participate

DTI ProjectNo:TP/153421

Report Headline

Addressing Mobile Phone Diversity in Ubicomp Development

Work package:1 (Games)
Deliverable: Relating to Technology Roadmap

Principal Authors (Partner): Nottingham University - Mixed Reality Lab
Chris Greenhalgh, Steve Benford, Adam Drozd, Martin Flintham, Alastair Hampshire, Leif Oppermann, Keir Smith, Christoph von Tycowicz, Alan Chamberlain

Version: 1
Release Date: March 2007
Status: Consortium

Deliverable Identification Sheet

DTI Project No	TP/153421
Acronym	PARTICIPATE
Full Title	Pervasive Computing for Mass Participation in Environmental Monitoring
Project URL	http://www.participateonline.co.uk
DTI Project Officer	Lee Vousden
DTI Monitoring Officer	Nigel Wall

Deliverable	NA
Work Package	1.2 (Games)

Date of Delivery	March 2007	
Status	Version 1	Final x
Nature	Report x	
Dissemination Level	Consortium x	

Authors (Partner)	Nottingham University			
Principal Author	Chris Greenhalgh		Email	cmg@nott.ac.uk
	Partner	MRL	**Phone**	

Abstract (for dissemination)	NA
Key Words	

Version Log			
Issue Date	**Rev No**	**Author**	**Change**

Executive summary

Summary. Mobile phones are a widely-available class of device with supporting communications infrastructure which can be appropriated and exploited to support ubicomp experiences. However, mobile phones vary hugely in their capabilities, especially with regard to communication, context sensing and application hosting. This report explains how a single dimension of 'phone application type' embodies the critical trade-off between capability and availability, i.e. between what can be done and the fraction of potential participants' phones that can do this. The report describes four different mobile phone ubicomp experiences that illustrate different points along this continuum (SMS, WAP/Web, and J2ME, Python and native applications). It then introduces a common software platform that has been co-developed to support these experiences. From this, and supported by a survey of the core experience developers, it then identifes four strategies for addressing mobile phone diversity: prioritise support for server development (including web integration), migrate functionality between server(s) and handset(s), support flexible communication options, and use a loosely coupled (data-driven and component-based) software approach.

Contents

Addressing Mobile Phone Diversity in Ubicomp Experience Development

Chris Greenhalgh*
Steve Benford*
Adam Drozd*
Martin Flintham*
Alastair Hampshire*
Leif Oppermann*
Keir Smith (University of New South Wales)
Christoph von Tycowicz (Hochschule Bremen)
Alan Chamberlain*
(* Mixed Reality Lab, University of Nottingham)

1 Introduction

Anyone wishing to deploy a ubicomp experience outside of a lab setting has two options for technology platform: they can either deploy their own devices and infrastructure, or they must appropriate and build upon existing devices and infrastructure. The mobile phone is an excellent – and popular – example of such an existing device and supporting infrastructure.

This report describes the trade-offs which must be made when targeting mobile phones as a technology platform for ubicomp experience development. It illustrate these trade-offs using four different mobile phone experiences and the common software platform that has been co-developed with them. From this the report identifies four strategies for effectively exploiting mobile phones.

It begins by charting the potential availability and utility of the mobile phone as a platform for ubicomp experiences.

Perhaps the most compelling argument made in favor of using a mobile phone platform is its almost ubiquitous status in many countries today. For example, in 2006 in the UK mobile phones were owned by 85% of the adult population [1], were present in 90% of households [2, p.157] (the same fraction as have fixed line telephones) and had coverage across 99.9% of the country (2G) [2, p. 17]. For many users their mobile phone is an important and integral part of their everyday life: in the UK in 2005 73% of mobile users rated their mobile phone as “essential”, while 31% of all telecoms users used their mobile as the main method of making calls (up from 21% in 2004) [2, p.158].

It then identifies two main ways in which ubicomp researchers can – and do – build upon and exploit mobile phones as a technology platform. The first approach uses the mobile phone as a system component which is readily available and relatively cheap (due to the large market). For example [3] uses mobile phones as personal interfaces to an augmented card-playing table, while [4] uses a mobile phone as a mobile client/interface for a mobile mixed-reality game. In each case the handsets are programmed and provided as part of a complete system and lent to users or provided in situ.

The second approach uses – or seeks to use – people’s own mobile phones as the technology platform. For example ContextPhone [5] augments users’ smart phones with a context-sensing software platform to support (e.g.) augmented contact management and tagged media sharing; [6] shows how mobile phones with local radio (Bluetooth or RFID) could be used to create a wide-scale object location system; [7] trials a location-based reminder application on mobile phones; and [8] explores the use of a mobile phone’s camera as an interaction device. Using people’s own mobile phones reduces the resources to be supplied by the ubicomp researcher, offers the possibility of a very large-scale deployment and means that the device is already part of a user’s everyday activities and routines.

However, mobile phone handsets are extremely diverse, varying widely in display (e.g. screen size, resolution, colour), input (e.g. keypad, touchscreen, camera, location), memory capacity, processor speed, support for downloaded applications (e.g. J2ME MIDlets, native applications) and networking (e.g. GSM, SMS, MMS, GPRS, 3G, Bluetooth or Near Field RF). Many of the current projects which target mobile phones as a “ubiquitous” infrastructure actually target a very specific subset of handset capabilities. For example, [5], [6], [7] and [8] all require smart phones running a custom application (native or J2ME), [6] requires local radio (Bluetooth or active RFID), [5] and [7] (like many phone-based location-sensitive

applications) require access to GSM cell ID which is only available via a native application on a particular range of operating systems and [8] requires an integrated camera and supporting API.
In practice this restricts a project to some combination of: a small minority of current handset owners; giving or lending suitable handsets to participants; or a possible future in which those handset capabilities will dominate the market. In the first case the handset is still part of the user's own everyday routine, however the potential scale of the deployment is limited, and some areas of research are almost impossible (e.g. "viral" recruitment of social contacts who have different handsets). In all cases the potential scale of any current practical deployment (and therefore of any non-simulation evaluation) is quite limited.
The remainder of this report is structured as follows. In section 2 it is suggested that the key dimension which can be used to characterize handset capability, and identify some of the key trade-offs inherent in targeting different points along this dimension. Following this four mobile-phone-based ubicomp experiences at different points along this dimension (section 3) and the common supporting software platform that has been co-developed with these experiences are introduced (section 4). Section 5 then identifies four significant strategies for addressing mobile phone diversity in system and experience development. Finally, section 6 identifies areas of ongoing and possible future work.

