Technical Pre-study for the ExMS project

Fredrik Bromée, HUMLE laboratory, fbr@sics.se

Supervisor: Per Persson, HUMLE, perp@sics.se

2001-03-19

SICS Technical Report T2001:03

ISSN 1100-3154

ISRN: SICS-T-- 2001:03

SWEDISH INSTITUTE OF COMPUTER SCIENCE

SICS

Abstract

This report aims to give an overview of software and hardware platforms available now or in the near future for building a prototype of an ExMS application (for an overview of the ExMS project, see Appendix). The report also gives an overview of the different technologies for building third-party mobile client software applications that are in use today.

The report is composed of three sections. The first section is a general discussion on mobile client software and the different technologies that can be used to develop third-party mobile client software. The next section continues with a specific discussion on ExMS and answers the following questions: What is the general architecture of the ExMS application? What alternatives exist for implementing the ExMS prototype? The final section of the report is a recommendation of hardware and software platform for building the ExMS prototype.

Recommended reading:

- If you are interested only in the recommendation for the ExMS prototype, read section 3.
- If you are interested in the different alternatives for the ExMS prototype, read from section 2 onwards.
- If you want more background information on any of the technical terms read from section 1 onwards.

Revision history

Date	Version	Author	Comments
2001-03-06	0	Fredrik Bromee	Document created
2001-03-09	1	Fredrik Bromee	First version finished

Table of Contents

1 Technologies for mobile client software development

When developing a third-party application for a mobile client, three things that have to be evaluated for a target platform are these:

What hardware platform do we want the application run on? Which protocols shall be used to access the network? What software platforms exist for third-party applications? This section will try to list different answers to these questions in order.

1.1 Hardware platforms

This is a list of features that are critical for the hardware platform. If a platform does not have all of these features, it cannot be a candidate for the ExMS prototype:

- Possibility to add third-party applications
- Network access (for telephony of course, but also some kind of data access for the delivery of the ExMS)
- Some graphics capability for the ExMS animations
- Some persistent memory (to store ExMS configuration and messages)

Apart from these features good CPU and main memory resources are also desired but not vital.

1.1.1 GSM phone

Unfortunately most GSM phones of today have limited, if any, capabilities for third-party applications since they run on proprietary locked operating systems. This makes the normal GSM phone a virtually impossible choice for an application prototype with more than trivial features.

1.1.2 Personal Digital Assistant (PDA)

A modern PDA has all of the features above but one - the network access. There are different ways to add this feature. Some PDAs connect to a separate mobile phone or computer via serial or infrared interface, while others have PC-card slots for GSM phonecards or wireless-LAN. PDAs have excellent CPU and main memory resources – for instance the Compaq Ipaq 3630 has a stunning 206 MHz Strongarm processor, not far behind a stationary computer.

1.1.3 J2ME phone

This is a normal GSM phone with a ‘sandbox’ for third-party applications written in the Java language compliant with the Java 2 Micro Edition or the PersonalJava specification (see following sections). Since the unability to add third-party applications was the only disadvantage noted with the GSM phone, this seems to be an attractive alternative. One question remains, though. It is availability – even though Java phones have been commercially available for

months in Asia we have yet to see a product reach the market in Europe. Motorola and Nokia both claim to release products within the next few months.

1.2 Network access protocols

How can the mobile client connect to a network to fetch data and communicate with other clients? This is a list of the most common protocols:

- SMS (Short Message Service) - can be used to send text messages to GSM phones. Pushes data to the client. If the client is not online, the SMS server stores the message and delivers it when the client goes online.
- MMS (Multimedia Messaging Service) – successor to SMS which can contain video clips, sound etc. No products support this yet.
- TCP/IP - this is the network protocol of the Internet, used by web browsers, e-mail applications, news servers, and all applications that access the Internet. When you use a GSM phone as a modem to open a data connection you can access the TCP/IP protocol stack.
- HTTP (HyperText Transfer Protocol) - stateless protocol used for fetching web pages. It has been tweaked to support different functionality. Why? Because it is let through most firewalls and is thus available from most locations. Runs generally on top of TCP/IP.
- WAP (Wireless Application Protocol) – stateless protocol used for fetching WML pages to wireless clients.
- GPRS (General Packet Radio Service) – brings immediate data access to the GSM phone.

