Java Community Process and Embedded Java User Interface

Patrick Mueller
IBM WebSphere Device Developer Runtimes Architect
pmuellr@acm.org

This paper is presented as a position paper for the OOPSLA 2003 Workshop on Pervasive Computing.

Introduction

Java is often presented as a language suited for "Pervasive Computing" and since there is an entire area of the Java Community Process (JCP) (http://www.jcp.org) focused on the "embedded" space, namely the J2ME (Java 2 Micro Edition) technology platform, it seems like a natural place to look for Java technology relating to Pervasive Computing.

This paper's focus will be on user interface, and will examine what work the JCP has done, and has not done, in the user interface area of embedded java.

What kind of UI is needed for Pervasive Computing?

NIST (http://www.nist.gov/pc2000/) defines Pervasive Computing as:

> *Shorthand for the strongly emerging trend toward:*
> *- Numerous, casually accessible, often invisible computing devices*
> *- Frequently mobile or imbedded in the environment*
> *- Connected to an increasingly ubiquitous network structure*
> *The aim is for easier computing, more available everywhere it's needed*

When I think of pervasive computing, I think of interacting with devices that I wouldn't normally think of as computers, but that could easily *have* computers embedded in them. Examples are refrigerators, car dashboards, my raincoat, and my date book. These devices already have a tried-and-true user interface associated with them, and are typically distinctly different than computer user interfaces. My refrigerator has no keyboard, my car dashboard has no mouse, my raincoat has no display, and my date book has no speakers or microphone.

To venture down the path of all the sorts of user interface that might be possible is quite a large task, so let's narrow it down to some obvious ones:

Visual: many devices will have some way of displaying information to the user. Rather than being a set of windows with menus, icons, scroll bars, and buttons, it would seem that having a display of controls much like existing hardware controls is more friendly to the user. Also, because of the coarser

tactile systems available (touch screen), visual displays that require fine control, such as scroll bars and menus, can be quite hard to use.

Tactile: Popular computer tactile interfaces include keyboards, mice, and touch screens. Of these, only the touch screen is likely to find favor in pervasive devices. Both mice and keyboards require too much physical real estate as well as typically having the user be situated in a particular position to allow comfortable inputting.

Vocal: Voice input, although having been available for years, seems to finally be taking off to some extent in the voice response market (phone menu systems). Continued work on recognition software and hardware (microphones and noise cancellation) will make voice input a more practical option for pervasive devices in the future.

What does J2ME provide?

Browsing through the list of Java Specification Requests (JSRs) available at http://www.jcp.org, we can see some number of interesting looking JSRs that might be appropriate for embedded user interface.

JSRs 37 and 118 – Mobile Information Device Profile 1.0 and 2.0

The Mobile Information Device Profile (MIDP) provides two user interface stories. One story, the low-level API, is a traditional Graphics on a Canvas story, which is very similar to doing low-level drawing in AWT or other UI toolkits. The other story, the high-level API, uses Forms, Items, and Commands, all specified in a fairly generic fashion.

The low-level API is intended primarily for games, and the high-level API is intended for applications.

The low-level API literally provides nothing but primitive APIs such as drawing lines, filling rectangles, and drawing images, and so is not itself a viable platform to develop high-function applications. Of course, it can be used as a basis to implement higher-level APIs.

The high level API provides for an interesting set of widgets, however provides limited layout, sizing, and color control. As such, it provides a highly portable but generally inelegant UI library.

The intended target for MIDP is cell phones, and is targeted to some extent for devices with a very small display, and ITU-style (phone) keypads with a limited number of hard buttons. As such, the resulting API is useful for devices basically like cell phones, and not much else.

The 2.0 version of MIDP adds additional gaming user interfaces to the low-level API.

JSRs 62 and 216 – Personal Profile 1.0 and 1.1

Personal Profile (PPro) provides much of the java.awt package functionality from the J2SE platform. While there are some 'restrictions' allowed, such as limiting the size and location of windows, aspects of the specifications are still very much desktop centered. For instance, the specification calls for supporting mouse move and drag events, even when a device may not have an input device that supports this type of action.

Due to the size of the library, PPro requires a level of hardware that is not appropriate for many pervasive devices.

The 1.1 revision of Personal Profile is currently under development.

JSR 113 – Java Speech API 2.0

This JSR is a the follow-on to the Java Speech API (JSAPI) 1.0 specification, which was published 5 years ago, in 1998. The 1.0 specification has numerous non-J2ME friendly aspects, such as requiring speech engines to synchronize with the AWT event queue, references to the Locale class, and references to security classes.

