A framework for device capability on demand and virtual device user experience

R. Y. Fu
H. Su
J. C. Fletcher
W. Li
X. X. Liu
S. W. Zhao
C. Y. Chi

The proliferation of mobile devices is gradually making it possible to access information anywhere at any time. However, the physical capabilities of the mobile device still greatly limit the experience of users because functionality has usually been traded off for ubiquity. Nonetheless, the enormous growth rate of new information appliances heralds the dawning of a device-rich era. In this paper, we propose a framework for augmenting mobile device capabilities with surrounding devices. We then discuss a possible approach to represent an augmented device as one single virtual device.

Introduction

All of the promises of the Internet rely on fast and convenient access to information. While certain benefits have already been realized by connecting desktop-class personal computer (PC) users, the full potential can only be exploited by connecting the broadest number of people via the widest range of mobile devices. Existing ubiquitous computing middleware has significantly extended the reach of the computing infrastructure to mobile users, but satisfactory user experiences and profitable business models have not yet been exhibited. Research momentum is shifting toward using server power more efficiently and seeking ways for multiple client devices to collaborate.

User acceptance largely determines the success of any system. In many parts of the world, users have primarily experienced desktop systems. Compared with desktop solutions, the restrictions on user experience and capabilities brought about by smaller devices are key inhibitors to an enterprise-ubiquitous computing adoption. Current trends for improving user experience largely focus on the development of powerful, multifunctional, but still handy and mobile devices. Enterprises are accustomed to the *single-device* model, where the capability of one device largely determines the overall user experience and quality of service for the enterprise applications. Meanwhile, middleware today must accommodate a vast variety of devices, which, because they continue to be resource-constrained, limit the possibilities of the middleware. A solution would be to shift the model to one that allows us to distribute the single application across multiple devices, leveraging the strengths of each physical device for the application.

Over the past few years, significant progress has been made in improving mobile device capability. These activities have leveraged both single-device and multidevice approaches.

The single-device approach emphasizes improving inherent device capability (e.g., continual hardware and software upgrades and multiple functions all in one). This approach entails costly solutions that usually cannot be replicated, are not at all flexible, and do not solve the essential problem—the limitations of single-device capability. In addition, while many future devices will carry faster processors and larger memories, the human–computer interaction will remain a problem because mobile devices, by their nature, will not become physically large enough to provide typical PC-level interactions. In a study of everyday appliance examples, Buxton questioned the value of *super-appliances* [1]. He suggested breaking away from the one-size-fits-all approach and evolving to tools built for specific purposes. The study conducted by Buxton concludes that devices will become sufficiently inexpensive to accomplish this, and they will be distributed around certain locations appropriate for certain activities.

0018-8646/04/$5.00 © 2004 IBM

Meanwhile, new wireless technologies such as Bluetooth** [2] have emerged to improve short-range communication between devices. This simplifies the task of providing distributed application capability that can enhance user experiences by overcoming single-device limitations. The Universal Plug and Play (UPnP**) Forum [3] is an industry initiative to enable simple and robust connectivity among standalone devices and PCs from different vendors. UPnP architecture offers ubiquitous peer-to-peer network connectivity for devices, but without specifying how two connected devices interact together with server-side applications. Other similar activities, such as Jini** [4] and Salutation** [5], have similarly addressed the requirements for interdevice communication.

Myers [6] describes the Pebbles project, which aimed at collaboration between personal digital assistants (PDAs) and PCs. A toolset has been developed for the PDA user to manipulate remote desktop applications. Pebbles also provides a simple method for a group of users to take turns controlling applications running on a PC. In this application, PDAs are used as remote controls and as additional input devices for the PC.

Several researchers have also focused on methodologies for building multimodal interaction systems. The QuickSet project [7] involved a collaborative multimodal system that employed a distributed, multiagent architecture to integrate not only various user interface components, but also a collection of distributed applications. Current multimodal technologies duplicate interactive content on different channels (usually graphics and voice) in order to provide interaction alternatives. Johanson et al. [8] describe a multibrowser system that provides a framework for exploring multiple heterogeneous displays to view and browse information simultaneously. The UbicompBrowser project [9] explored the use of a handheld device to access and control surrounding output devices so that devices such as a standard television can be used to display Web pages.

Siemens Corporate Research [10, 11] conducts research activities for mobile and ubiquitous multimedia access with small screen and composite devices. Robertson et al. [12] describe the user interface issues related to a system in which the PDA is used as the remote control for an interactive television. Problems such as how to split information between devices are addressed as well. The ICrafter project [13] involved a framework for service user interface aggregation. This framework facilitates the creation of user interfaces for combinations of services, providing users the convenience of controlling several services simultaneously.

Multibrowsing, UbicompBrowser, ICrafter, and the Siemens research activities demonstrate a new perspective for mobile device augmentation. However, topics such as how to dynamically associate an application with the unpredictable user environment and how to develop the kinds of applications that can leverage heterogeneous discrete device capabilities have not yet been explored.

As described above, many researchers are actively developing multidevice collaboration systems. However, few attempts have been made to build user-centric or capability-on-demand systems that give mobile users enhanced experiences while leveraging the current infrastructure. The device-capability-on-demand (DCOD) framework and virtual device service gateway (VDSG) architecture, combined with the underlying virtual-device-oriented programming model presented in this paper, taken together, represent, we believe, the first attempt to build such a system.

The remainder of this paper is organized as follows: First, we present the motivation and vision for the project, along with sample application scenarios. Potential benefits to end users (i.e., experience improvements) are then presented, followed by an overview of a conceptual VDSG architecture. We then present technical details regarding the necessary gateway, the client-side components, and the virtual device service programming model. Finally, we present demonstration implementations and discuss possible future work.

Motivation

With the research knowledge we have gained, we observe and project that

- Mobile devices will continue to evolve and will become more powerful, but limitations on human–computer interaction will persist.
- Appliance design will evolve toward a special-purpose model [1]. Computer appliances will become inexpensive and widely deployed.
- Emerging location-sensing technologies will enable more accurate presence detection. In addition to infrared, radio frequency identification (RFID) [14], global positioning system (GPS) [15], Bluetooth, and the upcoming ZigBee** [16], numerous research activities are being conducted for presence or location detection with ultrasonic waves [17] or video cameras [18].
- Service discovery will be the cornerstone of any future platform that will provide user-centric services. Specifications such as Composite Capability/Preference Profiles (CC/PP) [19] and Bluetooth Service Discovery Protocol have been introduced for device capability description and service discovery.
- Wireless technologies are emerging as cable replacements and may potentially serve as system buses in a dispersed system. Wireless-enabled appliances—such as television [20], projectors [21], and printers—are already in the marketplace.

