# Mobile Collaborative Virtual Environments: A Paradigm Shift from Desktop to Mobile Online Communities

**Umer Farooq**
ufarooq@vt.edu

Thesis submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
**Master of Science in Computer Science**

December 13th, 2002
Blacksburg, VA USA

**Keywords:** Mobile Collaborative Virtual Environment (M-CVE), Mobile Education (M-Education), Computer Supported Cooperative Work (CSCW)

# Mobile Collaborative Virtual Environments: A Paradigm Shift from Desktop to Mobile Online Communities

Umer Farooq

(ABSTRACT)

There are myriad examples of virtual communities and environments available for collaborative activities. However, most of these environments are confined to the desktop and thus preclude collaboration while users are on the move. Through a scenario-based design process, this article establishes the importance of mobile collaborative environments that are readily accessible for users on mobile devices. The element of mobile accessibility for collaborative environments renders them ubiquitous—they can be used anywhere and at any time. A working prototype is then presented that has been developed to supplement an existing desktop-based online virtual community. The prototype illustrates a generic, extensible and platform-independent architecture for translating a desktop collaborative environment into a mobile system. Based on the prototype, we also foresee its application for users in fieldwork settings, particularly for learning and educational activities of teachers, students, and peers through collaboration in a distributed environment.

## Acknowledgements

I would like to thank Craig Ganoe for his valuable input on using handheld technology for collaboration. It wouldn’t have been possible to test the application on the actual devices without his assistance. I also want to thank Chandresh Chhatpar for convincing me to take a thesis option. Other sources of guidance were Wendy A. Schafer, Rishi Panday, Dheva Raja, and Con Rodi. I am grateful to my committee for their continuing supervision. Finally, I would like to express my deepest gratitude to my parents for their moral support through the course of my studies.

## TABLE OF CONTENTS

## LIST OF FIGURES

## 1. Introduction

The CHI community has long been interested in collaboration, and one of the considerations to support collaborative activities are online virtual communities (or environments). Virtual communities, or networked communities as referred by Mynatt et al. [1997], are a form of technology-mediated environments to cultivate collaboration and interaction among users, that is, a sense of community. Among others, virtual communities include MASSIVE [Greenhalgh and Benford 1995], Jupiter [Curtis et al. 1995], CAVE [Cruz-Neira, 1992], MOOsburg [Carroll et al. 2000a], etc. Most of these collaborative virtual environments require the user to use the desktop (e.g. MOOsburg), and if not, then be located in a surrounding environment like a physical room (e.g. CAVE). In any case, such environments do not foster users who are statically present at a physical location; rather, they are on the move continuously as a function of time. As mobile computing devices become easier to network and acquire, it becomes practical to integrate them into virtual network communities. In this article, we rationalize the importance of having accessibility to statically located collaborative environments for mobile users, of course assuming that they possess mobile devices such as PDAs and cellular phones as the hardware medium. The motivation for such collaborative facilities for mobile users is established through a scenario-based design process [Rosson and Carroll 2002].

For materializing our vision of anywhere, any time collaborative environments, we have developed a prototype of a mobile collaborative virtual environment (M-CVE)—a CVE that is available on mobile devices—based on an existing online community known as MOOsburg [Carroll et al. 2000a]. MOOsburg can only be accessed from desktop computers and not mobile devices due to issues such as limited bandwidth, incompatible user interfaces, etc. The M-CVE prototype, called MOOsburg++ (analogous with C to C++, denoting additional features), supplements MOOsburg by providing accessibility to the virtual community from internet-enabled mobile devices. The MOOsburg++ prototype affords a subset of the functionality available in MOOsburg. MOOsburg++ not only provides a concrete implementation of our paradigm, but also delineates a reusable, extensible, and platform-independent architecture for collaborative environments in general. The platform-independence facet of this flexible architecture allows users to utilize any device from a range of tiny commodities including cellular phones, pagers, and PDAs.

As wireless technology expands and becomes more affordable for the common user, academic applications of wireless computing will help revolutionize educational computing. Using this very idea as a motivational foundation, we present a possible application domain for MOOsburg++: the use of wireless and mobile technologies in education. This concept encourages distributed peer collaboration over mobile devices and desktop computers to create novel opportunities for discovery and education in the field and community. Mobile education or M-Education is significantly different from existing mobile learning systems in that it leverages its collaborative activities from an existing desktop-based online virtual community (MOOsburg), and thus offers an array of interactions, such as asynchronous and synchronous modes of collaboration with peers, and viewing or changes to persistent data.

This article is structured as follows. Section 2 starts off by presenting a background of MOOsburg, and then emphasizes the importance of mobility in context to CVEs. Section 3 outlines the technology and architecture for MOOsburg++, describes the functionality and what we learned from it. Section 4 discusses the application domain of using wireless and mobile technology for education, and rationalizes how MOOsburg++ may be instrumental in this field. Section 5 presents an overall discussion of this article's contribution and future work, and section 6 ends the article with some conclusions.

## 2. Background and Motivation

### 2.1. MOOsburg

MUDs (Multi-User Domains) and MOOs (MUDs Object-Oriented) are computationally-based environments that provide access to a persistent, online world. MOOs are fundamentally spatial where users may navigate the information structure. Users in a MOO can collaboratively interact with each other, using chat for instance. The underlying database is persistent: users can create, modify and manipulate objects, changing the state of the MOO for subsequent users.

MOOsburg is a community-oriented MOO that models the geography of the town of Blacksburg, Virginia; its intended users are the residents of the town and surrounding area [Carroll et al. 2001]. Thus, MOOsburg is not merely spatial; it is place-based. It is not for the purpose of fantasy-oriented entertainment or informal social activity [Curtis 1992]; it is community-oriented and is often used for educational purposes. MOOsburg provides an interactive map of the virtual community, and a range of collaborative tools that provide access to shared content such as chat, message boards, etc. The MOOsburg infrastructure consists of the following core components (see http://java.cs.vt.edu/dev/linc/sburg/dev/):

1. *Landmarks*: A landmark is a place in the MOO. Landmark is a generic term, with "rooms", "street corners", and "park benches" being possible specific kinds of landmarks. Currently, landmarks show up as small dots on the map. A pop-up menu on the map allows users to create, edit, and move to landmarks. Each landmark has a place-based chat associated with it.

2. *Spaces*: Spaces are landmarks that contain other landmarks. Space is a generic term, with "building" and "park" being possible specific kinds of landmarks. Each space has its own map (and possibly other navigation mechanisms), so entering a space changes the map.

3. *Characters*: Characters represent users in the MOO. Characters have one and only one location within the MOO.

4. *Things*: Things are objects that exist at a landmark that can be manipulated by users at that landmark. Things can typically also be "taken" and then "dropped" at another landmark. Things are collaborative objects such as shared files, web links, whiteboards, etc.

The infrastructure of MOOsburg seeks to preserve the most useful aspects of traditional MOOs, while extending both the client-side user interface and server-side components to take advantage of emerging internet technologies [Carroll et al. 2001]. Like traditional MOOs, the MOOsburg environment consists of a selection of *objects*. Objects in MOOsburg refer to the core components, i.e. landmarks, spaces, characters, and things.

These objects are stored in a persistent database, and can be created/manipulated by users. For example, a user may create a message board at a particular location, and the contents of the message board are preserved persistently every time some user provides an input. MOOsburg supports synchronous and asynchronous interactions, as users can interact with other users in real time, see the preserved effects of past actions, and/or manipulate objects for subsequent users.

MOOsburg is significantly different from traditional MOOs as it provides richer interactions instead of simply relying on text. Traditional MOOs governed that interactions such as navigation and communication be performed through simple input and output text commands. MOOsburg not only provides an interactive map for navigation, but also supports enriched awareness cures and communication mechanisms. In traditional MOOs, users would perhaps request the system for co-located users, whereas in MOOsburg, the system provides automatic visual cues when a person enters or exits a location. The creation, manipulation, and deletion of objects in MOOsburg are also done through graphical interactions.

MOOsburg also reflects a significant departure from traditional MOOs because it is spatial, that is, provides support for well-defined *hierarchies* of places within the MOOsburg environment [Carroll et al. 2001]. Places in MOOsburg are organized in the form of a tree-like structure, starting from the root node. The root node in MOOsburg is a reference point for the locations. The hierarchy can be thought of as containers, hence there exists the concept of spaces and landmarks. The first level under the root comprises spaces and landmarks. The space in this first level may contain more spaces or landmarks (second-level under the root), and so on. Since navigation is performed using a graphical map, vague ideas of distances can be inferred from physical experiences, as MOOsburg models a real town.

The goal of MOOsburg is to enhance community development by supporting improved access to local information and to new kinds of collaborative activities. Three versions of MOOsburg have been developed to date: a classic text-based MOO, a MOO extended to drive a web-browser, and a Java-based system. However, all these versions of MOOsburg leverage its use from a desktop.

## 2.2. Importance of Mobility

The World Wide Web has provided a common platform for users all around the world to interact in. With the increasing number of users of mobile devices, new research issues are spawned, particularly relevant to wireless and distributed collaboration. The increased availability of communication facilities has seen a shift in the nature of mobile computer systems and applications, and hence, it must be taken into consideration that mobile devices behave differently and offer different interaction possibilities depending on the particular context in which the system is being used [Dix et al. 2000]. This research area poses numerous questions that we attempt to answer with our efforts, such as:

• Can user needs in a collaborative environment be leveraged from mobile devices?