2 Characterising Mobile Phone Diversity

The mobile phone handset market is very diverse, and several factors tend to maintain this diversity: users' preferences for different handset sizes and form factors, maintaining brand identity and distinctiveness, choice of device capability versus ease of use (e.g. the current "backlash" against phone complexity in some market segments [9]), and high levels of competition and cost-sensitivity in most market segments.
Th Mixed Reality Lab has been involved in designing and developing a number of ubicomp experiences based on mobile phone platforms, in partnership with other organizations. The most significant choice that has had to be made in each experience has been the kind of handset application which the end-user will employ within the context of the ubicomp experience, with the main trade-off being between the potential numbers of users on the one hand and the richness of functionality and interaction supported on the other hand. Figure 1 summarises the main options and trade-offs associated with this dimension of choice.

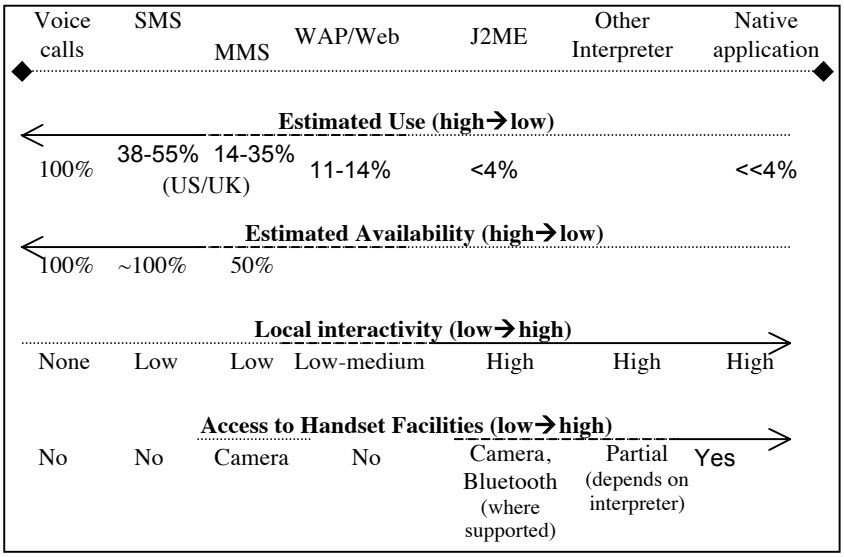

Figure 1. Main options for mobile phone application type and associated trade-offs.

Most handsets can be organized along this single dimension of handset application type: voice calls; SMS/text messaging; WAP/web browser; J2ME MIDlet; other cross-platform interpreter; native smart-phone application. In many cases there are further sub-options or variations (some of which are considered below), however this dimension is the starting point for assessing potential availability and scale. Each option is now considered in turn.

2.1. Voice Calls

Voice telephony is the only universally available service on mobile phones. However the use of voice telephony in ubicomp experiences appears to be limited to direct communication between players/participants without system mediation, e.g. [10]. Richer experiences could be supported by integrating VOIP (Voice Over IP) functionality into the system more generally, or by using phone operator location services which are available commercially on an explicit opt-in basis. However voice-based interaction may still be problematic, being relatively intrusive, subject to interference from surrounding noise, technically challenging (e.g. to achieve speaker independent recognition) and unsuitable for maps and other inherently visual information.

2.2. SMS/Text messaging

After voice telephony, SMS or text messaging is the next most widely available option for incorporating mobile phones into a ubicomp experience. However availability and take-up varies worldwide, for example use of text messaging in the UK in 2006 was 85% of mobile phone subscribers but only 38% in the US [11]. Compared to voice, text messaging is much easier to integrate into a complete system (via one of the many commercial SMS gateway service providers). Text messaging is also generally less intrusive than voice interaction, with received messages being held until dealt with.
A further sub-option here is MMS or photo messaging. On a phone with integrated camera this provides a richer capture capability for user content and context, or for interaction and location when used with server-based glyph reading technologies (as in CONQWEST [14]). However, take-up of picture messaging is substantially lower than for text messaging, with about 30.7% of UK subscribers and 14.5% of US subscribers in 2006 having used it [11] (36% and 38% of the numbers using text messaging in the UK and US, respectively). About 50% of subscribers actually have a handset with an integrated camera [19].
SupaFly [12] is one example of an experience (a pervasive game) based on SMS communication with players. Botfighters (version 1) (see [15]) combines SMS communication with operator location services. Of the experiences described in this report DoF (section 3.1) also relies solely on SMS for mobile phone interaction, while LoveCity (section 3.2) uses SMS plus operator location as its “mass” participation option. SMS is also mentioned for direct player interaction in [10].

2.3. WAP/Web Browser

The next step up in capability is dynamic WAP/Web pages accessed via a (pre-installed) mobile phone browser. The main sub-options here are WML over WAP [13], which is more widely available on handsets (especially entry-level handsets) but limited in graphical richness, and XHTML over HTTP, which is currently limited to more capable (often smart phone) handsets but gives full HTML functionality and composition options. WAP/Web-based interfaces allow more complete interactive applications to be delivered via mobile phones, with graphics, navigation, forms, etc. On some networks and contracts this may also be cheaper than using SMS.
However, as well as requiring a sufficiently capable handset, there are other potential obstacles to their actual use: they require a contract which includes data services; this often has to be configured by the user (at least selecting the correct “access point”); they require reasonable network connectivity throughout the interactive session (each exchange with the server, which may be every page); and interaction latency is generally high (of the order of 3-5 seconds for even simple pages). The take-up of WAP/web-based services is much lower than for text messaging, with 14.4% of UK subscribers and (relatively higher) 11.7% of US subscribers using their mobile phone to browse news and information in 2006 [11].
WAP/Web interfaces can be combined with WAP push/SMS to push new experience elements to the user, and can also be combined with operator location services.
Of the experiences described below Prof. Tanda (section 4.3) uses XHTML/HTTP as its main mobile phone interface, although in that case it is supported by an additional Python “trigger” application also running on the handset.

