Presented at Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'11), Nottingham 26th July 2011

Towards an Object Oriented Ambient Computing Model

Idham Ananta[a,1], Vic Callaghan[b,] Jeannette Chin[c]
[a] *Computer Science and Electronic, Universitas Gadjah Mada, Indonesia*
[b] *Computer Science and Electronic Engineering, University of Essex, UK*
[c] *Institute of Social & Economic Research, , University of Essex, UK*

Abstract. As creative creatures, people like to change and customise their environments. In the computing world, this has led to a growing demand for people to be able to customise their 'electronic spaces such personal computers and mobile phones'. In this work-in-progress paper, we argue that this reasoning can also be applied to AmI (Ambient Intelligence) Environment. However, existing computational models have significant shortcoming that act as a barrier to implementing the concept of end-user development in AmI environments. This paper presents a scenario that illustrates the need for a more functional and robust underlying computational model. We argue that OO (object-oriented) concepts could form the basis of such a system and, to these ends, present preliminary ideas for an object oriented end-user development system for building AmI applications.

Keywords. Object Oriented, End-user Development, AmI Applications

Introduction

Lieberman [1] defined end-user development as: "*a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers to create, modify, or extend a software artefact.*" He provided some supporting arguments for end-user development research, mentioning that there will be exponential growth in the number of end-user developers compared to the number of software professionals [2]. Lieberman's approach could radically change the software development model, noticeably from professional developers to the application end users. Empowering end-users and allowing domain experts to directly program or customize their digital environments would have significant advantages, especially when coping with dynamically changing environments [3].

Blackwell [4] argued that there is an urgent requirement to develop facilities to enable end-user development for more complex

[1] idham@ugm.ac.id.

longer-term needs, rather than servicing trivial tasks with short lifetimes. He also advocated psychological research to create novel programming systems and new theoretical characterizations of human problem solving. He provided a valuable user perspective on several end-user technologies such as scripting languages, visual programming, spreadsheets, and programming-by-example.

Enabling end-user development of Intelligent Environments is not an easy task. There has been some discussion related to how intelligent environment applications can be developed or be programmed. Callaghan et al [5] suggest two approaches: embedded-agent-based approaches and end-user programming based approaches. The embedded-agent-based approach utilizes artificial intelligence techniques to reduce the user's cognitive load, whilst the end-user programming based approach is directly programmed by people, which advocates of this approach argue allows more creative input and adds some transparency, engendering a sense of trust in the system. For example, Ball conducted an online study on users preferences and found almost 70% of users preferred end-user programming to agent control [6] which was consistent with a general finding of numerous studies that concludes a fundamental requirement of users is to be in control of their environment, rather than to be controlled by it. These studies are described exhaustively in Chin seminal work on end-user programming in digital homes which outlined the main arguments in favour of the end-user approach which may be summarised as being:

1. End users demand a full control over their environment
2. User wish to customize their technology and, in particular, the functionality of smart-homes (personalising homes is an age old tradition)
3. People wish to understand why home technology does what it does (ie the operation of personal technologies needs to be transparent).
4. People were worried about losing too much human control in digital homes.

Of course, as we mentioned previously, there are also arguments in favour of autonomous agents, the most powerful being as a way to manage the complexity of the technology (i.e. reduce the cognitive load on people. Thus, later, Ball suggested an alternative paradigm; adjustable autonomy, which he hoped, might offer the best of both approaches [8]. Beyond such considerations there is the issue of providing appropriate lower level infrastructural support for end-user programming paradigms; for example, how are the basic components,

and their aggregations implemented so provide the required portability, scalability and mobility required. In the remainder of this paper we will argue that OO (object-oriented) concepts could form the basis of such a system and, to these ends, present preliminary ideas for an object oriented end-user system for building AmI applications.

1. Related Work

Chin [7] introduced Pervasive Interactive Programming (PiP) as an alternative method to empower end users to customize Digital Homes. PiP is a form of end-user programming and provides a computational model that introduces the concept of a Virtual Appliance (i.e. an appliance constructed by aggregating network services), Meta-Appliance/Applications (MAps, virtual appliance data object representations), and a supporting ontology called dComp (Decomposed Community Programming). PiP used the Programming by Example (PBE) paradigm to bring programming activities to non-technical end-users.

