Learning Technology Research Taskforce:

Middleware for Pervasive and Proactive Computing

James Powell
jpowell@vt.edu

Abstract

This paper explores proactive and pervasive computing, and looks at the challenges faced by middleware for pervasive computing. Pervasive or ubiquitous computing is a vast area of research that has implications for networking, applications, development environments, user interfaces, remote sensing, and even the form of what we today know as a computer. It seeks to connect computers to one another and to their environment, and insulate users from mundane integration tasks while empowering them through proactive services. This paper focuses primarily on middleware for pervasive computing, and various aspects of the pervasive computing environment that affect its design. It also takes a brief look at the as-yet unresolved privacy issues surrounding some elements of the pervasive computing infrastructure.

Keywords

Middleware, proactive computing, pervasive computing, ubiquitous computing, context, task awareness, transparency, Radio Frequency Identifier (RFID), e-tag, wireless networking, Web Services, security, privacy.

Introduction

Imagination is often the spark that ignites new trends and computing is no exception. Many credit the idea of ubiquitous or pervasive computing to Mark Weiser, a scientist at Xerox-PARC. Xerox-PARC is the birthplace of numerous other world-altering technologies including the pinnacle of the interactive computing era – the graphical user interface. In his visionary paper entitled "The Computer for the 21st Century", Weiser breathlessly proclaimed that "the most profound technologies are those that disappear" [1]. He goes on to describe a world in which computing power and smart devices are so common as to merit less attention than a personal day planner or even yesterday's newspaper, but simultaneously one in which the impact of "silicon-based information technology" has become as profound as that of the written word.

The pervasive computing environment is one in which computers and computing devices greatly outnumber users, yet manage to remain unseen and able to act with little or no human knowledge, intervention or supervision. User context is no longer restricted to the virtual world of computer software, networks and network-accessible resources, but can incorporate physical location and the presence of others.

The pervasive or ubiquitous computing environment is characterized by a predominance of embedded computing devices that are constantly aware of their context, including where they are, what they are doing, who is using them, and what resources are nearby [2]. Users however are seldom, if ever, aware of these devices. It is a world in which many (eventually, most) objects have e-tags that can be detected and communicated with via other computing devices, initiate information exchange, and respond to their environment. Active e-tags such as MEMS (MicroElectroMechanical Sensors) will "sense" various aspects of the physical world. Much current research into pervasive computing focuses on smart environments [3,4], but the long-term goal is connecting with and augmenting the human experience of the physical world. The pervasive computing environment is like a rainforest – it is an ecosystem just as intricate, interdependent, and at times, unpredictable. It extends beyond the current boundary between the virtual and the physical world. Not only is it like the rainforest, it can also be a part of the rainforest – recording data about migration patterns of tropical birds, keeping a watchful eye out for endangered predators, and detecting climate changes through extensively deployed sensor nets.

Moore's Trends

Over the last 30 years, Moore's Law has changed our lives. All of us have benefited from an exponential growth in the availability, affordability, and power of general purpose computing devices. Because of the proliferation of personal computers, this has been referred to as the era of personal or interactive computing. Personal computers (PCs), personal digital assistants (PDAs), set-top Internet boxes, and Web-enabled cell phones have all grown up together. Users have been the focal point of personal computing software and hardware product offerings, but users are also the primary integrator of many of these devices. For example, cell phones can now store data and run applications, but they rely on the user to initiate data transfer between the phone and other computing devices. The user has to anticipate the need for data exchange, and initiate the transfer. So it is far from certain that this revolution will continue to benefit users. It may instead overwhelm them.

The demand for and prevalence of special-purpose chips has increased even more significantly than demand for computational processors (processors destined to be used in general purpose computers like PCs and servers). In the year 2000, there were over 150,000,000 computational microprocessors shipped. In that same year, there were three times as many microcontroller units, four times as many digital signal processors, and twice as many embedded microprocessors shipped compared to computational microprocessors [5]. These processors go into everything from cars to toasters. These devices are often connected to the physical environment through some specialized sensors, but they are rarely networked, so the data they collect is not readily available outside of the object in which they reside. So they too depend upon people to mediate integration with other computing devices and access the data they collect.

Likely to soon outnumber both general-purpose computers and special-purpose processors are auto-identification devices. In particular, smart tags, or e-tags, are becoming smaller, cheaper and more common at a phenomenal rate. There are already e-tags smaller than a grain of sand [6], and the European Union is considering embedding them in paper Euro note [7]. E-tags can identify an object and various properties related to it. Some also have the ability to store new data, and are equipped with sensors that allow them to sense various aspects of their environment. E-tags are built around radio frequency technology that allows the e-tag to be accessed at a distance. Passive e-tags can only report information in response to a query from a reader – the signal provides them with enough power to respond. Active e-tags contain a battery and can broadcast information about themselves as needed, but active e-tags are more costly and have a limited lifespan. E-tags represent the cornerstone of what Gartner refers to as the "supranet". The supranet is an environment in which "the physical and the virtual worlds overlap … making computing transparent (that is, invisible) to humans and creating an 'intelligent' environment around us, populated by 'smart objects'" [8]. Gartner's supranet is in fact essentially a subset of the pervasive computing environment.