• Is there a general architecture for developing collaborative mobile applications?

• How do users interact from different platforms and devices?

In this article, the targeted group of people for M-CVEs are mobile users; users that cannot access desktop computers because they are on the move. In order to effectively realize the significance and necessity of M-CVEs, we need to profile the impacting difference between users working from a desktop and mobile devices. Desktop users are typically in their office or home, where they have an increased sense of awareness and knowledge about the surrounding milieu. This feeling of environmental certainty is further enhanced by accessibility to resources such as documents, people, information, etc. This is precisely the goal of MOOsburg: to support users' awareness of location, other participants, and opportunities for action [Carroll et al. 2001]. And this raises the question: how are all these aspects (awareness, familiarity, resources, etc) sustained for mobile users, and what new opportunities and applications do they create due to mobility?

To answer that question, we need to highlight here the key difference between the mobile and desktop users, namely *context*. While users are on the move, they do not have access to peers or resources. They are in an utterly different contextual situation, because everything within the environment, including itself, has changed. This leads to a state of uncertainty, or *heterogeneity* as referred by Kristoffersen and Ljungberg [1999]. The greater unpredictability of the contextual constraints within which mobile work must take place means that mobile workers have less control over the configuration of their environment, and therefore the way they manage their work [Perry et al. 2001].

In light of the above-mentioned complications encountered by mobile users, one of the major premises of mobile technologies is to remove the bindings between a fixed space and a person's information and communication resources [Perry et al. 2001]. Perry et al. [2001] also argue that by supporting access to these resources wherever they go, the uncertainty associated with the contextual constraints while mobile is removed. Based on these foundations, we present our ideas for enriched mobile collaboration using M-CVEs to provide a context channel for mobile users.

As mobility in collaboration is emerging as a research topic in itself (e.g. Luff and Heath [1998]), it is imperative that researchers in this field explore new methods of interaction and novel applications. Our concept of collaboration using M-CVEs and its application in education (elaborated later as M-Education) takes researchers one step further in realizing the potential synergy between mobility and collaboration. The concept of M-CVEs is not entirely new, as subsets of this notion exist in previous and current research. For example, Satchel [Lamming et al. 2000] is a system that provides access to any document, any time, and anywhere. Although it is not an M-CVE, but the underlying groundwork transpired from the necessity for sharing resources from anywhere and at any time. The underpinnings of Satchel rely on a scenario involving a professional away from office who needs to access a remote document, and cannot do so efficiently in a timely fashion due to the very nature of mobile work. The Satchel scenario draws attention to the fact that while sharing resources (such as documents) during mobility is important, it is even more crucial to generalize the scheme towards collaboration of not only resources but also information in broader terms (such as chat, message boards, etc) and people.

An M-CVE would offer opportunities such as synchronous and/or asynchronous interaction with people and data for mobile users. One might argue that Instance Messaging (IM) systems, specifically the ones that may be accessed from mobile devices (e.g. [AOL], [MSN]), fall in the category of M-CVEs also. Although mobile IM clients exist and in general, they support interaction and outeraction [Nardi et al. 2000], studies show that their use seems to be tethered to a computer terminal and consequently do not provide anytime and anywhere communication [Grinter 2002]. This is largely attributed to the fact that although IMs follow the history of internet-based communication systems including MUDs and MOOs, the key difference is that the latter offers richer interactions [Grinter 2002], in that they provide synchronous and asynchronous communication not only with people but also with data that may be persistent.

In contrast to M-CVEs, there are many other technologies that come close to what we are aspiring for. For example, the Short Message Service (SMS) is the ability to send and receive text messages to and from mobile telephones [GSM]. This is not only limited to interaction with text messages, but also confined to just mobile phones. Very similar to the services we envision for M-CVEs is another system known as Blackberry [RIM]. Blackberry, a system specifically designed for the enterprise, provides quick, easy access to a person's email, contacts, calendar and task list on the move. However, Blackberry does not support collaboration with other peers; it simply provides access to one's own information content. In addition to being a software artifact, Blackberry is also the commercial wireless handheld device on which the software runs, thus limiting the use to specific handhelds rather than mobile devices in general.

Porting systems like MUDs and MOOs to mobile devices would give even more importance to place-based presence and data organization, because the desktop users are taken to a higher level of interaction that may be performed anytime, anywhere on anything. Wireless messaging systems such as HandiMessenger only provide tight coupling between awareness and contact capabilities [Hibino and Mockus 2002], and do not suffice for scenarios in which they emphasize on place-based presence and data organization. For example, a study group regularly meeting in a virtual community may like to refer to discussion notes from a previous meeting. Unlike IM-style chat sessions, email exchanges, or even shared web sites, the discussion notes offer a means of persistent data organization and its importance is further highlighted due to its association with a specific place (a virtual meeting location such as a library where the study group meets regularly). The motivational scenario in the next subsection provides credence to the above arguments in support of M-CVEs.

## 2.3. A Scenario

Consider the following scenario in which three colleagues are meeting virtually to collaboratively complete a task.

*During thanksgiving break, three colleagues who are unaware of each other's physical location, are meeting virtually using MOOsburg to discuss documents and architectural drawings for a class project due the following week. Being stuck in a traffic jam, Joe is late to the meeting and does not wish to delay the work for his two colleagues. He logs into the M-CVE version of MOOsburg through his Palm handheld and navigates to the virtual meeting room to find out that his two colleagues have not yet arrived. Joe posts a message to the notice board that he is on his way, enters the URL where he uploaded his contributions for the project so the others may view it upon arrival, and activates an event notification tool through which a message will popup on his handheld when any one of his two colleagues arrive.*

The above scenario is an example of an *exception* scenario where technology is being used to improvise an online meeting in a virtual community. Normal interactions have failed to take place such as being on time for a meeting and the unavailability of a computer. In this scenario, Joe is struggling to get home from work so that he can log into MOOsburg to attend his virtual meeting. Being unaware of his colleagues' locations and their phone numbers, Joe is unable to call his colleagues. Joe could have conveniently sent an email from his handheld, but that would restrict him to a one-way asynchronous interaction since he wanted to be notified of when his colleagues show up for the meeting. This scenario illustrates how an M-CVE can bridge the technology gap between users with desktops and mobile devices in both synchronous and asynchronous modes of interactions. Moreover, the scenario brings to light the importance of place-based presence (colleagues meeting at a common place) and data (posting of the URL as a persistent data artifact).

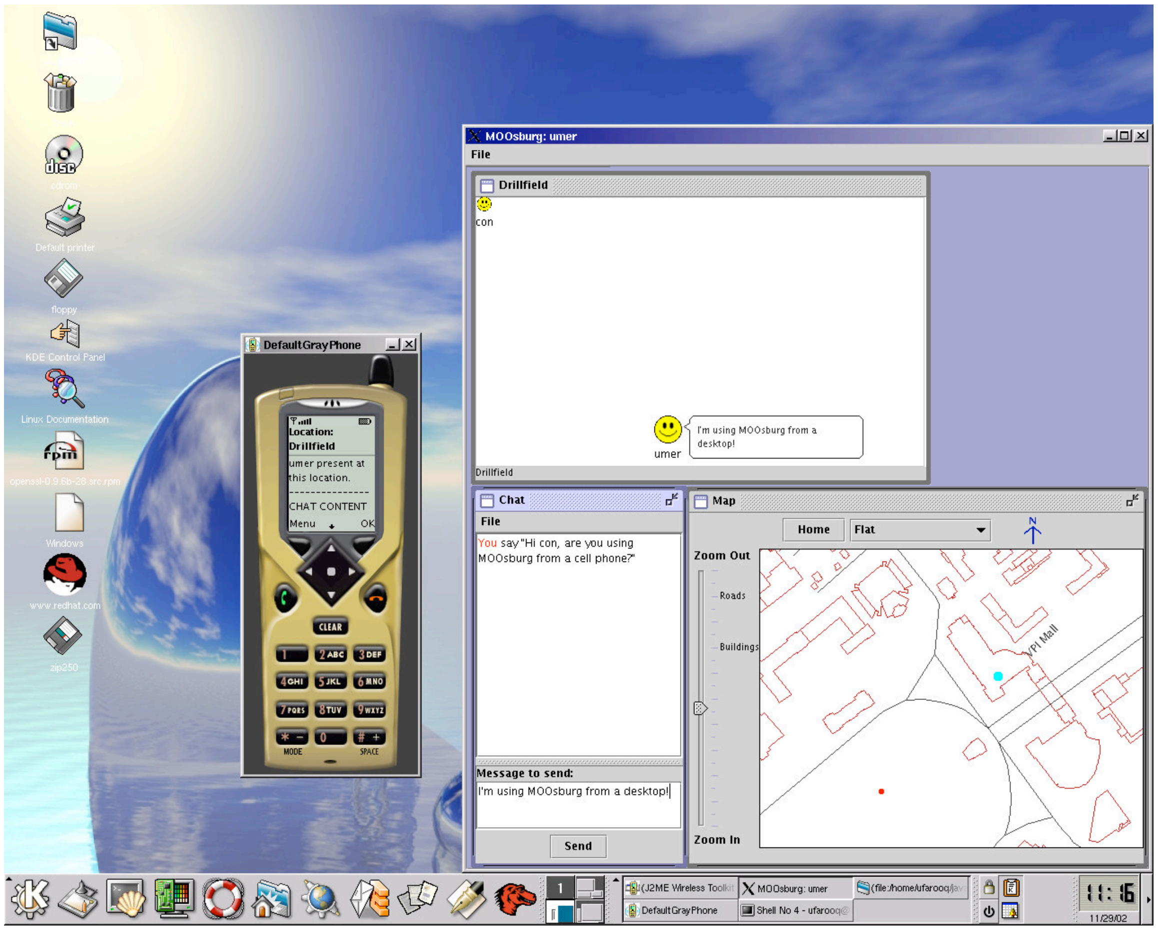
Through our arguments and scenario, we rationalize the need and importance of M-CVEs. M-CVEs can bridge the gap between mobile and desktop computing as users from different devices (e.g. desktops and handhelds) can collaborate in a common environment and have access to the same shared data, irrespective of the form factor. Community mobile computing is feasible because the number of users of wireless communication services is rapidly increasing due to the falling prices of mobile terminals and cellular phones [Nishibe et al. 1998]. It has been shown, for instance by the ICMAS'96 Mobile Assistant Project [Nishibe et al. 1998], that mobile computing technology integrates well with conventional desktop computing technology. The following section sheds light on a prototype we have developed to realize our vision.

