.
6.5 Change in CHI models

A good case for study is Norman’s [28] model of interaction that is widely used in HCI: the so called execution-evaluation cycle. The execution-evaluation cycle splits the interaction into a sequence of sub-actions, each of them being a result of a specific user intention. Initially the user is forming the goal, and then forming a sequence of intentions, followed by specific actions, the user proceeds in executing these actions, and perceiving the state of the system after these actions are performed (change state, communicated to the user via appropriate feedback). The user then interprets the new state of the system and finally evaluates the outcome comparing it to his/her initial goal (to what extent the goal is achieved).

Bellotti [2] attempts to re-think Norman’s ’7 stage’ interaction model, focusing more in interactions that are are more appropriate for AmI environments (assuming the interactions are not GUI based). Bellotti suggests the following five interaction challenges for AmI researchers and exposes a subsequent number of design challenges that designers should address:

- **Address** - how to direct communication to a system. (Disambiguate signal to noise; Disambiguate intended target system; How not to address the system)

- **Attention** - establishing that the system is attending (Embody appropriate feedback so that users are aware of system’s attention; Then direct feedback to user’s attention)

- **Action** - defining what can be done with the system (Identify and select a possible interaction object; Identify and select one action and bind it to the object(s) and avoid unwanted selection; Handle complex operations (i.e. multiple objects and actions, more abstract functions)

- **Alignment** - monitoring systems response. (make the system state perceivable; Direct timely and appropriate feedback; Provide feedback on the response)

- **Accident** - avoiding errors and misunderstandings, or recovering from them. (Control or cancel system action that is in progress; System intervention in case of error; Disambiguate what to undo in time).

Some pitfalls that are applicable for End User Development in AmI, which are identified in the Bellotti [2] model are: unintended actions (leading to undesirable results), failure to execute an action, limited operations available, wasted input effort in a non-attentive system, inability to detect mistakes, difficulty in evaluating new state, inability to detect mistakes and to recover the previous state.
6.6 Visibility, Reversibility of Actions, 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. In CollaborationBus, feasibility checks automatically deactivate inadequate operations. 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. Some actions are upon a remote environment, and cannot be undone. Therefore error prevention, tolerance and reversibility all pose grave but interesting system design challenges [2] [5] [25].

6.7 Concerns about AmI

Research main focus is on technology development that enables AmI (i.e. sensors technology, miniaturization, processes, networking, middleware, energy provision), and not so much in the HCI issues occurring from AmI. There are insufficient existing HCI practices, theories and models regarding to HCI for AMI while User Centered Design is at its very early stages in this field of research.

AmI vision tent to present people as passive consumers accepting happily an increasing dependability on AmI systems. AmI sees people increasingly and comfortably relying on AmI for a number of activities (for reminders, surveillance, health monitoring, entertainment, home automation etc) [5]. Yet, it remains unclear how AmI systems are maintained. This vision does not seem realistic because AmI systems cannot be absolutely problem free.

It is not clear whether new technologies will provide a means to escape gracefully from being always connected [33], [21]. Moreover it is questionable how people will accept living and evolving within AmI technological environments and how this 'nurture' will impact human nature (e.g., why train one’s memory if AmI gives reminders?) [33]. Social issues relate to how people’s individual as well as social behavior will evolve by living in AmI environments, and thus developing certain expectations, assertions, and habits [24].

Ubiquitous networks will need to track and collect significant data from users’ activities. A pervasive network of interconnected devices and communications will mean that the quantity of information in circulation will increase greatly and beyond the human perceptive capability (infor-
information overload). Studies are that try to respond to understanding the broader implications of AmI (such as Digital Territories Study (DT), [6], or Safequards In The World Of Ambient Intelligence report (SWAMI [33]) are still at early stages to be taken into practice.

AmI system design has to face serious challenges regarding the system’s usage, as to how can users control the AmI system (i.e. with appropriate means, or aided by intelligent agent interfaces), how can they predict what the complex networked artefacts will do, and how the whole AmI system in turn will function appropriately and unobtrusively, providing for qualitative experience and safeguarding its user’s privacy.

7 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 (and at times collaboratively) 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.

This new generation of prototyping tools can help shape the future of ubiquitous computing and eventually accelerate the development of next generation ubiquitous applications. Tools aimed at end users need to be researched in their own merit, as they constituting a very important part of the Ambient Intelligence vision. The research community needs to work on defining a roadmap towards appropriate, efficient and effective editing tools for creating of AmI applications.
Bibliography


End User Development in AmI: a user centered design overview of issues and concepts

Springer Lecture Notes on Computer Science (LNCS3295), page 243-254