Pervasive Computing is not Re-centralized Computing

Microsoft Windows dominates the personal computing platform. In addition to Windows, Microsoft also offers operating software for personal digital assistants, set top Internet appliances, and video game appliances, among other computing devices. But despite the fact that this is an advantageous position in which to find oneself as a software company for selling product, it hasn't helped them establish Windows as a pervasive computing environment. In theory, Microsoft could develop and leverage a homogenous software layer among various platforms they support, which in turn could make it easier for them to develop software for pervasive computing. But instead, Microsoft offers a collection of appliance-oriented environments of varying complexity, interoperability, and functionality with a relatively high cost of ownership. So despite their predominance in the operating system arena, it is very difficult to achieve a meaningful level of integration or easy mobility across devices running Microsoft operating environments.

The state of the art in Microsoft desktop integration is Active Directory, but to call it "active" is a misnomer. A more descriptive reference would be "Delegated Directory Management Services." A great deal of the product revolves around exposing and supporting delegated management of information about people, computers, and resources to systems administrators within an organization. In other words, Active Directory was designed to work the

way large businesses and IT organizations work. Active Directory offers an acceptable framework in which to centrally manage desktop security and shared network accessible resources, but falls far short of enabling pervasive computing. In its current form, it is a dead-end. That is, for as long as the company places greater emphasis on creating hooks in other applications that require customers to run Active Directory [9] and on "standards speculating", that is, introducing new interoperability standards and then expanding them with proprietary extensions.

Microsoft's closed-platform, centralization-minded dominance of the desktop environment is in sharp contrast to fledgling pervasive computing efforts such as the peer-to-peer focused Project JXTA (AKA Project Juxtapose) and its pervasive computing-oriented objectives [10]:

- *Interoperability - across different peer-to-peer systems and communities*
- *Platform independence - multiple/diverse languages, systems, and networks*
- *Ubiquity - every device with a digital heartbeat*

Sun introduced the JXTA development platform in mid 2001 in an effort to position Java as the pre-eminent platform for peer-to-peer. Described by Sun as "25,000 lines of code that could sit inside just about any computing device from servers to cell phones" [11], it can be used to build peer-to-peer applications for file sharing, communication, and collaboration. More recently the environment has also been used in conjunction with Sun's own grid computing platform, Sun Grid Engine. Sun's JXTA implementation thus has the potential to play a significant role in a pervasive computing environment by offering a shared, platform-independent abstraction for peer-to-peer communication.

Need for Middleware

JXTA is an example of middleware that can enable pervasive computing across heterogeneous networks, software platforms, and devices. The pervasive computing environment relies heavily on proactive middleware rather than interactive applications. This is not just a convenience for the user – it is essential, because the user will not be able to continue to mediate interaction between all of these computers, devices, smart objects, and sensors. Proactive middleware for pervasive computing manages communication between devices connected to the real world, anticipates user needs, and initiates actions on the user's behalf transparently, assisting the user without distracting him from the task at hand.

The pervasive computing environment depends upon standards to enable communication between all its occupants. Some of these standards exist now, and they include IPv6 (with its vastly expanded address space), Web Services (for discovery and interaction with distributed computing services), specialized XML applications such as SensorML (a structured data interchange format for dynamic sensing devices [12]) and the Physical Markup Language (PML), Object Name Service (ONS), and the MIT-developed EPC standard [13, 14] (which allows any RFID tagged object to have a unique identity). These are all standards that can be used by middleware applications as part of a pervasive computing infrastructure.

Middleware is a largely invisible but important component of most computing infrastructures that offers a range of integration services to users and applications. By its very nature it is transparent to the end-user. It bridges the gap between heterogeneous computing platforms and applications. Middleware will be key to pervasive computing, because it is the middleware layer at which much of the supervised and unsupervised interactions will occur. It is also the layer in which many applications and services that today are not connected and not network aware will move into as proactive middleware services.

Software today behaves as if it were designed for life in a zoo. Programmers turn requirements into code that implements support for predefined usage scenarios. If something goes wrong, it may be perfectly acceptable for the software to simply reject input and refuse to proceed until "correct" input is supplied. In a pervasive, decentralized, proactive environment, loosely coupled event driven software is needed in order to provide an environment in which the user can rely upon the infrastructure rather than having the infrastructure depend upon the user. This software will anticipate user needs, negotiate for access to resources, and implement

contingency plans when things go wrong. It will not expose low-level errors to the user, it will not interrupt user activities in order to ask for information related to interaction with other software or devices, and it will not require that the user be constantly aware of its presence.

According to Grim, et al [15], any software designed for a pervasive computing environment should be able to do three things. It should expose and accommodate change. It should be capable of composing and assembling services dynamically. It should maintain a separation between data and functionality. There are technologies and techniques that address each of these issues to varying degrees today. Event driven architectures, such as those that employ messaging, are by definition designed to expose and respond to change. The Web Services architecture includes specifications for discovering and dynamically addressing Web Services, either singly or sequentially to perform complex tasks. Application architectures that depend upon a database or XML document sets to store data, and separate application code to perform tasks with that data represent a system for separating code from data, although the code still depends upon a fixed, predefined structure for the data.