1.3 Software platforms

There are two basic types of software platforms: the native platforms and the different Java platforms. When using a native platform the application code is compiled into machine code specific for the processor and operating system of the target device. The Java approach is instead to use a virtual machine (VM) that interprets standardized *byte code* specific for the VM. The basic goal of VM technology is to provide application software with an interface that stays constant regardless of the processor and the operating system on the target device. In this architecture, the application code is isolated from the target device and its particular requirements.

There are advantages and disadvantages to both approaches. The most obvious advantage of the native approach is performance – since the virtual machine always adds an extra layer of abstraction on top of the physical machine, native code will always run faster than equivalent interpreted VM code. The biggest advantage of the VM approach is portability.

1.3.1 Java – specifications and runtime environments

Java is a program language that uses a virtual machine and the VM interprets bytecode. So why are Java programs written for different VMs similar? Because they use the same syntax, and also to a certain extent the same APIs. Why are not Java programs written for different VMs exactly the same? Because the

APIs that the VM support are not exactly the same. (some graphics library may be missing, for instance).

Why not use the same JVM that I have on my computer on my cell phone? Because the J2SE JVM is too heavy. A modern JVM needs several Mb of RAM and around even more Mb of persistent memory to work properly. The idea is to use a slimmed-down version of a standard JVM that supports a subset of the APIs that the standard JVM supports. This means that you do not have to port much of your application to make it run on different target platform.

1.3.1.1 J2ME – Java 2 Micro Edition

J2ME is a collection of specifications aimed for the embedded and handheld market. The different J2ME specifications are built up using 'Configurations' and 'Profiles'. Roughly speaking, configurations define runtime environments and profiles define the API scope.

The Connected Limited Device Configuration (CLDC) is the first J2ME configuration and uses a very limited version of the JVM called the KVM. It is aimed for devices with limited resources, like mobile phones or set-top smart boxes. One of its corresponding profiles is the Mobile Information Device Profile (MIDP). The CLDC-MIDP is the profile implemented by the coming Motorola Java phones.

The Connected Device Configuration is expected to be next configuration to be released. Combined with the also-to-come Personal profile it will be aimed for more potent devices like PDAs and feature a full JVM implementation.

More information:

http://java.sun.com/j2me/

1.3.1.2 PersonalJava

PersonalJava is another specification from Sun. Sun also provides beta reference implementations of this specification for some target platforms, like Windows CE. No further work is being done on this specification and Sun has announced that this specification will be replaced with the Personal profile of J2ME.

More information:

http://java.sun.com/products/personaljava/index.html - general info

http://developer.java.sun.com/developer/earlyAccess/personaljava/ - download of runtime environment

1.3.1.3 J9 – Visual Age Micro Edition (VAME)

J9 is a virtual machine from IBM that has been ported to several embedded platforms, including WinCE and PalmOS. It has a reputation of being fast and very stable. GUI support is at the moment restricted to the IBM proprietary GUI API called MicroView. This is not compliant with the AWT from J2SE or any other GUI library but has been ported to all target platforms that J9 has been ported to. Beta support has been announced for the CLDC configuration of

J2ME, and IBM claim that VAME will soon support more parts of the J2ME. J9 comes as a part of IBM Visual Age Micro Edition (VAME).

More information:

http://www.embedded.oti.com/ - general site for VAME

news: news.software.ibm.com – IBM's VAME newsgroup

1.3.1.4 Waba

Waba is a virtual machine ported to some embedded platforms. It can interpret a subset of the J2SE specification. Examples of what cannot be interpreted by the Waba VM is threading and exception handling. More information can be found at:

http://www.wabasoft.com

1.3.1.5 Jbed

JBed is a virtual machine from Esmertec. It comes with different APIs compliant with PersonalJava or the J2ME CLDC. It is however limited to PalmOS only at the moment. More information:

http://www.esmertec.com/p.html

1.3.1.6 JMS – A helper API

Java Message Service (JMS) is a helper API which is not part of the J2ME specification but it is mentioned here because it would suit the ExMS application very well. JMS could be used to mimic the store-and-forward functionality of SMS and would make it very easy to write a messaging application. A company called Softwired claims to have an implementation of JMS for mobile devices, the iBus Mobile. However due to shortage of time no further investigation has been made to find out exactly what they mean by this.