2.4. J2ME

The majority of ubicomp experiences on mobile phone platforms have a custom application running on the handset. Compared to the previous options, this can support richer interaction, reduce interaction time, give more complete access to the phone’s facilities (e.g. camera) and support peer-to-peer (e.g. Bluetooth)

and disconnected operation. J2ME appears to be the most widely available platform for running applications on mobile handsets at the present time however it is difficult to obtain reliable data in the public domain. In 2004 penetration of J2ME on 3G handsets was predicted to reach 75% this year [16], and 50% on 2.5G handsets. In the UK at the end of 2005 8.0% of households had a 3G handset/service [2]. In 2006, 4.2% of UK subscribers and 3.2% of US subscribers had actually downloaded a game (not necessarily J2ME-based) to their phone [11].
However J2ME is not a single option but rather a set of options. The minimal (most widely compatible) "profile" for J2ME is CLDC1.0/MIDP1.0, which is quite limited. Further sub-options include CLDC1.1 (adds floating point types), MIDP2.0 (adds better UI and more networking options), Bluetooth, Media (adds camera access) and so on.
J2ME development also suffers from several complications, including buggy and inconsistent virtual machine and package implementations (see [17] and [18]). It is also not possible to access native APIs directly from J2ME applications; where this is required a separate native application must be deployed on the phone that can be communicated with via (e.g.) local HTTP (for example, ContextPhone [5] and PlaceLab [22] support such local interfaces).
[3] and [7] both use J2ME phone applications, which in [7] is also combined with a small native application (to obtain phone cell ID). The phone client for MobiMissions (section 3.4) is implemented in the same way.

2.5. Other Cross-Platform Interpreters

There are also a number of non-Java scripting engines or interpreters available for mobile phones (in particular, smart phones). In general these are native applications which, once installed, allow other content and applications to be downloaded and run. Current options include a Python interpreter (for Nokia Series 60 phones), Flash Lite (a cut-down version of Adobe Flash, available for Symbian and BREW) and Nokia's MUPE multi-user application development platform [20] (the interpreter for which is a J2ME application).
These have the advantage of (generally) simpler development than for native applications, but typically require a large installation of the interpreter on the phone first, and (like J2ME) have more limited access to native phone facilities than a native application. LoveCity (section 3.2) has a Python-based smart phone client option, while Prof. Tanda (section 3.3) has a Python-based "trigger" application which runs in the background on the phone.

2.6. Native Application

As well as the general advantages of running a custom client, native applications are often essential to access the widest possible range of handset information. However, this also narrows the compatibility to very specific platforms and handsets. [17] and [18] deal at some length with issues of native application development and capability. At this level handsets are more diverse and less compatible than the common profiles of J2ME. The main platforms at present are Symbian (although the Nokia and Sony Ericsson UI layers differ), Windows Mobile and BREW (which is strictly a cross-platform development/deployment environment) (see [18]).
As already noted native helper applications are used in [7] and MobiMissions. The applications in [4], [5], [6] and [8] are all realized as native applications.
Note that installing a native or J2ME application raises additional barriers to participation beyond handset compatibility: the download may be slow (and expensive over the air), the user may not trust the application provider, or the user may not be familiar with this aspect of their phone's capabilities.

3 Introducing the Experience Projects

The previous section charted the space of options and the main trade-offs for exploiting mobile phones as a platform for ubicomp experiences. Briefly described are four experiences that illustrate different points on this continuum (see figure 2). These have been developed and deployed over the past two years in collaboration with three different user groups: Blast Theory for Day of the Figurines and Prof Tanda; Active Ingredient for Love City; and Futurelab for MobiMissions. They have been implemented using, and had a formative role in the development of, our common software platform which is described in section 4.

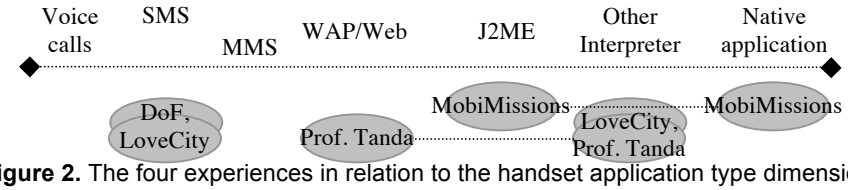

Figure 2. The four experiences in relation to the handset application type dimension.

3.1. Day of the Figurines (DoF)

DoF is a role playing game for mobile phones that employs text messaging in order to be widely accessible to large numbers of players who can use their own mobile phones. The game follows twenty four hours in the life of a small virtual town which are mapped onto twenty four days of real time. It is therefore a long-term, slow-paced game that unfolds in the backgrounds of players' lives, requiring them to send and receive just a few messages each day. Players take on the roles of refugees who have arrived in the town. They join the game by visiting a physical venue where they register their details and choose a plastic figurine to represent them. As part of an additional spectator interface to the game, human game operators move their figurine around a large physical game board that models the virtual city (figure 3), following arrows that are projected onto the board by a custom game client.

