Integrating the Physical World with the Web to Enable Context-Enhanced Services

Philippe Debaty, Patrick Goddi, Alex Vorbau
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-192
September 8th , 2003*

E-mail: philippe.debaty@hp.com, patrick.goddi@hp.com, alex.vorbau@hp.com

ubiquitous computing, pervasive computing, Web presence, context-aware, location-aware, smart spaces, Cooltown

In Cooltown, we believe that the integration of our physical world with the Web offers unique opportunities to enable ubiquitous computing applications. This paper describes our latest results in building a model and a software architecture called the Web presence manager (WPM) to support this physical-virtual integration. This software layer implements and specifies the services and information provided by Web representations of physical entities such as people, places, or things. We detail an extensive context-enhanced media-oriented application built on top of our platform. Our application enables mobile and context-aware access to personal contents and rendering on local appliances in a variety of ubiquitous computing environments.

* Internal Accession Date Only
Approved for External Publication

Integrating the Physical World with the Web to Enable Context-Enhanced Services

Philippe Debaty, Patrick Goddi, Alex Vorbau

Hewlett-Packard Laboratories
1501 Page Mill Road – MS 1138
Palo Alto, CA 94304
philippe.debaty@hp.com, patrick.goddi@hp.com, alex.vorbau@hp.com

Abstract. In Cooltown, we believe that the integration of our physical world with the Web offers unique opportunities to enable ubiquitous computing applications. This paper describes our latest results in building a model and a software architecture called the Web presence manager (WPM) to support this physical-virtual integration. This software layer implements and specifies the services and information provided by Web representations of physical entities such as people, places, or things. We detail an extensive context-enhanced media-oriented application built on top of our platform. Our application enables mobile and context-aware access to personal contents and rendering on local appliances in a variety of ubiquitous computing environments.

1 Introduction

In a little more than a decade the Web has exploded in capability and value. Where it once contained static web pages, and later dynamic personalized content it now is on the verge of massive integration with our daily activities. We communicate with our Yahoo buddies, are judged by our eBay reputation, and offered opportunities to sell our old Amazon purchases when someone else wants them. The last barriers of access are being overcome; the remaining barriers are the ways in which our physical world activities are integrated into the world of web information and services. In essence, whether I am at work, at home or on the road, I would like to say: 'I am here, what can I accomplish? What can I experience?' The Web is leaving the browser and entering the real world.

Where portals like MyYahoo organize around our online interests, new types of portals will organize around our physical environment. My new portal should take into consideration all the local capabilities, the capabilities I carry with me, and the services to which I subscribe to deliver a customized interface, and a capture and playback experience. It should not be decoupled from my physical environment, but instead leverage both the value of the web and the capabilities of the local environment.

To address this need, we introduce a distributed software framework to integrate the physical environment with the Web. Our approach is to establish and exploit a network of web presences for physical entities. A web presence is an independent piece of software that represents a physical entity such as a person, a place or a thing. A conference room, a printer, a projector, a stereo, a paper book, and a human user are all examples of physical entities that can have a web presence. Each web presence manages up-to-date information about its physical counterpart and presents it on the web for use by other web services or web users.

In the following sections, we first describe a concrete example scenario to motivate our work. Then we situate our specific contribution with respect to related work in the field. This brings us to the detailed description of the architecture and the open-source implementation of our software platform. Finally, to challenge the validity of our system, we present an extensive example application that we developed using our platform and that fully supports our example scenario.

2 An Example Scenario

Figure 1 Alex and Philippe in our "Web presence-enabled living room"

Patrick enters his living room carrying his PDA. The room is equipped with a Plasma display, a sound system and a digital picture frame. Patrick turns on his PDA and opens a browser window to his personal context-enhanced portal on the Web. The portal has already detected his physical environment and presents him with a view that allows him to locally play his personal music, videos, and pictures. With a tap on the screen, he chooses to play a U2 song on the local sound system. He receives a Web form on his PDA that lets him control the music playback.

Alex and Philippe enter the room and hear the music playing on the stereo. Excited, Alex says: "It reminds me of this U2 song that I really like. Remember, we listened to it last week when we were at Philippe's place."

Patrick looks at his PDA. His portal page also shows a set of hyperlinks to context-enhanced lists: "people in the room", "files used in this room", "files used with these people". He selects the last one and is now presented with a list of files recently used when Alex and Philippe were around. He quickly finds the song Alex was talking about and plays it on the sound system.

In the meantime, using his own PDA, Philippe controls the digital picture frame and the Plasma display to display his pictures and videos of the U2 concert he went to last weekend. They all start talking about it.

The next day, Patrick is at work and enters a conference room where he is going to give a presentation. The conference room is equipped with a printer and a projector. Using his PDA, he accesses his personal context-enhanced Web portal, which has adapted to his new environment. With a few taps on the PDA screen, he uses the projector to display his slides and prints a few supporting documents on the local printer.

