

# Supporting Collaboration in the Deployment of Ubiquitous Computing Installations

Stefan Rennick Egglestone<sup>1,A</sup>, Andy Boucher<sup>2,B</sup>, Chris Greenhalgh<sup>1,A</sup>,  
Jan Humble<sup>1,A</sup>, Andy Law<sup>2,B</sup>, Sarah Pennington<sup>2,C</sup> and Tom Rodden<sup>1,A</sup>

<sup>1</sup> Department of Computer Science, University of Nottingham, Nottingham NG8 1BB

<sup>2</sup> Goldsmiths College, New Cross, London, SE14 6NH

<sup>A</sup> {sre,cmg,jch,tar}@cs.nott.ac.uk

<sup>B</sup> {a.boucher,a.law}@gold.ac.uk

<sup>C</sup> sarahpennington@mac.com

Ubiquitous computing toolkits have been presented as part of a solution to some of the technical difficulties associated with the construction and deployment of ubiquitous computing installations. However, less has been said about their potential to support the collaborative processes that may be required during the lifecycle of a particular installation. In fact, we have found that the availability of such a toolkit has been a substantial aid in this collaborative process, and believe that this is an important benefit of the toolkit approach that warrants further discussion. We use this paper to describe three ways in which our collaboration has been aided by the presence of a toolkit, and to draw attention to the importance of considering support for collaboration when designing toolkits.

## 1 Introduction

The process of developing a ubiquitous computing installation often involves the solution of difficult technical problems, many examples of which are documented in the literature. One approach that has been adopted to facilitate this process has been the development of a number of ubiquitous computing toolkits. These promise to provide a set of standard solutions to commonly-encountered problems, and may have the potential to reduce the amount of effort required in this process. Popular examples of ubicomp-focused toolkits include ContextToolkit [2], iStuff [1] and the Equator Component Toolkit [5], and substantial numbers of publications are available that describe their architectures, implementations and the use to which they have been put.

However, there are other difficulties that relate to the design, construction, deployment, testing and evaluation of these installations that have not been discussed in as much detail. One such difficulty relates to the necessity of supporting collaboration between the set of diversely-skilled individuals who may be involved at different stages in the lifecycle of a particular installation. These might include artists, designers, educators, museum curators and ethnographers, along with individuals with technical computing skills such as system architects and developers. We have developed a belief that the production of genuinely interesting and useful ubiquitous computing systems often requires a close collaboration between such a set of individuals. How-

ever, difficulties can be present in such collaborations, and these difficulties commonly involve a lack of common concepts and language with which to discuss particular issues.

Over the last few years, we have been involved in such a collaboration, which has taken place as part of activities organized by the Equator Interdisciplinary Research Consortium. This collaboration has involved the modification and use of the Equator Component Toolkit (ECT) to deploy a number of ubicomp installations into the domestic environment. For an overview of this collaboration, the reader should consult [11]. We have found that such toolkit use has reduced the technical effort required to develop these installations. Additionally, however, we have also found that the use of this toolkit has significantly simplified the collaborative processes that were required between the authors, some of whom are from a computing background and some of whom are from a design background. We believe that this represents an important, and little-discussed advantage of the toolkit approach that warrants publication and further discussion. As such, this paper represents an informal discussion of the different ways in which we believe that ECT has assisted our collaboration, and hopefully precedes a more formal treatment of this topic in a later paper.

## **2. Overview of ECT functionality**

Although this paper does not focus on the technical details of ECT, it seems important to give a brief overview the functionality of this toolkit to aid understanding of the discussion that follows.

Firstly, ECT is constructed around the concept of a component, a self-contained item of software that is intended to be re-usable in a variety of contexts, and whose interface has been constructed using a particular interfacing technology. ECT is provided with a wide range of components, some of which have been designed to control commonly-used ubiquitous computing devices such as Phidgets [4]. Components also exist that implement software-only behaviour, with examples including queues, logic gates and email clients.

In addition, ECT provides an infrastructure that allows components to be distributed across multiple, networked computational devices, and that allows these distributed components to be organized into distributed installations. This functionality has been informed by previous Equator projects such as Ambient Wood [9] which have involved the construction of wide-scale, distributed and interactive ubiquitous computing installations.

In order to simplify interaction with this component infrastructure, ECT provides a number of graphical interfaces. Of these, the most commonly-used combination consists of the capability browser and graph editor. The capability browser, shown in figure 1 below, is used to locate and instantiate the components that are being hosted on the varied computational devices involved in a particular installation, and the graph editor, shown in figure 2, is used to configure these components and specify how they should interact.