## 3. Prototype

For materializing our vision of anywhere, any time collaborative environments, we will now focus our attention on a prototype for an M-CVE known as MOOsburg++, which is based on an existing online community known as MOOsburg [Carroll et al. 2000a]. MOOsburg++ supplements MOOsburg by providing accessibility to the virtual community from a range of internet-enabled mobile devices. MOOsburg++ is essentially a *lighter* version of MOOsburg, because unlike the latter, the former does not require a high bandwidth internet connection and provides a minimal user interface tailored for mobile devices. MOOsburg++ not only provides a concrete implementation of our paradigm, but also emphasizes a reusable, extensible, and platform-independent architecture for collaborative environments in general as we shall explain in the following subsections.

### 3.1. MOOsburg++ Features

MOOsburg++ provides basic implementation of shared collaboration. MOOsburg++ allows users to log into MOOsburg using their same user name and password, since the database tier is common to both systems. MOOsburg++ also displays a buddy list of all users currently in MOOsburg, therefore, a user from MOOsburg++ can interact with users from MOOsburg. A text-based interface to navigate the MOOsburg containment hierarchy is also provided. Moreover, MOOsburg++ supports for limited asynchronous and synchronous collaboration using chat and message boards. It is to be emphasized once again that all shared content (such as chat and message boards) is uniform in both systems, i.e. users from both systems see the subsequent changes.



I'm using MOOsburg from a desktop!

Location:
Drillfield
umer present at this location.
----------------
CHAT CONTENT
Menu OK

File

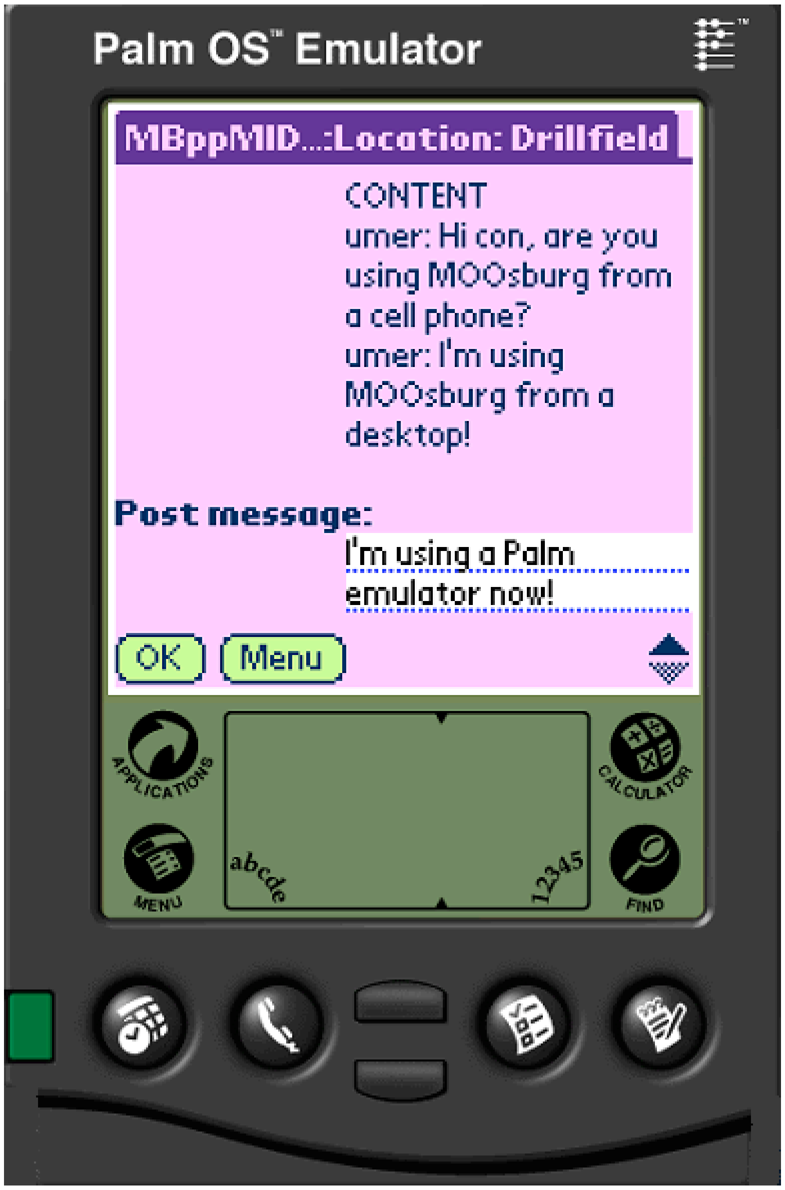
You say "Hi con, are you using MOOsburg from a cell phone?"

Message to send:
I'm using MOOsburg from a desktop!

**Figure 1. A chat conversation between user "con" in the MOOsburg++ emulator environment on a cellular phone and user "umer" in the desktop version of MOOsburg.**

Figure 1 shows user "con" logged in MOOsburg++ from the emulator cellular phone, and user "umer" using MOOsburg. For simplicity and testing purposes, the MOOsburg environment that is shown only consists of a chat window (bottom left) and an interactive map (bottom right) that has places marked by red (occupied location) and blue (unoccupied location) dots. Both users are having a synchronous chat conversation. Note that on the cellular phone, it is indicated that user "umer" is present at the location "Drillfield". Conversely, when user "con" entered the "Drillfield", the desktop version of MOOsburg shows a tiny avatar (top left) indicating that "con" is present.

Figure 2a shows the MOOsburg++ environment using the Palm OS Emulator (see http://www.palmos.com/dev/tools/emulator/) environment. Since Java 2 Micro Edition (J2ME) provides a generic user interface to be used on all Java-enabled handheld devices, MOOsburg++ is rendered differently on each device depending on the underlying operating system and support for user interfaces. This figure shows the same content as the cellular phone emulator: the location of the user ("Drillfield"), who is present at this location (scrolling will display this data), the chat content, and an interface to enter new chat messages.



CONTENT
umer: Hi con, are you using MOOsburg from a cell phone?
umer: I'm using MOOsburg from a desktop!

Post message:
I'm using a Palm emulator now!

**Figure 2a. Chat conversation in the Palm OS Emulator environment.**

Figure 2b shows the text-based navigation scheme in MOOsburg++. Since the map in MOOsburg used substantial bandwidth, it was not possible to port it to handheld devices. Therefore, a text-based navigation scheme was devised for MOOsburg++. Users in MOOsburg++ may select a location from the displayed choices, or filter locations by letter entry. The filtering option is necessary because hundreds of locations can be present in MOOsburg, and it is cumbersome to select a location from such a lengthy list. In this case, there are three locations that start with the letter "D". If the user enters "D" in the filter locations entry box, it will only display those three options, thereby reducing the overhead of searching for these locations.