Proactive Computing

Proactive computing is an enabler of and enabled by pervasive computing. In a pervasive computing environment, proactiveness is an essential characteristic of systems, software, and devices. As Bradley and Starner point out in their 1996 paper on the Remembrance Agent, there are three approaches to handling automated tasks [16]:

- *Perform a task only when specifically requested*
- *Lurk in the background but only act when a specific trigger occurs*
- *Run continuously*

Proactive software uses whatever data it may have available to anticipate user needs and provide some form of assistance to the user through the devices or software the user is using. This can happen at various levels including at the application level (e.g. spell and grammar checking found in recent versions of Microsoft Word), between applications (Remembrance Agent), at the operating system level (monitoring battery life in a laptop), or at the network level (what resources are nearby).

One of the major advocates for proactive computing is David Tennehouse, a researcher at Intel. He and his company have good reason to be interested in any trend that might sell more microprocessors. But his enthusiasm for the topic is genuine. He speaks and writes frequently about the topic. In a recent paper, he identifies three areas of research [5]:

- *Getting Physical - Proactive systems will be intimately connected to the world around them, using sensors and actuators to both monitor and shape their physical surroundings.*
- *Getting Real - Proactive computers will routinely respond to external stimuli at faster-than-human speeds.*
- *Getting Out - Shrinking time loops and sheer numbers demand research into proactive modes of operation in which humans are above the loop.*

The proactive computing environment and the software that supports it share many problems. The most significant of these is the limited resource of human attention. Garlan, et. al define human attention as “the ability of a user to focus on his primary task, oblivious to system-generated distractions such as failures and poor performance” [17]. One of the great challenges of developing proactive software is balancing proactiveness with transparency. A successful proactive application targeted at users should be context aware, task aware, and essentially invisible to the end user.

Context

Context consists of information about the user, the equipment and software being used, what it is being used to do, and the physical environment and location. Context can consist of any or all of the following aspects of the environment in which the computing device is located: who is around,

what they are doing, availability and speed of various network connections, what resources are nearby, what resources are contained within the device, what access the user has, what other people are around, the current time, and the user's schedule. If context information is monitored and provided by middleware, then it becomes available to various applications and devices.

Task Awareness
Task awareness is the ability for a piece of software to determine what the user is doing. In today's computing environments, task information is usually inferred from the application context, and available only to that application. Task information is not represented in any formal fashion and is not available to the operating system in which a particular application is running. Microsoft Word demonstrates primitive task awareness, in features such as date completion, grammar and spell checking, and by offering templates based upon the structure or content of a new document. Middleware that maintains task awareness can facilitate proactive task support between applications, and allow software to continue to support users as they move around in the physical and virtual environment.

Transparency
Transparency refers to the ability of a proactive or anticipatory application to avoid capturing the user's attention. A transparent application is one that the user is not aware of, one that requires no direct user input or supervision. Transparency requires that an application be fault tolerant and able to respond dynamically, so that it can silently cope with change. Transparency is also one aspect of a pervasive computing environment where many computing devices, sensors, and other resources coordinate activities to provide information and services without human intervention. Middleware for proactive computing can make it easier to build transparent proactive applications by providing a universal layer in which change can be managed.

"Getting Physical"
Weiser describes an interconnected world of devices both seen and unseen [1]. He talks about prototypes of three categories of devices with which he envisions people would interact regularly: boards, pads, and tabs. Weiser's tab embodies some of the characteristics of smart tags. He describes various prototypes and potential designs ranging from smart badges to PDA-like devices. Today PDA tabs such as the Sony Clio, Palm Pilot, and various Windows-CE based devices fulfill some aspects of Weiser's vision for tabs by offering personalized services, but since they largely mimic the desktop-interactive model of personal computing they fall short of his vision for smart context-aware, proactive tools. One reason for their shortcoming is their obliviousness to the physical environment. Weiser envisioned that tabs, pads, and boards would all be connected to the physical environment.

Only now is the physical world within reach through small, affordable and reliable sensors and smart tags. Modern smart tags were preceded by a variety of specialized devices that explored various challenges of smart tag design, including latency, accuracy, fault-tolerance, scalability, and low power consumption to offer users specialized functionality ranging from location tracking to collaboration to entertainment [18]. Devices such as MediaCup, Cricket, Progammable Beads, iBadge, WearBoy, Meme Tags, and the infamous Lovegety (a matchmaking device that originated in Japan), paved the way for innovations in and subsequent generations of e-tags. These smart tags could in some ways be considered early prototypes of RFIDs and MEMS.

