Middleware for Pervasive Computing

Vesa Kautto
HUT, Telecommunications Software and Multimedia Laboratory

vesa.kautto@hut.fi

Abstract

This paper investigates the role of middleware in the various pervasive computing scenarios. Middleware can be used provide an important transparency or abstraction layer between applications and lower layers that actually implement the requested services. Middleware services include messaging, RCP, synchronization and universal data access services to applications that run on devices of pervasive computing. After general discussion the paper concentrates on the middleware for wireless, mobile terminals.

1 Introduction

This paper takes its subject matter from the eighth chapter in the book “Pervasive Computing Handbook” (Hansmann et al., 2001) but the subject is extended by considering the role of middleware in pervasive computing and by including more examples of important middleware components. Let’s start by clarifying the two main subjects of this paper: pervasive computing and middleware.

1.1 What Is Pervasive Computing?

The past decade saw the emergence of new class of computing and communication devices like smart cards and labels, handheld computers (PDAs) and smart phones. The new devices, the advances in wireless technologies and visions of universal information processing services anywhere and anytime has created a new area of computing what we call here the pervasive computing. Nomadic and ubiquitous computing are two other closely related concepts to pervasive computing and according to Raatikainen: “they are — at least — very close to each other, if not different names for the same thing” (Raatikainen, 2001).

Mark Weiser has been named as the father of ubiquitous computing (Ubicomp) and has presented his vision (in Weiser, 1993b) in the following way: “Ubiquitous computing has as its goal the enhancing computer use by making many computers...
available throughout the physical environment, but making them effectively invisible to the user.” In his another paper (Weiser, 1991) Weiser predicts that there will be quite commonly hundreds of computers in one a room but by then they are so small and commonplace that they are virtually invisible to users. The vision of Weiser is so strong that people shouldn’t need to carry any device to be able to access information or get services but the technology today isn’t quite that mature yet so this paper looks at pervasive computing as it is possible today or in the near future. Today users still need to carry some kind of wireless, mobile device. Mobile Internet, which is marketed strongly by mobile terminal and infrastructure manufacturers like Ericsson and Nokia, fulfills a part of the Ubicomp’s vision.

1.2 What Is Middleware?

Any piece of software that glues together various other pieces of software can be labeled as middleware. The two most common functions handled by middleware solutions are messaging and data access services. A typical usage scenario is one where a graphical user interface (GUI) component needs to access a remote database. Usually the GUI part has to be independent of the actual database implementation and a middleware component or a set of middleware components provide that functionality to the GUI. Thus middleware provides a service layer in the software architecture that separates the details of implementation from users of middleware (fig 1). The typical users of middleware are application developers who build new applications to be deployed in the target environment.

![Middleware Diagram](image)

Figure 1: Middleware layer is placed between applications and hardware or operating system or network layer services (after Bernstein, 1996).

Other typical middleware services include message passing, transaction monitoring, directory lookup and object brokerage or other distributed computing environ-
ment services. Many of the middleware solutions in use today are application-specific or optimized for a set of applications but naturally there are also generic middleware solutions. Examples of current generic-purpose middleware solutions are CORBA (Common Object Request Broker Architecture), DCOM (Distributed Common Object Model), J2EE (Java 2 Enterprise Edition), J2ME (Java 2 Micro Edition) and WAE (Wireless Application Environment) (Raatikainen, 2001). Of these only J2ME and WAE are intended to be used on mobile devices. The remaining three are still suitable for server-side computing but they don’t adapt well to more challenging requirements of pervasive computing like automatic reconfiguration and service discovery or context-awareness on the device.

The second section provides further background on pervasive computing and middleware. In the third section we study various ways that middleware can support implementation independent development of pervasive computing applications, and the fourth section investigates some remaining challenges for pervasive middleware and points to a case study of how a middleware components could be used in a real life and discusses its implications. The fifth section presents the conclusions.

2 Background

This section provides a short introduction to pervasive devices, to networks that connect them and to applications of pervasive computing. It also starts the discussion of the role of middleware in pervasive computing.

There are three building blocks that will enable pervasive computing environment. The first part is formed by cheap, low-power computers that include equally convenient displays. The second part is formed by a network or networks that tie the pervasive devices together, and third part is formed by software systems that implement ubiquitous applications and services (Weiser, 1991). The software systems are clearly the hardest part since there are already quite usable devices, and networks with sufficient bandwidth are already available or will be available in the foreseeable future.