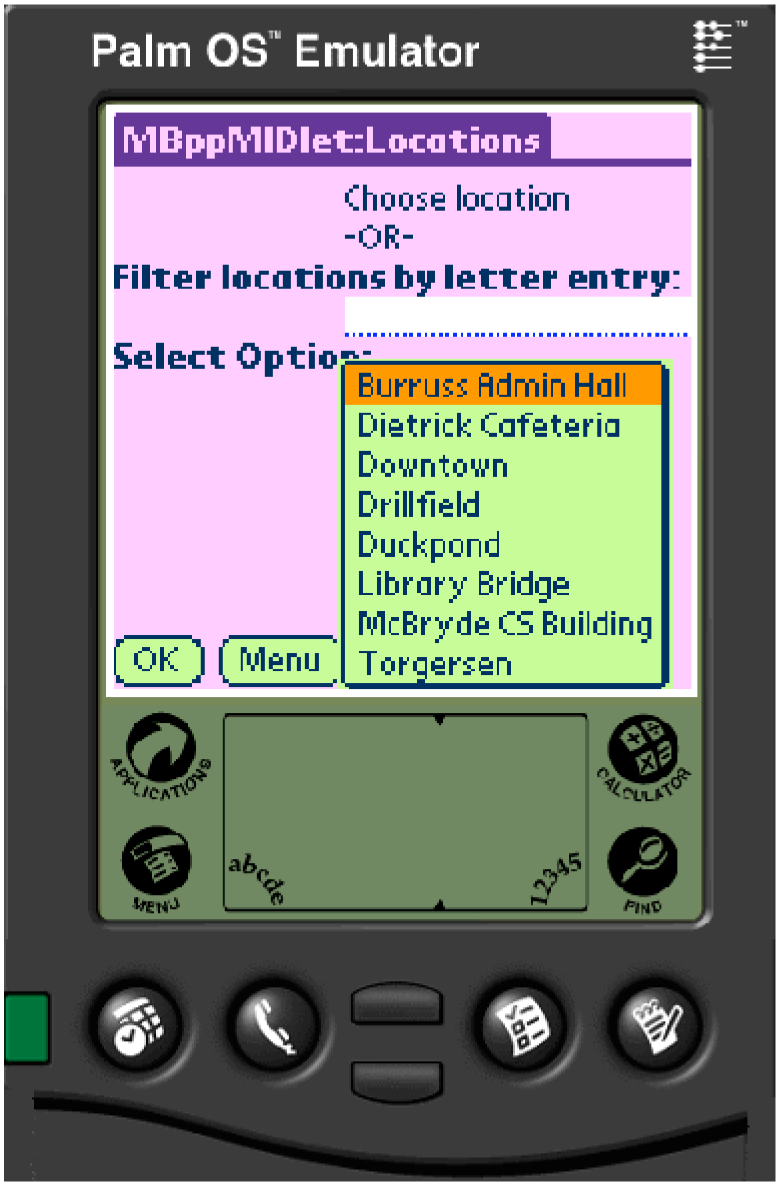


**Figure 2b. Text-based navigation scheme in MOOsburg++.**

Figure 3a shows a user working with MOOsburg++ on an actual handheld device. The device is a HandEra handheld running Palm OS version 3.5.3 equipped with the Symbol Wireless Networker card. The wireless infrastructure is an 802.11b network. To run MOOsburg++, the Palm has to be Java-enabled. Version 1.0 of MIDP for Palm OS was the Java platform for this device. Installing the Java platform and the application simply required a HotSync operation of two PRC files, which is the format of the Palm-ready applications.

Figure 3b displays the initial screen on the Palm when MOOsburg++ is started. The figure shows a login screen, prompting the user for the name and password.

**Figure 3a. User running MOOsburg++ on a handheld device with Palm OS and wireless Internet access.**

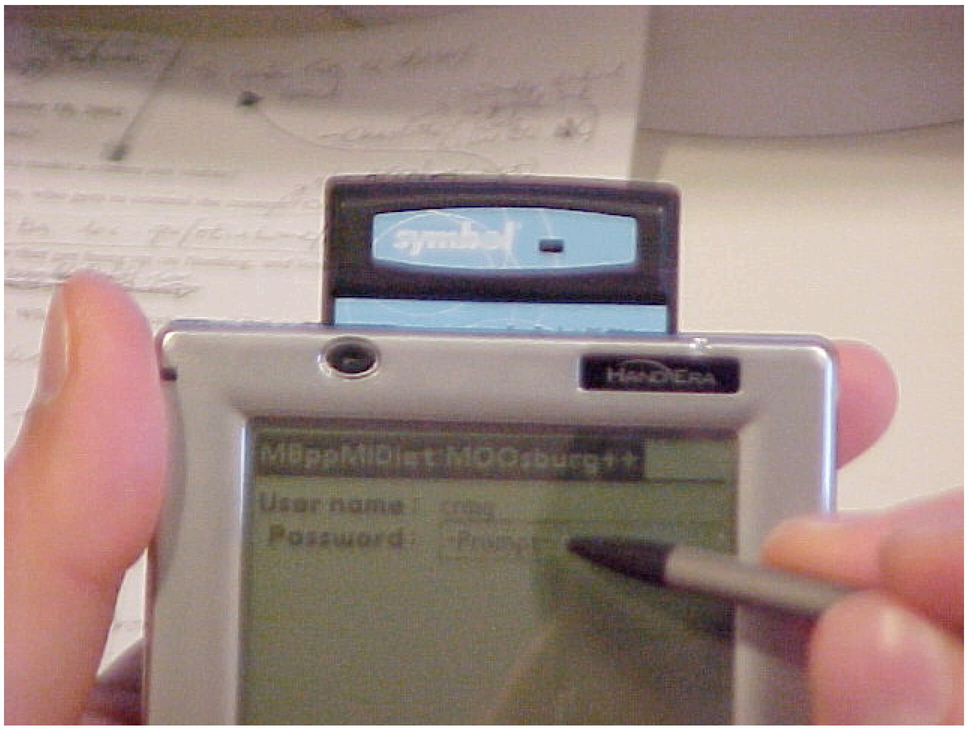


**Figure 3b. User entering name and password to login to MOOsburg++.**

## 3.2. User Interface for Text-based Navigation

In figure 2b, we saw the parallel scheme for graphical-based navigation in MOOsburg as text-based navigation in MOOsburg++. A graphical-based map can scale well to hundreds of locations, as the map can be interactively moved on an x-axis and y-axis, with additional capabilities of zooming at appropriate levels. Scalability in text-based navigation is achieved by simply appending more locations to the current list of locations. For seeking a location in a text-based map, the filtering option may be used to narrow down the spectrum of choices, provided the user knows the starting letter(s) of the location.

However, a graphical-based map provides *context* to the locations. Consider the following scenario:

*Greg wants to visit the sports shop that specializes in selling Cricket gear, but he cannot remember the name of the place. Realizing that last time, he walked a couple of blocks to the same shop after eating at the 15th Avenue McDonalds, he navigates to McDonalds on the map and finds his way to the sports shop.*

In this scenario, the restaurant provides a *sense of nearness* (sports shop being near to McDonalds) that Greg uses to find his destination. This capability should also be translated to the map-based scheme in MOOsburg++. One way of achieving this is to have an option to display nearby places within some defined physical boundary of another place. For example, choosing McDonalds would yield a search result of all the places in the surrounding two blocks.

## 3.3. MOOsburg++ Technology

The architectural foundation of MOOsburg++ is based on Sun Microsystems' (see http://www.java.sun.com) Java technology and their concept of "write once, run anywhere". This means that the program may be compiled into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java virtual machine (VM). That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on any platform such as Windows 2000, a Solaris workstation, or on an iMac.

MOOsburg++ uses the Java 2 Micro Edition (J2ME) technology to port the current version of MOOsburg to mobile devices. J2ME is a set of technologies and specifications developed for small devices like smart cards, pagers, cellular phones, and set-top boxes. J2ME uses subsets of the existing Java 2 Standard Edition (J2SE) components, such as smaller virtual machines and lightweight APIs.

J2ME provides the Mobile Information Device Profile (MIDP): a set of Application Programming Interfaces (APIs) to program for a particular mobile device. MIDP also describes minimum hardware and software requirements for a MIDP device. A MIDlet is a Java application developed to the MIDP specification for mobile devices. A MIDlet is used for developing the application interface of MOOsburg++. Therefore, on the client side, a MIDlet can be developed using the J2ME Wireless Toolkit and provisioned onto any MIDP-compliant device, whether it's a cellular phone, two-way pager, or palmtop [Sun a], which provides flexibility in choosing the mobile device for using the M-CVE.

With the delivery of the J2ME platform, Sun provides a complete, end-to-end solution for creating state-of-the-art networked products and applications for the consumer and embedded market [Sun b]. Granted that there exists other wireless technology that could have been used to develop MOOsburg++, J2ME was the right choice for the following reasons:

- The current MOOsburg architecture is based on Java object replication for communication between client and server [Carroll et al. 2001]. By using J2ME, the same object replication scheme is employed and users in MOOsburg can interact with users in MOOsburg++, i.e. the gap between the desktop environment and mobile devices is bridged. Of course, a simple HTML web browser could have been used on the wireless mobile devices; however, it would not provide the features of the Java-language platform such as threading, simple user interfaces, etc.

- The current MOOsburg application interface may be appended with MIDlets without changing the underlying architecture of MOOsburg. Therefore, the standardized, modular components of MOOsburg are reused. For example, some code from existing servlets that supported the web version of MOOsburg was reused to create new servlets customized to cater for MIDlets.

Another advantage of J2ME is that the application can be tested on a device emulator provided by the wireless toolkit. Therefore, the developer can avoid the complications of testing an application on a real device by first executing it on a desktop emulator, that provides a range of devices such as mobile phones, PDAs, etc. The following subsection presents the architecture of MOOsburg++, and how easily it integrates with the existing MOOsburg infrastructure.

## 3.4. MOOsburg++ Architecture

The underlying architecture of MOOsburg differs significantly from both traditional text-based MOOs and from transaction-based web applications. Each piece of data (people, places, and collaborative objects) in the environment is represented by a Java object and replicated across all interested clients using the Content Object Replication Kit (CORK), a toolkit for building web-accessible interactive distributed applications [Isenhour et al. 2001]. Following is an overview of the five primary parts of the CORK architecture: replicated objects, listener objects, change objects, a central database, and permissions objects.