RFID/Auto-IDs – A global partnership of corporations and universities have joined to form the Auto-ID center whose stated goal is to build "an Internet of things" [13]. The Auto-ID Center advocates the use of RFID-based e-tags and coordinates research into their design, but hardware design is only one aspect of the Auto-ID Center's activities. They have also been working on several standards and software for a global RFID infrastructure. The Electronic Product Code (EPC) standard is a numbering system that allows every tagged object to have a unique ID. The center is working with other standards bodies to develop a migration path from barcodes. An EPC has four parts: an 8 bit header that identifies the EPC version number, a 28 bit "manager" id that would likely identify the product manufacturer, a 24 bit object class that

identifies the product line, and a 36 bit serial number that uniquely identifies the object that has been tagged. Products descriptions will reside on Physical Markup Language (PML) servers, which can be accessed through an Object Name Service (ONS) server via the object's EPC. This model is similar to the HTML-URL-DNS model employed by the World Wide Web. PML is an XML application designed for tagging information about objects. It can contain information about the object (classification or type of object, composition, etc), as well as constantly changing dynamic data about the object (its current temperature), and temporal data (its current location). In the field, objects are tracked through a distributed middleware layer called "Savant." Each instance of Savant is responsible for collecting product data, managing tasks, and maintaining a real-time in-memory event database (RIED). These components allow Savant to proactively manage a collection of products (such as the inventory of a store) and tasks related to them.

Current areas of research and activity include e-tag design (chip and antenna design and manufacturing), bandwidth utilization (no global standards exist and some frequencies work better for some applications than others), reducing cost of various types of e-tags through new manufacturing processes, designing and building cheaper and better readers, refining the Savant model and PML, furthering the development and adoption of e-tag related standards, and encouraging the deployment of infrastructure components [14]. As Vince Stanford pointed out in a recent article he wrote for IEEE's Pervasive Computing magazine: "at the high end, RFID tags are wireless, networked, pervasive computers, successfully integrated into their environment" [19].

MEMS – Also referred to as "motes" or "smart dust", are miniaturized sensing smart tags. MEMS stands for microelectromechanical sensors. MEMS are a class of devices that range from active, sensor-equipped e-tags to nanotechnology that combines integrated circuits with mechanical systems smaller than a grain of pollen. MEMS are capable of collecting data about various aspects of the physical environment ranging from temperature, humidity, sound, light, or movement of objects, and communicating that data as a node in a "peer-to-peer wireless sensor network [20]. Sandia National Labs has developed a micromachine manufacturing process called "SAMPLES" and anticipates a $100 billion market for MEMS [21]. MEMS have the potential for a huge range of applications including military surveillance, weather monitoring, manufacturing, home automation, deep sea and space exploration, and medicine. Researchers at Sandia envision a revolution lead by MEMS [21]:

> *"…the hallmark of the next thirty years of the silicon revolution will be the incorporation of new types of functionality onto the chip; structures that will enable the chip to not only think, but to sense, act and communicate as well. This revolution will be enabled by MEMS."*

Challenges include reducing size, reducing cost, researching new manufacturing processes such as self-assembly, battery life/alternate power sources, and transmitting data over an ad-hoc wireless network. Areas of research include building networks that efficiently transmit data between MEMS using as little power as possible, packaging sensor data (e.g. by using a standard such as Sensor-ML – the Sensor Markup Language) building software to support querying of live data sets (archived and real-time sensor readings), and research into allowing MEMS to scavenge power from environmental sources of energy such as vibrations or ambient light [20].

Middleware for Proactive and Pervasive Computing

Internet2 and Middleware

Everyone has their own idea of what constitutes middleware and their own focus. Internet2 is one example. It is a consortium of 200 universities partnered with the private sector and government formed to develop software and standards for the next generation of the Internet. Most of Internet2's middleware initiatives are concerned with solving authentication and authorization problems. Internet2 defines core middleware as identifiers, authentication, directories, authorization, and certificates and PKI [22]. Internet2's narrow middleware focus has allowed them to develop or influence the development of important standards including the

eduPerson LDAP object class specification, and innovative middleware solutions such as Shibboleth, pubCookie, and openSAML. Currently Internet2 has little involvement in what they term "upperware" such as business application middleware, research middleware, or tools for ubiquitous computing. Nonetheless, the middleware solutions developed by Internet2 offer important core services for ubiquitous computing.

Gartner's Middleware Taxonomy

Gartner Group takes a broader view than Internet2 of middleware; this is reflected in their middleware taxonomy, which identifies two broad groups of middleware: basic and integration. They define basic middleware as software that "provides integration between data and applications built upon a consistent architecture" [23]. They subdivide the basic middleware category into the following categories:

Gartner Middleware Taxonomy: Basic Middleware
Communication middleware
Message oriented middleware (MOM)
Data management middleware including • Netware • NFS • ODBC and JDBC
Platform middleware, including • Application servers such as Jboss, BEA WebLogic, IBM Websphere. • Portal frameworks, portal application servers, and other Web integration services • Transaction processing monitors • Object request broker services

Gartner describes integration middleware as software that "helps connect applications and databases that are built on different software architectures" [23]. Integration middleware overlaps with basic middleware, and includes:

Gartner Middleware Taxonomy: Integration Middleware
Gateways such as database gateways, MOM gateways, and platform middleware gateways
Superservices – e-business platforms
Business process managers - such as workflow and process builders
Integration brokers – such as B2B integration services

Gartner's taxonomy is more comprehensive, but it too fails to explicitly recognize middleware for pervasive computing, which tends to incorporate some design concepts found in other types of middleware (and in fact builds upon some existing types), but emphasizes functionality required to address pervasive computing problems, such as computer ubiquity, context awareness, transparency, task awareness, proactivity, dynamism, device heterogeneity, physical environment awareness, and a high degree of fault tolerance.

