



Creating Entertainment Applications for Cellular Phones

PAUL COULTON, OMER RASHID, REUBEN EDWARDS,
AND ROBERT THOMPSON
Informatics, Lancaster University, Lancaster, U.K.

Cellular phones offer a whole range of interesting and exciting possibilities for entertainment systems coupled with a very resource-constrained environment. In this article we consider the possibilities currently achievable through the example of a networked sports service. Applications that keep users up-to-date with sports results and playing fantasy team games, based on the results of actual events, are well established on the Internet; they attract millions of subscribers world-wide. As yet, the sports results services on cellular phones, even within countries that offer 3G services, are by and large SMS or WAP-based, and there are no dedicated fantasy team game services for cellular phones. In this article we present a novel application using GPRS that not only keeps users up-to-date, wherever they are, with the events of the English Premier Football League, but also provides the opportunity of playing a real-time fantasy football game as these events transpire. As we are seeing moves by cellular phone manufacturers to adopt standardized operating systems, we compare application development in Symbian, Brew, and J2ME. Although J2ME is not an operating system, it has, up until recently, been the only means of cross-platform application development for cellular phones. This application not only takes advantage of the “nature” of the cellular network, rather than porting existing services off the Internet, but radically improves currently available services in terms of cost and efficiency. It also highlights the fact that resource constraints on a cellular phone are not a bar to creating compelling content. This technique could be applied to a whole range of sports from Formula 1 to baseball.

Categories and Subject Descriptors: J.7 [Computer Applications]: Computers in Other Systems—*Consumer products*

General Terms: Design

Additional Key Words and Phrases: Symbian, BREW, J2ME, sports services, cellular

I. INTRODUCTION

World-wide sales of cellular phones grew in 2003 by 20.5 percent compared to 2002. Sales figures thus far indicate a likely increase of 30 per cent for 2004 compared with 2003 [Richardson 2004]. The increasing functionality and capabilities of all cellular phones are fuelling a high demand (most notably by manufactures and operators), for applications that will continue to drive the market forward.

It can be argued that for any application to succeed it must address a particular market requirement or “need,” and preferably offer something that either does not exist or improves on the existing provision. Applications are often affected by the culture in which they operate; in other words, what succeeds in one country may not succeed in another. However, one cultural aspect, which is almost universal, is the general love of

Authors’ address: Informatics, Lancaster University, Lancaster, U.K.

Permission to make digital/hard copy of part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date of appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Permission may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036, USA, fax: +1-212-869-0481, or permissions@acm.org.

© 2005 ACM 1544-3574/05/0700-ART3B \$5.00

sport. Therefore, sports-related applications are the ideal platform to discuss both their provision in cellular networks and their development for the resource-constrained devices.

In order to facilitate application development, manufacturers and operators are now producing environments where each application is no longer developed for a particular make or model of phone [Vaughan-Nichols 2003]. To this end we are now seeing the wide-scale deployment of phones using standard operating systems (OS). The two principal systems are Symbian [<http://www.symbian.co.uk>] in Europe, and the Binary Runtime Environment of Wireless (BREW) [<http://brew.qualcomm.com/brew/en/>] in the United States. Previous to this we had only been able to develop cross- platform applications using a cut-down version of Java, the so- called Java 2 Micro Edition (J2ME) [<http://java.sun.com/j2me/>]. While everyone acknowledges the potential of the cellular phone application market the debate still rages as to which environment is the most advantageous and likely to dominate the market. Many developer forums are still fixated on this debate, which has led to reluctance on the part of software developers, currently using Java, to embrace either Symbian or BREW. This is a pity, as both systems can complement J2ME, in that both the latest versions provide the Mobile Information Device Profile (MIDP) 2.0 to run over the top of their basic OS, or offer facilities and opportunities beyond those provided by MIDP 1.0.