Java supports object serialization: the ability to read or write a whole Java object to and from a raw byte stream. Therefore, any Java object can be encoded into a byte stream that can be streamed over a network or to a file system. The advantage of this mechanism is that Java objects as a whole can be read or written without conversion to/from bytes and parsing. CORK supports *replicated objects*, which are simply Java objects that maintain the same state in MOOsburg. For example, if some user edits a shared whiteboard (replicated object), the change to the whiteboard is represented as a Java object that is then conveyed (through serialization) to all the other users using the whiteboard. Thus, the whiteboard is a replicated object having the same state across all users. For users who might use the whiteboard at a later time, the object representing the change can be stored persistently (e.g. in a file) and reconstructed when required.

In the example above, how do I know when to get the updated version of the whiteboard when someone else edits it? This is achieved by attaching *listener objects* to the replicated objects described above. Listener objects are simply objects that want to be informed of state changes to replicated objects. When some user accesses a whiteboard, a listener object is attached that detects any changes to the whiteboard, therefore keeping the replicated objects in synchronized.

The change that is transmitted to a replicated object is known as a *change object*. In the whiteboard example above, we talked about an object that represents the change of state for that replicated object—that object is called the change object, which is sent to all the users. When some user edits a whiteboard, other users receive the change object and apply the embedded change to the appropriate replicated object.

The changes made to replicated objects have to be persistent, and therefore, a *central database* is required. As we mentioned before, replicated objects may be stored persistently in a database due to Java serialization mechanism. When a change to a replicated object is made, current users receive a change object immediately. For users who use the system at a later time, the updated replicated objects are stored in the central database, and when required, are transmitted to appropriate users. This scheme assures that a user will receive the most updated version of a replicated object, as the central database stores the latest change to a replicated object.

In a collaborative system such as MOOsburg, privacy issues are raised. For example, if I create a whiteboard for my class project, I may not want other users to edit the whiteboard. Therefore, each replicated object in CORK has *permission objects* affiliated with it that deals with security concerns. Depending on the setting of these permission objects, a change to a replicated object may or may not be allowed.

Now, we shall see how a MIDlet simply appends the top layer of the MOOsburg architecture so that the underlying CORK-based infrastructure is preserved and reused. Figure 4 is a depiction of the MOOsburg layered architecture that also shows the integration of the MIDlet (highlighted in green) to give rise to MOOsburg++.

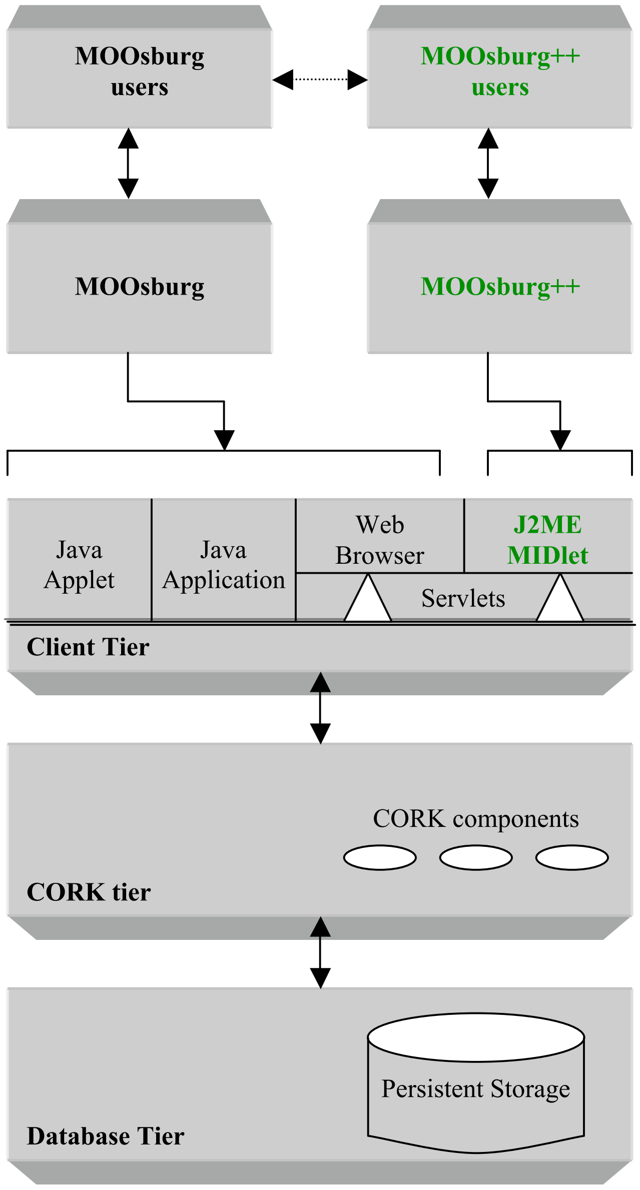


**Figure 4. The architectural integration of MOOsburg++ with MOOsburg.**

As seen from the layers in the CORK-based architecture used by MOOsburg, clients request object replicas (representing characters, landmarks, and things) from the CORK server (CORK tier). The server creates the requested objects or retrieves them from persistent storage. The client software then attaches appropriate user interface components that allow the user to modify the data and see the results of modifications made by other clients. The implementation currently supports several kinds of clients.

Java applets and applications can communicate with the server directly, using Java Object Serialization over a TCP connection. To support access from a broader range of devices, we have also developed servers that communicate with the CORK server and render object replicas as HTML, images, and other data types that can be transmitted via HTTP. This allows MOOsburg objects to be rendered in a standard desktop web browser, and also provides a suitable communication layer for interacting with MIDlets. Since HTTP has served as the cornerstone protocol for the WWW since 1990 [Qu et al. 2000], it bridges the communication gap between the MIDlet and the existing infrastructure provided by MOOsburg. The MIDlet communicates with the servlets that have all the business logic embedded in them, thereby allowing a universal protocol to be implemented. This architecture enables MOOsburg++ to share the same database tier as MOOsburg, so user interactions in different application interfaces relate to the same persistent data.

When the user interacts with the MIDlet, the application connects to the server by first authenticating itself with a user name and password, and then sending/receiving appropriate information from the server. Various connection protocols could have been used to connect to the server (such as Datagram), but we decided to use HTTP connection. There were two motivations for using an HTTP connection:

1. Other connection protocols are specific to the wireless device and the vendor. For example, the Datagram protocol can only be used on Motorola cellular phones.

2. Using HTTP connection provides platform independence, since all wireless devices define the general HTTP connection protocol.

In the HTTP request the MIDlet makes to the server, the application calls the doPost (…) method of the servlet. Each HTTP request that the client generates contains two parts:

1. Request property: The request property is the *header information* that the client uses to identify the type of operation that needs to be invoked on the server.

2. Actual message: The information content being conveyed from the MIDlet to the servlet and/or vice versa.

The request property is stored as a *key-value* pair, where the key is known to both the client and server. Depending on the different values of the key that the client specifies, the server executes respective operations. After sending the header information, the client uses *output stream* to write the actual message (operation or request) to the server. The client waits for confirmation from the server by using *input stream* to read the data that the server is sending. After the write and read operations are completed, the connection is closed and appropriate information is displayed on the screen.

To explain the information flow in Figure 1 from the database tier to the MIDlet, let us assume a user wishes to leave a chat message using MOOsburg++. The user activates the mobile device, launches the MIDlet, and enters the user name and password for verification. The user name and password are transmitted to the servlet (as the MIDlet knows the host and port of the servlet) using HTTP connection, and are verified. When the user is authenticated, the servlet receives the chat replicated object (CORK component) from the central database for the current location of the user. The servlet extracts the chat content from the replicated object, and sends it back to the MIDlet using the HTTP connection. The MIDlet displays the information to the user, and the user enters a message. This message is sent to the servlet, which then constructs a change object, applies it to the chat replicated object, and sends back the change to the user to update the information. The central database now maintains the latest chat replicated object, therefore any other user will have access to this updated chat content, whether from MOOsburg or MOOsburg++.

The architecture we have proposed provides uniform support for synchronous and asynchronous manipulation of replicated objects, reducing development effort required to support both modes within a single application. The primary advantages of our architecture are highlighted below:

- *Supporting multiple client tier applications:* This architecture provides a generic client tier that supports multiple application interfaces (such as Java applets, Java applications, web browser, etc). As we've established through our architectural design, porting MOOsburg to MOOsburg++ only required to develop an application interface (MIDlet) for the mobile devices with few changes to the servlets, where the other layers were completely unaltered.

- *Thin-client fat-server implementation:* Mobile devices have limited resources such as memory constraints, limited display capabilities, and a slow networking mechanism. Keeping these drawbacks in mind, especially the issue of narrow bandwidth, any application developed for these devices needs to have a thin-client fat-server implementation, where the *business logic* is embedded in the server, and the client tier is kept to a minimal implementation. This architecture rests its foundation on this notion, as the servlets deal with the business logic of the replicated objects, and the MIDlet is only responsible for displaying the information received from the servlets.

- *Adaptation of MOOsburg collaborative tools:* MOOsburg is an ongoing development effort with new collaborative tools being appended continuously. This raises an important question of how MOOsburg++ supports these new additions to MOOsburg. Using our architecture, this problem shrinks down to a minimal implementation of the user interface and information conveyance. Any new collaborative tool in MOOsburg is based on the CORK architecture; hence, it relies on Java's object replicated scheme. The servlets in the client tier have access to these replicated objects (CORK components), which brings the problem down to two issues. Firstly, how do we send this CORK component (or its information) to the MIDlet, and secondly, how do we display the information? Since the access of CORK components is limited only to the servlets in the client tier, the servlets extract useful information from the CORK component, and convey it to the MIDlet through a HTTP connection. The MIDlet, having received this information, obviously needs to display this information to the user. Therefore, with minimal alterations to the servlet and the MIDlet, any new collaborative tool in MOOsburg can be catered in MOOsburg++.