Table 1. Comparison of end-user techniques used in AmI Research

Research	Development Framework	End-User Techniques
PiP	Rule Based, Ontology	PBD/Visual
Herranz	Rule Based/ Agent	Script/Visual
Alfred	Goals and Plan Concept	Verbal & Physical
Hague	Rule based	Cube/Visual
Humble	Programming	Jigsaw & Puzzle

Herranz et al [9] have successfully separated the environment representation from the programming system to enable the design of an Intelligent Environments in a way that makes it easy to integrate and incorporate new technologies into the Environment. They have done this by creating a rule based agent mechanism as the kernel of a ubiquitous end user, UI independent programming system.

The MIT Alfred project [10] sought to allow users to compose a program via teaching-by-example, using a 'goals' and 'plans' concept. Their system proposed to make use of a macro programming approach that could be generated by verbal or physical interaction. Truong's CAMP project [11] utilized a fridge magnet metaphor and pseudo natural language interface to realize context-aware ubiquitous applications in the home. Hague [12] proposed a tangible media metaphor to represent programming logic in which programming was

undertaken by turning appropriate faces of cube. Humble [13] proposed a jigsaw puzzle like metaphor as graphical programming representation to build applications.

Table 1 compares, differing End-user approaches that have been applied to AmI environments. From the table it is clear that no other researchers have used an object-oriented framework to support end user development and, although not shown here, neither have they used it to create an AmI support framework. Instead most researchers have focused on the programming metaphors and ignored the underlying frameworks, which we argue are critical to enabling commercial deployment of these system in a robust and large scale manner befitting the vision for future AmI environments. In this paper, we will discuss how OO might be a good candidate to solve this challenge.

2. Motivations for Bringing End-User Programming into Intelligent Environments

Cypher [14] presented several examples to motivate end-user programming on the web. We believe that some of them are also relevant to AmI Environment, which we now describe:

1. More options and personalization. In a private domain, such as a home, apartment, or car, personalization will add a more colourful experience to ritual activities or daily routines. For example, a homeowner could be given more options for creating customized domestic appliances or, for example, creating a personalised care environment tailored to various disabilities.
2. Triggering automatic response. It is easy to imagine users creating a simple application that sends texts to a user when their security alarm rings but it may be more interesting to get alerts when beverages or items in their refrigerator run low.
3. Information Gathering. End-users could programme reports about the state and usage of their consumption of unhealthy food, wasteful use of energy or perhaps connecting food replenishment to a refrigerator's stock, or recipes.

Those examples are simply examples from a set of almost unlimited possibilities that could be creatively developed by end-user.

3. End-User Development Scenario

"Tony is a young executive living in his own home that is enriched with some programmable smart devices called AmbiO's. He had created a few bespoke AmbiO's a few years ago to help him around his home. One of the simpler AmbiO's is called the 'wake up' AmbiO. It uses a combination of network services to create his AmbiO's, namely his automatic window curtains, digital alarm clock, mp3 player, and hi-fi surround sound system. Tony created this AmbiO using the OOEUD, to draw the curtains in his room, and play some energetic song every weekday morning at 7am.

One day, Tony visited his friend Sarah, a young attractive girl who had created her own AmbiO's. Sarah told Tony that she also had developed a similar AmbiO to wake her up in the morning. However, Tony realized that Sarah's AmbiO was far more interesting than his, as her AmbiO played real-time news on her video displays to show the weather forecast and traffic news of the day. Also while she was still lying on her bed, her AmbiO turned on her toaster, which contains 2 slices of bread she inserted the night before, as well as her coffee machine next to it, before triggering her alarm (and if she didn't get out of bed, as a safety feature, it even turned them off). Tony was very impressed with Sarah's AmbiO. He asked her whether she was willing to share it with him. Sarah agreed and emailed it to him right away.