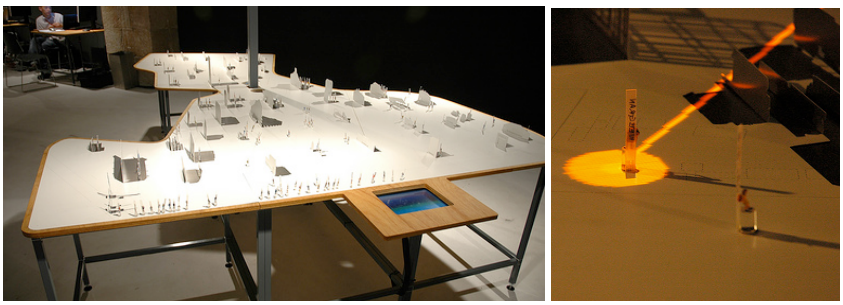

Figure 3. The DoF game board (left) with a digital annotation (right)

After they leave the venue, players control their figurine by sending and receiving SMS text messages, employing a simple grammar to specify different actions including: **go** to a destination; **say** a message to nearby players; **pick** up, **use** and **drop** objects; **find** another player and **leave town** (quit the game). In return, messages from the game convey pre-authored descriptions of fifty different destinations and various events that occur at different times, responses to players' actions, multiple choice dilemmas and chat from other players. Players are also allocated missions that combine these other elements. The following short exchange of messages is typical of SMS interaction in DoF:

> **RECEIVED**: 09:57pm, you've arrived at the Trafalgar Sq, Italian football is on the telly. BERNARD, EVE and SUCHDA are here. There are many PINTS here.
> **SENT**: Use pint
> **RECEIVED**: 09:57pm, you down the pint of Vale leaving a modest foam moustache on your top lip. You are feeling well.

The first message in this sequence demonstrates an aggregation technique in which details of nearby players and objects are dynamically appended to key event messages rather than generating a separate message every time a player enters or leaves a destination. This enables the game to fully utilise the bandwidth of each message and minimize the number that are sent so as to avoid flooding players and to reduce costs.

As part of a process of iterative development, DoF has so far been deployed in London (85 players), Barcelona (160 players), Berlin (145 players) and Singapore (141 players). In its most recent outing in Singapore, players sent a total of 12,685 messages to the game, received a total of 21,767 from it, and 75% of the 24 who responded to a questionnaire said that they would play again.

3.2. Love City

Love City is a location-based game for mobile phones that connects three different physical cities with and through a single virtual city. Players' movements through their local physical cities are mapped to their location in the virtual Love City, enabling them to encounter one another. For example, three players who are at the civic centres of their respective cities would meet at the centre of Love City. A player is given

one chance to send a message of love – an anonymous text message – to each other player that they meet who may then choose to accept or reject it (figure 4, left). Acceptance leads to a 'pair bond' being formed between these players. If a player manages to initiate pair bonds with players from both remote cities they achieve a 'menage a trois' at which point they gain a new offspring which they can instruct to send further messages to players that it encounters.
A central website enables players to gain an overview of the state of Love City including recent messages that have been sent, offspring that have been created and the virtual locations of other players (figure 4, centre).
The implementation of Love City includes two different mobile phone clients. In order to address as wide an audience as possible, the first uses SMS messaging only on the handset, supported by network operator location of the handset where available (on three out of the four major networks in the UK in the initial trial). The second provides a more sophisticated graphical interface delivered through a Python client application (figure 4, right). This utilises cell ID positioning in the client and communicates with the game server over HTTP. The artists who created the game engaged in extensive war-driving of the three cities, logging combinations of cell ID (for the Python client) and GPS coordinates (for operator location and geo-referencing) in order to define the different virtual city regions.

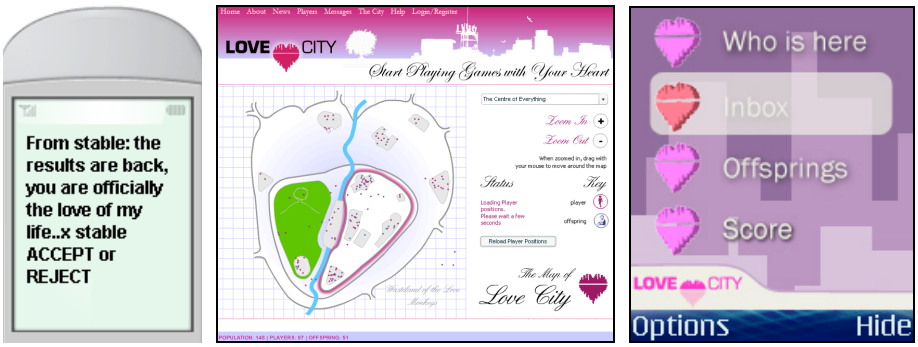

Figure 4. LoveCity text message (left), website/Flash view (middle), and Python client (right)

At the time of writing initial trials of Love City are underway. So far, over one hundred players are taking part after two weeks of a four week trial period.

3.3. Prof. Tanda

Prof. Tanda is a mixture of a game and survey and is intended to engage players during their daily routines, providing them with amusement and information in return for data about their lifestyle, environmental actions and attitudes. This information is then sifted and fed back to the public by broadcasters and campaign organisations.
Prof. Tanda is played up to twice each day for about ten minutes per session. The game is embodied through a quirky character called Professor Tanda who contacts the player, tries to guess where they are and what they might be doing, asks them questions and even gets them to undertake simple activities and experiments such as measuring the amount of water that they use when taking a shower by leaving the plug in the bath (see figure 5, right).
A distinctive feature of Professor Tanda is the way in which he tries to guess a player's current context whenever he contacts them. This draws on the time of day and also on handset location (cell ID). For example, if a player tells the game that they are at home in one session, then the game will associate "home" with the time and logged cell ID of that session and use this information to tailor subsequent sessions.
Prof. Tanda is currently realized as a "trigger" application implemented in Python and running in the background on the phone and XHTML pages generated by the server. Each night the trigger application downloads details of when it should next trigger a session from the server. It then monitors the time and current cell ID, and when the condition is met (or when the player explicitly "calls" the Professor from the trigger application) it directs the phone's web browser to a session-specific URL on the server, which generates the XHTML pages for the player's interaction with the Professor during that session.
This initial implementation of Prof. Tanda has recently been trialed by 20 players for two weeks. In general, they reported enjoying their interactions with the central character in the game; especially the way in which it engaged them in local activities, an aspect of the game that they would like to see extended in future

versions. Currently a second trial is planned with additional context information being logged and used in triggering (e.g. phone profile).

