The Stick-e Note Architecture:

Extending the Interface Beyond the User

Jason Pascoe
Computing Laboratory
University of Kent at Canterbury
Canterbury
Kent CT2 7NF
ENGLAND
+44 01227 764000 ext. 7754
J.Pascoe@ukc.ac.uk

ABSTRACT
This paper proposes a redefinition of the human-computer interface, extending its boundaries to encompass interaction with the user's physical environment. This extension to the interface enables computers to become aware of their context of use and intelligently adapt their activities and interface to suit their current circumstances.

Context-awareness promises to greatly enhance user interfaces, but the complexity of capturing, representing and processing contextual data, presents a major obstacle to its further development. The Stick-e Note Architecture is proposed as a solution to this problem, offering a universal means of providing context-awareness through an easily understood metaphor based on the Post-It note.

Keywords
context-aware computing, stick-e note architecture, mobile computing, ubiquitous computing, situated information spaces.

INTRODUCTION

In addition to studying the communication protocols between human and computer (e.g. GUIs), there is a huge potential in improving the user interface by augmenting its scope. That is, the interface between human and computer can be extended so as to encompass the user's own physical environment (or context). This knowledge of the user's environment, supplementing the user's commands, can then aid the computer in performing or operating in a way tailored specifically to the user's particular circumstances - the computer has become context-aware. To give a very simple example, when the computer is aware of the location of both the user and the networked printers, it could automatically send a document to the printer most convenient for the user.

DEFINING CONTEXT AWARENESS

Context-aware computing devices react to their changing environment in an intelligent way so as to enhance the computing environment of the user [5]. Mobile environments such as palm-top computers or PDA's (Personal Digital Assistants) are ideal candidates for exploiting this technology due to their very varied and continually changing context of use, and it is the mobile environment that we concentrate on in this paper.

Context-aware computing, rather than treating the mobility of a device as a problem to be overcome, instead seeks to exploit the nature of mobility. Rather than just transferring and adapting desktop based applications to work on mobile platforms, context-awareness facilitates the development of a new generation of applications that can intelligently interact with the user based on a knowledge of their environment.

There are many different types of context that a computer can utilize: location, presence of objects or people, temperature, blood pressure of the user, etc. In fact any environmental factor that might influence the activities of the computer can be used, provided there is some mechanism for capturing it.

Extending the User Interface

So how can the computer's knowledge of such contexts be usefully exploited in the user interface? Research projects to date have applied context-awareness to both enhance interfaces of traditional applications and also in creating new applications whose existence is only made possible through context awareness.

Existing interfaces can be enhanced with context-aware techniques such as proximate selection [5], a method of presenting data that is ordered by proximity to the user. A more application specific adaptation of the user interface is demonstrated in the Mobisaic system [6]. Mobisaic extends the World Wide Web architecture by incorporating facilities to make use of context. However, to fully exploit context-awareness new applications and user interface metaphors are required, such as *situated information spaces* [2].

Situated Information Spaces

The concept of situated information spaces proposes the idea of attaching information to objects within the user's physical environment. In effect the user-interface is extended beyond the computer screen and into the world of the user anchoring data to physical objects which act as retrieval cues.

Information can be overlaid onto the user's real three-dimensional environment. This information can then be viewed through a computing device (e.g. a palm-top computer) that acts as an electronic 'lens', bringing the data into view when it is positioned near the physical anchor. For example, when the user's computing device is positioned near the phone an electronic phone book which is anchored to the physical phone will appear. This technique can also help in preventing 'information overload' in the user by organizing information within the user's three dimensional environment rather than presenting all information through a single focal point such as the monitor.

This has much in common with the ubiquitous computing philosophy where the prime concern is in providing computing interfaces transparently throughout the user's environment so that computer services can be provided wherever and whenever the user requires them [7]. However, instead of providing many different computing devices throughout the environment, the situated information space's concept proposes that, although the information is situated, there is only one device that is associated with a user - their personal information lens.

A driving principle of situated information spaces is to embed computer interfaces throughout the user's environment whilst retaining the conventional physical interfaces that are beneficial to the user. In addition to the supplemental information displays such as the phone book attached to the phone, these interfaces can consist of controls that can manipulate the user's environment.

THE STICK-E NOTE ARCHITECTURE

Having illustrated the usefulness of context-awareness it seems odd that, although still an embryonic field, there are not more applications and interfaces that are taking advantage of such technology. I suggest that the primary reason for this is the difficulty and lack of any supporting infrastructure in capturing and processing context data, so hampering the development of context-aware systems.