3 Related Work

The work presented in this paper was conducted as part of a research initiative at HP laboratories called Cooltown [3]. Our research is exploring the convergence of nomadic [1] and ubiquitous [2] computing with the Web. Cooltown specifies adopting and extending Web protocols and standards for use in ubicomp environments. In particular, Cooltown describes a vision of people, places, and things having a Web presence [4]. A sub-project was started to explore information and services Web presences could provide to Web users and applications. Some of our early results explored issues related to dynamically generating Web pages that represented relationships between physical entities [6], and an exploration of using this technique for representing places [5]. In this paper we report on our more recent enhancements to the model and platform specifically looking at integrating the service model of the Web with Web-present people, places and things.

Many other past or present research and industry efforts are contributing to the fields of context-awareness and smart environments.

The Context toolkit project [7] (Georgia Tech), part of the Aware Home Research Initiative, aims at providing a distributed software framework to build context-aware applications. Their approach is to use programming paradigms similar to UI design. They introduce context widgets that provide applications with access to context information while hiding the details of context sensing. Our approach complements this work in that we use a Web paradigm to represent the physical environment instead of a UI design paradigm.

QosDream [8] (University of Cambridge) is a research platform for the development and management of location-aware multimedia applications. Like the context toolkit, it supports and hides the details of a variety of location sensing mechanisms such as mobile phones, GPS, active badges, etc. Their open source framework FLAME provides location awareness to applications through CORBA interfaces. Like the Context toolkit, this work does not attempt to integrate the physical world with the Web.

The Interactive Workspaces project [9] (Stanford) introduces an Interactive room operating system (iROS) to support the development and deployment of applications running in a smart environment. Their system and applications enable and exploit the collaboration and aggregation of devices within the bounds of a specific local physical space.

Similarly, the Easy Living Project [10] (Microsoft Research) was focusing on providing architecture and technologies for smart environments that allow the dynamic aggregation of devices into a single coherent user experience. Their architecture consisted of a middleware layer (inConcert) to facilitate distributed computing, an extensive geometric world model, sensing support and service description. Like the Interactive Workspaces project, their work was focusing on a room-centric approach in a highly sophisticated controlled environment.

Several other projects such as the RoomWare project [11] (GMD-IPSI, Darmstadt) or the GAIA project [12] (UIUC) take a similar approach. Although these systems support dynamic changes in the configuration of the environment, they are not focused on mobility issues such as supporting users constantly moving to many different environments with sometimes very limited capabilities. Mobility in a variety of environments is a main focus of our work. By integrating the physical world with the Web, we attempt to lower the barrier of deploying context-aware applications.

In the industry, the emergence of standards such as UPnP[18], RendezVous[20] or Jini[19] shows the current trend of considering devices and their services as peers of the network as opposed to peripherals of the PC. Those technologies address the issues of network discovery and control of these devices. However, they provide very limited physical discovery. A user can detect the devices on the network but cannot determine which ones are right next to him/her, unless the network is small enough to imply co-location. Our goal is to exploit these technologies when possible and complement them to enable fine-grained context-awareness.

4 Architecture

4.1 Overview

Our approach is to introduce a distributed software layer to integrate the physical environment with the Web. The goal of this layer is to enable and facilitate the development of context-enhanced applications. It can be viewed as a two-way bridge between the physical world of people, places and things and the virtual world of Web resources.

In the rest of this section, we first motivate our choice of using the Web as the basis of our architecture. Then we present the common needs and problems of context-aware applications. This brings us to describing the model of the physical world that we chose to address these needs. Finally, we detail the organization and the components of our architecture.

4.2 Building on the Web

This work has its roots in the Cooltown vision. As in [4], we believe that the Web is the most promising basis for ubiquitous computing systems because of its low barrier to entry, its device, language and service independence, its content centric model, its depth of features and its ubiquity. Building on the Web increases our chances of having a distributed, ubiquitous and scalable software layer. Our goal is to extend the Web to support dynamic context-awareness and physical world integration while preserving its desirable properties.

4.3 Context-Awareness

Context-aware applications usually need or use the following kind of physical context information:

- *Where*: The location of the user can be expressed in terms of absolute coordinates but also in a semantic way, in terms of room or place he or she is in.
- *When*: The current time can be useful in archiving and later retrieving information.
- *Who*: Who is the user and who is around the user?
- *What*: What things or devices are near the user? What can he or she do with it? What are the physical characteristics of physical entities around him?
- *How*: How are those objects and other persons organized. For instance, are they all in the same room? How can the user control a nearby device?