Vision

One critical factor regarding the fate of ubiquitous computing is the user's experience accessing information. While much work has been done to improve connection reliability and response time on both the server and network sides, there are still many areas in which we can enhance the device experience. With an awareness of the limitations of devices and the rapid evolution of the mobile computing environment, we initiated the research work on the DCOD framework to enable multidevice cooperation and federation. The framework is intended to turn various services, existing or future ones, into true user-centric services through these four approaches:

- *Sentience:* Perceive the user's ever-changing environment. With the help of wireless and sensory technologies in combination with service discovery protocols, we intend to make it possible to identify all available peripheral devices around a user for use in the potential service delivery.
- *Adaptation:* Obtain the user's instantaneous expectations. We will determine static user preferences, profiles, or habits as points of reference and apply common user-centric design principles to determine the proper methods for dynamic service delivery.
- *Association:* Instantiate a service in a given circumstance. We will select the suitable elements in the user's environment with respect to a specific relationship (e.g., location, ownership, privacy, or security policy) and link them together for a given service session.
- *Virtualization:* Provide the user with the most powerful virtual device available at the time of need. That is, we will make possible the collaboration of various different devices to deliver a service that produces the desirable user experience.

We envision a day in the future when many different devices can easily cooperate and federate to form useful and usable virtual devices to serve our daily life, providing a consistent single-device human–computer interaction metaphor. Thus, ubiquity and richness will be equally satisfied.

DCOD application scenarios

Despite the great progress made in network and communication technologies, business travel continues to be inevitable. The following scenarios are focused on a future businessman's customer visit:

- *Waiting for the flight:* John, a senior salesperson, is waiting for a flight to an important customer site. Suddenly, his mobile phone beeps, indicating an incoming message. It is an urgent e-mail from his colleague, Susan. The mobile phone is not capable of showing every detail in the message on its screen, since several large pictures are attached. However, it prompts John that a nearby private kiosk can do the job. John enters the kiosk and clicks the OK button on the mobile phone, and the entire e-mail message is shown instantly on the kiosk screen. Susan informs him that some changes must be made in the customer presentation. Unfortunately, it is time to board the flight.. . .
- *Work on the flight:* John remembers that he has to make some amendments to the customer presentation. However, his laptop is now "sleeping" in the overhead cabinet, and he hates to wake up the passenger beside him in order to retrieve it. However, by clicking an icon on his PDA, he makes a wireless connection to his laptop and starts editing the file using the on-flight personal liquid crystal display (LCD) and his PDA, while collaborating with Susan through voice and video.
- *In the customer building:* At the reception area in the customer building, John identifies himself and is granted a visitor badge which will take him to permitted areas and along a designated path only. He receives navigation instructions on appropriately located displays as he walks to the customer conference room. A central security system working behind the scenes tracks his movement and projects the directions on the displays along the route.
- *In the conference room:* By pressing an icon on the PDA, John wakes up his laptop sleeping in the bag and chooses the presentation file. The PDA prompts him that several wireless fidelity (WiFi)-enabled devices can be used to show the slides.
- *Making a presentation:* Several key managers and engineers enter the conference room, and it is time for the presentation. John selects the desired projector, and the presentation slides are shown on the projector screen. The PDA screen displays the speaker's notes, a presentation outline, and a projector control panel. When the agenda page of the presentation is about to be shown, John selects both the projector and the desktop screen, and the agenda appears on both. It stays on the desktop screen during the entire presentation. John has embedded several video clips in the presentation document to make it more animated. Whenever such a slide is about to be shown, the PDA prompts John that it will use the television in the room to display the video clip. While the video clip is playing, John uses his PDA as a remote control for the television.

The presentation is a success, largely because of the ease with which John is able to leverage the appropriate devices available to him in a simple yet dynamic manner.

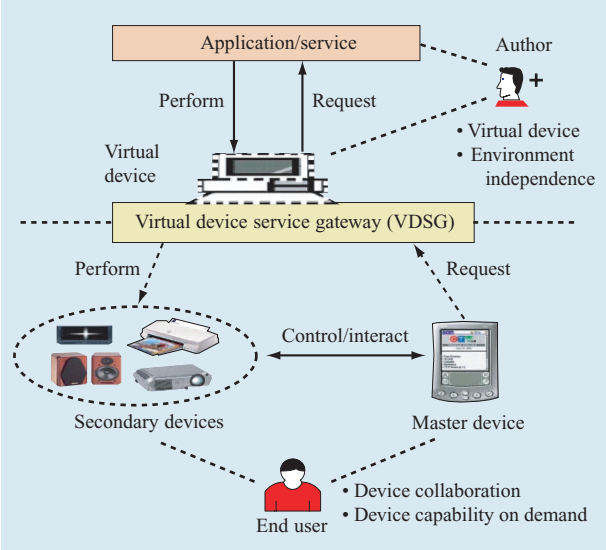


Figure 1

User experience of device capability on demand (DCOD).

DCOD user experience

Application capabilities and the experience of end users are two of the most important aspects of the life cycle of each information system, for their effectiveness largely determines the fate of the system. The DCOD framework aims to provide end users with a feature-rich experience through on-demand multidevice cooperation.

To avoid confusion when using multiple devices, the framework assumes that one mobile device carried by the end user acts as the master device. The master device is the initiator of the service request. When the master device is not capable of handling the service alone, it selects the appropriate secondary device or devices to assist. The master device then becomes the end user's control point to manipulate all selected secondary devices.

Since most application developers are familiar only with the single-device programming model, and most popular authoring tools are designed with this assumption, the DCOD framework introduces a new concept—a *virtual device.* A virtual device appears as one physical device, although its capabilities may be derived from many different physical devices. The virtual device conceals details of the user environment and multidevice collaboration. It reveals only the capabilities available to the end user, isolating the user from the underlying details of the multiple physical devices.

The framework provides a new programming model under which application authors may design applications according to specific needs, without binding the application to a single physical device for all of its runtime needs. At runtime, the DCOD framework maps the virtual device to the user's actual environment. **Figure 1** illustrates the new experiences the DCOD framework may bring to end users and authors.

VDSG architecture

As shown in Figure 1, a virtual device service gateway (VDSG) is required to deliver the promised user experiences of the DCOD framework. The VDSG is the core of the DCOD framework. By associating user environment capabilities with application-defined capability requirements, it assists applications that demand more capability than the primary device can offer.

As shown in **Figure 2**, the VDSG architecture comprises primarily the device manager, the security manager, the device federation engine, the content delivery engine, the administration interface, the virtual device service interface, and the virtual device Web adapter. The VDSG aims to support Web and non-Web applications, but since current efforts are focused on Web applications, the term *application* is used here to mean *Web application* unless otherwise specified.