This paper proposes an architecture for context-aware computing to provide the infrastructure for developers to use in designing context-aware applications and interfaces. Such an architecture will make it possible to easily incorporate context-awareness not only in new specially designed applications, but also in the design of existing user interfaces where the overhead of designing and constructing context aware elements, although helpful to the user, had too great a development overhead in the past.

The role of such an architecture is to provide the framework that can be used by the application to adapt its user-interface or perform some action based on the current context. The very application-specific nature of some of the potential uses of context-awareness make it difficult, and perhaps undesirable, to create a common display system that all of these technologies use. It is much more relevant to focus on the underlying metaphor and infrastructure that supports the visible part of the user-interface. This is what the proposed architecture in this paper attempts to do.

Defining a Metaphor

Most context-aware applications can be thought of in terms of attaching objects to particular contexts, so that should the user enter such a context the object will be invoked. These objects could take the form of information, actions or even user-interfaces.

This notion of attaching objects to contexts naturally lends itself to comparison with Post-It notes, the little yellow paper notes that can be written on and stuck to various parts of office furniture, walls, doors, etc. so as to provide information in a convenient or attention-grabbing location. An electronic version can be envisioned where the user types a message which is stored along with the current location so that the electronic note can be displayed again if this location is re-encountered. These electronic versions of Post-It notes we call *stick-e notes* [1].

Stick-e notes lend themselves to a much broader environment of use than their manual counterparts. This is achieved through:

i. a diverse use of contexts (where I am, who I am with, the time, temperature, etc..), and

ii. a diverse range of content (any type of media such as HTML pages, or even actions that have no visible representations but which allow designers to incorporate context-aware services into their own applications).

It is in bringing together this diverse nature of context-awareness, in both context and content, that the strength of the stick-e note metaphor lies.

Representing the Real World

Fundamental to the design of the architecture is how contexts are to be represented. The importance of extensibility is paramount here as this is the area of the architecture most likely to be extended in applications that pioneer new types of context. In the stick-e note architecture contexts are modeled in a hierarchy as shown in the figure 1.

The contexts become more specific as the hierarchy descends. The use of a generic context class as a base to the hierarchy enables the employment of features of the implementation language (C++) that allow clients of context objects to treat them as a homogeneous group. Therefore clients have the option to remain abstracted from particular context implementations and are thus shielded from changes in those implementations.

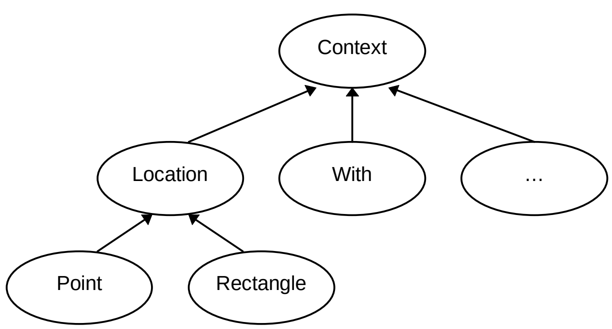

Figure 1 - Context Hierarchy

Within the stick-e note architecture, devices (i.e. objects such as GPS receivers [4] that capture context data) are treated as separate objects independent of the context classes, and are modeled in a separate hierarchy. The design of the device hierarchy is similar to that of the context hierarchy, enabling devices of the same general type (e.g. location) to be treated in a common way.

Potential clients of device classes may not actually require access to specific types of device, just a category of context. For example, an object may require the current location, the type of device used to retrieve that location is irrelevant. In order to achieve this level of generality an *Environment* class has been designed which provides access to the general context categories, such as location. This centralized resource (designed using the singleton object oriented template [3]) utilizes all the operational devices it can find to provide the most accurate data transparently to the client.

Imaginary Worlds

In addition to providing data on the current context, *pretend contexts* can enrich the stick-e note model by allowing the user to pretend to be in a particular context. This lets the user preview or review stick-e notes that will be invoked, or *triggered*, in a particular circumstance. For example, the user could go on a guided tour of Disney World without leaving England!

Defining a Stick-e Note

The stick-e note object is defined in terms of the context it is attached to (e.g. a place, person, time, etc.) and the content that it represents (e.g. information, actions, interfaces, etc.).

The context is recorded within the stick-e note to indicate the condition for which a note is invoked. This condition is called the *trigger-condition* and the process of invoking the stick-e note as *triggering*. The context in which a note is required to trigger may consist of many conditions, requiring a compound context, e.g. trigger a ‘go to the beach’ note if the location is the computing laboratory, it is a weekend and the temperature is above 25^{o}.