To address those questions, context-aware applications typically develop and use:

- *A model to represent the physical world*: The model simplifies the complexity of the world to consider only the information that is useful for the application. It also organizes and expresses this information so that it is usable by machines.
- *An implementation of the model*: Whereas the model is a mathematical abstraction, the implementation defines how the information is actually stored and exploited by computers and devices. For instance, an implementation can be distributed or centralized, use a relational database or a flat file system, etc.
- *One or more sensing mechanisms*: Sensors are used to gather raw data relative to the physical environment and automatically feed the model implementation. GPS, Cellular phone-based location systems, infrared beacons, RFID tags and readers, ultrasonic transceivers, Bluetooth and network-level location tracking are examples of commonly used context-sensing mechanisms.

In many context-aware applications the physical world model and implementation are tightly coupled to the sensing mechanism. For instance, when using a GPS device, the model and implementation are based on an absolute coordinates system. However, this approach does not allow the support of multiple sensing mechanisms. Most ubiquitous computing platforms, as those described in section 3, recognize this issue and decouple the model and implementation from the sensors. We make a similar choice. In the next section, we present a model driven by the typical needs of context-aware applications rather than by the sensor modality. Our approach is to lower the barrier of deploying and exploiting this model by leveraging the structure of the Web.

4.4 A Web-like Physical World Model

Our physical world model is designed to be tightly integrated with the Web and mimic its structure. The Web is populated with hyperlinked Web pages and Web services accessible by URLs. A user browsing the Web navigates easily from pages to pages using hyperlinks embedded in the pages. In parallel, computer applications also leverage the Web using technologies such as XML and SOAP to communicate and WSDL and UDDI for discovery

and advertisement. Our goal is for physical world entities to be integral parts of this highly interlinked virtual structure of the Web for both users and applications.

Thus, we consider the physical world as a dynamic set of interrelated *physical entities* such as people, places or things, just like the Web is a set of hyperlinked Web resources. Once implemented, this modeling approach will allow the interlinking between Web resources and Web representations of physical entities. Figure 2 illustrates our simple yet extensible model.

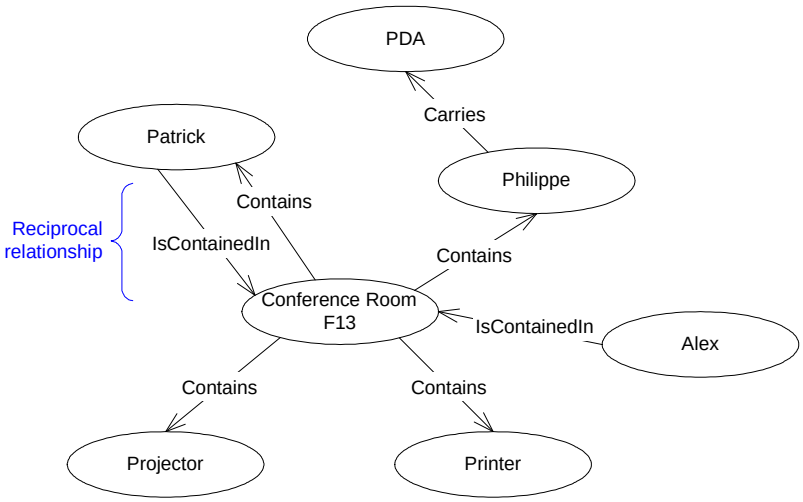

Figure 2 Example view of the physical world using our model

Each physical entity can be described by a set of characteristics such as its type, its identity, its physical attributes and state. This description is the equivalent of the information delivered by a Web resource. In addition, just like a Web resource is hyperlinked to other Web resources, a physical entity is related to other entities. For instance, a conference room is a physical entity and it has *relationships* with the physical entities it contains such as a projector, a printer, a few people, etc.

To be able to address the most common context questions presented in section 4.3, we characterize further those relationships by introducing a hierarchy of relationship types and an extensible set of properties. “Contains”, “isContainedIn”, “isNextTo”, “isCarriedBy” are examples of relationship types. A relationship type can be a subtype of another and inherit properties and behaviors from it.

Just like hyperlinks, relationships are directed: they have a *subject* and an *object*. For instance, when a room “contains” a person, the room is the subject and the person is the object. Note the similarity with RDF [21], which we could use to represent this model. A relationship is said *reciprocal* when another relationship of opposite type ties the object to the subject. Note that although in the real world, all relationships are reciprocal, the actual implementation of this model will be incomplete either because of a lack of sensor data, or by deliberate choice for privacy reasons. Indeed, a user might not allow that the store where he currently is shopping, is informed of his presence.

In addition to the subject, the object and the reciprocity of a relationship, there can be more properties and behaviors that characterize it. For example:

- The time of creation of the relationship.
- The maximum time to live of the relationship. After this time, the relationship is automatically deleted.
- The relative coordinates of the object within the referential of the subject. For instance, the local coordinates of the printer within a building or a room.
- When a relationship is deleted, it can automatically delete its reciprocal peer.