More information:

http://www.softwired-inc.com/

1.3.2 Native software platforms

1.3.2.1 PalmOS

PalmOS has good support for third-party application development with a relatively large body of developers and a good emulator, the POSE.

1.3.2.2 Microsoft Embedded (Windows CE)

Windows CE is the operating system shipped with the PocketPC devices. It has good support for third-party applications written in languages like C, C++ or VisualBasic.

More information:

- http://www.microsoft.com/windows/embedded/default.asp

1.3.2.3 Symbian (Crystal, Quartz, etc)

Symbian is an operating system for PDAs and advanced phones. It is being developed by the organisation with the same name – Symbian – that was founded and is still owned by a number of telephony companies, Nokia and Ericsson included. The OS comes in several different flavors (Qrystal, Quartz, etc) and provides some possibilities for third-party applications. The to-be-released Nokia Communicator 9210 is built on Symbian Crystal and provides a native C++ API as well as a Java API based on the PersonalJava specification.

More information:

- http://www.symbian.com
- http://www.symbiandevnet.com - developer's perspective
- http://forum.nokia.com/symbianforum/main/1,6668,1_49_1,00.html - Nokia 9210 specific

2 ExMS prototype

Before going into the discussion of the architecture and the different alternative platforms let us review the different abstraction levels on which one can choose target platform.

First, one can choose target platform based on *hardware*. Example: The application will run on this processor. Secondly one can choose target platform based on *operating system*, thereby saying that the application will run on all devices that have this operating system. Finally one can choose target platform based on *virtual machine* (VM), thus saying that the application can run on all devices to which this VM has been ported. For each level you win some and you lose some. For each level ‘up’ on the abstraction level you can target more types of devices and the application becomes somewhat easier to develop. At the same time, performance may decrease and native APIs and functionality might not be available.

2.1 Generic ExMS architecture

This section assumes some knowledge about the ExMS project.

From an architectural viewpoint it is rewarding to divide the ExMS application into two parts, one community / configuration part and one composing and sending part. In the community / configuration part users can choose and download skins and moods to their terminals (configuration) and also add user-developed skins to the community. In the composing and sending part users compose and preview their ExMS on their terminals, send them to another user who in turn can ‘play back’ the ExMS on his/her own terminal.

The reason for dividing the application into these two parts is because they can be developed independently from each other and that they do not to a great extent depend on each other. The community / configuration part would be a straightforward web application, even including the part where the user can add skins, since these could be restricted to be bitmap images of a certain size. These bitmap images could be imported via the HTTP put method on to the web server. Bitmapped images means that users could use most any drawing programs to create their own skins.