This article will avoid the philosophical software engineering arguments and show through the development of a novel sports application that all three systems have benefits for the developer. Furthermore, we will show that, as with any other task, designers should choose what is most relevant to their application and the market in which it will operate. In Section 2 we provide a brief introduction to the three environments considered in this article (Symbian, J2ME, and BREW) and highlight other potential environments for developing cellular applications that are currently vying for the attention of would-be applications developers. In Section 3 we describe a novel application for future development; in Section 4 we highlight generic implementation constraints presented by J2ME and BREW on the process of “application development.” In Section 5 we discuss the operational architecture of the application, before discussing the important issue of user interface (UI) development in Section 6. Finally, we draw our overall conclusions of the development process and highlight various aspects a developer should consider when selecting a particular environment

2. CELLULAR APPLICATION DEVELOPMENT ENVIRONMENTS

Fundamental device resource constraints have greater influence on programming cellular phone applications than do applications developed for the current PC market. Today, even the most basic PC has large amounts of memory and huge disk space, relieving developers from the need to minimize use of their particular application resources. Whereas cellular phones have limited memory and no disk drive storage, meaning that whatever space the application requires is taken away from overall data space [Kiely 2001]. Even the most advanced cellular phone has a significantly smaller screen size than a PC, which, when coupled with the different ways that users interact with handheld devices, means that developers must get their ideas across within a very small window [Kiely 2001]. There is a range of environments in which cellular applications can be developed, of which we have mentioned three; other possibilities include Windows CE and Windows Mobile [<http://www.microsoft.com/windowsmobile/>], Savaje [<http://www.savaje.com/>], and Linux [Longino 2004]. Phone manufacturers are

generally reluctant to allow Microsoft onto their cellular devices, no doubt fearing a repeat of what happened in the PC market, thus limiting Microsoft's penetration of the cellular market. Linux-enabled phones are still relatively few in number, and most current Linux developers are reluctant to make consumer applications [Longino 2004]. Savaje is a relatively new fully-featured native Java OS, which has thus far only been demonstrated on an iPAQ, although it has gained backing from both Vodaphone and Orange. In terms of illustrating cellular phone applications, we will now briefly introduce the important facts regarding the environments of J2ME, Symbian, and BREW.

J2ME was created by Sun and a group of partners to allow Java specifically to be used on smaller devices by limiting the number of Java core instructions and application program interfaces (APIs) to those features relevant to the target devices [Piroumian 2002]. J2ME generally incorporates the connected limited device configuration (CLDC), which is implemented on top of the OS, and serves as an interface between the OS and Java-based applications [Lawton 2003]. The CLDC uses a cut-down version of the Java virtual machine (JVM), known as the K-virtual machine (KVM), designed for small devices. The J2ME MIDP sits on top of the CLDC and provides a set of APIs that define the way that cellular applications interface to the phone [Lawton 2003].

Symbian was formed in June 1998 [Jipping 2002], and is a sophisticated, powerful, and reliable operating system, written mainly in C++ and designed for small battery-powered devices with extensive communications capabilities. Advanced cellular phones come in many forms, and Symbian differentiates among their user interfaces via phone series (e.g., series 60; series 80, UIQ). Applications developed for a particular series of phones will be compatible with all phones of that type.

BREW can be viewed as: a set of APIs that enable developers to create software applications for cellular phones, and a means of selling and delivering applications. BREW sits between a software application and the Qualcomm application specific integrated circuit (ASIC)-level software. BREW applications can be written using C, C++, or now, in Java, with the implementation of a MIDP 2.0 KVM. BREW is supported by the freely available Software Development Kit (SDK), which allows applications to be developed and tested using the BREW emulator [Barbagallo 2003].

To commence commercial development several additional costs must be incurred: first, in order to access tools fundamental to actual hardware, the developer must be authenticated, which requires a subscription; second, the CPUs used in BREW phones are currently produced by ARM. Since C/C++ applications run on the device, an ARM compiler is required; finally, an application must pass TRUE BREW certification before carriers will make it available on their networks, which again requires an additional cost [Barbagallo 2003]. However, once this process is finished, the application is made available for download on the networks, where for the developer all distribution and billing issues are taken care of. In the case of Symbian and J2ME, the SDKs allow development to the stage of commercial deployment without any additional cost.

3. APPLICATION DESCRIPTION

The English Premier League is arguably one of the best football leagues in the world, and fans in the UK support it passionately. On any match day, supporters want to stay up to date with latest news and running scores, and not only for the particular club they support, but also for the entire league. Current services offered to cellular users work

over the Short Message Service (SMS), and are generally limited to goal alerts, with the user paying by the number of SMS messages received. The application presented in this article to improve both the timeliness of information and the range of information available without increasing the cost to the user uses General Packet Radio Service (GPRS) as its data bearer. This is done via bit-level information exchange optimized for the service [Coulton et al. 2004] in order to minimize information transfer. By providing information such as goals, red cards, yellow cards, substitutions, and so on, this application offers comprehensive updates on all the football matches on a given day. All the information is displayed in an easy accessible format, where a user can simply scroll through the event-by-event coverage of all the league games on that day.