Our software layer represents the physical world using this model and uses a variety of sensing mechanisms such as GPS, Infrared beacons and receivers, RFID tags and readers, Barcode readers, Bluetooth, etc, to automatically update the state of the representation. In the next section, we will describe how this physical model is actually implemented and integrated with the Web.

4.5 Web Presences and Modules

To actually implement this physical model, we introduce the concept of *Web presence*. A Web presence is the virtual representation of a physical entity as an integrant part of the Web. To Web users, the incarnation of a Web presence is a set of interactive Web pages that provide information and control about a physical entity and its relationships to other entities. To computer applications, the incarnation of a Web presence is a set of Web services to learn and interact with the physical entity.

A Web presence is formed by a set of services called *modules*. Each module implements a specific functionality. We provide a software framework called the *Web Presence Manager (WPM)* to support the easy development, deployment and management of Web presences and modules. Our framework provides tools so that each module can easily deliver:

- An interactive Web user interface, accessible from any Web browser.
- APIs for local or remote programmatic access based on Web technologies.

An application programmer who wants to take advantage of our context-aware software layer can either write his application as a module and make it part of a Web presence, or write his application as an external service and use the interface of one or more modules to learn about the physical world and interact with it.

In the rest of this section, we will describe the core modules that we developed to provide the most fundamental functionalities of a Web presence.

4.6 Core modules

Description Module.

The description module stores and exposes information in XML format about the identity, the characteristics and the capabilities of a physical entity. For instance, the description module of a room contains its name, its location, its size, etc. The description module also provides an interface to query and modify this information.

To be understood and recognized by other applications, the XML description of a physical entity must follow a pre-defined vocabulary or XML schema. We defined a basic common vocabulary that physical entities of all type can use.

In addition, we introduced a simple hierarchy of physical entity types. Figure 3 shows a limited view of our hierarchy of types.

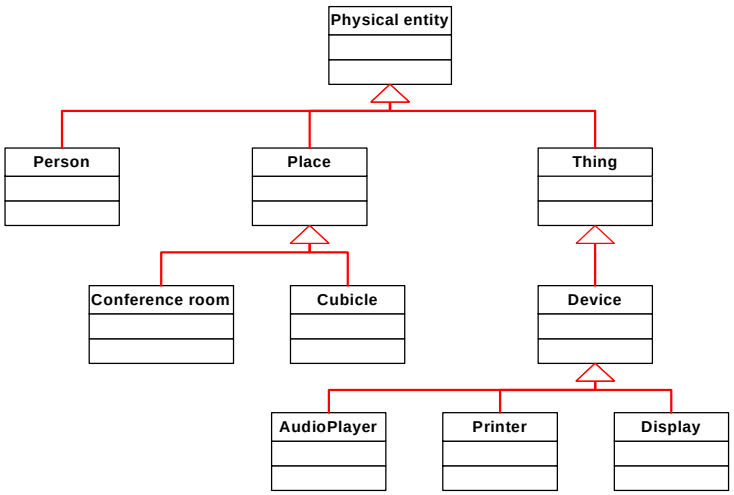

Figure 3 Hierarchy of physical entity types

For each type, we defined a specific vocabulary that extends the shared vocabulary and can be inherited by subtypes. We deliberately limited ourselves to a simple set of types and vocabularies that were useful for the kind of context-aware applications we develop.

Ultimately, we recognize that it is the role of industry consortiums and ontology projects to define an extensive hierarchy of types and complete vocabularies for each type. Such efforts are already ongoing in organizations like the UPnP forum[18] or the Standard Upper Ontology (SUO) working group [22] and go beyond the scope of our project. We plan on leveraging the outcomes of those efforts as they become widely accepted and stable.

Some characteristics of a physical entity (e.g. the temperature of a room, the status of a printer, the arterial tension of a person) can be automatically measured and updated by sensors. In this case, sensor wrappers can be implemented to update the description module through its API. Tools such as the Context Toolkit [7] can help in this task.

Directory and Discovery Modules.

Relationships between a Web presence and other ones are stored and managed by its directory module. For instance, the directory module of a room stores relationships to the Web presences of the things and persons currently inside the room. The directory module of a person stores relationships to the Web presences of things he or she carries and to the Web presences of the place where he or she currently is. The directory module also provides interfaces to query, add and delete relationships.

For each relationship, the directory module stores:

- A set of properties, such as the relationship type, its time of creation, its reciprocal relationship, the URL of the related Web presence, etc.
- Optionally, a cached copy of the XML description of the related entity. Keeping this cached copy increases the performances of queries.

The support of various sensing mechanisms to automatically update the relationships of the Web presence is handled by the discovery module. Several components are needed to perform this task.