Through the graph editor, details of the interface defined by a component are represented to a user. Component interfaces are defined by a set of named and typed

properties. Users configure and control components by using the interface to provide values to these properties, an action which may trigger the underlying component implementation to perform a particular behaviour. Component interaction is specified by the addition of dataflow linkages between these properties, with an installation being composed of a set of component instances and a set of property linkages. In the case of figure 2, an installation has been created involving three connected components. These are *Camera*, which is used to control a web-cam, *PhidgetInterfaceKit*, used to monitor a physical interfacing device (to which a physical button has been attached), and *TriggerConverter*. These components are connected together so that pressing the physical button triggers the web-cam to capture an image, with *TriggerConverter* being a necessary software-only intermediary required to make this installation work as expected.

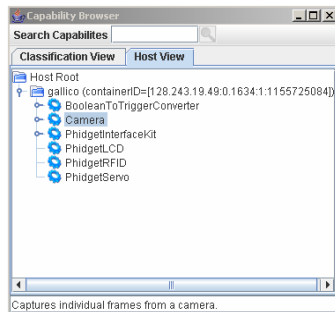


Fig. 1. Capability Browser

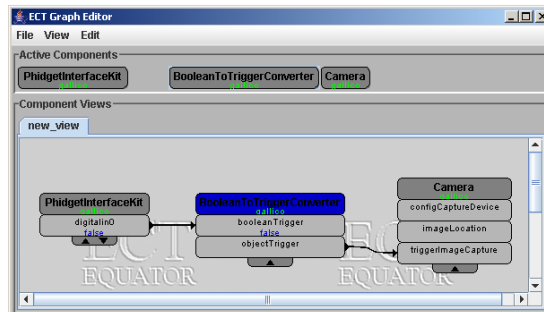


Fig. 2. Graph Editor

### 3. How has ECT supported collaboration between a diversely-skilled set of individuals?

Our collaborative work with ECT has focused primarily on installations that make use of the ECT component distribution infrastructure, and which have been constructed using the graphical interfaces introduced above. In constructing these installations, we can identify at least three types of situation in which the use of the ECT has facilitated collaboration between the authors of this paper. Each of these categories are described below, using real examples drawn from the work that we have been involved with.

#### 3.1 Supporting iterative development of components

The installations that we have been involved in constructing have all made use of a set of heterogeneous computational devices. Integrating such devices into an installation has required a close collaboration between a number of individuals with design skills, who have been mostly responsible for explorations of the potential uses of these devices, and number of software developers, who have mostly been responsible for developing reliable components used to control them.

Developing this type of component often requires an iterative process. For example, some software implementation is often necessary to allow developers to demonstrate the potential of devices to designers, who can then use this information to suggest potential uses for them. These suggestions for use might then feed back into details of the particular component interface. It is through support for this iterative process that ECT has simplified this collaboration. This is because adding a new component to ECT often only requires the addition of one file to a particular directory known to the toolkit installation. This means that iteratively-developed versions of components can be distributed to designers through mechanisms such as email, and designers can then easily install them and experiment with them by manipulating their interface using graphical tools such as the graph editor introduced above. Designers can then also iteratively feedback further requirements to developers for the next generation of such components.

A specific example of this kind of iterative design process involved the development of a component to control the display of text on an LCD screen. The first version of this component only allowed static text to be displayed on the screen, and use of this component by designers revealed the necessity of providing scrolling facilities. Later versions implemented a variety of scrolling mechanisms, which were then refined over a number of generations to suit the application that the designers were planning for these screens.

### **3.2 Supporting the transference of skills between collaborators**

Although the majority of components used in installations have been developed through the iterative process described above, a minority of components have been developed by designers who have learnt software development skills during the course of our collaboration. Pressure to learn these skills has been created by the academic environment, which requires developers to be continually involved with innovative technical work rather than being able to act as service providers for designers, and by the necessity of component development continuing even when developers are not available.

As such, features have been added to ECT to support the process of teaching designers how to develop software. Firstly, template components have been added by developers, which demonstrate simple functionality, and which can then be extended by designers as their development skills improve. These templates have been used in the development of a number of useful components by designers, including a set of software-only logic gate components intended to help in the specification of system behaviour. Designer-led development effort has also resulted in the contribution of an improved template component with an easier-to-understand software structure. Secondly, wrapper software has been added to ECT to allow scripts written in the Processing [12] language to be used directly as components. This language is popular in the art and design communities, and was learned by collaborating designers with minimal assistance from collaborating developers. These designers have used this facility to construct a number of graphical interface components, including one that renders a visualization of the lighting level in two homes connected through a particular ECT installation.

### 3.3 Supporting a sensible separation of tasks

ECT supplies a very clean separation between those facilities provided to aid component distribution and communication, and the interface that is exposed to component developers. In fact, such developers are never exposed to these underlying distribution and communication facilities, which simplifies the implementation process considerably.