Pervasive Computing Middleware

There are a number of pervasive computing projects that have developed middleware specifically for ubiquitous computing environments. Two common recurring themes are smart spaces and enhanced mobile computing. The following projects illustrate implementations of each.

Project Aura (*Carnegie Mellon*) –The goal of Project Aura is to "provide each user with an invisible halo of computing and information services that persists regardless of location" [24]. Two middleware building blocks of Aura are Coda and Odyssey. Coda and Odyssey are designed to facilitate mobile information access. Coda is an experimental filesystem that offers "continued access to data in the face of server and network failures" [25] by relying heavily on

caching. It was designed to support application-transparent adaptation by assuming all responsibility for adaptation, so that it would be completely backwards compatible with other applications (which is especially important for a filesystem). Odyssey is a collection of mobile computing extensions for UNIX. These include application-aware adaptations that offer energy-awareness and bandwidth-awareness to extend battery life and improve multimedia data access on mobile devices.

Researchers have proposed a comprehensive Aura system that would be a middleware testbed for software that enable proactivity and self-tuning with the goal of producing more effective systems with fewer system-generated distractions [17]. Plans are for the Aura system to incorporate or utilize tools such as Coda and Odyssey.

Gaia (*University of Illinois at Urbana-Champaign*) – The Gaia project aims to develop a middleware operating system that manages an Active Space. An Active Space is defined as an environment which "encompass the devices and physical space surrounding the machines" and as a result people and devices within them can "seamlessly interact" [3].

Gaia is modeled after operating systems. At its core is the component management architecture, which allows Gaia to create, destroy, or upload components. A Gaia component is a software module that can be executed on any device within an Active Space. Devices are referred to as Gaia nodes. Nodes organize components in containers, which can be created, browsed or deleted. The next layer in the Gaia environment offers a number of services, including a context service, an event manager, a presence service, a security service, and a component repository. Above that are the application framework, which can execute applications built from components taking into account the hardware resources available in an Active Space or with a particular device, and the quality of service framework, which aims to offer highly dynamic QoS services that take into account application configuration, resource availability, user mobility, and end-user preferences. At the top layer are "Active Space Applications", which expose services to users through various input and output devices within the space.

A testament to the robustness and flexibility of Gaia is the collection of applications that have been developed for it. Active Space applications include a scheduler, a PDF viewer, a PowerPoint viewer that can display multiple synchronized presentations, a synchronized whiteboarding tool, authentication services based on biometrics (finger print), and even an MP3 audio player.

one.world (University of Washington) –
one.world is a systems architecture for pervasive computing that embodies three design principles [15]:

1) *Expose change*
2) *Compose dynamically*
3) *Separate data and functionality*

This approach avoids potential limiting factors such as "tight coupling between major application components" which other approaches require in order for applications to adapt to changing contexts. Unlike existing environments, such as Microsoft's Windows and Active Directory products, one.world "does not rely on all-powerful administrators and well-maintained servers locked up in a machine room" [15]. one.world is an execution environment in which pervasive applications run on a device. It uses a shared data management system based upon tuples through which all applications share data. In their most basic form, these objects have unique ID and a set of metadata associated with them. The event processor allows applications to exchange events asynchronously through event handlers. one.world is implemented in Java and has a small binary footprint, so it can potentially be deployed on many different types of devices.

iROS (*Stanford*) – The developers of iROS (interactive room operating system), describe it as a "meta-operating system or middleware infrastructure" [4] built to support three types of tasks: moving data, moving control, and performing dynamic application coordination. iROS has three major subsystems: event heap, data heap, and iCrafter. The event heap is essentially a messaging queue. Applications send and receive messages about events to the event heap. This allows the environment to tolerate a high degree of dynamism and to survive device and

application failures. The data heap offers virtual storage for digital objects. Any object can be stored with an arbitrary number of metadata elements. Users locate their data by searching rather than attaching to or otherwise accessing physical storage volumes. The third major component of iROS is the iCrafter system, which supports service discovery and device-appropriate user interface generation services. iROS has been used to explore a number of concepts, including room-based cross platform interfaces such as room controller (for controlling environmental systems and displays within a room), PointRight (for multi-user control of shared devices, such as displays), and SmartPresenter, which includes a scripting language for coordinating multiple presentations across multiple displays.