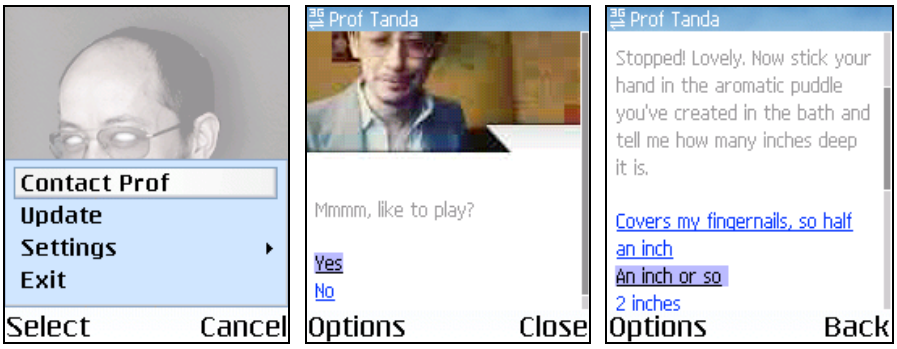

Figure 5. Images from Prof Tanda trigger app. (left) and XHTML pages (middle & right)

3.4. MobiMissions

MobiMissions is a game in which players use camera smart phones to create, complete and document real-world missions. It is a conceptual extension of the text-only game Hitchers [21]. The content and purpose of missions is left open-ended for players to define for themselves, each mission being defined by up to five photographs and/or five sections of text. A player creates a mission by using their phone to take a series of photographs and enter text instructions, and then drops the mission from their phone. The mission remains where it was dropped until it is found and picked up by another player (location being determined by the mobile phone's cell ID).

Players can search their current location to find missions which they can then choose to pick up, at which point the mission is transferred to their phone (figure 6). As a player carries out a mission (potentially over several sessions of play) they document their progress by capturing up to five photographs and adding short text annotations. When they are happy with their response they submit it, which causes their response to be uploaded to the game server and the mission to be dropped from their phone. They also rate how good the mission was.

The mobile game is supported by a website which allows players to browse missions and responses, rate other players' responses to a mission and leave comments.

The MobiMissions phone client is implemented in J2ME, specifically CLDC1.1, MIDP2.0 and the media API (for camera access). Communication with the game server is over HTTP. Cell ID is determined using the Placelab server (a native Symbian application) that runs on the handset [22], limiting the experience to Symbian Series 60 phones.

An initial trial of MobiMissions has been conducted with a group of seventeen 16-18 year old users who created 75 missions which generated 123 responses over a period of five weeks. Feedback through participant diaries and interviews showed that these players generally enjoyed the experience, although many would have preferred to spread missions from player to player through social networks rather than via location-based play.

Figure 6. Available missions, mission details and a response from the MobiMissions client

4 Introducing the Common Software Platform

The four ubicomp experiences introduced in the previous section have been implemented using and co-developed with a common software platform, EQUIP2, which is introduced in this section.

The core of EQUIP2 is a "dataspace" API, i.e. a tuple-space [23] which stores strongly typed (programming language) objects rather than un-typed tuples. As the name implies a dataspace can be thought of as an information space into which objects can be placed and from which they can be retrieved or removed. Most retrieval/removal operations on a dataspace (or tuple-space) are based on pattern matching (i.e. "find objects like this…"). A single application may have one or many dataspaces, and each can be regarded as a sharable *model* in the sense of the Model-View Controller pattern/paradigm [25].

The dataspace API has two main parts. The first is a synchronous data storage and query interface which is very similar to an object database, and which was inspired in part by the Hibernate open source Java Object/Relational mapping system. Like Hibernate, and unlike previous versions of the platform, the objects placed in the dataspace do not have to support particular (Java) interfaces or extend particular base classes. In most cases they are simple JavaBeans or "Plain Old Java Objects" (POJOs), and tend to be entirely passive data holders (with no internal threads or overridden methods).

The second part of the dataspace API is an asynchronous pattern-based notification facility, which allows sections of code to register an interest in certain kinds of objects being placed into, changed or removed from the dataspace. This provides facilities comparable to a content-based publish-subscribe event system such as ELVIN [24], but tightly integrated with the dataspace's state management facilities: the "published" events are actually all changes made to the dataspace.

EQUIP2 is written in Java, using language features and classes common to both J2ME (for use on mobile phones) and J2SE (for use on more capable devices and server machines). Consequently it cannot make use of J2SE reflection facilities (which Hibernate does) since these are not present in J2ME, but has to provide its own simple reflection-like facilities (through a system of dynamically loaded "helper" classes). Similarly, it has to provide its own object marshalling framework (J2SE object serialization is again dependent on reflection), which currently supports XML and binary encoding options.

The dataspace API is common across J2ME and J2SE, but different dataspace implementations are available on each platform. These include a transient (in memory) implementation common to both J2ME and J2SE, a RMS-backed persistent implementation for J2ME and a Hibernate-based persistent implementation (over a relational database) for J2SE.