Once the entire day's events become available to the application, the second part of this application becomes possible: a real-time fantasy football game. Fantasy football is played with a typical maximum budget of around £80 million for purchasing players. Players are selected from a list featuring over 500 footballers from Premiership clubs. The players are assigned to specific playing positions, so typically a user can select: one goal keeper, four defenders, four midfielders, and two strikers. When users purchase players, they are restricted to having only two players from any one club. Points are gained or lost depending upon the players' performance during the season's matches; the user whose team has the most points at the end of the season wins. During the season a user can evaluate player performance and decide whether to make changes to the team.

These games are played by large on-line communities, e.g., 300,000 on the BBC last season [bbcfootball.fantasyleague.co.uk/ 8th July 2004]. In the online games, player performance is normally updated only once or twice a week; but in this cellular version, updates occur as the real-time events happen.

The application requires a constantly updated source to receive new updates. The client server architecture for receiving the live updates is shown in Figure 1.

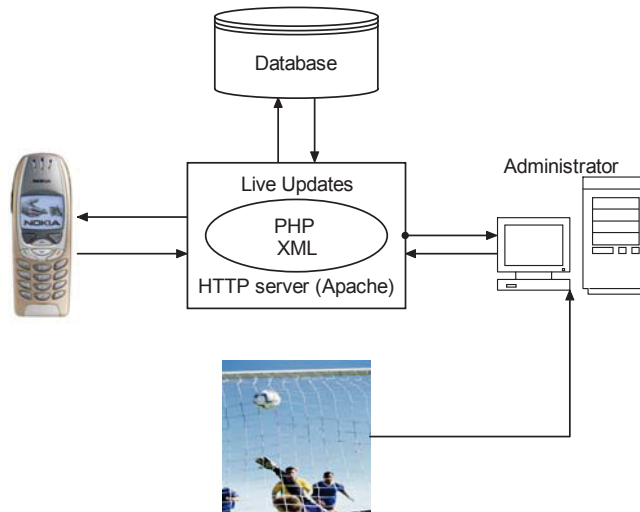


Fig. 1. Client server architecture for live updates.

The live update component, running on a Java-, Symbian-, or BREW-enabled phone, sends a request to the server with the update ID. The server receives its live information, which is then stored in an XML file, from a particular game via updates from the administrator. Each event is given a unique ID that is stored alongside the details about the event (e.g., goal, red card, yellow card, or substitution), the name of the team, and the name of the player involved in the event. All the updates at the end of a day's play are stored in a database. A request from a client is received by a PHP script that checks if the update ID received from the phone matches one on the server's. If the ID matches, then the PHP script parses the XML file containing the event data and returns it to the client. If the ID does not match the current ID of the server, indicating that the client has missed events, the server will send events to the client until its ID matches that of the server. Once the response has been received at the phone, it is interpreted and displayed [Coulton et al. 2004].

MIDP 1.0 implementations are not required to support socket-based connections. In fact the only connection protocol they are required to implement is the Hypertext Transfer Protocol (HTTP) 1.1. Although both Symbian and BREW support socket-based connections, HTTP was selected as a common communication model for all three versions of the applications where market penetration and the delivery of information to the client in the least possible time at an economical charge (currently the application can handle 100 updates for the same cost as one SMS) is of utmost importance.

4. IMPLEMENTATION CONSTRAINTS

To facilitate the forthcoming comparison of the application developed in J2ME, Symbian, and BREW, we shall first discuss the more generic advantages and disadvantages of an application running over these environments.

Size. Java developers are restricted to a limited application size of a few kilobytes (typically 30-64K), compared to Symbian and BREW applications, which can be in the multiple megabytes. However, for an installation over the air (OTA), where users simply select the application they want to purchase and download it to their phones, there are often limits (128k or 256k are typical) defined by the operators, which negates some of the advantages of Symbian and BREW.