RCSM (Arizona State University) – RCSM stands for Reconfigurable Context Sensitive Middleware. RCSM takes a different approach than some of the other types of pervasive computing middleware, in that it maintains a separation between context information and application logic. In RCMS, context changes directly trigger actions within an application object [26]. One benefit of this approach is that applications can be easily adapted to use new contexts. Interface maps specific contexts with specific actions within a given application. CA-IDL (Context Awareness Interface Definition Language) allows generation of an appropriate interface for a given context in a number of different languages, so developers can use the tools they are familiar with and focus on the application requirements without having to also build context sensitivity into the application. Objects that implement these generated interfaces in effect expose actions (methods) so that context code can make direct calls to those methods as appropriate. Another focus of the project is to facilitate ad hoc spontaneous networking between two devices, and context-triggered communication [27].

Project Oxygen (*MIT*) –
The goals of MIT's Project Oxygen are [28]:

- *building applications using composable, distributed components,*
- *customizing, adapting, and altering component behavior,*
- *replacing components, at different degrees of granularity, in a consistent fashion,*
- *person-centric, rather than device-centric, security, and*
- *disconnected operation and nomadic code.*

There are a number of software projects that fall under the Project Oxygen umbrella. "MetaGlue" is Oxygen middleware that supports software agents. These agents offer collaboration and discovery services in a sensor-equipped "intelligent room" environment. MetaGlue is written in Java and supports the Jabber messaging protocol and an expert systems rules engine called Jess. Also of note is "The Search project" which is an NSF-sponsored project to investigate systems architectures and protocols that could be used to secure interaction between wearable and embedded devices in a pervasive computing environment [29].

The Open Knowledge Initiative (OKI)

OKI is in some respects a complement to the Internet2 middleware initiatives. Their main focus is defining Open Service Interface Definitions that are tuned to the needs of academic computing. They also oversee reference implementations for middleware that supports each OSID they have identified. OSIDs are intended to offer vendors and open source developers a standard suite of middleware services. OSIDs include typical middleware services such as authentication, authorization, and database connectivity, as well as middleware that supports higher level abstractions including [30]:

- Dictionary – supports domain or cultural context through language dictionaries or domain-specific nomenclatures.
- Filing – platform independent hierarchical file management.
- Scheduling – a mechanism for managing events in shared calendars.
- Usermessaging – provides user-to-user messaging and notification.

- Workflow – allows for management of interdependent successions of activities where each may have its own completion constraints.

User-facing applications as Middleware

As this diverse collection of projects, libraries, tools, and practices for pervasive and proactive middleware illustrates, a fundamental problem in middleware development is how to identify new functionality and design software that offers this functionality in a usable fashion. This is driving an emerging trend in proactive computing, which is the migration of user-facing application functionality down into the middleware layer, where it becomes infrastructure upon which other applications can be built. Edwards, et al define describe infrastructure as [31]:

> *"software that supports construction or operation of other software… that enables applications that could not otherwise be built or would be prohibitively difficult, slow, or expensive."*

and they later describe the problem with designing infrastructure:

> *"…even though the technical features of the underlying infrastructure are visible in- and to a large degree, even determine – the features of applications created using it, we often lack criteria for designing or evaluating the features of the infrastructure itself."*

In proactive computing, to determine what infrastructure is needed one merely needs to look at software the user uses today, and what tasks users are forced to perform that could be anticipated and handled automatically. The core functionality of many user-facing applications is in many cases an excellent candidate for being re-architected as middleware for a proactive computing environment.

Remembrance Agent (RA) is an example of software that straddles the boundary between a proactive application and middleware infrastructure. RA monitors keystrokes and dynamically constructs queries that it submits to a personal search engine, which maintains an index of the user's documents, seeking out content that might be relevant to the task at hand [16]. RA takes a task that many of us perform regularly (searching) and makes it an automated, proactive middleware service. It is an example of a user-facing application that has been pushed down into the middleware layer, but it predates much of the aforementioned work on pervasive computing middleware environments.

Web Services provides a suite of technologies that make it relatively easy to expose the individual functions that make up an application, and offer secure, direct access to them. It also offers a discovery mechanism that allows for dynamic composition of a chain of tasks that access these individual functions in a particular order, and with dynamically generated input (i.e. output from one function serving as input to another). These same Web Services can serve as the underpinnings of existing applications that use this functionality. Imagine a Web-based file storage system, with a typical folder metaphor interface for managing remotely stored files. If this application were re-implemented as a set of Web Services, it could still maintain this user-facing Web interface. But other applications could also call the file management through individual Web Services. This would enable other programs to interact with the storage in much the same way that users do (e.g. save a purchase receipt or class ticket to the storage). Proactive applications could use these middleware storage services to transparently support users as they moved between spaces and devices in a pervasive computing environment.

Privacy and Pervasive Computing

Privacy is the Achilles' heel of pervasive computing. In a world full of smart objects, it is all too easy for middleware and backend business systems to collect information that reveals more than the typical user might expect. But even now, we have less privacy than we think. Today, every credit car purchase, every phone call, every Web search and page view is recorded. Parking passes, building access cards, subway cards, and toll road pass cards are

logged every time they are used. Some cards and cell phones can report their user's location via GPS or triangulation with cell towers. We constantly trade privacy for convenience.