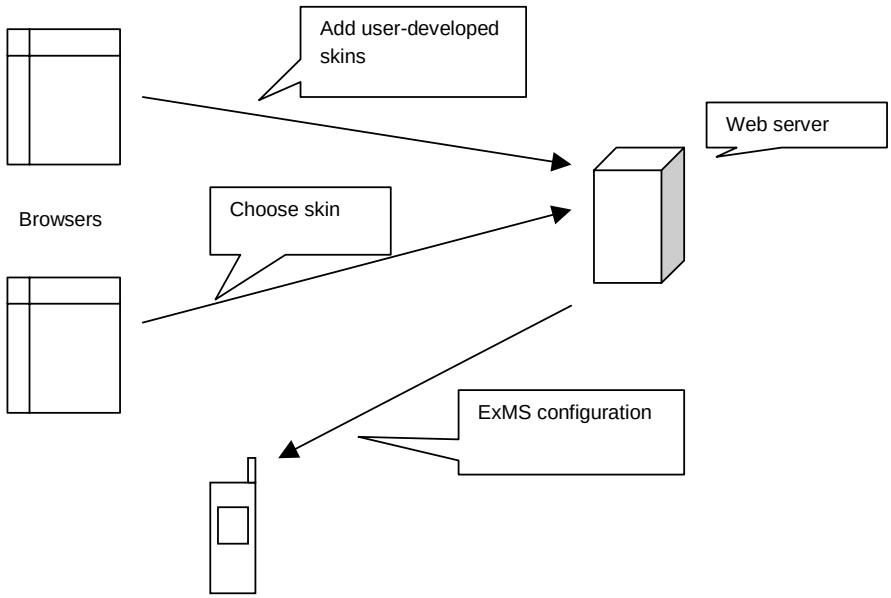

Figure 1. Community / Configuration

2.1.1 Community / Configuration

The only vague part of the community / configuration part is where the ExMS configuration is downloaded to the terminal, the mobile device. How this should be done depends on how the sending part will be implemented. Choices for the configuration download include SMS, HTTP and TCP/IP sockets. It would also be possible to make the whole prototype simpler just by simply cutting functionality. The question is, how important is the community function for the user study? If there would be no functionality for adding user-defined skins, all possible skins (say 20) could be preloaded into each terminal that will participate in the tests. In this case there would be no need for sending raw image data, it could suffice with MoodML scripts both for messages and configuration, with image number id:s referring to the actual images. However, as soon as this functionality is needed (users adding their own skins), the need for sending raw image data will arise. It is a question about what is important for the user study. Please note however that even if the community / configuration part does not pose any greater technical problems it will take some time to develop given all functions needed.

Recap: Community / configuration part is not technically difficult to implement, except for the downloading of the ExMS configuration to the mobile device. How this should be done depends on what mobile device will be chosen and how ExMSs will be sent and delivered.

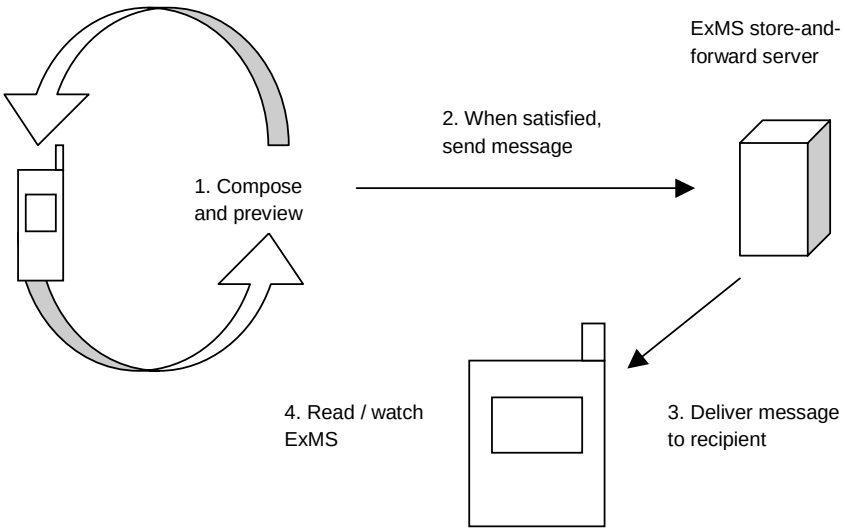

Figure 2. Composing and sending

2.1.2 Sending

The sending part of the ExMS application is where the technical challenges of this project lie. These are the desired characteristics that we want from the ExMS application's sending part:

- Graphical display to compose, preview and watch ExMS animations on.
  - One of the most challenging obstacles of the ExMS project is to create an interface that is easy enough to use for the clients - it has to be as easy (rather easier) to use as normal SMS. This implies that we should choose platform based on the GUI capabilities. Is it easy to create GUIS on this platform? Is it easy to change the GUI for this platform? Could we develop several different GUIs for the users to try? Requirements for animators depend on what device is chosen and cannot be specified here. Since animations should be kept at a few frames a piece, no standard is needed for these. Connected bitmapped images with timing information should be enough.
- One device.
  - It would be best if this device would be the participant's normal GSM phone, but this is unfortunately not possible. The second best option would be a device that can replace the participant's normal GSM phone, which provides the telephony and SMS capability of the old device plus the new functionality of the ExMS application.
- Low cost
  - Money is always an issue and if the devices are cheap one can do a larger user study.
- Asynchronicity of SMS
  - An important characteristic of SMS is that it is asynchronous, that the data is *pushed* to the client. It is vital that the ExMS prototype keeps this characteristic, so that it at least appears as if data is pushed to the client.