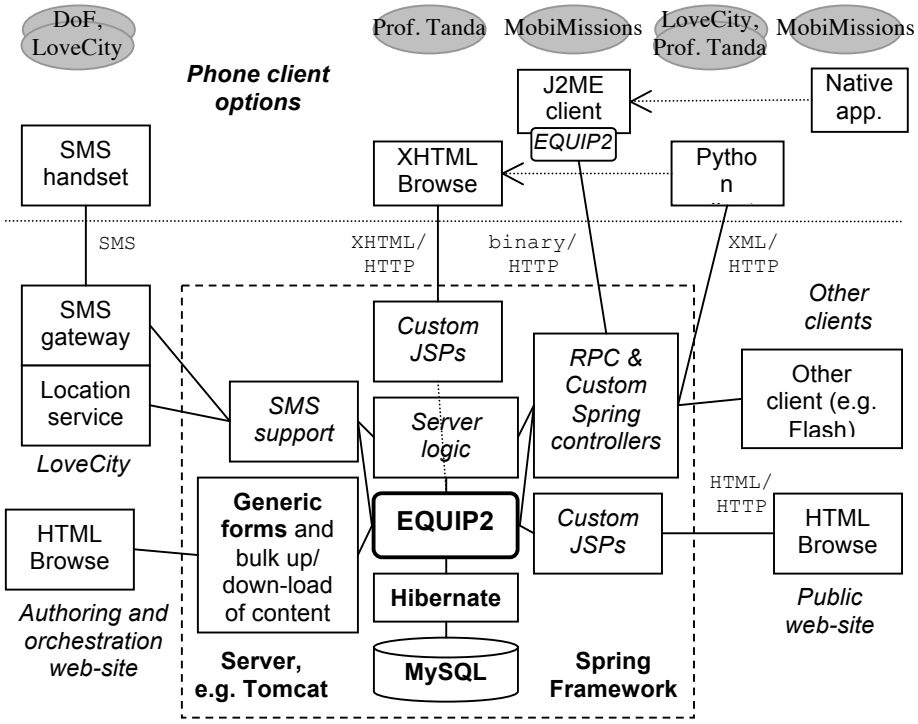

Figure 7. Common and *optional* elements of the EQUIP2 server platform.

All of the experiences presented have an EQUIP2-based server component, which is based on a common template web application for use in a J2EE (Java 2 Enterprise Edition) servlet container such as Apache Tomcat. As illustrated in figure 7, this combines: a persistent EQUIP2 dataspace based on Hibernate over a relational database (in these cases, MySQL); an EQUIP2 Java Server Pages taglib for dataspace access from JSPs; a set of generic form views for browser-based viewing and editing the dataspace content; and forms for bulk XML and binary upload of the dataspace content. This also uses the Java Spring Framework [27] for declarative application composition and web-based MVC support.
The template web application provides a common starting point for each experience, which is then specialized by defining experience-specific data classes, followed by iterative development of experience-specific interfaces (for participants, authors, operators and analysts) and application logic. Client applications can be developed concurrently as server capability matures, making use of EQUIP2 where appropriate (e.g. as in the MobiMissions J2ME client).

5 Strategies

Four ubicomp experiences have been briefly described which employ mobile phones in various ways and the common software platform that has supported and been co-developed with them. From this, the following section identifies four significant strategies for addressing mobile phone diversity in the development of ubicomp experiences: prioritise support for server development, migrate functionality between the server and the handset, support very flexible communication options, and use a loosely coupled (data-driven and component-based) software approach.
In addition to drawing on our own direct experience of development it was also appropriate to survey the five core developers who worked on these four experiences using an optionally anonymous web-based questionnaire. These developers were not primary developers of EQUIP2 itself, and it was made clear to all respondents that an impartial response was desired, and that responses would be treated anonymously and used only for evaluation and planning for future extensions to the software platform.

5.1. Prioritise Strong Server Support

At the present time the only truly mass-scale applications available and used on mobile phones are voice and SMS/text messaging. In both cases all custom application functionality must exist off the handset, normally on one or more “server” machines which together comprise the “engine” for the experience. Even with J2ME or native phone applications many experiences still have a large server element, for example for communication and coordination. The importance of server support is apparent in all four experiences described in this report.
In addition, many experiences of this kind have a web-based component (accessed from regular PCs) in addition to the phone-based component of the experience. This is also true of all four experiences described here (and of others such as [12]). A standard web-based interface such as this can provide richer and more extensive content than the phone interface (e.g. larger screen, various plug-in options), and is also useful “behind the scenes” for management, monitoring, authoring and “orchestration” of the experience (as in [26]).
Consequently it is argued that a platform to support development of experiences delivered on mobile phones should prioritise support for server development.
This has been a major area of development in the EQUIP2 platform, in particular the template web application and associated reusable elements (taglib, forms, Hibernate-based dataspace implementation, SMS support). All of the respondents to our developer survey thought that the use of EQUIP2 made experience monitoring and orchestration faster than it would have been with alternative technologies, and three of the five respondents reported that the generic database forms were one of the best elements of using EQUIP2; these are both server-based elements of the experiences.

5.2. Migrate Functionality from the Server to more Capable Handsets

It is relatively difficult and slow to develop applications on mobile phones, for example due to limited debugging support, inconsistencies and errors in virtual machines and libraries, and slow development cycles (see [17], [18]). Consequently, especially given good support for server development (section 5.1), the opinion could be held that it is a good strategy to develop initial functionality on the server, and then to migrate that functionality to higher-capability mobile phone client applications as required. This also allows

the initial server-based version(s) to be used with less capable handsets. The experiences described illustrate several actual and possible examples of this migration.
In LoveCity there are both SMS and Python phone client options. They both communicate with the same server, which maintains common state and handles multi-user experience elements. However, the Python client supports a richer (graphical) interface, and handles some elements of user interaction (e.g. detail browsing) without needing to communicate with the server. The SMS-only option is the default for public participants, who would otherwise be limited to Nokia Series 60 phones and installing a substantial application.
In MobiMissions all possible user interaction has been migrated to the J2ME phone client. This client application is built around a persistent EQUIP2 dataspace running on the phone, which caches information from the server (missions) to minimize communication with the server. The phone dataspace also accumulates new information and images (captured with the phone camera) as the user performs those missions, which can be uploaded to the server at a later time. The developer who worked on the phone client for MobiMissions rated the use of EQUIP2 for this as faster, less complicated and more flexible than the likely alternative(s) (in this case, J2ME without EQUIP2).
In Prof. Tanda the trigger application represents a migration of one key function to the handset: monitoring certain aspects of user context (time and cell ID) in order to trigger a new interactive session. At present other phone interactivity is provided by the server-generated XHTML pages, however the server implementation of this uses Java over the EQUIP2 API and is designed so that in a future version this could be run directly within a J2ME MIDlet on the phone, similar to MobiMissions.
The same approach allows (non-UI) code intended for a phone-based (J2ME) application to be initially developed and tested within desktop J2SE applications and test cases.