First, various sensor wrappers need to be implemented to hide the details of the sensors and provide usable data. Depending on the sensor modality, the wrapper provides different types of information. For example, RFID and Barcode readers provide an identifier, GPS or indoor location systems provide coordinates, infrared or Bluetooth beacons can provide custom information.

Secondly, to create a relationship, the discovery module needs at least a reference to the Web presence to relate to. Infrared or Bluetooth beacons can be customized to send the URL

of a Web presence. However, when a sensor wrapper can provides identifiers, a resolution engine is necessary to map the identifier to the URL of a Web presence. The resolver module provides this functionality. It is based on [13]. It stores a set of bindings and provides interfaces for resolving identifiers and creating and editing bindings.

Finally, properties can be specified to characterize the new relationship. The discovery module can be customized to determine those properties either by prompting the user, exploiting extra data from the sensor or making an educated guess based on the current status of other modules.

Autobiographer and Observer Modules.

In many situations applications or users need to be notified of a change in the environment rather than have to pull this information from the system. The autobiographer and observer modules tackle this need for eventing in context-aware applications. The autobiographer module is a log where other modules or external services record actions that they have performed. This serves two purposes.

First, it keeps a history of events related to a physical entity. For instance, the autobiographer module of a room keeps track of people entering and leaving the room.

Secondly, it is constantly monitored by the observer module, which automatically triggers actions when specific criteria are met. The observer module consists of a set of rules. Each rule specifies a condition and a resulting action. The condition is constantly challenged against the events logged in the autobiographer. When the condition is met, the resulting action, which could be for example a call to another module or to an external service, is triggered. By creating custom rules, external applications can be automatically notified of a change in the physical environment.

Control.

Using the functionalities of the previous modules, applications and users can learn about the physical environment. Another common need of context-enhanced applications is to be able to control and interact with the physical environment when possible. For instance, an application or user might want to remotely turn a light on or off or send a document to a printer or projector. Various types of controller modules provide this functionality. Often times, network protocols to control a device already exist and a controller module can be implemented to act as a bridge between those protocols that are usually proprietary and standard Web protocols. When no network protocol exists, a controller module can be embedded in the device to control it directly. Different types of controller modules are necessary to control different types of devices. In essence, a controller module is similar to a UPnP service [18] and we offer a way to implement it using UPnP.

Other Modules.

Our framework enables the creation of custom modules. Several of them were implemented to extend the features of the WPM or to satisfy the need of specific applications. For example, a security module was implemented to provide basic authorization mechanisms to secure other modules. An indexer module and a listkeeper one will be presented in section 6.

4.7 Builder Service

There is a profusion of tools to easily create and author contents on the Web, from HTML editors to Blogging tools. The barrier to contribute to the Web is getting lower and lower and this contributes to its success. Similarly, to lower the barrier of deploying and leveraging our software layer, we provide a builder service to easily generate new Web presences. Just like a module, the builder service provides a Web-based user interface as well as APIs for local or remote access. It takes as an input the type of physical entity as well as information destined to be stored in the description module. It then generates a Web presence for this physical entity with all its appropriate modules.

5 Our Implementation

The implementation of this architecture, called the Web Presence Manager (WPM), has evolved from a simple prototype to an extensive and robust research platform. When installed and run on a machine, the WPM instantiates a Web presence container that can host and manage several Web presences and their modules. It is scalable enough to manage more than a thousand web presences but also lightweight enough to fit in a device with limited resources. Web presences hosted on one machine in the network can keep relationships with Web presences hosted on another machine. Our software can actually be embedded in devices such as printers, which can therefore host their own physical entity. But a company can decide to host all its conference rooms, offices and employees on one centralized server behind a firewall for easy maintenance and security. The physical entities hosted on the centralized server can then interact with the one hosted on the printers.

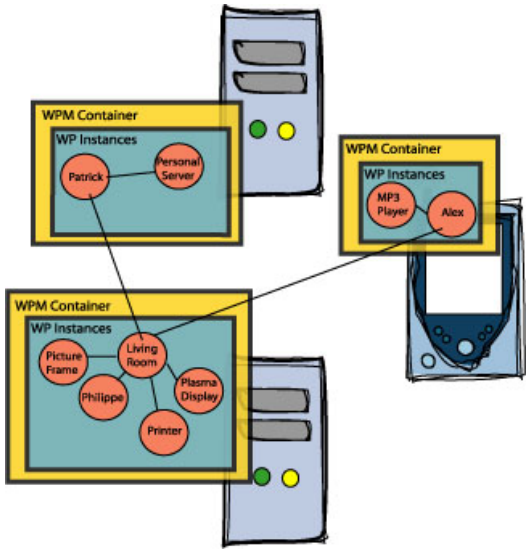

Figure 4 Example deployment of WPM Containers on various machines or devices.