Back home Tony examined the AmbiO Sarah had sent him. Because it was an object, customising it was simple, as he simply created a new "wake AmbiO" that inherited Sarah's, functions, and then using an intuitive graphical interface, manipulated it to create a new "wake AmbiO". Tony noticed that he did not have a digital toaster, so he disabled that feature. When Tony was satisfied that his AmbiO worked (by running it on his object simulator), he saved it before instantiating his 'wake up' AmbiO object straight away.

The next day Tony realized that tomorrow was his cousin's birthday. Suddenly he got a brilliant idea; why not send his cousin an AmbiO as a present? He then set about modifying his 'wake-up' AmbiO. He instantiated a new version of his 'wake-up' AmbiO, inheriting the functions of his original AmbiO and then used his graphical interface to disable the news feed, and change the video stream to one that played a happy birthday video from YouTube. He also added a special 'pizza order' function, billed to his account, and mailed this new "birthday surprise' AmbiO to his cousin's email address. It allowed his cousin, after playing the video, to choose his favourite pizza menu, using touch screen services via his cousin's interactive screen. Mike, his cousin, was thrilled to get such a thoughtful birthday present from him."

The scenario illustrates the how OO concepts assist the end-user develop AmI applications. Whilst it doesn't illustrate all the advantages of OO, it introduces some such as the portability of applications and suggests some requirements to provide development environments for end user that:

1. Reduce/eliminate duplicate codes/logics
2. Maintain high degree of reusability, and use interchangeable component
3. Manage various level of access and privileges
4. Able to distribute application across different platform (portability, and heterogeneity)
5. Allow mobility of applications and devices
6. Provide a robust computational framework

4. Motivations for Bringing Object Oriented Concept on End User Development

Brad J Cox [15] said object-orientation represented a major change in how programmers would do their jobs. Most interestingly, he also speculated on encapsulating hardware as a means to create worlds populated by heterogeneous mixes of soft and hard objects. As far as we know, nobody has succeeded in realizing this vision, which is a major motivation underpinning my work. The main benefits in applying object-oriented concepts as the underlying computational model for building end-user AmI application are:

1. The power of inheritance provides end-users with reusable components, allowing them to avoid rewriting the code from scratch rather they just "extend" their class to "inherit" all of attributes and services. If they want to customize functions (methods in OO terms), they edit the customized part. In AmI there are numerous similar objects in terms of attributes and functions. Inheritance allows similarities to be described in one central place, whilst differences can be managed in local object instantiations. This is provides a tractable way to deal with managing changes to objects, especially systems comprising massive numbers of objects, as is the vision for AmI calls for. Since generic functionality is made once, but used many times, inheritance boosts productivity. Thus, in the end-user programming world, inheritance could bring efficiencies to the development process.

2. The power of encapsulation shields the end-users from the need to understand the system at a detailed code level (they simply need to know what an object does, not how it does it). Encapsulation is a mechanism to protect attributes/data in an object with procedures that shield them from improper use or invalid access. Some sophisticated procedures built by suppliers, or other end-users, could have complex logic, so the advantage to end-users is that they don't have to understand "how" procedures work in detail. Encapsulation also can be used to set a secure boundary of values an object can have. For example, an encapsulation mechanism could prevent an application from conducting unauthorised actions (e.g. missetting parameters or a malicious access). In AmI application, this is really an important aspect.

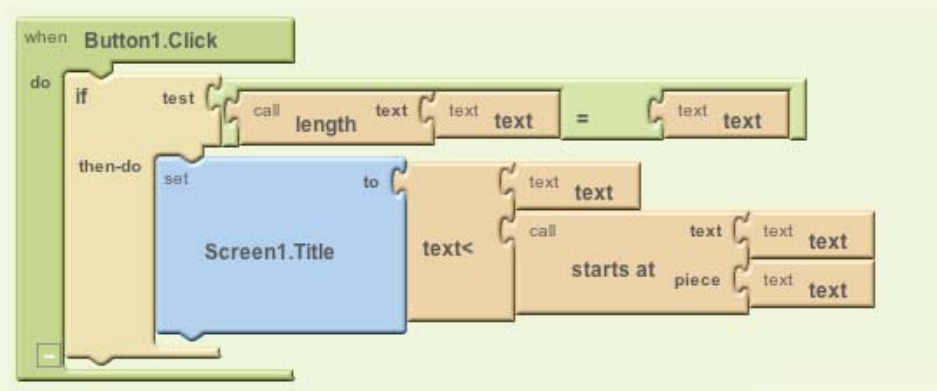

Figure 1. OpenBlock used by Google AppInventor.