2.1.2.1 ExMS Carrier

The ExMS carrier is the network access protocol used for:

1 Sending ExMS from the device.

2 Delivering the ExMS to the recipient device.

3 Notifiying the recipient that a new ExMS has arrived.

Implicitly the ExMS carrier is also used for downloading ExMS configurations. Please note that whichever carrier is chosen, MoodML, is always used for the encoding of the messages. There are three different options for ExMS carriers: SMS, HTTP or similar, and TCP/IP sockets. These can be combined in different ways to deal with all the the three steps above. See the following sections for a discussion on a few possible combinations suitable for the ExMS prototype.

2.1.2.2 SMS for sending, delivering and notification

In this option SMS is used for both sending, delivery and notification. Advantages with this option is that there would be no need to develop an ExMS store-and-forward server, since we would use the GSM network's own SMS server for this. There are a few disadvantages with this approach, though. One is that the SMS functionality may not be accessible for third-party developers, and even if there exists an API for SMS functionality, using this API would bind the prototype to this platform only. There is also the matter of limited payload. SMS are limited to 160 characters a piece and this is not enough for an ExMS MoodML message. Concatenating SMS solves the problem, but only if the community option of adding skins is removed. If this option is not removed there has to be some way of sending skins to the client, i.e. sending bitmapped images. To send these over SMS would require many concatenated SMS, as well as some encoding algorithm.

2.1.2.3 SMS for notification, HTTP or similar for sending and delivery

This option mimics the asynchronicity of SMS by using an SMS to notify the user that he/she has received an ExMS. When the user tries to read the ExMS the application makes an HTTP request to the ExMS server and displays the result to the user. This means that a data connection has to be open when the user tries to read an ExMS, and in practice it means that the device has to be GPRS since waiting for a normal GSM data connection would probably be unacceptable. With this option there would be no problem sending new skins, since there would be no payload restrictions. Same goes for the configuration download, which would also be a HTTP request.

2.1.2.4 TCP/IP Sockets for notification, HTTP or similar for sending and delivery

This third option has the 'push' characteristic of SMS, but puts high demands on the mobile device since it would need to be GPRS and also multi-threaded. If the mobile device is truly multithreaded an ExMS application could run in the background listening on a TCP/IP socket or similar socket if one exists on the platform. This implies that a data connection would always be open when the application is running. Unless it is a GPRS device this would mean that the device is blocked from incoming calls, not to mention the cost of keeping a

connection alive. So, this option is only feasible if the multi-threaded mobile device is also a GPRS device. Another thing to think about if choosing this way is that it is said that some GPRS networks do not allow mobile terminated data, i.e. that data is pushed to the client. The reasons for this being are two: billing, and lack of IP-addresses. Since the billing plan is to charge per megabyte sent or received from the terminal someone 'spamming' a terminal with unwanted data would generate unwanted costs for the client. The lack of IP-addresses arises since each terminal would be assigned a static IP-address. IPv6 would solve the address problem. The only GPRS operator in Sweden (Europolitan) **does** in fact support mobile terminated data. The only reservation is that the user opens a GPRS session first, but after that the data connection session is open until you close it or turn the phone off.

2.2 Alternatives

These are the hardware and software alternatives that have been found reasonable or possible to implement the ExMS prototype on.

2.2.1 GSM phone

Comments: This first option is not possible to implement as a prototype but rather to give an idea how this application would be implemented in a production environment.

Hardware platform	GSM phone
Software platform	Proprietary or open
Network access protocol:	
Sending / Delivery	MMS
Notification	MMS
Configuration download	MMS
Benefits	One device
Disadvantages	Device does not exist

2.2.2 PDA combined with PC-Card GSM or GPRS phone

Hardware platform	PDA combined with PC-card GSM or GPRS phone
Software platform	WinCE **or** J2ME **or** PersonalJava
Network access protocol:	
Sending / Delivery	SMS **or** HTTP
Notification	SMS **or** HTTP
Configuration download	SMS **or** HTTP
Benefits	One device, good display, extremely good processing power
Disadvantages	Device a bit clumsy, expensive

2.2.3 GSM J2ME phone

Hardware platform	GSM **or** GPRS J2ME phone
Software platform	J2ME **or** PersonalJava
Network access protocol:	
Sending / Delivery	HTTP
Notification	SMS **or** HTTP
Configuration download	HTTP
Benefits	One device, true telephony device
Disadvantages	

2.2.4 PDA with separate GPRS or GSM phone

Hardware platform	PDA connected to GSM/GPRS phone
Software platform	WinCE **or** PalmOS **or** Symbian **or** J2ME **or** PersonalJava
Network access protocol:	
Sending / Delivery	SMS **or** HTTP
Notification	SMS
Configuration download	SMS **or** HTTP
Benefits	Relatively stable technologies, products on market now
Disadvantages	Two devices

2.3 Product availability

These are examples of the products mentioned above. Cost per unit in SEK, VAT excluded at Dustin (www.dustin.se) for products available now. Estimated availability by vendor otherwise.