5.3. Support Very Flexible Communication

A software platform to support ubicomp experience development for mobile phones needs to support very flexible communication in at least three respects.
First, any server(s) may need to communicate with different capabilities of phone client (SMS, WAP/Web, J2ME or native application). For example, the LoveCity server supports both handsets using SMS only and handsets running a Python client which communicates via XML RPCs (Remote Procedure Calls) over HTTP.
Second, in the case of custom applications running on the handset, it must be possible to carefully structure and optimize the communication between that client and the server. There are several reasons for this: communication with mobile phones is relatively high latency and low bandwidth; it may be expensive for the user; and there may be repeated or extended periods when a connection cannot be obtained or maintained. All of these can have a profound impact on the user's experience, which may need to be designed to explicitly account for them and reflect them to the user.
Third, the server may also need to communicate with diverse other clients in addition to mobile phones. For example, both DoF and LoveCity include public visualization interfaces implemented in Adobe Flash, and DoF also has the board augmentation client (figure 2) which is written in C++.
As a result there will be no single best solution for communication in these kinds of experiences. Consequently, a supporting platform should provide help with realizing and integrating communication options, rather than a specific solution (or a closed set of solutions).
EQUIP2 reflects this because, unlike its predecessor, it has no specific built-in communication mechanism. Instead it provides a number of supporting facilities that can be assembled and tailored in different ways.
For example, in DoF there are a set of data classes and supporting web interfaces for linking to commercial SMS gateways which have been reused in LoveCity. DoF and LoveCity both use EQUIP2's object marshalling framework with a simple generic (reflection-based) RPC server skeleton to support interaction with the Flash-based visualizations already mentioned.
In MobiMissions the protocol used between the J2ME phone client and the server again exploits the marshalling framework (this time the binary encoding, for performance). However the protocol has been carefully crafted in tandem with the user interface and interaction. For example, it is relatively common for the upload of a completed mission to fail part way through. If the user then goes to the server web site they may see the mission as completed, or not, depending on whether the failure occurred during the final acknowledgement phase of the upload. Consequently the protocol, data model, user views and user interaction all reflect the fact that a mission upload may have succeeded, definitely failed, or be in an unknown state.
All respondents to our developer survey rated EQUIP2 as making networking development more or much more flexible compared to alternative technologies they might have used, and two respondents rated it as

much less complicated and much faster than the alternatives (of the other two respondents to these questions one rated both neutrally and one – who worked on with the very first release of EQUIP2 – more complicated/slower). One respondent reported speed of networking development as a main reason for using EQUIP2 in the future, while one respondent suggested *integrated* flexible network support as a key future development.

5.4. Use a Loosely Coupled (Data-driven and Component-based) Software Approach

The last strategy found it to be effective is a software system design approach which is data-driven and component-based, creating a loosely coupled and flexible system.
Section 4 explains how each dataspace can be regarded as a model in the model-view-controller pattern/paradigm. The dataspace(s) which underlie an application can then serve as the linking and integration points between the various sub-components which comprise the whole application.
For example, the SMS "component" from DoF effectively forms a link between a dataspace and an external SMS gateway. However it does not "care" (or need any re-coding to deal with) how the data objects corresponding to incoming and outgoing text messages are processed or generated within the rest of the application – it just adds a new received message object to the dataspace when a SMS arrives, and requests the sending of a new message by the SMS gateway when a new message request object appears in the dataspace.
In the same way, communication between application dataspaces can be decoupled from most interaction and view logic. For example, the web application's dataspace forms can be used to monitor, populate and manipulate the dataspace from a browser, while still triggering appropriate application logic. Similarly, communication between the MobiMissions J2ME client and the server can continue concurrently with user interaction involving the same dataspace.
A number of elements contribute to this flexibility. First, there is the dataspace (tuple-space) paradigm itself, and its use of pattern matching, which is a well-established strategy for constructing loosely coupled applications. Second, the integrated notification facility, which is present in a subset of tuple-space systems, provides additional support for decoupled but timely coordination. Indeed, four of the five respondents to our survey identified this as one of the best elements of using EQUIP2. Third, the build process for the template web application encourages the application developer to begin by defining the data types (object classes) to be used in the application. These can then be used as a common "language" for defining and implementing the rest of the application. This was put forward by two of the five respondents as an important element. Fourth, the template web application makes extensive use of the Spring Framework [27], which in turn embodies a particular approach to component-based software development called "dependency injection", which makes it relatively easy to separate and then reuse the components (objects) that make up an application, without sub-classing or re-coding. Fifth, the consistency of the EQUIP2 API across different platforms and implementations makes it easy to create unit tests (e.g. using JUnit) against test dataspaces without the overhead of creating and populating dummy relational databases or J2ME record stores.

6 Ongoing and Future Work

The common software platform described is still under active development, and being used in other projects and activities. It is freely available under the "new" BSD open source license. One area of ongoing work is support for use in languages other than Java. Currently there is an experimental auto-translated C++ version which was used by the DoF board augmentation client.
From the initial developer feedback reported here priorities for future development are: support for scripting within EQUIP2 applications; more "standard" networking and distribution options; and simpler ways to get started using EQUIP2 and the web application template (which was reported as having a steep learning curve). There are plans to develop support for "web 2.0" interfaces (such as blogs), while the review in section 2 suggests that support for integrating voice telephony with the server might be a promising direction for some classes of ubicomp experience.