The WPM relies on Web standards and web technologies. It is entirely written in Java for portability and ease of development. It runs in a Java Servlet container such as Apache Tomcat [23] or Jetty [24]. The various web pages representing a Web presence are dynamically generated using JSP. Custom JSP tag libraries are provided to access web presence related functionalities from a JSP page. Web presence data is stored and exchanged mostly using XML.

The WPM code is available open-source on our developer portal[14].

6 Example application: context-enhanced personal media portal

6.1 Overview

Along with developing a Web presence platform for ubiquitous computing research we also developed an extensive context-enhanced application that would serve as both a test of the platform and an example for service developers. We wanted to build a service sophisticated

enough to utilize the rich capabilities of WPM, but simple enough that any Web service developer could easily replicate it.

We chose to develop a mobile context-enhanced personal media portal. It provides the user with content selection and control through a PDA interface, but allows rendering on any compatible appliance discovered in the environment. We focused on delivering spontaneous appliance use, because we targeted mobile users interacting with many environments. We limited ourselves to a specific set of common content types including MP3 audio files, MPEG 2 and AVI video files, JPG image files, MS PowerPoint files, Adobe PDF files, and MS Word files.

The service we developed fully supports the reference scenario described in section 2. Here is a short list of its main features:

- *Personal:* We wanted to give access to personal contents stored on any Personal Computer, Mobile Appliance, or web-enabled appliance. We chose to implement a new WPM module called "Indexer" to support this capability.
- *Mobile:* The users are only required to carry a wireless PDA to render their contents in ubiquitous computing environments.
- *Context-aware:* The service uses information from the physical environment to organize the portal views and render a person's media contents.

6.2 Components

Hardware Setup. To experiment with the application, we installed two physical environments similar to the ones described in the reference scenario of section 2.

- A living room with a plasma video screen, a custom built digital picture frame and an HP digital entertainment center as sound system.
- A conference room with a printer and a projector.

Ideally, each device should have been a standalone Web enabled peer of the network. However, due to a lack of commercially available such devices, we simulated some of them by using biscuit PCs. The digital entertainment center was the only purely standalone Web-enabled device.

We used HP pocket PC Jornadas as PDAs. Each PDA had wireless and infrared capability. Each room was covered by 802.11b wireless network access. To enable the automatic detection of the user environment, we used a set of standalone infrared beacons able to regularly broadcast a piece of XML via infrared. A beacon was placed in each room.

WPM Extensions. To fulfill the needs of this application, we developed a set of generic and reusable modules to extend the core functionalities of Web presences.

To make local files available by nomadic users with Web access, we implemented an i*ndexer* module, designed to be part of a person's Web presence. After a user selects which local directories should be published, the indexer module recursively scans the directories and records metadata about each file in an XML database. The indexer module can be distributed to gather meta-data about files from various machines (e.g. the user's home computer, his work computer, his Yahoo briefcase…). The module then provides interfaces to query this meta-store and retrieve the files from the Web. It supports organizing the files in a directory/file hierarchy or by other organizing methods like album/artist for music, etc.

It is also able to query the directory module to determine the devices available in the environment of the user and determine for each file a list of local devices able to render the file. To support this feature, the capabilities of each device are embedded in the description module of its Web presence. These capabilities are expressed in terms of supported or unsupported Mime-types. For each file, the indexer module filters the available devices based on the Mime-type of the file.

To organize information into contextual lists, a l*istKeeper* module was developed. It uses past and recent events available in the autobiographer module, as well as current relationships

available in the directory module to create lists of information and content that might be useful in the current context.

Sensor Wrapper. To exploit the infrared capability of the PDAs, we developed a software application (BeaconReceiver) so that a PDA would pick up infrared beacons and interact with the discovery module of the Web presence. We implemented a tray application in the Pocket PC operating system that could run in the background listening for beacons on the infrared port. When a beacon was discovered, the XML data it broadcasted was passed via HTTP to the discovery module of the user's Web presence. The response from the discovery module was a URL that the BeaconReceiver opened in the PDAs browser. This URL could request approval or additional information from the user, or could simply be the URL of the person's web presence portal view.

Device Control. To enable the rendering of content on our simulated Web Appliances, we developed a Microsoft Windows application called the E-Squirt Viewer (ESV). This application consists of a web browser with an embedded Web server. The Web server allows remote control of the browser including loading URLs, page forward, back, etc. It supports rendering all content types we selected for the demo. For the HP digital entertainment center, we implemented a Web accessible audio player software.