Description	**Product**	**Price SEK**	**Available**
PDA w/ PC-card slot	Compaq's IPAQ 3630 + expansion pack for PC-card.	6 000	Yes
PC-Card GSM phone	Nokia CardPhone v2.0	3 000	Yes
PC-Card GSM/GPRS phone	Xircom GPRS CreditCard		2Q 2001
J2ME GSM phone	Nokia Communicator 9210		2001
J2ME GSM/GPRS phone	Motorola Accompli 008		1Q 2001

GSM/GPRS phone	Motorola Timeport	4 000	Yes

3 Recommendation

Prototyping for the future

It is important to keep in mind that the ExMS prototype is exactly this – a prototype and not a commercial application. Even if the technical aspects are not the main objective of the ExMS project this is a real opportunity for the participants to learn and benefit from the prototype. These are arguments that clearly speak for the device-independent approach of Java and against using native APIs that binds the prototype to a specific device.

Device unavailability

Much will be decided in the nearest months. What device will come to the market? What will be their exact features? If the Motorola Accompli 008 is released first quarter 2001 as quoted on their website it would be very interesting to use it with its J2ME MIDP support. If it proves to be multithreaded it would be the ideal device, however the chances for this are slim. Another alternative would be the Communicator 9210. A PDA with GPRS card phone would also be interesting indeed. However, if you can afford to wait a few months it is best just to wait and see but not stay inactive.

Commit to runtime environment

Commit to a technology that will fit on most platforms, with little changes. Java. With its large developer's base and its huge industry support it is bound to succeed, even if it might not be the best technical solution. In one way, Java would be a 'safe' way. Choose Java now and you're sure not to be wrong, at least. So, conclusion would be to choose Java and work with an emulator on the device that seems most promising. Try to keep application independent from target platform. Commit to platform sometime in summer.

Tough choice

OK, so Java it is. Then what runtime environment? If J9 had supported some other GUI library than its own MicroView it would have undoubtedly been a first choice. Now, it is not so certain. Using MicroView binds you to J9, which binds you hardware platforms that J9 is ported to. It would be much more attractive to use J9 as a development platform if they provided support for one of the J2ME profiles like Personal, Foundation or MID. They have beta support for CLDC right now and claim that they are working on other parts of the J2ME specification, so this wish might be answered.

So, how about PersonalJava? Developing against this option would keep the door open for both the PDA + GPRS as well as the Nokia 9210. Both of these options provide a rich graphical interface and are reasonably attractive solutions.

The MIDP is equally attractive, with the Accompli 008 as target platform. Good emulators are supplied both from Sun and Motorola until the device is released.

Whatever road taken, care should be taken to keep the GUI as detached as possible from the rest of the application in order to make it as easy as possible to port the application to a different runtime environment if this becomes necessary. This would also make it possible to develop several GUIs to see how well they work.

Alternative

If Java would not be an alternative the recommendation would be to use the PDA + GSM cardphone, specifically the Ipaq. The one disadvantage of this is the tight link to operating system and the device (both the PDA and the PC-card) that would make the application. Hard to port the application, hard to learn from and benefit from it.

Summary

The recommendation is to develop the ExMS application according to one of the J2ME specifications, either the PersonalJava specification or the MID profile. Which one depends on which device is the most attractive, the Ipaq or the Nokia 9210 for PersonalJava, or the Accompli 008 for the MIDP.

4 References

Please use the links below as a help to learn more about these technologies.