Device manager

Discovery of device presence and capability is the cornerstone of any environment-dependent application, such as those required for this project. The primary job of the device manager is to discover and maintain information about devices that are available to and accessible by the mobile user. This is essential to support the ability of the device federation engine (see the description below) to map service requirements to the proper devices. An agent active on each device presents the capabilities of its host and its status using the CC/PP format (see the description below). The DCOD framework makes use of the location-based service gateway [22] to determine the vicinity relationship between master and secondary devices.

Device discovery is critically important, but, given the breadth of devices in use today, it is also difficult to implement. Each type of device has its own set of special capabilities that must be identified and prioritized to enable effective use of the available secondary devices.

Device federation engine

The device federation engine (DFE) matches device capability requirements from the application with the available secondary device information maintained by the device manager and selects proper candidate devices for application runtime binding. The DFE composes the appropriate aggregated best set of device capabilities from the set of devices available, given the specifications provided by the application. As shown in Figure 2, the DFE comprises the following entities.

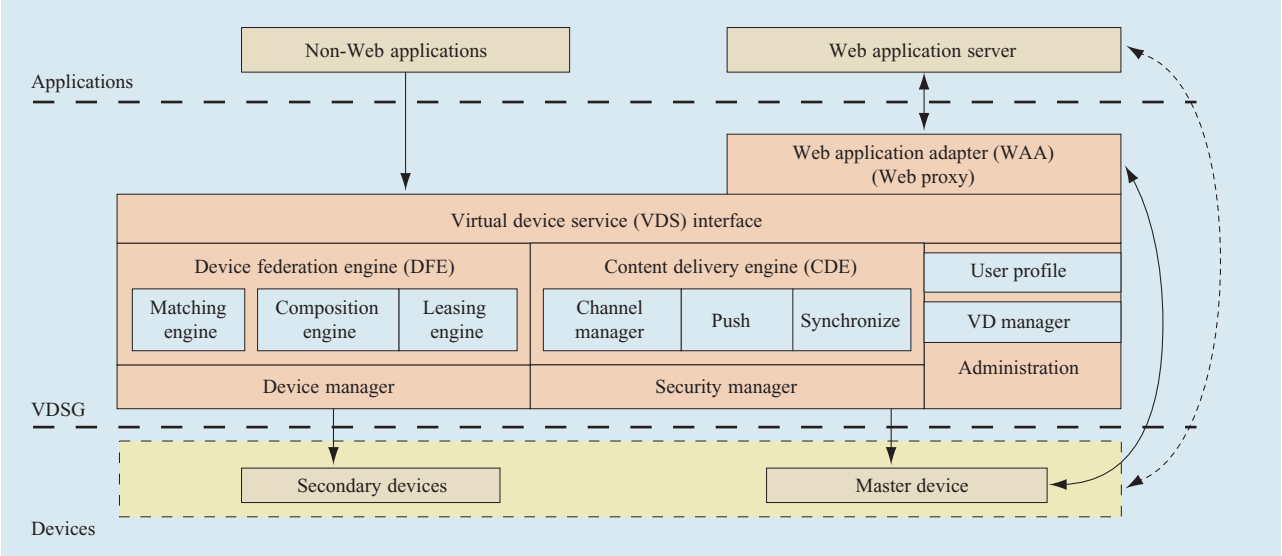


Figure 2

Virtual device service gateway (VDSG) architecture.

Matching engine

The matching engine takes application requirements and secondary device information as input parameters and selects candidate devices (device groups) that can be content object containers. Given that an application is composed of a set of pages, with each page containing element groups and elements, a content object may be a page, an element group, or an element. **Figure 3** illustrates the high-level workflow of the matching engine.

The application author uses a document profile (a CC/PP-like description—see the section on the virtual device programming model) to define the requirements of a content object. The device manufacturer (or, more specifically, the underlying device agent) uses CC/PP to describe the hardware and software capabilities of the device. The matching engine first attempts to discover all secondary devices that can directly execute a selected content object, and then seeks those devices that can process the object solely through media adaptation. The matching engine may also request the composition engine (see the description in the following section) to integrate discrete capabilities from different devices. Through adaptation and composition, the matching engine can provide more candidate devices (device groups) for the selected content object.

For example, suppose that an input element is about to be rendered in a Web page. The information is presented in the following format: *<input ref="username"> <label> user name </label> </input>*. Actually, this simple

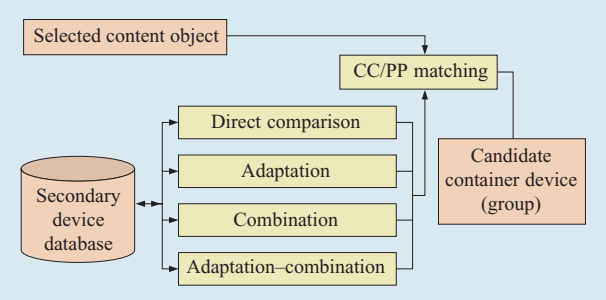


Figure 3

High-level workflow of matching engine.

content object requires both input and output capabilities. Suppose that the secondary device database currently contains four potential devices: a PDA, a screen, a microphone, and a speaker. **Table 1** illustrates all possible candidate devices for executing the input element.

For an application composed of a set of pages, the author may use a document profile to define the device capability requirements of each individual page. When associating devices with application requirements, depending on a user profile, the matching engine will first attempt to identify a single device that can execute the page. If a single device is not available, it will apply a *divide-and-conquer* tactic by separating the page element into groups over which the same matching algorithm will

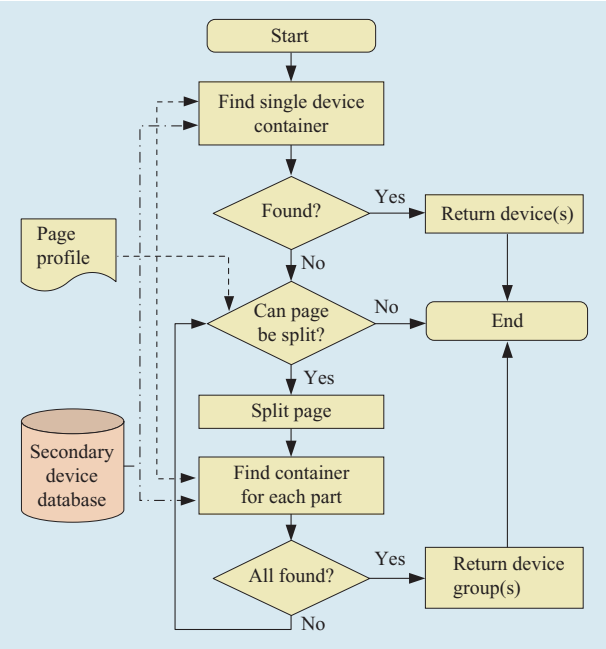


Figure 4

Page matching workflow process.

be executed. **Figure 4** illustrates the workflow of page matching.