2.1 Pervasive Computing Devices

The device categories for pervasive computing according to Hansmann et al., 2001 are

1. *information access devices* including cellular phones, pagers, PDAs and all kinds of mobile phone+PDA combinations,

2. *intelligent appliances* including household appliances, vending machines, point-of-sales terminals, ATMs, medical monitoring devices and car embedded devices,

3. *smart controls* including home controls, smart card and labels, manufacturing process controls, and
4. **entertainment systems** including MP3 players, digital cameras and TVs, various set-top boxes and game consoles.

Burkhardt *et al.* , 2001 (chapter 3) gives a very good introduction to most popular mobile information access devices and their operating systems as well as typical ways to program them. At this time the most popular mobile devices are PalmOS-based devices, Windows CE-based Pocket PCs, Symbian-based smart phones and small notebooks.

Many smart controls provide intelligent sensor functionality and they are usually networked to some kind of central control unit that monitors and instructs them. Smart cards and labels on the other hand can be used to offer sophisticated cryptographic and identification functionality.

Intelligent appliances are another class of devices that can be enhanced by embedding more intelligence and connectivity into them. For example, networking a smart alarm clock with the electric heater of the automobile to start heating automatically can be quite useful in the cold Nordic countries. Other applications implemented using smart appliances could be remote diagnosis and operation of these devices. Car-based pervasive systems belong also this category and there are already many prototypes including applications like dynamic navigation systems (based on GPS), in-vehicle networks and infotainment services.

Entertainment systems are used mainly for entertainment, and future TVs and game consoles could be also connected to the infrastructure of home networks. Applications visioned for systems with two-way communication channel include e.g., integration of all kinds of communication software (e-mail, chat) into them, web browsing and various kind of personalized TV programming. The very clear trend is that soon interactive digital TVs and game consoles are becoming computing appliances with sophisticated abilities to communicate.

Smart controls and most intelligent appliances are beyond the scope of this paper since the application(s) on them are typically pre-programmed on the pervasive memory and once started they run the whole lifetime of the device. Thus there is no real for middleware since a general-purpose middleware solution is always more efficient and customized solution an application or two.

“Mobile computing devices such as PDAs, laptops, cameras and mobile phones impose new requirements for middleware platforms. These devices face temporary loss of network connectivity when they roam; they need to discover other mobile devices in an ad-hoc manner; they are likely to have scarce resources, such as battery lifetime, processing power and memory; they are required to react to frequent and unannounced changes in the environment, such as high variability of network bandwidth, new physical locations, and so on” (Emmerich *et al.* , 2001).

Since mobile devices of pervasive computing put special demands on the middleware that can be used with them the existing middleware needs to be modified to be able to cope with the new environment, or completely new middleware solutions must be created.
Because of limited processing power and memory restrictions the applications of pervasive computing tend to be distributed systems that often follow the client-[proxy]-server architecture.

2.2 Networks of Pervasive Computing

Pervasive computing devices can be connected to each other using three types of networks. *Wireless Wide Area Networks* use typically digital cellular radio technologies from the end user devices to base stations. *Short-range Wireless* technologies can be used typically indoors since the range is usually just a few tens of meters. The third type of networks can be found at residential and office environments where they connect controls and appliances. More info about all of the networks and technologies used in them can be found from Chapter 12 of Hansmann et al., 2001.

2.3 Conclusions

Two kinds of applications can be run pervasive computing devices: *standalone* applications that use only local data and computing resources, and *networked* applications that utilizes a network connection. Networked applications can naturally just fetch or send data, or do both. But even in principle standalone application can need some networked services like a text editor application requesting a printing service from a near-by printing device. As mention earlier the network connectivity can not be guaranteed always, and it would be a quite odd situation if every single application that needs some networked services implements the checking of all possible transport options and then availability of specific services, etc. In multi-application environment communication with networked services should be done using middleware.

In short, the promise of pervasive computing: “access to any piece of information at anytime at a reasonable price” is still a fantasy but it is turning into a reality. Maybe not as fast as some people want it but surely. However there are several competing technologies or standards in almost all categories of this evolving technology, and there are no sure winners in any category. In that kind of situation it is wise to bet on already established or well-supported standards. Unfortunately, in this fast changing world there are not too many standards that can be relied on. One wise way to develop new applications is to use appropriate middleware that make the underlying technologies transparent to the application under development. Thus middleware provides the required ability to adapt to changing operating environments.