6.3 Putting it all together

Configuration. We used our WPM software to represent each of the physical objects with Web presences. The digital picture frame, plasma video screen, digital entertainment center, printer and projectors all had a Web presence available on the Web. The two rooms also had a Web presence as well as each person (Patrick, Alex, and Philippe). Some of the Web presences were hosted by a Web presence container embedded on the local appliances while others were hosted on some server available on the Web. Then we manually created (using the Web interface of the directory module) "contains" relationships between the rooms and the contained devices. This allows users to recursively discover devices in the space just by making a connection with the room's Web Presence.

Discovery and Registration. Now we are ready for our mobile user. Patrick enters the living room with his PDA. On the wall is an infrared beacon that is broadcasting the address of the room's web presence. With his PDA, Patrick receives the beacon XML string, which is sent by the PDA's BeaconReceiver software to his personal Web Presence for analysis. The discovery module in his WP determines that the beacon is of type "place" and returns to Patrick's PDA a link to a web page with basic information about the living room and a button to press if he wants to establish a web-presence connection with the room. The PDA's BeaconReceiver launches a web browser and displays this web page. All of these transactions are recorded in the room's and Patrick's autobiographer module for future reference. Patrick presses the "register" button. His Web presence performs a handshake with the room's WP and establishes a reciprocal relationship. Figure 5 illustrates this process.

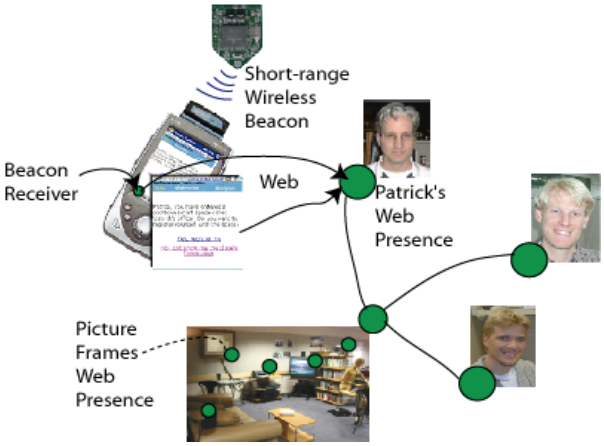

Figure 5 Discovery and registration process

Portal page. Patrick's Web presence queries device information from the room and content information from its own indexer module and presents Patrick with a home page that blends location-specific information with his personal information. We call this home page his WPM portal home page (Figure 6).

Figure 6 WPM portal home page

The portal home page, generated by JSPs in Patrick's WP, provides basic information about the place, links to his digital content on various computers, links to present people and devices, and lists of context-specific information that he may find useful. Clicking on the "related web presences" link will allow him to navigate the relationships between the place, other people in the place, and devices in the place. Each content store can be navigated like a traditional hierarchical file system or other organizational models. The context-specific lists are generated by the listkeeper module, which uses the observer module to respond to a "register" event. Although there are many potential contextual lists, we show four in the picture above.

Content View Page. By selecting a content store or a contextual list, the user will see a page that shows the corresponding contents and the local appliances that can render each file (Figure 7). Many UI designs were possible, this was chosen for simplicity. By selecting the

filename the user can download the file on the PDA. By selecting a local device next to a file, the device will render the corresponding file. For example the picture “bahnhof_str.jpg” can be sent to the digital picture frame by selecting the frame icon in the action column in the file row. A drag-and-drop interface would have been equally possible.

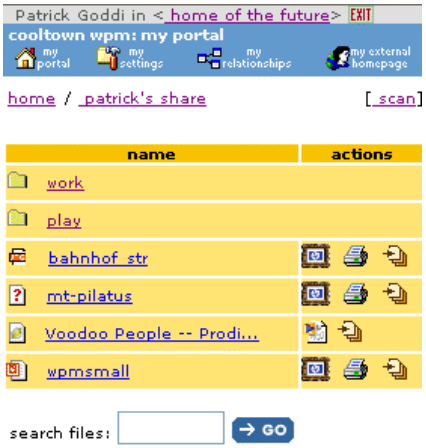

Figure 7 View of the contents of a directory

Figure 8 shows an example contextual list, in this case the “files used with these people list.” The people in the room are listed at the top of the list, followed by content that was used in contexts where these people were together. Just like in the content store view, the user can select context specific actions directly from the list.

Figure 8 View of the contents of a contextual list

Pass-by-reference and Control. When a user selects an application to user it to render the file, the control module of the digital picture frame will receive a message containing the URL of the jpg and it will process this request be retrieving the URL and displaying it on the frame. Thus, the file is only sent by reference and the appliance is responsible for fetching the file data pointed by the reference. After the file is sent by reference to the device, its control module returns a Web form, which is embedded as a frame in the portal view. The user can then control the device with this Web interface.