The matching engine may generate more than one candidate device (a device group) for a given page. To facilitate decision-making, the DCOD framework must consult the user profile information to determine the variable wishes of the end user.

Composition engine

When the matching engine encounters a content object that cannot be executed by any single device but can be executed by using the combined capabilities of more than one device, the combination of devices is selected. For example, if it encounters a 10 × 10 data collection table with only a PDA and a projector available, it will use both devices to process the table. The composition engine is designed to create new virtual capabilities from more than one physical device. In this example, the composition engine combines the input of the PDA and the output of a projector to display the table and handle the application input. **Table 2** illustrates possible composition types. The first composition type, a keyboard/mouse from one device with a graphical user interface (GUI) being output to another, has been implemented in the DCOD framework.

Pebbles [6] provides a simple method of using a PDA to control a PC by using soft keyboard and stylus input on the PDA screen to emulate real PC keyboard and mouse operations. In this solution, the PDA and the PC will each install an agent to communicate with each other and interact with the local operating system, making their solution platform-dependent.

Under the DCOD framework, we chose a browser-based approach. The input objects that are run on the browser of one device capture local keyboard and mouse actions and turn them into browser document object model (DOM) events [23]. The DOM events are then transferred to the browser on the output device for rendering. **Figure 5** illustrates the high-level workflow of the input and output device composition.

Leasing engine

Under the DCOD framework, access to many of the secondary device capabilities is based on the concept of leasing. A *lease* is a grant of guaranteed access over a requested time period. Each lease is negotiated between the consumer and the provider of the capability. Leases are either exclusive or nonexclusive. Exclusive leases ensure that no one else may use the resource during the leasing period; nonexclusive leases allow multiple users to share a resource.

Table 1 Examples of a match engine. (I: input; O: output, CE: composition engine; TTS: text-to-speech engine; SR: speech recognition engine.)

Possible containers	*Method*	*Requirements*
PDA	Directly compare	None
Screen (O) + PDA (I)	Combine	CE
Speaker (O) + PDA (I)	Combine and adapt	CE + TTS
PDA (O) + microphone (I)	Combine and adapt	CE + SR
Screen (O) + microphone (I)	Combine and adapt	CE + SR
Speaker (O) + microphone (I)	Combine and adapt	CE + TTS/SR

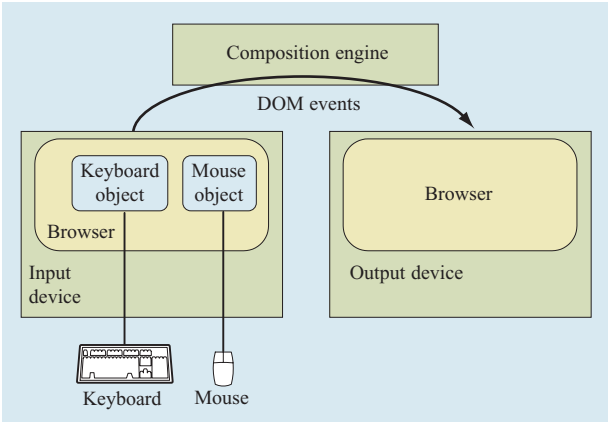


Figure 5

High-level workflow of the input and output device composition.

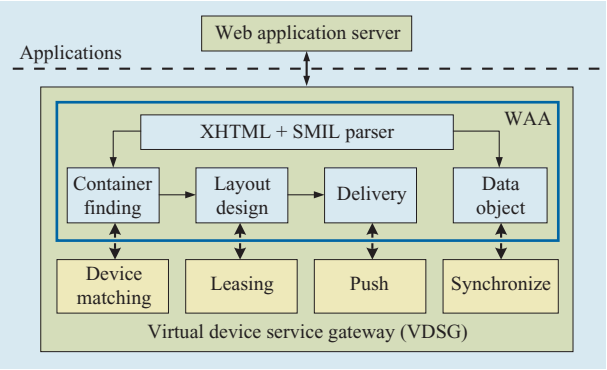


Figure 6

High-level architecture of the Web application adapter (WAA). (XHTML: Extensible Hypertext Markup Language; SMIL: Synchronized Media Integration Language.)

Content delivery engine

The content delivery engine (CDE) is responsible for delivering applications to the user environment devices that can provide the required functions. Each device function is defined as an application delivery channel (ADC). An ADC is registered to the CDE when the appropriate function is selected, and the CDE maintains life cycles of all ADCs related to an application. Since an application page might be split and executed on multiple devices, the order of page part delivery and the event exchange among page parts are most important. In addition, the variables, especially global variables that appear in multiple page parts, must be synchronized. The synchronization component of the CDE controls this critical process.

The push component of the CDE is responsible for delivering page parts to appropriate ADCs. ADCs can be categorized as *fat* or *thin* according to the intelligence and complexity of their providers. The provider of a fat ADC is smart enough to process complex communication protocols and data stream formats (e.g., Web browser, media player, word processor). The provider of a thin ADC can provide only simple data stream types over the network. How a page part is delivered to an ADC depends largely on the ADC type. For example, when delivering a page part to a fat ADC, the content may be kept in its original format, but for a thin ADC, the page part has to be transformed into raw data streams appropriate for that device.

Web application adapter

The Web application adapter (WAA) acts as a Web proxy server so that when the master device sets the WAA as its Web proxy, the WAA can capture responses of the service requested by the master device. The WAA is the bridge for Web applications to fulfill virtual device functionalities. **Figure 6** illustrates the high-level architecture of the WAA and its relationship with other components of the VDSG.

Table 2 Possible device composition types.

Composition type	*Example*
GUI (O) + Keyboard/mouse (I)	Wall-screen (O) + PDA (I)
GUI (O) + voice (I)	Wall-screen (O) + speaker (I)
Voice (O) + Keyboard/mouse (I)	Speaker (O) + IBM ThinkPad* (I)
MultiGUI (O)	Two small screens to create a bigger one
Multivoice (O)	Multiple speakers play different parts of the same source
Multikeyboard/mouse (I)	Use PDA handwriting and keyboard of a PC to input
Keyboard/mouse (I) + voice (I)	Use keyboard and microphone to input