E-tags offer another opportunity for tracking products and potentially the people who use them. On the surface, this sounds like a scary prospect. Bennetton, an Italian clothing manufacturer, was intimidated into reversing plans to use e-tags by privacy advocacy groups who launched the "I'd Rather Go Naked" boycott campaign [35] when word leaked out of their plans to embed e-tags into apparel for inventory tracking. These same advocacy groups fail to realize that it is far easier to track a person through a store by monitoring their cell phone signal than it is to track them with a passive e-tag that has no independent ability to broadcast information at all, and an effective read range of only a few feet when contacted by a reader. And many consumers opt into programs that already allow retailers to track their purchases. Simpler (and cheaper), passive e-tags are basically barcodes for the 21st century. Yet, there are outspoken opponents with an incomplete understanding of the technology or its potential benefits who are greatly concerned about the perceived loss of privacy and eager to whip others into a frenzy, as the following commentary from Mary Starrett, a regular contributor to NewsWithViews.com illustrates [36]:

> *"Chipsters say the technology will only be used to help retailers keep track of inventory - like bar codes. But privacy-loving consumers question the very concept of a device that sends out radio waves to 'readers' that not only identify the article, but where and with whom it's going… the Big Brother implications of this thing need little hyping to get your skin crawling.'*

She concludes by invoking a name synonymous with big brother:

> *"If RFID gets off the ground as planned, that would make George Orwells' predictions off by just 20 years"*

Consumers Against Supermarket Privacy Invasion and Numbering (Caspian) is supporting legislation that it says will protect consumer privacy in the age of RFIDs. The "RFID RIGHT TO KNOW ACT OF 2003" demands that all products which bear RFIDs bear labels [37]:

> *(1) stating, at a minimum, that the package contains or bears a radio frequency identification tag, and that the tag can transmit unique identification information to an independent reader both before and after purchase; and*
> *(2) in a conspicuous type-size and prominent location and in print that contrasts with the background against which it appears.*

The proposed law seems on the whole reasonable. But RFID is just one example of a wave of new technologies that make up the pervasive computing environment. Will each new device or technique require its own legislation? Only time will tell.

Nonetheless, pervasive computing is an unstoppable trend that is largely already with us. Cars no longer just have one computer – they have a network of computing devices controlling fuel injection, braking, wheel traction, monitoring safety systems and tire pressure, and increasingly replacing mechanical controls with drive-, brake-, and even steer-by-wire systems. Some vehicles come equipped with GPS/cellular units that allow for personalized driving directions and vehicle tracking (OnStar, TeleAid). Homes can be equipped with burglar and fire monitoring equipment that "knows" when you open and shut any door or window that is monitored by the system, regardless of whether the system is armed. Personal video recording units such as TiVo and Replay know (and report) what you watch on TV and how often. We fight for privacy in the grocery store, but willingly surrender it in our cars and homes.

Privacy need not become extinct in the world of pervasive computing. Weiser anticipated this threat and suggested, "a well-implemented version of ubiquitous computing could even afford better privacy protection than exists today" [1]. He foresaw the emergence of "digital pseudonyms" which have since enabled millions, both friend and foe to maintain online anonymity while availing themselves of information resources such as search engines and e-journals, free e-

mail accounts, and even online dating services. And in fact there are already examples of just the type of privacy-protecting technologies Weiser envisioned, of which Shibboleth and tools and systems that implement the Liberty Alliance specifications are two examples. Technology is not innately evil – but sometimes it is necessary to create new checks and balances in order to ensure that neither man nor corporation use it in intentionally or unintentionally harmful ways. Open solutions based on open standards are easier to monitor for their potential to reduce privacy. Well-informed vigilance can preserve the benefits of smart objects to consumers and protect privacy.

Conclusion

The software and technologies behind pervasive and proactive computing are enabling technologies. The risks are not insignificant but the potential rewards are great. But to reap those rewards, and avoid the pitfalls, we need to become actively involved in the development and deployment of pervasive computing infrastructures. The only way we can do this is if we take the advice of Tennehouse and "declare victory on office automation." The Supranet-Pervasive-Proactive-Ubiquitous computing era is coming. Aura, Gaia, one.world, Savant, Labscape, RFIDs, SensorML, EPC, PML, ONS, proactive computing, and the pilot projects that this paper suggests represent areas we could begin to explore in order to better understand the infrastructure needs, as well as the policy, security, and privacy issues that will be common in this new era of computing.

References

1. Weiser, M. (1991, September). The Computer for the 21st Century. Scientific American 265, (3).

2. Schilit, B., Adams, N., Want, R. (1994). Context-Aware Computing Applications. IEEE Workshop on Mobile Computing Systems and Applications.

3. The Gaia Project. (2003). University of Illinois at Urbana-Champaign. <http://choices.cs.uiuc.edu/gaia/>

4. Johanson, B., Fox, A., Winograd, T. (2002). The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. Pervasive Computing. <http://swig.stanford.edu/pub/publications/iwork-overview-layout.pdf>

5. Tennehouse, D. (2000, May). Proactive Computing. Communications of the ACM, 43 (5).

6. McCullagh, D. (2003, January 13). RFID tags: Big Brother in small packages. CNET. <http://news.com.com/2010-1069-980325.html?tag=nl>

7. Chai, W. (2003, May 22). Radio ID chips may track banknotes. CNET. <http://news.com.com/2100-1019_3-1009155.html>

8. Magrassi, P., Berg, T. (2002, August 12). A World of Smart Objects: The Role of Auto-Identification Technologies. Strategic Analysis Report: Gartner Group.