However, this separation has also simplified the collaboration necessary to install experiences into individual homes, by suggesting a sensible separation of tasks. For example, individuals with technical computing skills can focus on designing and configuring networks and communications architectures to be installed into the home, or on modifying existing networking architectures, but these architectural details will be hidden from individuals who have been given the task of piecing together the component level aspects of the system from a set of pre-written components.

A clear example of this situation has been provided by a particular installation into two homes [11], both of which had existing broadband connections. This installation was designed to investigate means by which inter-home awareness could be increased, so our toolkit installation had to allow for the distribution of components across both homes. The clear separation between networking and component implementation allowed collaborating developers to focus on issues such as configuring and testing domestic routers and performing device driver installations, allowing the component-orientated aspects of our system, such as an externally-visible graphical visualization of lighting levels in both homes, to be put together by collaboration designers, who also pieced together an input device to be used as a button by hacking into a wireless keyboard.

## 4. Discussion

Section 3 above has introduced a number of situations in which our use of the Equator Component Toolkit has supported a collaboration between a set of individuals with a diverse set of skills. We neither claim that our categorizations of such situations are complete, or that the ability to support collaboration is unique to our toolkit. However, we believe that the observation that toolkits can support collaboration is a useful one to make, which might potentially help sensitize toolkit designers and users to the collaborative potential of any future toolkit software that they are involved with.

Furthermore, we would like to draw attention to the specific ways in which ECT has supported collaboration. Firstly, by providing a simple mechanism for the distribution, installation, introspection and testing of components, ECT has facilitated a process by which the details of these components and their interfaces can be *discussed*, *negotiated* and *improved*. The graphical representations of component interfaces has been particularly important in this process, as it has simplified the process by which these interfaces could be examined. In general, in aiding discussion and negotiation, ECT has acted as an artifact around which common language and conceptualizations could be *constructed* by collaborators.

Secondly, ECT has acted as a useful *co-ordination point* for particular items of work. For example, when learning to develop components, developers have distributed template components, designers have extended these components and distributed these extended versions, and developers have then been able to demonstrate how to fix bugs and improve the design of these extended components. Acting as such a co-ordination point, ECT has therefore acted as an artifact around which collaborating individuals can *negotiate* items of work. It should be noted that such co-coordinating abilities are particularly useful when a collaboration between individuals is taking place over a distance, as is the case in our ongoing collaboration.

Of course, having made these observations, it seems important to indicate the areas of work which we feel they could influence. Firstly, an obvious conclusion seems to be that toolkits designers should consider carefully how to support collaboration in future design work. However, building really useful support for collaboration into a toolkit is probably in itself a large research area, so we leave discussion of this topic to future work and future research papers. Secondly, we hope that sensitizing toolkit collaborators to the nature of the collaborative work that they are involved in will help them to refine their modes of collaboration, perhaps by facilitating discussion about procedures and approaches taken during collaborative toolkit projects.

## 5. Related work

Papers discussing the collaborative nature of work with ubiquitous computing toolkits seem to be rare, as a review of existing publications relating to both ContextToolkit and iStuff produced no publications on this topic. This may be a product of a common pattern of working, in which toolkit designers seem to have also acted as both component implementers and experience creators, which may have simplified the collaborative process to an extent to which it has become unremarkable. We have already stated in this paper, however, that we feel wider collaborations involving a more diversely-skilled group of individuals are important in moving ubiquitous computing research forward, and if this is the case, it seems likely that collaborative issues will become more remarkable as more researchers begin to work in this way.

Moving outside the ubiquitous computing domain, however, there does seem to be some limited recognition of the collaborative potential of toolkits. Dourish and Edwards [3] suggest that toolkits offer “a common conceptual framework for application development, which can aid both application designers and users”, although they do not comment on the social processes by which such frameworks develop, and their discussion focuses on scenarios in which there is little *direct* collaboration between toolkit designers, component developers and toolkit users. Other authors, such as Mackay [7], recognize that collaboration is commonly involved in end-user customization of software. This suggests that collaborative opportunities in ECT might be widened by allowing end-users to customize component interfaces directly, rather than requiring component developers to perform this task, a feature which might be usefully added to ECT in future design iterations.

Additionally, some more generic discussions of collaborative issues have been provided by the Computer-Supported Co-operative Work (CSCW) community.