• *Asynchronous and synchronous support:* This architecture provides a unified mechanism for real-time updates, late-joiner support, and content persistence. Any interaction of the mobile user resulting in sending information back to the servlet consequently updates all shared content in MOOsburg++. Since the underlying server runs a Java servlet, any update on the desktop will result in a *transactional* interaction for the mobile client, i.e. the mobile client will have to perform a request to the servlet in order to be updated. The transactional nature of servlets implies a tradeoff in terms of automatic updating on the mobile client. The potential of Java threads should be explored to poll the servlet for any change, which would result in automatic updates, where the time is dependent on the frequency of polling.

From our architecture and prototype, we have established that CORK is a reusable and extensible architecture, not only limited to desktop computers (though that was the original intention) but also handheld devices. Another architecture similar to ours is the one commercially used by Microsoft. Their MSN Instant Messenger service can be used from desktops and mobile clients. However, our architecture does not depend on any special type of vendor support. MSN Messenger demands vendor specific requirements; it can only work with Verizon Wireless web-enabled mobile devices [VZW].

Among the various above-mentioned facets we learned from MOOsburg++ and its architecture, we also encountered a drawback of just using the J2ME platform. J2ME provides generic user interface features, independent of the underlying hardware device (be it a PDA, cellular phone, etc). When a J2ME program such as MOOsburg++ is ported on different devices, the user interface does not render aesthetically well. The rendering is not in control of the user nor the programmer; it purely depends on the Java operating system on the specific device. However, having experienced this challenge of J2ME, we propose the use of User Interface Markup Language (UIML) (see http://www.uiml.org) with J2ME as a future direction to be explored for rendering user interfaces according to the various devices.

Having motivated the necessity of M-CVEs and presenting our prototype (architecture and application), the following section talks about a prospective application domain for MOOsburg++ that bridges the gap between mobile and desktop computing.

## 4. Application Domain

Networked computers and corresponding applications facilitate distributed education with the mediation of learning activities by a constellation of various tools (such as shared spaces, whiteboards, etc) having appropriate pedagogical approaches to collaboration and social interactions [Fjuk and Smørdal 2001]. MOOsburg is one such example, which is a part of the Learning in Networked Communities (LiNC) project that has developed and evaluated software tools and applications for collaborative learning activities [Carroll et al. 2000b]. Research has compellingly established the importance of learning communities. At the same time, mobility, flexibility and instant access of handheld devices add considerable freedom for people to collaborate anywhere, anytime [Soloway et al. 2001]. However, not enough research has been done in integrating the two concepts, for example trying to coordinate the use of desktop computers and handheld devices. In this article, we propose the concept of Mobile Education—or M-Education—which is a new way of using wireless and mobile technologies for education by extending access to a desktop-based online virtual environment such as MOOsburg to handheld devices used as part of a mobile collaborative community. This section brings to light the potential of M-Education, and how the power of handheld computing can be combined with the traditional use of desktop computing to realize a new improvement in education.

### 4.1. M-Education

Our premise of M-Education is based on the idea by Pascoe et al. [2000]: "Using While Moving", which is the basic ability fieldwork users require of a mobile computer system. M-Education encourages distributed peer collaboration over wireless devices and desktop computers to create opportunities for discovery and education in the field and community. It is a novel approach that will use a wireless virtual community to facilitate the learning activities of teachers, students, and peers through collaboration in a distributed environment. M-Education is significantly different from existing mobile learning systems in that it leverages its collaborative activities from an existing desktop-based online virtual community (e.g. MOOsburg), and thus offers a range of collaboration opportunities, such as synchronous and asynchronous interactions with peers, and viewing or changes to persistent data. This will enable users who are interacting from either handheld devices or desktop computers to merge their learning experiences in a shared collaborative environment, both synchronously and asynchronously, with reference to the same underlying data. The communication between a handheld and desktop is similar to that between two desktops.

Numerous efforts are being made in the direction of using handheld devices for educational purposes. In cases where efforts consider possible coordination between handheld and desktop environments, none have proposed the rich interactions we envision in M-Education. By examining a few related applications and concepts, we shall see how M-Education takes learning using wireless and mobile technologies one step further.

## 4.2. Related Work

KNOWMOBILE is an exploratory and research project [Lundby et al. 2002] that is conducting experiments on medical students during their field assignments. It focuses on how its users, in the context of their local environment, use handheld devices to access web-based medical knowledge and information. Although this project facilitates distributed and context-specific access to information, it makes no effort to coordinate such activities with other educational activities or peers.

Wireless Internet Learning Devices (WILD) [Roschelle and Pea 2002] offers another vision for how one might use handheld devices in classrooms for computer-supported cooperative learning (CSCL). It is offered as a substitution for replacing CSCL applications that use desktop/laptop computers, a sort of paradigm shift. In contrast to our vision, the use of WILD in CSCL replaces—rather than integrates with—the use of desktop computers for distributed learning.

Perhaps the closest effort to the M-Education concept emphasized in this article is by Luchini et al. [2002], which uses handhelds to support collaborative learning. The authors merely suggest that handhelds may be used with desktops when the disadvantages of the former such as limited screen space become a considerable issue. M-Education takes the counter approach, emphasizing that when desktops are not available, collaboration is still possible using handheld devices providing the same enriched interactions as available on a desktop computer. Our vision is not simply to supplement desktop user interfaces, but rather to explore the new and varied educational activities that become available in a mobile computing setting.

## 4.3. The M-Education Vision

One of the target applications for MOOsburg is an ongoing community project called Save Our Streams (http://www.iwla.org/SOS/). Save Our Streams is a national watershed education and outreach program that uses hands-on activities, such as cleaning up stream corridors and monitoring stream health, to help restore watersheds. Through these activities, community members learn about the importance of protecting their local watershed and become more educated about the environmental, economic, recreational, and public health benefits of clean water.

The national Save Our Streams program consists of over 300 local chapters that coordinate activities for their local citizens. In Blacksburg, the Museum of Natural History at Virginia Tech organizes field trips for grade school children and offers training sessions that teach others how to monitor and adopt a stream section. One of the major activities is an assessment of the stream's health through biological sampling, such as insect counts. Participants on such trips learn about stream ecology and how to assess water quality. The data collected on these outings are often provided to the local and state government to augment their knowledge of the stream's condition.

Currently, the local Save Our Streams project conducts biological sampling at seven distinct locations. We have modeled each of these locations in MOOsburg, such that all of the data related to a particular site is available online. Through the use of synchronous and asynchronous chat tools, Save Our Stream leaders as well as other community members, can discuss interesting findings such as the overall condition of the local stream.

In this article, M-Education underscores the importance of merging the handheld devices with desktop computers for educational purposes by analyzing the need of a wireless online virtual community for the Save Our Streams group, or in general, any group working in the field that requires a collaborative channel for achieving an educational goal.

Assessing a stream through biological sampling is a particularly interesting activity for M-Education. In organizing an educational trip to the stream, Save Our Stream leaders want to engage people in hands-on activities and informative discussions about the local watershed. One way to encourage learning in this setting is to compare data collection results, explore trends, and communicate with other stream experts. Upstream activities affect downstream collection results and a stream can change over time. Accessing previous data while adjacent to the stream can foster an educational discussion of these properties and encourage participants to learn about what affects a stream's health. Also, communicating with other stream experts not on the trip can provide additional insight. These features are available through MOOsburg, yet they are not readily available in the field when people are learning.

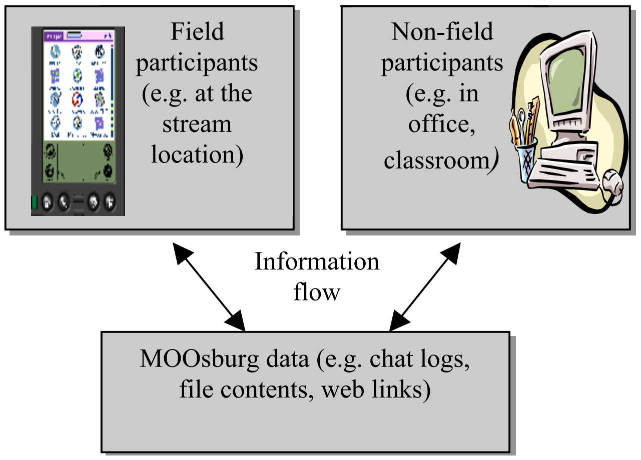

Field participants (e.g. at the stream location)
Non-field participants (e.g. in office, classroom)
Information flow
MOOsburg data (e.g. chat logs, file contents, web links)

**Figure 5. Coordinated use of handheld and desktop computing for learning.**

A high-level depiction of the combined interaction between handheld devices and desktop computers using MOOsburg is given in Figure 5. In this figure, participants in the field can perform synchronous and asynchronous interactions with peers (non-field participants) and data, where data may consist of insect count readings stored in a file or the health condition of the stream a few weeks back that was discussed during a chat session. In the following subsection, possible scenarios are outlined and evaluated in light of the requirements of the Save Our Streams group.