Interaction with Phone Features. J2ME (MIDP 1.0) has no direct access to IrDa, Bluetooth, phone dialing, a phone book, or calendar, thus limiting the scope and the ability of applications to interact. MIDP 2.0 provides support for Bluetooth in the form of JSR-82 [Piroumian 2002], although the numbers of cellular phones in the market that are MIDP 2.0-compliant are few. Applications developed for Symbian and BREW can access IrDa, Bluetooth, SMS, MMS, calendars, contacts, and can even dial the phone. Generally J2ME access to SMS can be overcome by vendor-specific APIs.

Application Speed. Since Symbian and BREW application code is compiled to machine code, they are innately faster compared to their J2ME counterparts. In this particular application, speed is not a great issue; but in other cellular phone applications, such as games, it is much more significant, particularly considering that the typical processor speed of a cellular phone is around 100 MHz.

Data Storage. For J2ME applications, persistent data storage on the phone is achieved through a record management system (RMS), which allows the application to record its data as individual entries, although it has to be contained within the allotted

memory. Unlike Symbian, Brew has a simple in-built database system, while Java database connectivity (JDBC) is an optional package if an embedded database is to be used. Java storage is also limited by the fact that it must reside in the allotted application memory, while Symbian and BREW have a much larger amount of available memory, with a further provision for expansion. In this particular design a flat file structure was used to minimize data storage and information transfer, with BREW and Symbian being more efficient, as they allow bit-level manipulation of data.

Multitasking. Symbian is a multitasking OS, which means that more than one process or server can be started to suit the needs of various applications. This is a distinct advantage over J2ME applications, as the application can be made to run as a background process. Although one way that two J2ME applications could use the same data is to give them access to the same record store. Unlike its predecessor, MIDP 2.0 incorporates push functionality, which means that a MIDlet can be launched in response to an incoming message. At present, BREW only supports cooperative multitasking, in that applications must be developed to interact in this way. While this is an improvement over J2ME, it is a long way short of Symbian. We discuss the implications of multitasking on the application architecture in more detail in Section 5.

Cross-Platform Flexibility. Although J2ME is well documented and has a huge developer community, one of the most critical issues is that it has no facility to automatically scale the drawing area to the screen size. This causes serious problems for the developers, as they have to design different versions of the same application to suit the needs of phones with various screen sizes. Since Symbian has a number of series relating to user interfaces, it is often necessary to change some application code to make it compatible with a variety of phones. But an application developed for the Nokia series 60, for instance, will run on all series 60 phones without modification.

Market Penetration. Java-enabled phones are much more common in the market, even when we exclude those running Java over Symbian and BREW. Theoretically, this gives an application a wider user base, which can affect the success of an application. As BREW is currently restricted to the Qualcomm chipset it has fairly limited base. In terms of smart-phone sales, Symbian (with Nokia) is dominant [Garfield 2004], with 41.7% of the market; its nearest rival RIM has a mere 12.7%.

Application Security. In terms of the potential threats that standardized OSs present in terms of Malware [Jamalludin et al. 2004], Symbian and Java are currently the most vulnerable, particularly in the European market, where they can be readily installed OTA with minimal restrictions. The Java Application Descriptor (JAD) file specification of MIDP 2.0 provides an extra layer of security between the device and the application; as for successful OTA installation, the information contained within the JAD file has to be very specific. The MIDP 2.0 JAD introduces the concept of security domains, which can either allow or prevent access to data, network resources, or even other applications. BREW certification aims to ensure that carrier networks remain free of viruses and malicious or unstable applications. Symbian has recently introduced its own voluntary authentication (signed) process; although many operators are already limiting their portals to signed applications. The latest version (9.0) of Symbian [Seizen 2005] will have, in effect, an in-built firewall that will restrict nonsigned applications from communicating through the network.