Johansen [6], for example, has categorized a number of computer-mediated collaborative tasks as being either asynchronous or synchronous, and as involving collaborators who are either co-located or working at a distance. Such a categorization then creates four classes of computer-mediated collaboration, which are:

- |                             |                                |
|-----------------------------|--------------------------------|
| 1. synchronous, co-located  | 2. synchronous, at a distance  |
| 3. asynchronous, co-located | 4. asynchronous, at a distance |

We have encountered instances of toolkit-assisted collaboration that fit all of these categories other than category 3, which seems to indicate both the diverse nature of collaborations that may be required in assembling ubiquitous computing installations, and the wide range of collaborations which toolkits can assist with. The following are a number of illustrative scenarios that have been drawn from our collaboration, with one scenario for each of categories 1,2 and 4:

1. A component developer and a designer both sit at a workstation, experimenting with a component through the graph editor interface. The component developer points to changes in the structure of the interface since the last implementation of this component. The designer experiments with these changes, and makes some suggestions for how they might be improved.
2. A developer and a designer both talk over the phone about a new component. Both are running a local installation of the component, and examining it through the graph editor. Having this interface in front whilst talking to a collaborator on the telephone assists them in developing ideas about how the component might be used and improved.
4. A developer adds a new component to ECT, but is unable to contact any potential users of this component to tell them about it. Instead, he adds documentation to the component [using documentation features provided by ECT], which, from previous experience, he has pitched at a level at which he knows the components users will understand. At a later date, a user examines the component, and reading the documentation assists them in learning how to use it.

Having identified collaborative categories in which toolkits have been involved, items of literature suggesting useful design approaches for supporting each category of collaboration become relevant. Mørch and Mehandjiev [8], for example, suggest that, for collaborations in category 4, making the user interface “a point of convergence” for a number of representations of a particular application item can help users to understand such an item. To an extent, this is already the case in ECT – representations of components used in the graph editor can include both a graphical representation of the component’s interface and a set of documentation about how to use it – but these authors suggest that additional representations such as design rationale documents should also be integrated into an interface as well. Additionally, they suggest that, in order to reduce the workload on developers, the production of such documents should be integrated into the process of developing an application unit, a suggestion which could potentially inform component-development aspects of future design work with ECT.

## 6. Conclusion

The provision of generic solutions to common technological problems is a frequent motivator for toolkit designers. However, we have established in this paper that, in the ubiquitous computing domain at least, another useful (but rarely recognized) feature of toolkits is their ability to support the collaborative processes that are often required in the construction of ubiquitous computing installations. Having made this observation, two challenges become apparent – cataloguing in detail the ways in which existing toolkits support collaboration, and using this input to inform future toolkit design efforts. We hope that, by developing support for collaboration in toolkits, we can increase the efficiency of future ubiquitous computing collaborations, and help this field of research to progress.

## References

1. Ballagas, R., Ringel, M., Stone, M. and Borchers, J. “iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments” *Proceedings of the CHI 2003 conference on human factors in computing systems* Fort Lauderdale, Florida, 2003
2. Dey, A., Salber, D. and Abowd, G. “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications” *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing, Volume 16 (2-4)*, pp 97-166, 2001
3. Dourish, P. and Edwards, K. “A Tale of Two Toolkits: Relating Infrastructure and Use in Flexible CSCW Toolkits” *Computer Supported Co-operative Work, Volume 9 (1)*, p.33-51, 2000
4. Greenberg, S. and Fitchett, C. “Phidgets: easy development of physical interfaces through physical widgets” *User Interface Software & Technology, CHI Letters, Volume 3(2)*, pp 209-218, 2001
5. Greenhalgh, C., Izadi, S., Mathrick, J., Humble, J. and Taylor, I. “ECT: A Toolkit To Support Rapid Construction of UbiComp Environments” *Proceedings of the UbiSys04 workshop at UbiComp04*, Nottingham, UK, 2004
6. Johansen, R. “Groupware: Computer Support for Business Teams” 1998, New York, The Free Press
7. Mackay, W. “Patterns of sharing customizable software” *Proceedings of the CSCW90 Conference on Computer-Supported Cooperative Work*, Los Angeles, 1990
8. Mørch, A. and Mehandjiev, D. “Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units” *Computer Supported Co-operative Work, Volume 9 (1)*, p.75-100, 2000
9. Rogers, Y., Price, S., Fitzpatrick, G., Fleck, R., Harris, E., Smith, H., Randell, C., Muller, H., O'Malley, C., Stanton, D., Thompson, M., & Weal, M. (2004). “Ambient wood: designing new forms of digital augmentation for learning outdoors” *In Proceedings of 2004 conference on Interaction design and children (IDC2004): building a community*, Maryland, USA, 2004.
10. “Equator Component Toolkit project website”  
Internet: <http://www.equator.ac.uk/technology/ect> (verified 18/08/2006)
11. “The curious home”  
Internet: <http://www.goldsmiths.ac.uk/interaction/curious.html> (verified 18/08/2006)
12. “Processing project website”  
Internet: <http://www.processing.org> (verified 16/06/2006)