Unfortunately, there has been no public activity for this JSR since the Expert Group formed in early 2001.

JSRs 129 and 217 – Personal Basis Profile 1.0 and 1.1

Personal Basis Profile (PBP) is essentially Personal Profile, without any but the most primitive Component classes, namely Component, Container, Window, and Frame. As such, and like the low-level MIDP library, this library isn't suitable for writing applications with, although it is perhaps a suitable library to build another library on top of, with it's own user interface controls.

The 1.1 revision of Personal Basis Profile is currently under development.

JSR 209 – Advanced Graphics and User Interface

JSR 209 intends on supplying "Swing, Java 2D Graphics and Imaging, Image I/O, and Input Method Framework" to the Personal Profile and Personal Basis Profiles. Obviously this is a fairly large story, and the prerequisite of PBP or PPro rules out all but fairly high-end devices.

Gaming/Multimedia related JSRs

There are also a set of JSRs that relate to gaming, multi-media, and 2D and 3D graphics. While these JSRs certainly might be useful in pervasive applications, they are not really capable of being the sole user interface mechanism.

JSR 134 – Java Game Profile
JSR 135 – Mobile Media API

JSR 178 – Mobile Game API
JSR 184 – Mobile 3D Graphics API
JSR 226 – Scalable 2D Vector Graphics API

What does J2ME not provide?

So what has the JCP not yet considered?

GUI via markup languages

While markup languages themselves don't provide any special expressive power, the fact is that graphic artists instead of programmers will design many of the user interfaces for pervasive devices. This is because it is often desirable to emulate an existing physical design or impart a signature design in the user interface. Graphic artists are not familiar with GUI libraries, programmers are. However, many graphic artists are now familiar with markup languages due to HTML and the web. It seems a natural fit to allow a graphical user interface to be designed largely in a markup language.

Examples where this is already occurring are Mozilla's XUL (http://www.mozilla.org/xpfe/xptoolkit/xulintro.html) and HTML itself (for instance, eating the output of Java Server Pages (JSPs).

Domain-specific libraries

Pervasive user interfaces, which are analogues of physical devices, have radically different interfaces than what can be provided by typical GUI libraries. Examples are 3-state buttons which provide different visual appearance for not pressed, pressed, and just released events; sliders and progress meters implemented as circular knobs/displays; toggle buttons. While libraries implementing such abstractions are straightforward to implement, having a standard library of such widgets would prevent every application developer from having to develop their own.

Non-standard input devices

This includes 'hard buttons', rotary input, pressure sensitive devices, primitive audio and visual input devices, all manner of text input with limited devices, including predictive text input, and the entire area of haptics. In some cases, existing input methodologies can be used to map these input devices. For instance, hard buttons can be mapped to keystrokes, as can rotary input. But with some complex haptic devices, no such mapping is suitable.

Middle-tier libraries

In the J2ME world, devices are either as dumb and limited as a cell phone, or as highly functional as a desktop. J2ME specifically does not have a decent library for the current crop of PDAs, such as Palm devices, Microsoft PocketPC devices, and the Sharp Zaurus devices. These devices are more functional

that cell phones, however have neither the horsepower to handle desktop applications, nor can they support desktop semantics.

Some of the characteristics of PDAs not accounted for are: applications that consume the entire display; collapsible windows due to soft keyboard pop-ups; desktop integration; automatic screen rotation (portrait->landscape); desktop synchronization.

As PDAs become more capable, some of the issues may become moot, however, this also implies that devices such as cell phones will become as capable as PDAs today, and thus they will suffer from the lack of the middle-tier libraries.

Skin-ability

These days, everything is skinnable, from your online TV Guide to calculator faceplates. In many cases, this is done purely for fashion, but in some cases, this adds a significant usability advantage to a device. Examples include being able to change the font size to allow us old folks be able to read text on a little device, and being able to change color schemes to allow reading a display under a variety of lighting conditions. Note that skin-ability is typically not provided by a 'skin' library, but is intrinsic to a user interface library itself.

Summary

While the existing J2ME JSRs concerning user interface span an interesting range of devices, from 'dumb' cell phones up through desktop computers, there is a noticeable lack of specifications for mid-level devices.

The good news for researchers, developers and entrepreneurs is that there is plenty of low fruit to be picked.