So far, the triggering condition of a stick-e note, whether consisting of a singular or compound context, has been defined for only one particular state of the environment. That is, there is only one set of conditions that may trigger the note. However, it will often be desirable to attach stick-e notes to multiple discrete contexts. For example, defining a stick-e note detailing an idea that triggers when I meet either Glenn, Vince, Helena, or Geraldina. The stick-e note will trigger if any (or all) of these conditions are met.

The Application Model

The stick-e note application architecture subscribes to the model-view-controller design pattern [3]. This concept separates the code for processing the display, i.e. the view, from the internal representation of what is being displayed, i.e. the model. The third entity in this concept, the controller, is an intermediary which actually drives the operation of the model.

We have defined the model of the architecture by providing classes that represent the stick-e notes and the contextual states that they are attached to. The controller element is provided through a manager class which is responsible for the continual trigger-checking process and the routing of triggered notes to displays. The architecture’s view class is defined in abstract terms of the interface it should support and the activities it should perform. This abstract nature is a result of the tremendous variation in potential display requirements, in fact some view objects may not be displays at all - just a party interested in being informed of the triggering of a stick-e note. For example, an interface could derive some of its functionality from a view object so that it can be event driven through the triggering of stick-e notes.

A Stick-e Note Tour of Disney World

To illustrate the virtues of using the stick-e note architecture, the development of a basic tourist guide application is examined. This proposed application requires the use of a mobile computer, attached to a GPS receiver, to display information on the various attractions as the visitor walks around Disney World.

Firstly, the developer creates the information displays in some graphics package. Then a set of stick-e notes are created which attach the graphics to their associated locations. This is achieved simply by specifying the file location of the graphics as the stick-e note content and then adding a rectangle object to the note specifying the location in which the stick-e note should be triggered, e.g. attaching a display about the Run Away Mine Train to its location in the complex. All the data preparation for the program is now complete.

The developer now needs to write a GUI for the application which is derived from the stick-e note architecture's view class. A function is incorporated in this GUI that responds to the notification that will be sent by the manager when a stick-e note is triggered; the developer simply uses the graphics filename supplied in the triggered stick-e note to retrieve the graphic display and present it to the user.

The basic application is now complete. However, to exploit the full potential of context-awareness the interface could be further enhanced by utilizing many other contexts such as the length of queues on rides, or even the user's heart rate!

CONCLUSION

It is desirable to extend to the concept of the human-computer interface to incorporate the user's physical environment into the interaction. The selection of proposed applications illustrates the great potential in developing new types of application and enhancing current interfaces through exploiting context-awareness.

The major obstacle to the development of such context-aware applications is the complexity of obtaining, representing and manipulating contextual data. The proposed stick-e note architecture overcomes this by providing a flexible and extensible infrastructure that hides these details behind a simple metaphor inspired by the use of Post-It notes in office environments. This architecture makes feasible the development of novel applications and interfaces that exploit context-awareness.

FUTURE WORK

A prototype implementation of the stick-e note architecture for a palmtop computer environment has been completed. Development of novel applications that use this stick-e note architecture are planned, in particular for use in the areas of archaeological and environmental sciences field work. These studies will help evaluate the features provided in the architecture and suggest further enhancements and additions that can be made.

Increased support for different types of context and their corresponding context-capturing devices will be investigated, in order to provide the computer with a more comprehensive interface to its physical environment. This will include a survey as to the types of context that can be used and how they can be best exploited. Additionally, an investigation will be carried out as to how these various contexts can be visually represented and manipulated by the user.

REFERENCES

An introduction to context-aware computing is given in [5]. The application of the stick-e notes in electronic publishing is discussed in [1]. Various applications of context-awareness are proposed in [2], [6], and [7]. [4] provides a comprehensive description of the GPS technology. Some of the object oriented design patterns used in the design originate from [3].

1. Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. *Proceedings of EP'96* (September 1996) [yet to be published].

2. Fitzmaurice. Situated Information Spaces and Spatially Aware Palmtop Computers. *Communications of the ACM*, Vol 36, No 7, 38-49.

3. Gamma, E. Helm, R. Johnson, R. Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley 1995, ISBN 0-201-63361-2.

4. Global Positioning System Overview, Available as <http://www.utexas.edu/depts/grg/gcraft/notes/gps/gps.html>

5. Schilit, B. Adams, N. Want, R. Context-Aware Computing Applications. *IEEE Workshop on Mobile Computing Systems and Applications,* (December 8-9 1994), 85-90.

6. Voelker, G. M. Bershad, B. N. Mobisaic: An Information System for a Mobile Wireless Computing Environment. *IEEE Workshop on Mobile Computing Systems and Applications,* (December 8-9 1994), 185-190.

7. Weiser, M. The Computer for the 21st Century. *Scientific American,* (September 1991), 66-75.