6.4 Observations and Other Applications

This service was used fairly extensively in team meetings and demonstrations over a six-month period. We found the system to be quite useful while it was supported. The primary downside was the burden of creating relationships and Web presences for each person wanting to use the system and each device in the environment. We have done some follow-on work to address this problem as part of a more recent project using UPnP and Bluetooth discovery. This application demonstrated the power and flexibility of the architecture, and served as a good training example for other groups and partners.

Our software has been successfully used as an underlying platform for several other internal and external research projects and pilots. A project called PLACTA [15] exploited the WPM to create a personal context-enhanced Web proxy. This proxy could modify the Web pages that a user is browsing to embed contextual information. A ubiquitous computing system deployed and tested in the San Francisco Exploratorium[16] leveraged our research results. The Hotspot server[17] project used the WPM to deliver services in an augmented wireless access point.

7 Conclusion and Future Work

We have outlined a model and software architecture to enable and facilitate the development and deployment of context-enhanced applications. We presented an extensive example of such application that was built on our platform. This application showed how a mobile Web portal could be enhanced to integrate the physical environment of the user.

While much work remains before pervasive support of ubiquitous computing will be a reality, we hope this paper and the open-source implementation of the Web Presence Manager will be useful to those exploring the integration of the physical world with Web services and particularly the development and delivery of Web services in ubiquitous computing environments.

There are many areas for further research that we are currently considering and exploring:

- Security and privacy of Web presence data and services.
- Resource sharing between multiple users and support for groups of collaborating users.
- Exploration of the possibilities and implications of distributing a single Web presence across multiple devices. For instance, the Web presence of a printer could run partly in the printer itself, partly on the manufacturer web server and partly on the reseller web server.

8 References

[1] Kleinrock, L., “Nomadicity: anytime, anywhere in a disconnected world”, Mobile networks and applications 1(4), 1997, pp. 351-357.

[2] Weiser, M., “Some computer science issues in ubiquitous computing”, Communications of the ACM, 36(7), 1993, pp. 74-84.

[3] Kindberg, T. and Barton J., “A Web-based Nomadic Computing System”, Computer Networks, Elsevier, vol 35, no. 4, March 2001, pp. 443-456.

[4] Kindberg, T. et al., “People, Places, Things: Web Presence for the Real World”. Proc. 3rd Annual Wireless and Mobile Computer Systems and Applications, Monterey CA, USA, Dec. 2000. p 19.

[5] Caswell, D. and Debaty P., “Creating Web Representations for Places”, Proc. of HUC2K, Bristol, UK

[6] Debaty, P. and Caswell D., “Uniform Web presence Architecture for people, places and things”, IEEE personal communications magazine, Aug. 2001.

[7] Dey, A., Salber D. and Abowd G., “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”, Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, pp. 97-166.

[8] Naguib, H. and Coulouris, G. “Location Information Management”, Proceedings of Ubicomp 2001, Atlanta, USA, October 2001.

[9] Johanson, B., Fox A. and Winograd T., “The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms”, IEEE Pervasive Computing Magazine 1(2), April-June 2002.

[10] Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. "EasyLiving: Technologies for Intelligent Environments", Proc. of HUC2K, Bristol, UK

[11] Streitz, N., Tandler, P., Mller-Tomfelde, C., Konomi, S., “Roomware: Towards the Next Generation of Human-Computer Interaction based on an Integrated Design of Real and Virtual Worlds”, Human-Computer Interaction in the New Millenium (2001), Addison Wesley, pp. 551-576.

[12] Román, M. et al., ”Gaia: A Middleware Infrastructure to Enable Active Spaces”, IEEE Pervasive Computing, pp. 74-83, Oct-Dec 2002.

[13] Kindberg, T. “Implementing physical hyperlinks using ubiquitous identifier resolution”, 11th International World Wide Web Conference (2001).

[14] http://www.cooltown.com/dev/download.asp

[15] Debaty, P. and Duebendorfer, T., “Aggregating local appliances and services to extend the functionalities of a Web browser”, proceedings of the first appliance aggregation workshop (aaw'02).

[16] Fleck M. et al., “From Informing to Remembering: Deploying a Ubiquitous System in an Interactive Science Museum”, IEEE Pervasive Computing Magazine, April-June 2002.

[17] Das, D., Manjunath, G., Krishnan V., Reddy, P., “HotSpot! A Service Delivery Environment for Nomadic Users System Architecture”, HPL technical report HPL-2002-134.

[18] UPnP: www.upnp.org

[19] Jini: www.jini.org

[20] RendezVous: http://www.apple.com/macosx/jaguar/rendezvous.html

[21] Resource Description Framework (RDF): http://www.w3.org/RDF/

[22] Standard Upper Ontology (SUO) Working Group: http://suo.ieee.org/

[23] Apache Tomcat: http://jakarta.apache.org/tomcat

[24] Jetty: http://jetty.mortbay.org/jetty/index.html