7 Acknowledgements.

8 References

1 TimesOnline, “Mobile madness consumes the UK”, 21 May 2006, http://technology.timesonline.co.uk/article/0,,19510-2189680.html (verified 2007-03-05)
2 Ofcom (Office of Communications), “The Communications Market 2006”, 10 August 2006, http://www.ofcom.org.uk/research/cm/cm06/main.pdf (verified 2007-03-05)
3 C. Floerkemeier and F. Mattern, “Smart Playing Cards – Enhancing the Gaming Experience with RFID”, in Proceedings of PerGames 2006, Duplin, pp. 27-36.
4 Cheok, A.D.; Anuroop Sreekumar; Lei, C.; Thang, L.N., “Capture the flag: mixed-reality social gaming with smart phones**”,** Pervasive Computing, IEEE, Volume 5, Issue 2, April-June 2006 Page(s):62 – 69
5 Mika Raento, Antti Oulasvirta, Renaud Petit, Hannu Toivonen, “ContextPhone - A prototyping platform for context-aware mobile applications”, *IEEE Pervasive Computing*, 4 (2): 51-59, 2005.
6 C. Frank, P. Bolliger, C. Roduner, W. Kellerer, “Objects Calling Home: Locating Objects Uising Mobile Phones”, To appear in proc. 5th Intl. Conference on Pervasive Computing (Pervasive 2007), Toronto, Canada, May 13-16, 2007.
7 T. Sohn, K.A. Li, G. Lee, I. Smith, J. Scott, W.G. Griswold, “Place-Its: A Study of Location-Based Reminders on Mobile Phones”, Proc. Ubicomp 2005, Springer-Verlag (2005), 232-250.
8 Ballagas, R.; Borchers, J.; Rohs, M.; Sheridan, J.G., “The smart phone: a ubiquitous input device”, Pervasive Computing, IEEE, Volume 5, Issue 1, Jan.-March 2006 Page(s):70 - 77
9 Springwise, “Phones for Boomers and Their Parents”, 23rd May 2006, http://www.springwise.com/telecom_mobile/phone_for_boomers_their_parent/ (verified 2007-03-05)
10 I. Smith, S. Consolvo, A. Lamarca, “The Drop: Pragmatic Problems in the Design of a Compelling, Pervasive Game”, ACM Computers in Entertainment, Vol. 3, No. 3, July 2005, Article 4C, pp. 1-14.
11 M:Metrics, “M:METRICS UNVEILS INDUSTRY'S FIRST DEFINITIVE MOBILE MARKETING METRICS”, 3rd October 2006, http://www.mmetrics.com/press/PressRelease.aspx?article=20061003-sms-shorttext (verified 2007-03-05)
12 Jegers, K., Wiberg, M., “Pervasive gaming in the everyday world”, Pervasive Computing, IEEE, Volume: 5, Issue: 1, Jan.-March 2006, page(s): 78- 85
13 Kumar, V.; Parimi, S.; Agrawal, D.P., “WAP: present and future”, Pervasive Computing, IEEE, Volume 2, Issue 1, Jan-Mar 2003 Page(s):79 - 83
14 “CONQWEST”, http://homepages.nyu.edu/~dc788/conqwest/ (verified 2007-03-06)
15 Rashid, O., Mullins, I., Coulton, P., and Edwards, R. 2006. “Extending cyberspace: location based games using cellular phones.” *Comput. Entertain.* 4, 1 (Jan. 2006), 4.
16 Sun Microsystems, “Sun in Telecom”, 2004, http://www.sun.com/aboutsun/media/presskits/ctia2005/Sun_Telecom_External_V2.pdf (verified 2007-03-06)
17 M. Huebscher, N. Pryce, N. Dulay, P. Thompson, “Issues in developing ubicomp applications on Symbian phones”, Future Mobile Computing Applications, 2006. FUMCA '06. International Workshop on System Support for, Sept. 2006, page(s): 51-56, Irvine, CA, ISBN: 0-7695-2729-9
18 Coulton, P., Rashid, O., Edwards, R., and Thompson, R. 2005, “Creating entertainment applications for cellular phones.” *Comput. Entertain.* 3, 3 (Jul. 2005), 3-3.
19 M:Metrics, “CAPTURED BY CAMERA PHONES”, http://www.mmetrics.com/press/PressRelease.aspx?article=20060807-photo-messaging (verified 2007-03-06)
20 R. Suomela, E. Räsänen, A. Koivisto, J. Mattila, “Open-Source Game Development with the Multi-user Publishing Environment (MUPE) Application Platform”, Entertainment Computing – ICEC 2004, pp. 308-320, Springer.
21 A. Drozd, S. Benford, N. Tandavanitj, M. Wright, A. Chamberlain, “Hitchers: Designing for Cellular Positioning**”,** Ubicomp 2006, pp. 279-296
22 Hightower, J., Consolvo, S., LaMarca, A., Smith, I., & Hughes, J., “Learning and Recognizing the Places We Go”, *Ubicomp 2005*, Tokyo, Japan. September 2005.
23 D. Gelernter, “Generative Communication in Linda”, ACM Transactions on Programming Languages and Systems, Volume 7, Number 1, January 1985, pages 80-112.
24 Bill Segall, David Arnold, Julian Boot, Michael Henderson and Ted Phelps, “Content Based Routing with Elvin4,” Proceedings AUUG2K, Canberra, Australia, June 2000.
25 Trygve Reenskaug, “The Model-View-Controller (MVC). Its Past and Present”, JavaZONE, Oslo, 2003, http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf (verified 2007-03-06)
26 Crabtree, A., Benford, S., Rodden, T., Greenhalgh, C., Flintham, M., Anastasi, R., Drozd, A., Adams, M., Row-Farr, J., Tandavanitj, N., and Steed, A., “Orchestrating a mixed reality game 'on the ground'”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM Press, New York, NY, 391-398.
27 “Spring Framework”, http://www.springframework.org/ (verified 2007-03-06)

ABOUT PARTICIPATE

Participate explores convergence in pervasive, online and broadcast media to create new kinds of mass-participatory events in which a broad cross-section of the public contributes to, as well as accesses, contextual content - on the move, in public places, at school and at home.

Participate is a three year collaborative Research and Development project, supported through theTechnology Programme with grant funding from the Department of Trade and Industry (DTI) and the Engineering and Physical Sciences Research Council (EPSRC).

Our consortium blends expertise in online services, pervasive computing, broadcast media, sensors, event design and management, and education. Our partners are BT , Microsoft Research Cambridge, BBC, Blast Theory, ScienceScope, University of Nottingham and the University of Bath.

For more information on Participate please visit:
http://www.participateonline.co.uk/

For more information on the Technology Programme and EPSRC please visit:
http://www.dti.gov.uk/innovation/techprioritiesuk/about_the_programme/index.html
http://www.epsrc.ac.uk/

BLAST THEORY BBC Creative R&D Microsoft Research BT

UNIVERSITY OF BATH The University of Nottingham science scope

dti EPSRC Engineering and Physical Sciences Research Council