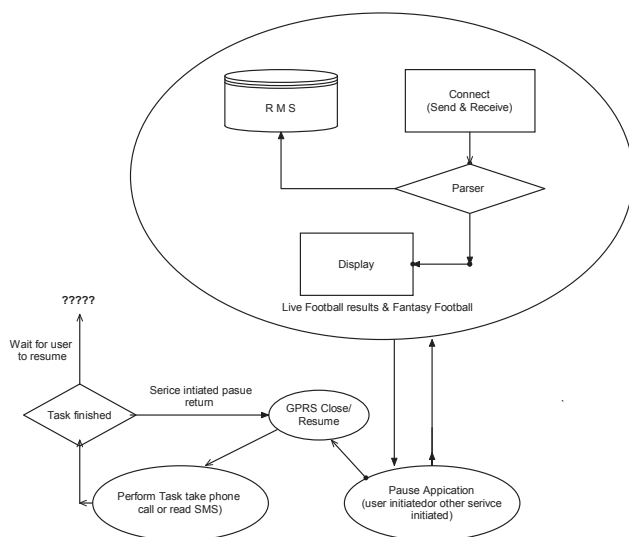


Fig. 2. Application architecture in J2ME and BREW.

5. APPLICATION OPERATIONAL ARCHITECTURE

In this section we discuss how the operational architecture of the application in the three environments differs. Figure 2 shows the main components of this application along with its behavior in relation to the rest of OS

If, when running the application, the phone encounters another service that has greater priority, such as a call or SMS, it will automatically “pause” the application and take the user to the main display, to either accept or reject the call or view SMS. During this transition the GPRS connection is suspended. Once the new task has been performed, the application returns from its pause state and resumes the GPRS connection. If the pause is initiated by the user, the application will simply save its state, allowing the user to access the other features of the phone. Even after the user has finished other tasks, the application cannot resume unless the user goes back to the applications menu and selects restart (resume the application), which will cause the application to wake up and resume from the last saved state. This is the scenario that will be commonly seen on BREW- or J2ME-enabled phones. If the application was running as J2ME over Symbian, the phone could use Symbian’s multitasking capability and spawn another process to run the other task.

Figure 3 shows the operational structure of the application created in Symbian, where the main application architecture has been kept the same as the one created in J2ME and BREW. In this case a user- or service-initiated task is dealt with by the OS, which creates a multitasking environment where the user can switch between different applications by simply holding down the menu key. The GPRS connection can stop or resume depending on whether or not another application requires the connection.

The foregoing explanation illustrates some of advantages and disadvantages outlined earlier; although for the three applications the major difference is due to the accessibility of other phone features. Only the J2ME application has access to Bluetooth directly (and

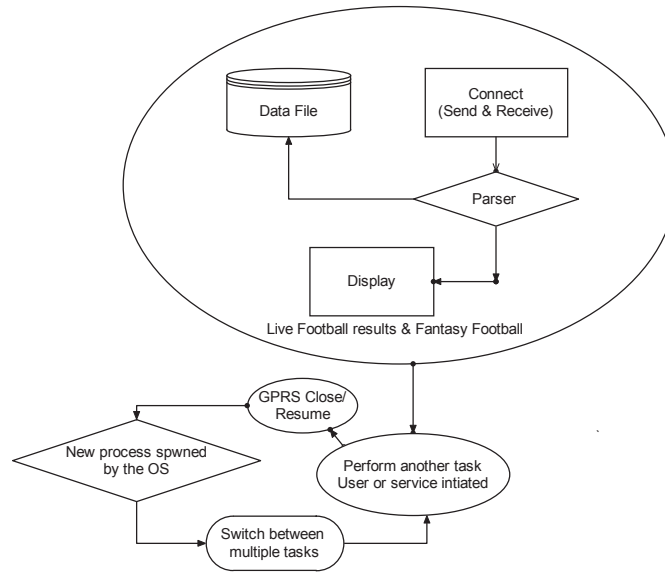


Fig. 3. Application architecture in Symbian.

only in MIDP 2.0-enabled phones), whereas Symbian and BREW can access Bluetooth, IrDA, SMS, and MMS [<http://brew.qualcomm.com/brew/en/>]. Access to the other commands and features can be used to enhance the application and produce wider benefits to the user:

- to invite a friend to try an application (SMS);
- to have a league of your own, compete with friends (the Bluetooth- or SMS-enabled fantasy football feature);
- to send the score or event to other friends (SMS-enabled feature allowing users easy access to the SMS to send the score or perhaps even another text);
- to make a phone call (for novice users); and
- to read news about a club and or league (access to a browser).

6. USER INTERFACE

The overall architecture of the application was kept the same for each operating environment, with the principal difference in user interaction (with the application via UI). So how different can a user experience be in different operating environments? In this article the experience is split into two classifications, i.e., physical constraints and UI (user interface) interactions.