## 4.4. Example Scenarios

The following scenarios illustrate the prospective role of MOOsburg++ in the use of wireless and mobile technologies for educational purposes in the field. The first scenario given illustrates how accessibility to data stored within MOOsburg can enhance an educational activity in the field.

*Save Our Streams leader, Bob, takes a small group of middle school students to a section of Stroubles Creek to conduct a stream assessment. As they begin to wrap up their invertebrate count at the creek, Bob is entering the numbers on his handheld and notices that the number of Mayflies is considerably lower than last year. The group discusses some of the possible reasons before Bob encourages the group to explore the data from some upstream sites. The students discover that there have been many water temperature changes over the past year and they discuss how these changes could affect the insect population.*

This scenario demonstrates the usefulness of M-Education for an educational field trip. A traditional trip would have the students count invertebrates and record their data on a record sheet. But M-Education allows the lesson to continue. By entering the data and examining other results in the field, the group learns about the larger preservation effort they are contributing to. They learn that it is important to share their findings with others and that different groups can benefit from one another. Also, looking at the previous data helps to put their results into perspective and encourages further questioning and discovery activities.

In this scenario, the group discussed past insect counts and water temperature changes. Walking up the stream might help them investigate the cause of these changes, prompting another educational discussion. Access to the community database in the field provides both of these opportunities for Save Our Streams leader. It allows them to educate people about the larger watershed protection effort and encourage hands-on learning related to streams.

In addition to data access, M-Education also enables peer-to-peer collaboration and coordination between handhelds and desktops, as in the following scenario.

*While training a couple of community members on how to conduct a Save Our Streams trip, Julie notices that the stream area they are visiting could use some improvement. She uses this opportunity to discuss these issues with the trainees and introduce the online virtual community. One of the issues raised concerns about the large amount of trash that is nearby. Unfortunately, Julie did not bring any supplies, such as garbage bags and gloves, to address this problem. The group also notices that the stream lacks shade and that planting some small bushes could be an easy solution. Julie does not want to have to remember these suggestions for the next visit, so she shows the group how to* *use their handheld devices to share their idea with others in the Save Our Stream community. Save Our Stream leaders often check with this online environment before going on a field visit so that they can see what has happened there recently. Shortly after they post their ideas, a new message appears from another stream expert. Apparently, he will be visiting this area soon with a local youth group and will be able to make the improvements.*

This scenario demonstrates the usefulness of MOOsburg both in the field and in an inside setting using a traditional desktop PC. Providing access in both locations allows the collaborators to continue their communication despite their physical location. It also allows them to exchange ideas in a timely fashion and not wait until they have access to a desktop PC.

Supporting these interactions in the field adds new dimensions to the educational experience. In this example, the trainees quickly learned about the network of stream experts that they would be joining. They learned about how the larger group works together and discusses various issues concerning the watershed. Similarly, a group of college students could have benefited from a synchronous exchange with another stream expert. Possibly this person could not join the trip but they offer additional knowledge about the stream. Having access to MOOsburg through both handheld devices and desktops makes these collaborations possible.

M-Education is a new conceptual paradigm in the use of mobile and wireless technologies for education. Its consequence is that teachers, students, and peers in a distributed field environment can interact seamlessly with their counterparts in a desktop environment.

## 5. Discussion and Future Work

The need for mobile users to use mobile devices for collaborative purposes does not arise from the fact that they are mobile, but from the tacit implication of mobility, that is, they do not have access to conventional means of collaboration through their desktops. Perry et al. [2001] support this argument by relying on evidence suggesting that much of what determines what mobile users do can be explained in terms of the limited resources available to them. As mobile devices and their networking become affordable and reliable, people are gravitated towards collaborative computing on a range of platforms. Computer-Supported Cooperative Work (CSCW) itself has branched a research interest known as *Mobile CSCW*. These changes have motivated a break from the traditional model of computation to a ubiquitous model that makes the users' entire environment available wherever it is required [Dearle 1998].

With the addition of MOOsburg++, MOOsburg as a collaborative environment in general is now accessible anywhere, any time, and on all platforms (provided it supports Java). Users from both MOOsburg and MOOsburg++ can interact seamlessly, since the collaboration is transparent. This means that MOOsburg users collaborating with MOOsburg++ users are not aware that they are interacting from MOOsburg++, and vice versa. Refer back to Figure 1, which shows both users are unaware of each other's collaborative environment. This *transparency* between the two systems caters for a normal mode of interaction—interaction that is same in all permutations of the scenarios—between the users, as none of them know which system is being used. Of course, this also has obvious disadvantages. Firstly, *non-transparency* may lead to awareness cues and further on to useful information. For example, if a MOOsburg user knows that he/she is interacting with a mobile user, the collaboration goal may be reached faster and efficiently, as the MOOsburg user realizes that interaction needs to be kept minimal due to the intricacies involved in using a mobile system. Secondly, non-transparency may also transpire a discussion between the MOOsburg and MOOsburg++ users (or MOOsburg++ and MOOsburg++ users) of the actual physical location of each of the users. This leads to critical repercussions, perhaps advantageous or not, such as issues concerning privacy. Currently, the systems are transparent, and in this article, we are not arguing that it is necessarily the ideal approach, but simply stating that transparency is a core issue in collaborative systems on different platforms and it needs to be explored further by researchers in this area.

Another appealing issue that has cropped up is the tendency of the MOOsburg++ to lean towards the traditional MOOs. A working example of this exists in our prototype, that is, the map. In MOOsburg, the map is an interactive graphical widget, whereas in MOOsburg++, navigation is performed in a text-based fashion, where the user is given a tree-like hierarchy of spaces to choose from. In the interest of bandwidth (and not as an argument), an interactive map was not provided for MOOsburg++. At the moment when bandwidth is limited for mobile devices, a text-based map is set by default. However, in future when network technology advances, it would be interesting to explore the *mode of navigation* that is suitable for M-CVEs in general and specifically, MOOsburg++. Due to limited screen space of mobile devices, users "may" prefer the traditional MOO-like navigation schemes to graphical interactions.

Place-based awareness for mobile users may be further enhanced in M-CVEs in general and specifically MOOsburg++ through the use of Global Positioning System (GPS). The use of GPS [first suggested by Herring 1996] in MOOsburg++ would spawn new research possibilities and suffice for richer user interactions. Consider the following scenario:

*Melissa, having forgotten to collect the laundry twice already, logs into MOOsburg++ and activates a reminder in the virtual Blacksburg Laundromat location. The next day while driving to the grocery store, Melissa passes by the Laundromat and her cell phones rings to display a reminder for collecting her laundry.*

By using GPS in MOOsburg++, this scenario provides a token in a virtual world that connects to an action to be performed in the real world. This vision provides endless possibilities, and we would like to reinforce on this concept with future development efforts on MOOsburg++.

Technologically speaking, the MIDlets on which the MOOsburg++ application interface is built have to be explicitly downloaded on the wireless devices, i.e. they do not cater for over the air provisioning. Wireless Application Protocol (WAP) and J2ME are not competing technologies; rather they complement each other [Mahmoud 2002]. We wish to explore this aspect by extending our MIDlets for integration in a WML page that can be called from a WAP-enabled browser. Consequently, the MIDlet would get installed on the device.

Bluetooth is yet another wireless technology that enables links between mobile computers, mobile phones, portable handheld devices, and connectivity to the Internet. Bluetooth aims to replace the cables used to connect computing devices with wireless radio connection. Bluetooth is clearly tackling a hardware networking issue whereas J2ME is on the software side of things [Morrison 2001]. As of now, it isn't clear how Bluetooth will impact J2ME, but we anticipate some synergy in the near future.

Two other important issues for mobile applications in general are security and connection reliability. Though MOOsburg++ need not be a security-intensive application, security concerns need to be addressed in future. To transmit sensitive information such as passwords and account information, Secure Socket Layer (SSL) may be used. We have already explored the possibility of incorporating SSL into MOOsburg++ by verifying that that the J2ME Wireless Toolkit allows URLs beginning with *https://* (for HTTPS connections). Connection reliability is often a hurdle for mobile applications, and currently, we are beginning to research solutions to this problem. In circumstances of poor or non-existent connections, an illusion of connectivity needs to be maintained [Ebling et al. 2002] so the user does not feel cognitively disconnected from the environment. We are currently considering the importance and incorporation of *translucence* [Ebling et al. 2002] for an M-CVE such as MOOsburg.

## 6. Conclusions

Realizing that MOOsburg is an online virtual community limited to desktops, we rationalize that MOOsburg++ is a necessity to suffice for richer synchronous and asynchronous interaction with people and data (that may be persistent). Through our scenario-based design, we believe that being (or having data) virtually in a place has advantages over more abstract notions of online sessions available like IMs. Moreover, we propose an architecture for MOOsburg++ that is based on the reuse of existing infrastructure of MOOsburg. MOOsburg++ serves as a model for building wireless virtual communities that are collaborative using the proposed architecture. Through our research efforts, we hope that MOOsburg++ is a universally accessible application anytime, anywhere, on anything that will not only serve as an architecture for other developers to follow but also a motivation towards a paradigm shift from desktop to mobile collaborative virtual environments.

We have also presented our vision of M-Education; an application domain for integrating the use of wireless technologies into an existing collaborative environment. The consequence is that teachers, students, and peers in a distributed field environment can interact seamlessly with their counterparts in a desktop environment. They are also able to examine and modify shared data maintained in the online community. The basic idea is in place and we are beginning to develop and evaluate scenarios of the sort described here.