3 Overview of Mobile Middleware Categories and Solutions

One of more formal definitions of what is meant by middleware or *middleware services* is given by Bernstein (in Bernstein, 1996): A middleware service is a general-purpose
service that sits between platforms and applications. Bernstein also lists a set of properties for middleware, and the list is reproduced below for convenience.

1. A middleware service is defined by the APIs and protocols it supports.

2. A middleware service meets the needs of wide variety of applications across many industries.

3. A middleware service must have implementations that run on multiple platforms.

4. A middleware service is distributed. This means in practice some kind of remote access to provided services and implies also often that the middleware service includes a client part and a server part that run on separate address spaces (i.e., on different systems).

5. A middleware service should support a standard API.

Middleware has been identified already as “mission-critical” software in distributed computing that protects end users and application developers from the ever-changing hardware and operating system changes since both can change freely since the software is written to be only dependable on the middleware layer and the services that it provides.

It can be said that “middleware proves the ’plumbing’ necessary for applications to exchange data, regardless of the environment in which they are running. Transactions, data broadcasts, and XML data often ride on middleware in the enterprise.” (Talarian, 2000)

Talarian Corp. has also divided the middleware market into five different segments (for enterprise computing). The segments are:

1. Transaction processing monitors that are not typically used for general-purpose program-to-program communication but are used by transaction-oriented applications that often access relational databases.

2. Remote Procedure Calls (RPCs) that use synchronous communication and a request/wait-for-reply programming model.

3. Object Request Brokers (ORBs) that are language-independent, object-oriented RPCs. There are two main competing standards for ORBs: CORBA and DCOM.

4. Homegrown middleware solutions that are developed in-house for specific middleware needs but are neither very expandable nor cheap to maintain. On the other hand to solution can be a very efficient and the development teams has full control of all aspects of it.

5. Message-oriented middleware (MOM) which is the current state of the art for general-purpose middleware solutions. It can be run in two different “modes”
(even simultaneously if needed). The more traditional is message queuing where program-to-program communications takes place via a queue (which is typically a file). The other mode has been labeled as message passing and it is easiest described by publish-subscribe programming model. Any interested program can subscribe to a subject and all messages sent on that subject will be forwarded to all subscribers. A publisher of messages can be any node of this ad hoc network.

Talarian’s list may identify the main segments of current enterprise computing but there are many important middleware services in pervasive computing that haven’t been mentioned in the list. The short list of these middleware services includes (data) security services, service discovery middleware services, multimedia streaming services and device-type specific standard APIs.

One more battleground can be drawn in the markets of current middleware platforms. That is the choice of overall computing platform, and the main choice is between .NET platform defined by Microsoft and Java 2 platform mainly defined by Sun Microsystems Inc. This treatment favors Java 2 platform because of more advanced mobile device support and better operating system independence.

3.1 Persistent Data Access Middleware

Data persistence can be arranged at least in two ways. Data can be stored either in a file or some kind of database. For files there is rarely need for a middleware solution but in the case of databases a solution is needed especially for relation databases because of their popularity. There are two main solutions available: Open Database Connection (ODBC) and Java Database Connection (JDBC). Both provide platform independent access to relational databases. On mobile devices both can be used to access the local database with probably quite limited capabilities but also remote database with additional arrangements like synchronication servers or gateways.

3.2 RPC and ORB Middleware

In this category Simple Object Access Protocol (SOAP) is nearly the only option in the pervasive computing scenario since both CORBA and DCOM have been labelled as only good for server-to-server communications with a careful (manual) maintenance plan for network configuration (Box, 2000). SOAP is an XML-based typed serialization format that uses HTTP as the messaging transport. Both of these factors make it a much more suitable to be used in pervasive computing since they support portability and especially HTTP’s lightweight and good Internet-compatibility make it an excellent choice compared with the other two.

Problems of SOAP usage in pervasive middleware are mostly related the fact there is no formal specification for the API, and “SOAP packets tend to be larger on the wire and they can be somewhat more resource intensive to parse/generate.” DevelopMentor,
The latter can be a big problem on devices with very limited resources so SOAP cannot be used everywhere.

RMI (Remote Method Invocation) of Java 2 platforms could be a serious contender in this category but unfortunately it forces the use of Java on both ends of communications which decreases at least the portability.