Physical constraints are defined by the cellular phone hardware, for example screen size, processing power, and memory. To gain maximum market penetration, an application may have to operate on many cellular phones, with screen performances ranging from 176*208 color displays to 100*80 monochrome displays (also, screen orientation is portrait rather than landscape, which could effect some applications). These factors affect all application environments equally. But because Symbian is mainly associated with “feature-rich” phones, it circumvents some of these problems.

In the following paragraphs we are primarily concerned with the user interface experience.



Fig. 4. User interface (J2ME).

J2ME. The trade-off for J2ME’s cross-platform flexibility comes in the form of program structure and usability. This is due not only to the difficulty of accessing Bluetooth, SMS, MMS, etc., but to the actual look-and-feel of the application.

Most notably, with J2ME we lose the ability to trigger dynamic menus in conjunction with a main application screen. Therefore, every time an option has to be presented to the user, it must be accessed from a main menu screen. For example, in the football application, if a user wishes to update the selection process via the team update page, he or she has to do so via the fantasy football selection on the options page. If a user then wants to go back to the live updates screen, he or she must navigate back through the menu structure. This gives the application a much more rigid and less flexible feel.

To highlight the major benefit of J2ME’s market penetration, we have chosen the Nokia 6310i as a platform to run the application on, as it offers the minimal required features. This phone is very widely distributed and easily available, even on the cheapest subscriptions. Figure 4 shows the application running on this device.

Symbian. In the Symbian application the user is able to navigate more easily, primarily due to the use of “soft keys” for the navigation and menu structure. The application’s main screen is considered somewhat independent of the menus executed by the soft keys (although menu content can be dynamic with respect to the context of the main screen), allowing operations and navigation to be carried out without having to first navigate back to a main menu/options screen. This allows a “free flow” movement throughout the application. The soft keys can contain instructions not only for the application but also for functions inherent to the phone itself. An example of this is evident in the fantasy football aspect of the application, where a user, playing the game via GPRS, may also choose to enhance the gaming experience by engaging, with selected friends, in a mini-league via a Bluetooth connection. Likewise, a similar system could also be set up using SMS for a “friends league” played over a wider area.

Figure 5 shows the main screen of the Symbian application running on the Nokia 3650, but in practice there can be many “main screens” layered on top of one another that can be selected by tabbing. For example, multiscreens to display different data elements such as “Your Team Updates,” “Premiership Updates,” and the “Fantasy Football” game.

The dynamic structure of the menus in Symbian presents the user not only with the possibility of enhanced features but an easier navigational experience.

BREW. The functionality of BREW allows the developer to produce a UI structure with a fluid feel similar to the one described previously for the Symbian application, shown in Figure 6.

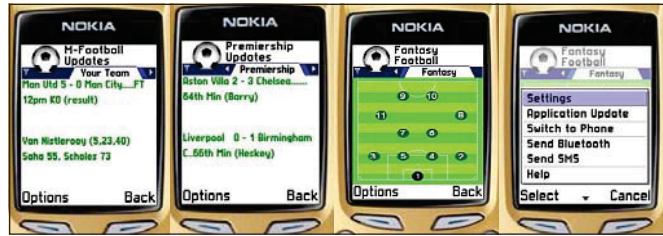


Fig. 5. User interface (Symbian).



Fig. 6. User interface (BREW).

Menus and list can be executed by the use of ‘soft keys’ (on the majority of ‘typical’ cellular phones they are two keys just below the display) allowing program flow to be unrestricted within the application or inherent phone functions such as answering phone calls. Further, as with Symbian, BREW is capable of utilising dynamic menus, allowing items within the menus to change with respect to application functions. On a final note we say that although Brew presents the same opportunities as Symbian for the developer at present there are fewer built-in functions to allow the developer to produce complex UI.

7. CONCLUSION

In Section 2, we identified the most prevalent environment from the wide range of possibilities for developing cellular entertainment applications that currently exist. We selected three in particular, Symbian, J2ME, and BREW, for a more detailed analysis, as we believe they currently hold the strongest positions in the marketplace. However, we recommend that developers consider very carefully the requirements of their application and the commercial operating environment in which they will operate, as this area is still in its relative infancy, and environments and devices are evolving remarkably quickly.