The core of our argument is not about the choice of end-user programming metaphor, but that there are significant challenges faced by the underlying computational model, such as portability, mobility, heterogeneity or even inheritance, that need to be solved to make end-user programming (of any form) a commercial success. In particular, we argue that OO provides a more effective computational model to support higher-level end-user programming paradigms. We also argue that end-users would find OO concepts such as inheritance, encapsulation and polymorphism easy to understand, as this concept is derived from nature and the world we are all familiar with as Cox eloquently has argued [15]. This view is further supported with the recent appearance of products such as MIT scratch [16], Google AppInventor [17] (figure 1), and the

simplified Object Oriented development environment, Greenfoot [18] (figure 2).

5. OO Based Model For Pervasive Computing

In real the world, people interact with real objects, physically. Chin's work has demonstrated that people prefer to work with real world representations rather than abstract descriptions; in her case she used the notion of virtual appliance or MetaApps.

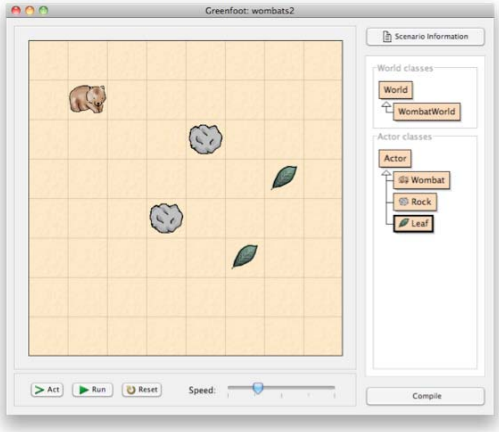

Figure 2 Greenfoot, Simplified OO Development Environment

Here we have the same view except we add to this by proposing that by adding object-oriented formalisms to Chins MetaApps, we introduce a more formal mechanism to support the wider development and maintenance needs via mechanism such as inheritance, encapsulation and polymorphism. That is why we believe that interacting with objects somehow provides both a natural and formal model (the best of both worlds). Like Chin's MetaApps, our "AmbiOs" (Ambient Objects) can also represent, not only physical abstractions of appliances inside digital homes, but also external soft entities such as information, media or higher order abstractions (e.g. a library), etc.

However, whilst Chin's work, had introduced conceptual support for application mobility, as it stood, it hadn't addressed how these concepts would be translated into a practical framework to support mobility. Thus, in practice, Chin's implementation couldn't practically accomplish the end-user development scenario above. Thus, our OO model advances

this area by providing a computational framework to enable portability and mobility of what she termed 'virtual appliances' and what we term 'AmbiOs' (Ambient Objects). Also, although Chin's work on MetaApps (Meta-Appliance/Applications) provided a way of aggregating abstract services (e.g. information, deconstructed software etc), apart from a MP3 player, she never pursued this line of research, which will be a main thread of activity in our follow-on work.

Therefore, the research described in this paper takes the best of PiP's work by Chin (that supports end-user programming in AmI environments), and marries them to the best of OO computational model concepts to come up with a novel solution that we tentatively call, OOPc (Object Oriented Pervasive Computing).

Figure 3 shows OOPc model for building AmI applications in our iSpace research facility that functions as follows. It has resources that connect to the digital home network and are managed by middleware (UPnP in our Essex iSpace). The Object Palette Space discovers resources (network services) via the middleware. These resources are structured as embodied objects. For example, the embedded-internet devices (Tini boards) run an AmbiOs virtual machine (Avm) that acts as a standardised interface to the device, emulating encapsulation and allowing any inherited customisations from parent objects to be subsumed. Soft data objects such as media or text files can be managed through a process that is equivalent to the Avm or through more conventional mechanisms such as cast them as objects with functionalities that support inheritance and encapsulation.