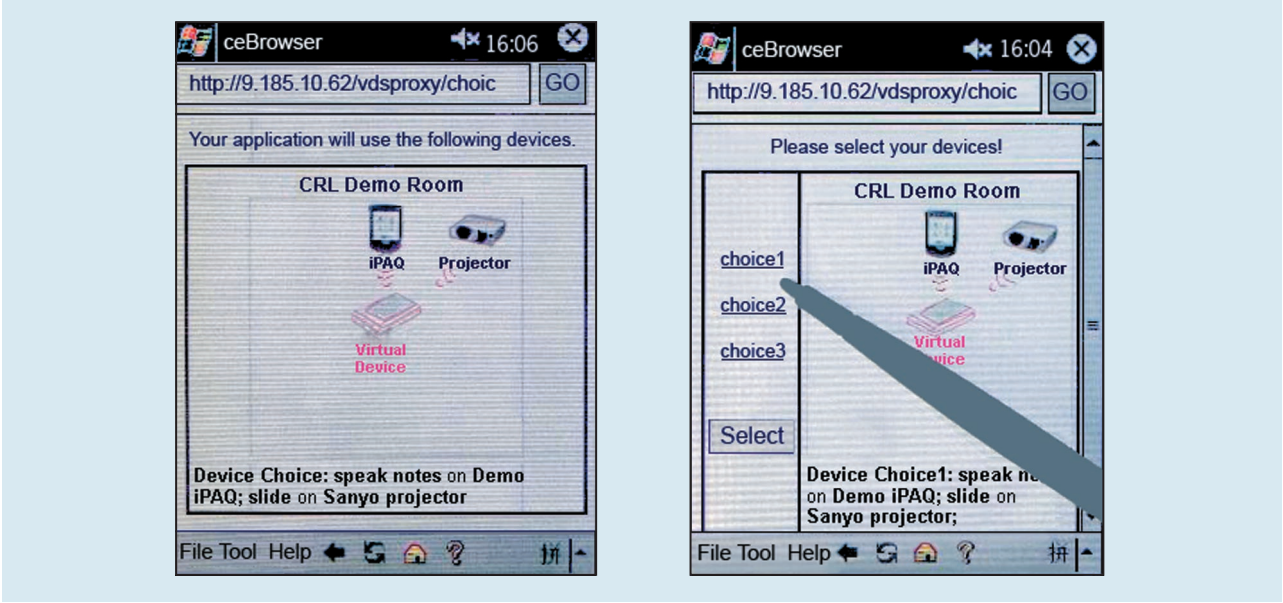


Figure 7

Confirmation page and selection page.

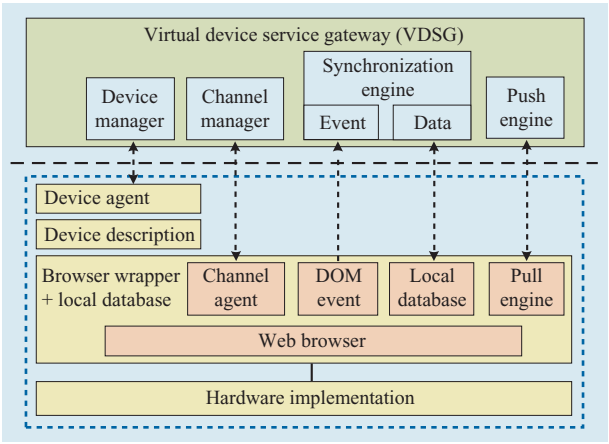


Figure 8

Client-side components.

The WAA parses the service response transmitted via HyperText Transfer Protocol (HTTP) and abstracts service requirements. Next, the WAA calls functions of the device matching engine to find an appropriate candidate device (device group) to process the service response. When the leasing component has ascertained that the selection is appropriate, the WAA redesigns the layout of the service interface according to the interaction features of the candidate device, and then calls the push component to deliver the service response to the selected device. For those variables that exist in two or more separated page parts, the WAA replicates the information in each of the related parts and utilizes the synchronization component to maintain their consistency.

Administration

The administration component maintains a user profile for the device selection preferences, including place, service, and device selection, in order to help the system select a proper candidate if the matching engine provides more than one choice. Once a choice is made by the system, the administration task generates a confirmation interface to inform the end user of the secondary device choice. The administration task also provides an alternative approach for manual device selection. This allows the end user the flexibility of making a personal choice. **Figure 7** shows the wizard interface of the confirmation and selection page generated by the administration task.

Security manager

In principle, the role of the security manager is to monitor communication among the master device, the secondary devices, and the VDSG, and to handle security across multiple devices. The security manager includes support for sensor networks used by the VDSG to detect the presence and status of secondary devices and the networks interconnecting the devices.

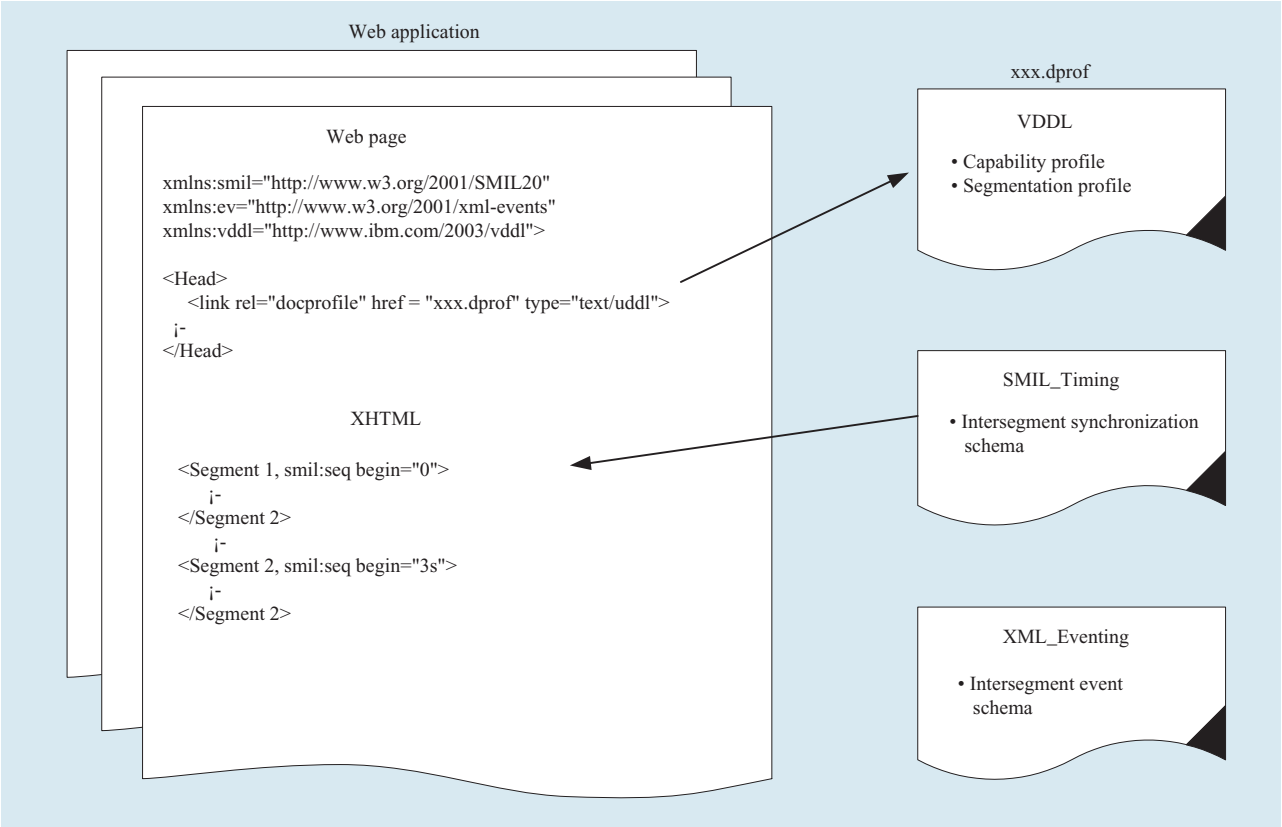


Figure 9

Framework of a virtual device application. (XHTML: Extensible Hypertext Markup Language; VDDL: Virtual Device Description Language; SMIL: Synchronized Media Integration Language; XML: Extensible Markup Language.)