J2ME

- http://java.sun.com/j2me/
- http://developer.java.sun.com/developer/products/j2me/index.html
- http://www.microjava.com/
- http://developer.java.sun.com/servlet/SessionServlet?url=http://developer.java.sun.com/developer/earlyAccess/personaljava/
- http://www.onjava.com/wireless/

Other Java runtime environments

- http://www.embedded.oti.com/ - J9
- http://www.wabasoft.com - Waba
- http://www.esmertec.com/p.html - JBed

JMS

- http://www.softwired-inc.com/

GPRS and GSM card phones

- http://forum.nokia.com/cardphoneforum/main/1,6668,1_6_2_1,00.html
- http://www.mobilegprs.com/developers.asp
- http://www.xircom.com/ - GPRS Creditcard
- http://www.wavecom.com/home.html - Wismo 2c-2 GPRS module
- http://www.option.com/ - Triband GPRS pc-card

Windows CE

- http://www.microsoft.com/windows/embedded/default.asp

PalmOS

- http://www.palmos.com/dev/

Symbian

- http://www.symbiandevnet.com
- http://forum.nokia.com/symbianforum/main/1,6668,1_49_1,00.html

5 Appendix: Project Plan Expressive SMS (ExMS)

Stockholm & Helsinki

January, 2001

Per Persson, Jussi Karlgren, Turkka Keinonen & Panu Korhonen

5.1 Introduction

In the beginning of February 2000, the HUMLE laboratory initiated a small project with Nokia Research Center (Helsinki) called *MobiPal*. The objective was to investigate and develop some ideas about avatar-based expressive messaging in a mobile context, both from a technical as well as user perspective point of view. The project resulted in a conceptual framework for development of a prototype together with a mock-up demonstrator to illustrate the ideas. The project has been presented to Nokia's business development departments, and a paper is currently under review for scientific publication.

5.2 SMS, avatars and emoticons

In some parts of the world, mobile phones are everywhere and used by everyone. Phones are not only used for synchronous and asynchronous voice communication, but also – increasingly – for written communication. In spite of complex and non-intuitive interfaces, Short Message Service (SMS) messages are sent and received by a huge number of customers, mostly young, in many European countries. In Finland, the mobile phone users sent in total 650 million text messages in 1999 (Ministry of Communication, Finland, 1999).

SMS differs from voice communication in many respect, which has generated a fair amount of interest and research, mostly from an ethnographic perspective. First, it is not primarily used for serious and task-oriented communication, but for expressive, social and emotional functions, e.g. 'how are you doing?' and 'whatsup?' messages. Humor, flirts, gags and play are central objectives of textual messaging. In spite of the low bandwidth (up to 160 characters, typically in low-resolution monochrome), there is a surprisingly high degree of expressivity in SMS communication, due to the systematic reliance on a rich and shared awareness of situation, preferences, sense of humor, and social context between sender and recipient. SMS messages are typically not sent to strangers, but used in peer-to-peer communication between friends, lovers, or family members. In addition, the composition and reception of SMS messages often take place in a collaborative setting with a group of people gathering around the device (Larsson, 2000; Weilenman, 2000). One of the objectives of the *ExMS* project is to build on our current understanding of SMS usage, and deepen it by studies of SMS text content and situational usage. In the continuation project we want to maintain the special features of SMS messaging, but expand it into other areas of interest. The basic research and design questions are: What is expressivity? How it can be enhanced?

Many collaborative multi-user environments allow people to use an *avatar* – a virtual representation of personal presence and personal characteristics. The way people choose to represent themselves in a virtual world is not always straightforwardly predictable from their physical world appearance and character – the virtual world allows users to play with their identities (Turkle, 1995). Our design project allows users to employ animated characters to enhance expressivity of SMS messaging. However, the *synchronous* nature of most virtual avatars makes fine-grained expressiveness difficult since users will

be unable to control the behavior of the avatar online over a simple numeric keyboard. In this project, we explore *asynchronous* avatar usage.

In this respect, our project draws more from *emoticons* than from avatars. Combining a textual message with a semi-imagery representation of a face adds new layers of meaning (e.g. irony), guides the recipients' interpretation of the message, and expresses the sender's emotional state. Creating animated emoticons that move and transform over the temporal flow of the message will merge modalities in similar ways to cartoon and animated film. The efficiency of emoticons also shows that expressivity lies not in the realism of the imagery, but in the combination of modalities, and, again, the rich and shared context of sender and recipient.