Object oriented software mechanisms are more developed and so how we handle this in hardware, or in hardware-software hybrids is an area we intend to research. These objects (hardware and software) are presented to End-User Development Interface. The interface works with the OOPc management that provides object discovery, repository, and management. From an end-users viewpoint, objects, take the form of visual representations using blocks and animations, or entities. These can be managed by a variety of end-user interaction modules (see top row of figure 3) ranging, for example, from PiP, Jigsaw to voice command. They might also be combined with interaction modes such as gesture or VR to support more sophisticated end-user experiences.

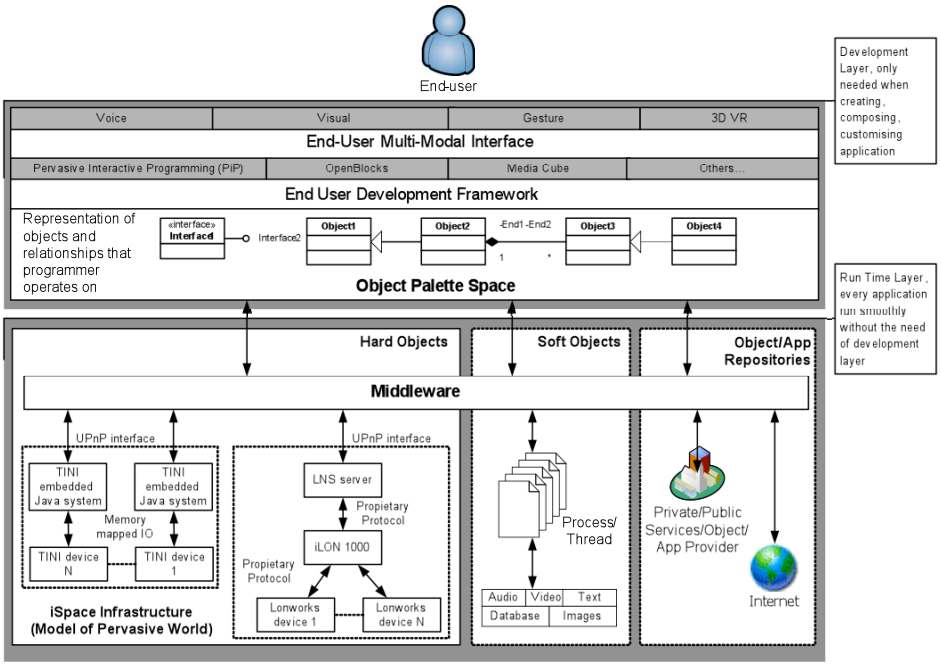

Figure 3 Pervasive End User OO World Model based on iSpace Infrastructure

Figure 4 The iSpace (our testbed)

For out pervasive world test-bed we will use the iSpace (Figure 4). Features of our out OOPc model may be summarized as follows:

1. **Hybrid Aggregation** - Applications in our model can be developed by combining multiple objects of differing types (eg hardware or software) and from differing sources, such as local repositories remote Internet repositories, or even commercial providers, etc.
2. **Inheritance** - Inheritance, make it possible for end-users to create customised objects easier by modifying the functionality of

objects that previously worked well to get more personalised applications. We envisage this working with single (atomic) or aggregated (compound) objects.

3. **Mobility** – An important feature of pervasive computing is the movement of people and devices across different spaces. Our AmbiOs (akin to Chin's virtual appliances) are compound objects made from collections of objects. Mobility presents a particularly difficult problem for the movement of sub-objects that are members of AmbiOs, in that replacements are needed to allow the overall system to continue to work, Likewise, people moving presents a similar challenge to reconstruction of AmbiOs. Our model well seek to cope with this by utilise encapsulation to provide a standardized object interface, rather than the need to deal with endless variations of hardware and software.
4. **Security** – security is a number one concerns for consumers. Encapsulation provides the basis of a perfect mechanism to build a security layer such that, for example, when an object is placed inside a home, it can be considered as "private" object where only the owner has privileges to modify it.