Security issues are another significant area in the DCOD research activities, which are needed to address problems such as multidevice authentication and multidevice encryption. No major breakthrough has yet been made, but we have identified that the working scope should focus on *ad hoc* network security.

Client-side components

As described in the previous section, a client-side device is composed of hardware and one or more ADCs that represent the capabilities of the device. **Figure 8** illustrates the architecture of the device. A client-side device comprises primarily a device agent, a device description, an extended Web browser, and hardware.

The device agent implements the discovery protocol used by the client device to detect the existence of the VDSG and register itself. The client-side stack also includes a leasing protocol to support leasing requests.

The device description component is used to describe the capabilities of the client-side device, including its hardware characteristics and the features of each ADC. CC/PP is used to describe the capabilities.

A Web browser is now embedded in many computing devices, but common browsers must be enhanced with a wrapper to become an available ADC. The wrapper comprises a channel agent, a DOM event manager, a local database, and a proactive pull engine. The channel agent registers the ADC to the VDSG when selected by the matching engine; it then takes control to provide for channel life cycle management. Both the DOM event component and the local database component are used to synchronize with other browsers. The proactive pull engine component enables the Web browser to support push operations.

Virtual device programming model

The DCOD framework provides mobile users with enhanced capabilities by breaking the physical boundaries of devices, enabling delivery of a service to multiple devices. To determine which devices should be selected

```
<?xml version="1.0" ?>
 <vdsgroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="VDS_VDDL_schema.xsd">
<vdsgroup>
 <groupset>
        <group>
                <group_id>slide</group_id>
                <source>/html/frameset/frameset/frameset/frame[@id="show"]</source>
                <device_requirement>
                        <requirement_item>largeScreen</requirement_item>
                        <requirement_item>mediumScreen</requirement_item>
                </device_requirement>
                <device_requirement>
                        <requirement_item>fullColor</requirement_item>
                </device_requirement>
                <device_requirement>
                        <requirement_item>movie</requirement_item>
                </device_requirement>
        </group>
</groupset>
<groupset>
        <group>
                <group_id>navigationbar</group_id>
                <source>/html/frameset/frame[@id="navbar"]</source>
                <device_requirement>
                        <requirement_item>smallScreen</requirement_item>
                </device_requirement>
        </group>
        <group>
                <group_id>outline</group_id>
                <source>/html/frameset/frameset/frame[@id="outline"]</source>
                <device_requirement>
                        <requirement_item>smallScreen</requirement_item>
                </device_requirement>
        </group>
        <group>
                <group_id>note</group_id>
                <source>/html/frameset/frameset/frameset/frame[@id="note"]</source>
                <device_requirement>
                        <requirement_item>smallScreen</requirement_item>
                </device_requirement>
        </group>
</groupset>
</vdsgroup>
```

Figure 10

Document profile sample.

and how to split a service across devices, the framework provides a language to describe device capability requirements and instructions on how to split a service, known as the Virtual Device Description Language, or VDDL. VDDL is a markup language that includes a set of device capability classes and Boolean expressions. The application developer uses VDDL to define the capabilities of his desired device and the virtual device, and then designs the service modules on the basis of virtual device requirements. At runtime, the system associates the virtual device with the user environment, enabling the virtual device environment.

The basic virtual device programming model is XHTML + VDDL + SMIL + XML, as shown in **Figure 9**. XHTML is used to structure and lay out application modules; the SMIL timing mechanism is used to define the order of segment delivery, and the XML event mechanism is used to define intersegment events for service delivery and data synchronization. VDDL includes two parts, the capability profile and segmentation profile, which together define the service profile.

Figure 10 illustrates a sample service profile written in VDDL. In this example, the system recognizes that the presentation documents can be split into two parts. One

part shows the slide, which requires a large screen with full color and movie support. The other part shows the navigation bar, the outline, and the speaker notes, which require only a small screen. In a real environment, the large screen content might be delivered to a projector and the speaker notes to a PDA.

Implementation and demonstration

We have completed two DCOD demonstrations for the future facilitation of *Office Environment* and *Mobile Worker*. The first demonstration shows an enterprise visitor browsing the enterprise Web site and presents slides by using a dynamic combination of his Pocket PC PDA and surrounding equipment. The second demonstration shows a mobile worker who receives, browses, and replies to e-mails via his mobile phone with the help of a nearby flat-screen television.

The first demonstration system consists of a Pocket PC PDA, some networked utilities with RFID tags, and an application server. RFID is used for presence, location, and rough orientation detection. The application server provides a Web interface for users to browse enterprise services that are available in their current area. We put client-side software components into utilities such as the printer, wall-screen, television, projector, and speaker interconnected by a common access mechanism over HTTP. In this system, the PDA is used as a console device to access information and services from the enterprise application server. The VDSG first determines available resources surrounding the PDA by tracking its position. Next, it invokes proper viewers to render multimedia content on selected displays and prints document handouts on nearby printers in order to augment the projector for an advanced presentation scenario. Since the implementation is based on HTTP, only an extensible Web browser is required on the PDA.

In this demonstration, an enterprise visitor is granted a PDA with a wireless connection to enterprise services. The visitor can then learn more about the local environment through the PDA browser. Web pages customized for the PDA are shown on directly on the PDA screen, but when the system encounters rich content, such as large pictures, video, or audio clips, it picks proper secondary devices and renders the content. **Figure 11** is a snapshot of the demonstration.

In another demonstration scenario, the visitor makes a presentation. In this demonstration, the visitor can make use of all necessary devices in the conference room—e.g., a projector for the slides, a PDA for the speaker notes and to use as the control panel, and a television for embedded video—to make an impressive and effective presentation. Compared with the previous scenario, this demonstration includes an additional application, a presentation service, which runs on the application server.