Cyberspace guru Howard Rheingold reinforces our vision of M-CVEs by predicting that during the first decade of the 21st century, the combination of the unique characteristics of virtual communities and mobile communications will spawn powerful hybrids, just as the merger of the PC with the telephone network created a wholly new medium, the Internet [Rheingold, 2001]. He foresees mobile virtual communities being used to coordinate actions of groups in geographic space and be social arenas, business tools, and political weapons. The result is that participants in online communities will remain in continuous contact over multiple platforms. Rheingold [2001] claims that the killer applications are going to be social, not technical, and we see our concept of M-CVEs as the first step towards this new paradigm of social interaction and collaboration.

## References

AOL Home Page. Available at: http://www.aim.com/index.adp/.

Carroll, J.M., Rosson, M.B., Isenhour, P.L., Van Metre, C.A., Schafer, W.A. and Ganoe, C.H. (2000). MOOsburg: Supplementing a real community with a virtual community. In *Proceedings of the Second International Network Conference: INC 2000*. pp. 307-316. Plymouth, UK: University of Plymouth/Internet Research.

Carroll, J.M., Rosson, M.B., Neale, D.C., Isenhour, P.L., Dunlap, D.R., Ganoe, C.H., Van Metre, C.A., Seals, C., Fogarty, J., Schafer, W.A., Bussom, T., Bunn, K., Davie, P., Freeman, M., Goforth, A., Mauney, S.M., Rencsok, F.C., Anderson, C., Hertel, M. and Svrcek, B. (2000). The LiNC Project: Learning in Networked Communities. *Learning Technology 2(1)*.

Carroll, J.M., Rosson, M.B., Isenhour, P.L., Ganoe, C.H., Dunlap, D.R., Fogarty, J., Schafer, W.A. and Van Metre, C.A. (2001). Designing Our Town: MOOsburg. *International Journal of Human-Computer Studies (54)*.

Cruz-Neira, C., D. J. Sandin, T. A. DeFanti, R. V. Kenyon and J. C. Hart. "The CAVE: Audio Visual Experience Automatic Virtual Environment," Communications of the ACM, 35(6), 1992, pp. 65-72.

Curtis, P. (1992). Mudding: Social phenomena in text-based virtual realities. Proceedings of the 1992 Conference on the Directions and Implications of Advanced Computing Berkeley, CA.

Curtis, P., Dixon, M., Frederick, R. and Nichols, D. The Jupiter audio/video architecture: secure multimedia in network places. Proc. MM'95. San Francisco. 1995.

Dearle, A. (1998). Toward Ubiquitous Environments for Mobile Users. *IEEE Internet Computing* 2(1): 22-32.

Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A., and Palfreyman, K. "Exploiting Space and Location as a Design Framework for Interactive Mobile Systems", *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 3, September 2000, pp. 285-321.

Ebling, M. R., John, B. E., and Satyanarayanan, M., "The Importance of Translucence in Mobile Computing Systems", ACM Transactions on Computer-Human Interaction, Vol. 9, No. 1, March 2002, Pages 42-67.

Fjuk, A. and Smørdal, O., "Networked Computers' Incorporated Role in Collaborative Learning", *CSCL 2001.*

Gamma E., Helm R., Johnson R., and Vlissides J. (1995). Observer. *Design Patterns*, pp. 293-303. Reading, MA: Addison Wesley.

Greenhalgh, C., and Benford, S., "MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading," in *Proc. IEEE 15th International Conference on Distributed Computing Systems (DCS'95)*, Vancouver, Canada, May 30 - June 2, 1995, IEEE Computer Society.

Grinter, R.E., “"Hanging Out With Computers:" The Role of IM in Teenage Communication”, *Human Computer Interaction Consortium 2002 Winter Workshop.*

GSM Association, An Overview of SMS. Available at: http://www.gsmworld.com/techno logy/sms/index.shtml/.

Herring, T. A. 1996. The Global Positioning System. Sci. Am. 274, 2, 32-38.

Hibino, S. and Mockus, A. “handiMessenger: Awareness-Enhanced Universal Communication for Mobile Users”, *Human Computer Interaction Consortium 2002 Winter Workshop.*

Isenhour, P.L., Rosson, M.B., and Carroll, J.M. “Supporting Interactive Collaboration on the Web with CORK”, *Interacting with Computers (13)*, Special Issue on "Interfaces for the Active Web", 2001, pp. 655-676.

Kristoffersen, S. and Ljungberg, F. 1999. Making place to make IT work: Empirical explorations of HCI for Mobile CSCW. In GROUP’99: *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work* (Phoenix, AZ, Nov. 14-17), ACM Press, New York, NY, 276-285

Lamming, M., Eldridge, M., Flynn, M., Jones, C., and Pendleburry, D., 2000. Satchel: Providing access to any document, any time, anywhere. ACM Transactions on Computer-Human Interaction, 7, 3, 322-352.

Luchini, K., Quintana, C., Curtis, M., Murphy, R., Krajcik, J., Soloway, E., and Suthers, D., ”Using Handhelds to Support Collaborative Learning”, *Proceedings of CSCL 2002,* pp. 704-705.

Luff, P. and Heath, C. 1998. Mobility in Collaboration. In *Proceedings of the CSCW’98 Conference on Computer-Supported Cooperative Work*. ACM Press, New York NY, 305-314.

Lundby, K., Smørdal, O., Larsen, A., Fjuk, A., ”Networked PDAs in a Community of Learners”, *Proceedings of CSCL 2002,* pp. 548-549.

Q.H. Mahmoud, “J2ME MIDP and WAP Complementary Technologies”, 2002, Available at: http://wireless.java.sun.com/midp/articles/midpwap/

Morrison, M., “Sams Teach Yourself Wireless Java with J2ME in 21 Days”, ISBN 0-672 -32142-4, Copyright © 2001 by Sams, page 16.

MSN Messenger Home Page. Available at: http://messenger.msn.com/.

Mynatt, E. D., Adler, A., Ito, M., and O'Day, V. 1997. Design for network communities. In S. Pemberton (Ed.), Conference Proceedings of CHI'97: Human Factors in Computing Systems (pp. 210-217). New York: ACM.

Nardi, B.,A., Whittaker, S., & Bradner, E. 2000. Interaction and outeraction: Instant messaging in action. In Proceedings of CSCW 2000 (pp. 79-88). New York, ACM.

Nishibe, Y., Waki, H., Morihara, I., Hattori, F., Ishida, T., Nishimura, T., Yamaki, H., Komura, T., Itoh, N., Gotoh, T., Nishida, T., Takeda, H., Sawada, A., Maeda, H., Kajihara, M., and Adachi, H. Mobile digital assistants for community support. *AI Magazine*, 19(2):31-49, 1998.

Pascoe, J., Ryan, N., Morse, D. 2000. *Using While Moving: HCI Issues in Fieldwork Environments*. ACM Transactions on Computer-Human Interaction, Vol. 7, No. 3, September 2000, Pages 417-437.

Perry, M., O’Hara, K., Sellen A., Brown, B., Harper, R. Dealing with Mobility: Understanding Access Anytime, Anywhere, ACM Transactions TOCHI, Vol. 8, No. 4, December 2001.

Rheingold, H. 2001. Mobile Virtual Communities. Available at: http://www.thefeature.com/index.jsp?url=articile.jsp?pageid=12070.

RIM Wireless Handhelds, “Technical White Paper Blackberry Enterprise Edition for Microsoft Exchange, version 2.1”, 2001.

Roschelle, J. and Pea, R., ”A Walk on the WILD Side: How Wireless Handhelds May Change CSCL”, *Proceedings of CSCL 2002,* pp. 51-60.

Rosson, M.B. and Carroll, J.M. (2002). *UsabilityEngineering: Scenario-Based Development of Human-ComputerInteraction.* Morgan Kaufmann, San Francisco, CA.

Soloway, E., Norris, C., Blumenfeld, P., Fisherman, B., Krajcik, J., and Marx, R. (2001), “Handheld Devices are Ready-at-Hand”, *Communications of the ACM*, 44 (6), pp. 15-20.

Sun Microsystems Corp., "Designing Wireless Enterprise Applications Using Java Technology," Available at: http://java.sun.com/blueprints/guidelines/designing_wireless_enterprise_applications/main.html/.

Sun Microsystems Corp., Java$^{TM}$ 2 Platform, Micro Edition (J2ME$^{TM}$ Platform), Available at: http://java.sun.com/j2me/.

Qu, C., Engel, T., and Meinel, C. "Implementation of an enterprise-level groupware system based on J2EE platform and WebDAV protocol", *Enterprise Distributed Object Computing Conference,* EDOC 2000, pp. 160-169.

Verizon Wireless with MSN. Available at: http://messenger.msn.com/support/verizon.asp/.

## Vita

Umer Farooq completed his Bachelor in Computer Science from FAST-ICS (Foundation for Advancement of Science and Technology-Institutes of Computer Science), Islamabad, Pakistan, now known as NUCES (National University of Computer and Emerging Sciences) in Fall 2000. Thereafter, he joined Virginia Polytechnic Institute and State University for Masters in Computer Science.

During his graduate work, Umer has worked as a graduate research assistant in the Center for Human-Computer Interaction. He is currently pursuing a PhD in Computer Science from Virginia Tech.