We believe that an object-oriented pervasive computing world will make the development of more complex end user applications possible, whilst supporting good levels of maintainability and portability.

6. Conclusion

In this work-in-progress paper, we have presented a scenario to illustrate the benefits of utilising OO for a pervasive computing computational model. For this we argued that inheritance and encapsulation ease the end-user development processes, by providing effective and robust means to support object sharing and mobility in a secure way within AmI applications.

Towards these ends, our research tries to marry the best of earlier work by Chin that provides an elegant concept for creating and programming virtual appliances with object orientation, to provide an easy to use, robust and secure way for end users to customise the functionalities of their own electronic spaces.

We believe that distributed applications, created by end-users, will change how people interact with their environment, enabling new lifestyles and business opportunities for people in the near future. Finally, our vision for a pervasive object oriented world (a heterogeneous mix of

soft and hard objects) can be likened to developing applications, in a very big computer, called '*the World*'!

Acknowledgements

This research is partly funded by Higher Education Directorate, Ministry of National Education, Indonesia.

References

[1] Lieberman, H., Paterno F., Klan M., Wulf V. End-User Development: An Emerging Paradigm, Lieberman H., Paterno F., Wulf V. Eds *End User Development*, Human-Computer Interaction Series. Volume 9, Springer, 2006, 1-8
[2] Boehm, B.W., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz, E., Modochy, R., Reifer, D. and Steece, B. (2000). *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall PTR.
[3] Costabile, M.F., Fogli, D., Fresta, G., Mussio, P. and Piccinno, A. (2002). *Computer Environments for Improving End-User Accessibility*. ERCIM Workshop "User Interfaces For All", Paris.
[4] Blackwell A.F, Psychological Issues in End-User Programming, Lieberman H., Paterno F., Wulf V. Eds *End User Development*, Human-Computer Interaction Series. Volume 9(Springer 2006), 9-30
[5] Callaghan V., Colley M., Hagras H., Chin J., Doctor F., Clark G. Programming iSpaces – A Tale of Two Paradigms, Intelligent Spaces, The Application of Pervasive ICT, Springer, London, 2006.
[6] Ball M., Callaghan V., "Perceptions of Autonomy" Intelligent Environments 2011, Nottingham, UK 25-28th July 2011.
[7] Chin, Jeannette., Pervasive Interactive Programming: Empowering End Users to Customise Digital Homes, Thesis, University of Essex, 2009
[8] Ball M., Callghan V., Gardner M., Trossen D., Exploring Adjustable Autonomy and Addressing User Concern is Intelligent Environments, Intelligent Environments 2009, Proceeding of the 5th International Conference on Intelligent Environments, IOS Press, Netherlands, 2009
[9] Herranz, M.G., Haya P., Alaman X., Towards a Ubiquitous End-User Programming System for Smart Spaces, Journal of Universal Computer Science, vol.16. no 12, 2010
[10] Gajos K., Fox, H., & Shrobe, H., "End User Empowerment in human centered pervasive computing", in Proceedings of Pervasive 2002, 1-7
[11] Truong, KN., et al "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home", Proceedings of Ubicomp 2004, 143-160
[12] Hague, R., et al: "Towards Pervasive End-User Programming". In Adjcunt Proceedings of UbiComp 2003, 169-170
[13] Humble J., et al "Playing with the Bits, User-Configuration of Ubiquitous Domestic Environments, Proceedings of UbiComp 2003, Springer Verlag, Berlin Heidelberg New York, 2003, 256-263
[14] Cypher Allen, "End User Programming in The Web", No Code Required, Giving User tools to Transform The Web, Elsevier USA, 2010
[15] Cox, Brad J., Novobilski Andrew J., "Object Oriented Programming, an Evolutionary Approach", Addison Wesley Publishing Company, 1991
[16] http://scratch.mit.edu
[17] http://appinventor.googlelabs.com
[18] http://www.greenfoot.org