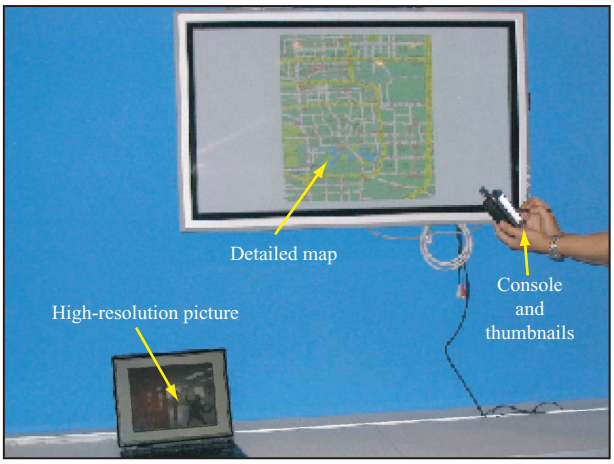


Figure 11

An information access demonstration.

Via a Web browser on his PDA, the visitor uploads his presentation documents, created with common tools such as Microsoft PowerPoint**, to the service, and the service then converts the presentation documents into XHTML format. Next, the service inserts VDDL tags into each of these XHTML pages. These tags literally separate each page into four groups (slide, navigator, outline, and speaker notes) and define the device capability requirements for each group rendering. Finally, the service publishes the processed XHTML pages on the Web server. The visitor can enjoy the same experience demonstrated in the previous scenario, but this time he makes a presentation. **Figure 12** shows the demonstration setup.

The second demonstration system is basically built on the same hardware and software environment as the first demonstration system, except that a mobile phone replaces the Pocket PC PDA and an additional mail application is provided. The mobile phone connects and registers to the VDSG via a Bluetooth gateway. The Bluetooth gateway also reports the VDSG position of the mobile phone, which then helps the VDSG to establish the vicinity relationship between the mobile phone and other equipment in the same room. The mail application runs on the application server and works as a mail server demon, monitoring incoming e-mail for registered users. When an e-mail arrives, the mail application sends a notification to the user's mobile phone via the Bluetooth gateway. The application then generates an XHTML page containing the message header, the body, and the interaction menu. The application then inserts VDDL tags into the XHTML page and publishes the page to the Web server. The uniform resource locator (URL) is contained in the message sent to the mobile phone to notify the

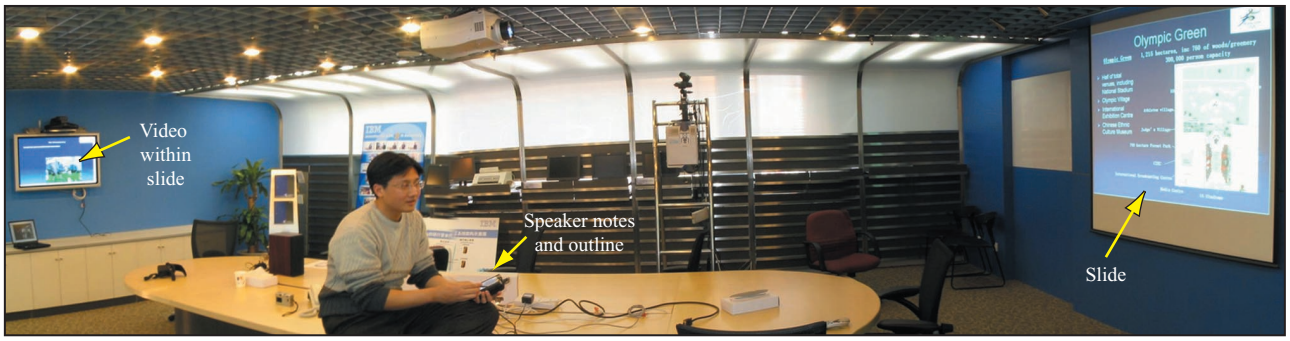
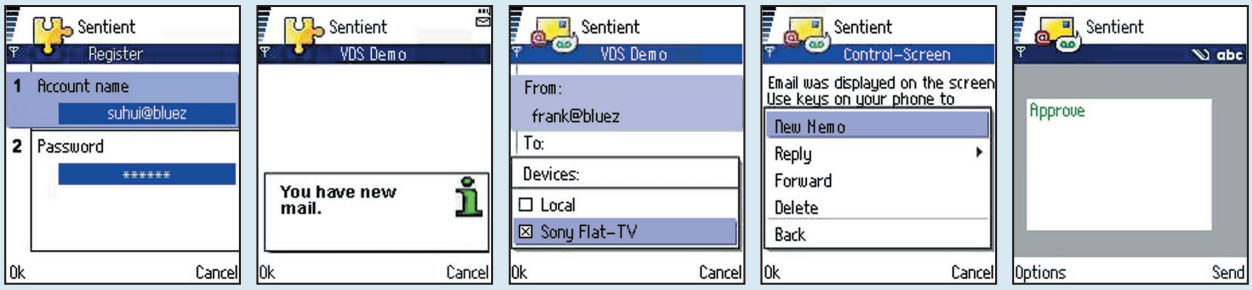


Figure 12

Presentation demonstration.



Register Notify Select Reply Input

Figure 13

Mobile phone screens in the mail process demonstration.

user that mail has arrived, The mobile phone agent redirects the request to the the VDSG and the XHTML page.

Interaction is the major enhancement in this demonstration. The user can use the e-mail application menu shown on the mobile phone screen to receive new e-mail, retrieve an existing e-mail, or delete a message. The user can also show the body of the e-mail on a flat-screen television, where it can be scrolled and the user can move through input fields by pressing the arrow keys of his mobile phone. Finally, the user can input mail content using the mobile phone keyboard. The input on the mobile phone can be synchronized with the television screen. **Figure 13** shows screen captures of the mobile phone screen in the demonstration.

Conclusion and outlook

This paper has presented an innovative framework that allows devices available to a mobile user to collaborate, enabling enhanced user experiences. The framework also gives developers a new authoring platform—the virtual device. The framework makes use of the virtual device service gateway, which collects device information in the user environment and associates service and application requirements with the proper devices in order to deliver enhanced services. The framework also defines a virtual device programming model for the application developer to create Web applications that benefit from federated client device capabilities.

To date, we have conducted a series of user studies and generated typical usage scenarios. We designed and implemented the prototype of the server-side VDSG, designed and implemented parts of the virtual device programming model for Web applications, and created the demonstrations. Other significant areas in device capability on-demand research which require further work include multidevice security mechanisms, a virtual device programming model for non-Web applications, and an embedded version of VDSG to support thin-client devices.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Bluetooth Special Interest Group, UPnP Forum, Sun Microsystems Incorporated, Salutation Consortium, ZigBee Alliance, or Microsoft Corporation in the United States, other countries, or both.