Section 3 describes a novel sports application, developed by the authors, which we believe highlights some of the major benefits of a cellular application, in that it can provide instantaneous information and entertainment at anytime and anyplace in a clear and user-friendly manner. It also highlights the fact that many existing web-based services can have their user experience greatly enhanced by incorporating the benefits of

cellular systems, rather than merely allowing access to them via the cellular device.

The resource constraints presented to developers by cellular phones and their environments provide the greatest challenge to producing compelling applications. In Section 4 we explore some of these constraints, and in particular their relevance to the operating environments of Symbian, BREW, and J2ME. Section 5 then highlights how the particular operating environments affect the operational architecture and functionalities within the particular application. These resource constraints often change with each new iteration of both the operating environments and the capabilities of particular handset design. This means that porting and testing can present very complex tasks for developers, and should not be underestimated.

The UIs of any cellular application must be designed very carefully, as these devices were developed with the principal aim of making and receiving calls, and thus place many restrictions on the developer. As with any application, the UI should be easy to use and as intuitive as possible. In Section 6 we discuss features within the three environments that affect UI design.

As previously stated, we do not wish to engage in a debate about which development environment is best, but rather to comment on our personal experiences derived from using these environments in teaching and developing both academic and commercial applications. Of the three, Symbian presents the greatest “learning curve,” particularly for developers inexperienced with C++ but familiar with more user-friendly development tools. However, Symbian offers developers the chance to access many features unavailable in J2ME, which can be used to greatly increase the ability to develop novel applications and enhance the user experience. BREW is relatively easy to learn and use, and we found it particularly well suited to game development, although it does not currently offer as many options for creating novel user interfaces as Symbian does. Of the three, J2ME is obviously the most established in terms of general programmer experience and the wide availability of handsets. This means that developers experienced in Java can quickly produce cellular applications, although with greater constraints than those experienced with Symbian and BREW.

Finally, we conclude that although developing cellular applications for entertainment can be very challenging, it is also highly rewarding in that it provides the opportunity to produce new and exciting entertainment experiences.

ACKNOWLEDGMENT

The authors wish to acknowledge the support of Nokia for the real-time hardware and software laboratory in Infolab21 at Lancaster University, where much of this work was carried out.

REFERENCES

- BARBAGALLO, R. 2003. *Wireless Game Development In C/C++ with BREW*. Worldware, 2003. [/bbcfootball.fantasyleague.co.uk/](http://bbcfootball.fantasyleague.co.uk/). July 2004.
- COULTON, P., RASHID, O., AND AHMED, H. 2004. Live information update services using GPRS. In *Proceedings of the IEEE ICTTA 04 Conference* (Damascus, Syria, May19-21, 2004).
- GARFIELD, L. 2004. Mobile phone sales up over last year. www.infosyncworld.co. May 2004.
- <http://www.symbian.co.uk>
- <http://brew.qualcomm.com/brew/en/>
- <http://java.sun.com/j2me/>

- JAMALUDDIN, J., ZOTOU, N., EDWARDS, R., AND COULTON, P. 2004. Mobile phone vulnerabilities: A new generation of malware. In *Proceedings of the 2004 IEEE International Symposium on Consumer Electronics* (ISCE_04, Sept. 1-3, 2004, Reading, UK). 1-4.
- JIPPING, M. J. 2002. *Symbian OS Communications Programming*. Wiley, UK, 2002.
- KIELY, D. 2001. Wanted: programmers for handheld devices. *IEEE Computer* (May 2001), 12-14.
- LAWTON, G. 2003. Moving Java into mobile phones. *IEEE Computer* (June 2003), 17-20.
- LONGINO, C. 2004. Mobile Linux looks set to grow. www.thefeature.com. May 2004.
- PIROUMIAN, V. 2002. *Wireless J2ME Platform Programming*. Sun Microsystems Press, 2002.
- RICHARDSON, T. 2004. Global mobile phone sales roar. www.theregister.co.uk. June 2004.
- SEIZEN, S. 2005. Symbian OS v9.1 functional description. Revision 1.1, Feb. 2005. www.symbian.com/technology.
- VAUGHAN-NICHOLS, S. J. 2003. OS battle in the smart-phone market. *IEEE Computer* (June 2003), 10-12.

Received February 2005; revised April 2005; accepted April 2005