5.3 The *MobiPal* demonstrator

We want to allow senders to express a basic emotional undertone in the message through the actions of their avatar. Users start composing a message by selecting from a palette of *moods*, e.g., happy, distressed, angry and busy. This makes the avatar perform some action to reflect the mood in a few seconds of animation, which will then be looped during the length of the message. In parallel, the textual message is shown as a cartoon balloon (Figure 1).

Figure 1. Example screen.

In addition to mood animations, avatars should be able to perform simple *events*, e.g. jump, dance, smile, laugh and weep (possibly connected to sound effects). Events can be added at specific points in the message. In preview mode, the animation can be stopped between the looped mood animations and the user can choose between events in a list. Since all mood and event animations start and stop at a neutral position frame, continuity of movement can be ensured without involving the user. When the sender has inserted event(s) and previewed, the package is sent off.

Events need to be paced to match the written message – timing is central in conveying punch lines or other emotional content. To this end, we have sketched a standard for encoding expressively enhanced messages: the Mood Markup Language or *MoodML*.

Expressivity lies, however, not only in the temporal composition of text, moods and events, but also in the graphical design of the *skins* and movements. Most users will probably make use of prefabricated skins. Some standard skins may be included in the service package. Others may be produced by professional firms for business marketing: just like product placement in films, users can get professional skins for free, if they allow their mobile avatar to wear a *Budweiser* T-shirt in all their messages.

Some users, however, will want to design their own skins using their own graphical software. To this end, open standards and APIs are absolutely essential. No single organization will have the stamina to produce and uphold the interest of the potential user community: the creativity of the users must be tapped into. Thus, it is important to create a community in which sharing scripts and skins are encouraged and awarded.

However, in order to ensure compatibility between skins, scripts, and users, there have to be some minimal requirements for any given skin. For instance, if you make a skin publicly available or start using it yourself, it has to perform a minimum list of mood and events.

Configuring the interface and the characteristics of the skin, archiving message scripts, sharing scripts and skins, community building and other administrative aspects of the system are accessed by the user from a Web interface rather than over the mobile device: the mobile device is used solely for composing, transmission and receipt of messages.

5.4 Research Questions

The research issues we intend to address span over several subject areas. There are non-trivial *system issues*, which are not the focus of the project. We expect to choose a convenient solution over a stable one for the prototype phase. Examples include the continued design and formulation of MoodML, animation issues where realistic image quality must be traded off for real-time performance, design of the interaction dialogue, and trade-offs between competent and lightweight client software.

Some of the *interface issues* cut deeper into the functions of the system, however. We must put serious thought into how we best can support users in constructing *ExMSs*. We must balance the competence of the web-based configuration tool, the mobile client, and the skin development kit. We must take into account the situational context of message composition. And in future releases of the system, we can envision complexity issues in configuring the actions of the interface, if the avatars are allowed some limited form of autonomy to act based on system understanding of the message.

Expressivity and *social issues* are the focus of the current project. We must address a number of relatively vaguely understood notions, and first and foremost gain some provisional understanding of expressivity in a social context. How can embodied interface avatars enhance it? How does expressivity relate to role playing and an individual’s sense of self and personal and social identity? On a social level, we want the expressive messaging functionality to be a shared notion in a community of users. How can we support the growth and development of such a community? And on another level of

expressivity, as regards the message content, we need to understand the relation between discourse flow, gestures and situation.

5.5 References

1. Koskinen, Topi (2000) Mobile Asynchronous Communication: Exploring the Potential for Converged Applications, *The Journal of Personal Technologies*, 4:45-53. [http://www.sics.se/~perp/andraspublikationer/Koskinenr3.pdf]

1. Larsson, C. (2000) *En mobiltelefon är inte bara en mobil telefon. En studie i tonåringars användande av mobiltelefoner*, Magisteruppsats, Department of Informatics, University of Gothenburg.

2. Ling, R. (1999) "We release them little by little": Maturation and gender identity as seen in the use of mobile Telephony, *Telenor FoU report*.

3. Ministry of Transport and Communications, Finland (1999) "Suomen Tekstiviestimarkkinat", Edita, Helsinki.

4. Weilenmann, A. (2000) Negotiating Use: Making Sense of Mobile Technology, *The Journal of Personal Technologies*, no. 4, Springer Verlag.

5. Turkle, S. (1995) *Life on the Screen*. New York: Touchstone.