References

1. W. Buxton, "Less is More (More or Less)," *The Invisible Future: The Seamless Integration of Technology into Everyday Life*, P. J. Denning, Ed., McGraw-Hill Book Co., Inc., New York, 2001, pp. 145–179.
2. The Official Bluetooth Web site; see *http://www.bluetooth.com/*.
3. UPnP Forum; see *http://www.upnp.org/*.
4. Jini Community; see *http://www.jini.org/*.
5. The Salutation Consortium; see *http://www.salutation.org/*.
6. B. Myers, "The Pebbles Project: Using PCs and Hand-Held Computers Together; Demonstration Extended Abstract," *Proceedings of the ACM Conference on Computer–Human Interaction (CHI 2000): Human Factors in Computing Systems*, April 2000, pp. 14–15.
7. P. R. Cohen, M. Johnston, D. McGee, D. Smith, S. Oviatt, J. Pittman, L. Chen, and J. Clow, "QuickSet: Multimodal Interaction for Simulation Set-up and Control," *Proceedings of the 5th Applied Natural Language Processing Meeting*, 1997, pp. 31–40.
8. B. Johanson, S. Ponnekanti, C. Sengupta, and A. Fox, "Multibrowsing: Moving Web Content Across Multiple Displays," *Proceedings of the 3rd International Conference on Ubiquitous Computing*, 2001, pp. 346–353.
9. M. Beigl, A. Schmidt, M. Lauff, and H.-W. Gellersen, "The UbicompBrowser," *Proceedings of the 4th European Research Consortium for Informatics and Mathematics (ERCIM) Workshop on User Interfaces for All*, 1998.
10. T.-L. Pham, G. Schneider, and S. Goose, "A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access Using Small Screen and Composite Devices," *Proceedings of the 8th ACM International Conference on Multimedia*, 2000, pp. 323–331.
11. G. Schneider, S. Djennane, T.-L. Pham, and S. Goose, "Multimodal Multi-Device UIs for Ubiquitous Access to Multimedia Gateways," *Proceedings of the 17th International Joint Conference on Artificial Intelligence: Workshop on Artificial Intelligence in Mobile Systems (AIMS)*, 2001, pp. 61–64.
12. S. Robertson, C. Wharton, C. Ashworth, and M. Franzke, "Dual Device User Interface Design: PDAs and Interactive Television," *Proceedings of the Conference on Human Factors in Computing Systems*, 1996, pp. 79–86.
13. S. R. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments," *Proceedings of the Third International Conference on Ubiquitous Computing*, 2001, pp. 56–75.
14. AIM RFID Web site; see *www.aimglobal.org/technologies/rfid/*.
15. GPS World Web site; see *http://www.gpsworld.com/gpsworld/*.
16. The Zigbee Alliance; see *http://www.zigbee.org/*.
17. A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Pers. Commun.* **4,** No. 5, 42–47 (October 1997).
18. B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, "EasyLiving: Technologies for Intelligent Environments," *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing*, 2000, pp. 12–29.
19. Composite Capability/Preference Profiles (CC/PP); see *http://www.w3.org/Mobile/CCPP/*.
20. Ars Technica Newsdesk; see *http://arstechnica.com/archive/news/1062698930.html*.
21. Wi-Fi Planet; see *http://www.WI-FIplanet.com/news/article.php/1566601/*.
22. IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China; location-based service project; see *http://www.research.ibm.com/beijing/updates/lbs.html*.
23. Document Object Model (DOM); see *http://www.w3.org/DOM/*.

Received October 16, 2003; accepted for publication February 4, 2004; Internet publication September 7, 2004

Rong Yao (Frank) Fu *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (furongy@cn.ibm.com).* Mr. Fu is a Staff Research and Development Engineer in the Future Client and User Paradigm Research Department. He received B.S. and M.S. degrees in computer science from Northwest Polytechnic University in 1995 and 1998, respectively. Subsequently, he joined IBM, where he has worked on ubiquitous computing. Mr. Fu received an IBM Research Division Award in 2000 and IBM Invention Achievement Awards in 2001 and 2002. He has filed or published 13 patents.

Hui Su *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (suhui@cn.ibm.com).* Dr. Su received B.S. and Ph.D. degrees from the Institute of Information at Tsinghua University in 1992 and 1996, respectively. He has been a Senior Research Member and manager of the Future Client and User Paradigm Research Department since 1996. He became a member of the Association for Computing Machinery in 2001. Dr. Su's research interests include interaction technologies between humans and computers, user experience, the impact on the ubiquitous computing infrastructure and programming model, and knowledge-worker experience. He has filed or published 18 patents.

James C. Fletcher *IBM Ubiquitous Computing Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (fletchj@us.ibm.com).* Mr. Fletcher is a Senior Technical Staff Member and chief architect in the IBM Ubiquitous Computing Division. He received a B.S. degree in computer science from North Carolina State University in 1976. He later joined IBM and spent much of his career focused on computer networking prior to joining the Ubiquitous Computing Division, where he focuses on enterprise solutions. He has received numerous IBM Outstanding Technical Achievement Awards during his career. Mr. Fletcher has authored more than 40 technical journal articles and coauthored three books. He holds numerous patents in the areas of networking and ubiquitous computing.

Wei Li *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (weil@cn.ibm.com).* Mr. Li is a Research Staff Member in the Future Client and User Paradigm Research Department. He received an M.S. degree in computer science from Tsinghua University in 1998. He subsequently joined the IBM China Research Laboratory, where his work has focused on ubiquitous computing. Mr. Li has published more than ten patents in the area of ubiquitous computing.

Xiao Xi Liu *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (liuxx@cn.ibm.com).* Mr. Liu is a Research Staff Member in the Future Client and User Paradigm Research Department. He received B.S. and M.S. degrees in electronic engineering from Northwestern Polytechnical University in 1998 and 2000, respectively. Mr. Liu subsequently joined the IBM China Research Laboratory, where he has worked on ubiquitous computing and relevant research.

Shi Wan Zhao *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (zhaosw@cn.ibm.com).* Mr. Zhao is a Research Staff Member in the Future Client and User Paradigm Research Department. He received B.S. and M.S. degrees in computer science from Tsinghua University in 1998 and 2000, respectively. Mr. Zhao subsequently joined the IBM China Research Laboratory, where he has worked on multimodal interaction and ubiquitous computing.

Chang Yan Chi *IBM Research Division, IBM China Research Laboratory, 2/F, Haohai Building, No. 7, 5th Street, Shangdi, Haidian District, Beijing 100085, People's Republic of China (chicy@cn.ibm.com).* Ms. Chi is a Staff Research and Development Engineer in the Future Client and User Paradigm Research Department. She received a B.S. degree in electronic science and technology from the Harbin Institute of Technology in 1990, and an M.S. degree in electronic engineering from the Harbin Engineering University in 1993. In 1998 Ms. Chi joined the IBM China Research Laboratory, where she works on multimodal interaction and ubiquitous computing.