9. Bailey, C. (2003, January 2). Exchange 2000 migration: Plan now or pay later. CNETAsia. <http://asia.cnet.com/itmanager/trends/0,39006409,39100547,00.htm>

10. CollabNet Services Team. Project JXTA. <http://www.jxta.org/>

11. Vance, A. (2001, April 25). Sun launches JXTA development platform. Java World (April 2001). <http://www.javaworld.com/javaworld/jw-04-2001/jw-0427-iw-jxta.html>

12. Botts, M. (Ed.). (2002, December 20). Sensor Model Language(SensorML) for In-Situ and Remote Sensors. Open GIS Consortium Inc. <http://www.opengis.org/techno/discussions/02-026r4.pdf>

13. Cover, R., Cover Pages: Physical Markup Language (PML). <http://xml.coverpages.org/pml-ons.html>

14. Auto-ID Center. (2003). Auto-ID Center: InDepth Look. <http://www.autoidcenter.org/aboutthetech_indepthlook.asp>

15. Grimm. R, Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., Wetherall, D. (2001). Programming for Pervasive Computing Environments. University of Washington, Department of Computer Science and Engineering.

16. Rhodes, B., Starner, T. (1996) Remembrance Agent. The Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology. <http://www.bradleyrhodes.com/Papers/remembrance.html>

17. Garlan, D. Satyanarayanan, M., (2000). Proactive Self-Tuning Systems for Ubiquitous Computing. <www.ngi-supernet.org/lsn2000/CMU-Smailagic.pdf>

18. Plymale, B. (2002). A Survey of Smart Tags.

19. Stanford, V., (2003). Pervasive Computing Goes the Last Hundred Feet with RFID Systems. Pervasive Computing. <http://www.computer.org/pervasive/pc2003/b2009.pdf>

20. Hoffman, T. (2003, March 24). Smart Dust. Computerworld. <http://www.computerworld.com/mobiletopics/mobile/story/0,10801,79572,00.html>

21. Brito, S. (2003). Vision for MEMS. Sandia National Laboratories. <http://www.sandia.gov/mstc/technologies/micromachines/vision.html>

22. Internet2 (2003). Internet2 Middleware. <http://middleware.internet2.edu/>

23. Hess, D. (2001, April 10). A Graphical Middleware Taxonomy. Research Note: Gartner Group.

24. Satyanarayanan, M. (2000). Project Aura. <http://www-2.cs.cmu.edu/~aura/>

25. Satyanarayanan, M. (1996, February). Mobile Information Access. IEEE Personal Communications. <http://www-2.cs.cmu.edu/~odyssey/docdir/ieeepcs95.pdf>

26. Yau, S., Karim, F. (2001, October). Context-Sensitive Distributed Software Development for Ubiquitous Computing Environments. 25th Annual International Computer Software and Applications Conference (COMPSAC'01). IEEE Computer Society. <http://www.computer.org/proceedings/compsac/1372/13720263abs.htm>

27. Yau, S., Karim, F., Wang, Y., Wang, B., Gupta, S., (2002). Reconfigurable Context-Sensitive Middleware for Pervasive Computing. Pervasive Computing. <http://www.computer.org/pervasive/pc2002/b3033abs.htm>

28. Garland, S. (2003). MIT Project Oxygen: Software Technologies. <http://oxygen.lcs.mit.edu/Software.html>

29. Computation Structures Group. (2003). MIT: Laboratory for Computer Science. <http://csg.lcs.mit.edu/projects/?action=viewProject&projectID=14&projectGroup=Security>

30. Open Knowledge Initiative. (2003). General Overview (OSID). <http://web.mit.edu/oki/specs/OSID_table.pdf>

31. Edwards, W. K., Bellotti, V., Dey, A. K., Newman, M. W., (2003). Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Infrastructure. Conference on Human Factors in Computing Systems (2003).

32. DeBonis, M. (2003, June 19). Neighborhood Watch. <http://vtmig.w2k.vt.edu/progress.htm>

33. Frequently Asked Questions about SETI@home. (2003). SETI@home <http://setiathome.ssl.berkeley.edu/faq.html>.

34. Arnstein, L. Borriello, G., Consolvo, S., Hung, C., Su, J. (2002). Pervasive Computing. <http://labscape.cs.washington.edu/labscapeIEEE.pdf>

35. Boycott Benetton. (2003). CASPIAN. <http://www.boycottbenetton.org/>.

36. Starrett, M. (2003, June 11). Big Brother Comes to Walmart. NewsWithViews.com. <http://www.newswithviews.com/Mary/starrett14.htm>.

37. RFID Right to Know Act of 2003. (2003). Consumers Against Supermarket Privacy Invasion and Numbering (CASPIAN). <http://www.nocards.org/rfid/rfidbill.shtml>.