3.3 Messaging-oriented Middleware

Although JMS (Java Message Service) is actually a J2EE technology it can be used in the many more advanced pervasive devices. It defines an abstract API that can be used to relay (private) messages inside an organization in an asynchronous way. The API and the framework defined by it then must be first implemented and deployed before it can be used. IBM’s MQ Everyplace is an example of a JMS implementation.

Messages are often triggered by some events and since the service guarantees delivery of messages in all circumstances this middleware solution can be used to build portable, message-based applications. The software architecture of this application is modelled typically by message producers and message consumers instead of more familiar client-server architecture.

An example of a message-driven pervasive application could be a news reader application that has been configured to receive all news stories related to some topic. The topic can be anything supported by the message producer but a typical example topic might be “Manchester United” (a soccer team) or “DaimlerChrysler” (an exchange-listed company). When the message producer publishes a news story to a subscribed topic it is delivered to all subscribers as soon as it is possible.

3.4 Device API Middleware

Middleware in this category doesn’t fulfil the 4th requirement of Bernstein’s definition (“A middleware service is distributed”) but there is clear need for these kinds of abstractions of operating system specific dependencies.

J2ME was earlier mentioned as one generic middleware solution and as such it is a good choice. But for actual new application development special purpose APIs are often needed that taken into account the needs of the application a specific application category. As example of this let’s take a look at JavaPhone API, which can be used by many telephony related services on a mobile phone that supports it.¹

A wireless phone with JavaPhone API together with PersonalJava² enables developer access to packages shown in figure 2.

Of these packages, for instance, datagram messaging can be used develop simple client-server applications using datagrams. The API allows applications to be developed independently of the physical network or bearer supported by the device. Version

¹Nokia Communicator 9210/9290 might be the only one at the moment
²Parallel technology to J2ME, to be replaced by J2ME and Phone Profile
Figure 2: JavaPhone middleware layer is placed between applications (top) and basic Java platform (PersonalJava). Courtesy of Sun Microsystems Inc. 4.

1.0 of JavaPhone API supports three concrete addressing (communication) protocols: 
\textit{sms}, \textit{udp}, and \textit{wdp} (WAP datagrams via GSM SMS message[s]). To provide true transport independence the addressing and selection of the actual messaging protocol can be abstracted well by using \textit{DatagramAddressService}.

Others device type specific APIs are, for instance, JavaTV API to be used in digital TV sets, and OpenCard Framework that can be used to abstract the problems related to very primitive (although standardized) UPDU\textsuperscript{5} and actually used and possible proprietary communication protocol.

4 Remaining Challenges for Middleware

Weiser identified nearly a decade ago several research areas for Ubicomp from the different fields of computer science and many of them have been solved. One problem that remains unsolved and could be solved by appropriate middleware is that of related to changing of the display or whole UI of an application from one device to another in automatic way, e.g. from PDA to TV or PC monitor. This \textit{window migration} problem hasn’t received as much attention as the development of the other required infrastructure, possible because of until now there hasn’t clear enough need for it (Weiser, 1993a).

Bardram & Christensen, 2001 is a white paper and a case study how pervasive computing could be integrated into daily routines of healthcare professionals in a hospital environment. It presents four very interesting real-life scenarios, which includes several window or view migrations from one device to another, several context-sensitive services (e.g., notification service) and automatic both network and device reconfig-

\textsuperscript{5}Application Protocol Data Unit - exchange format for communication between smart cards and their reader devices
urations. Like most projects today the computing is still device-centric (in practice a PDA), which provides at least identification and location services to the overall architecture. In many cases the handheld device also provides the necessary controls (e.g. for zooming or editing) for the visual user interface that might be shown using a completely different device (e.g. TV or SmartBoard). If and when all the presented functionality is implemented it will definitely require new, innovative middleware solutions. Overall the planed pervasive computing architecture is very interesting but unfortunately it doesn’t scale very well outside the hospital environment (on the other hand it doesn’t necessarily need to either).

Raatikainen, 2001 reviews present research activities in mobile, distributed computing and presents a research agenda for future software infrastructure of mobile information society. It is based on the belief that (all wide area) telecommunication networks will be based on the IP (Internet Protocol) in the near future. The roadmap section of the paper presents a framework for future research the area (cfg 3).

![Layered architecture of middleware and Internet protocols](image)

Figure 3: Layered architecture of middleware and Internet protocols. Courtesy of Kimmo Raatikainen.

The framework embraces open standards like whose developed by IETF and W3C but gives room also to technologies that are popular but less open. The place of “IP middleware” and Wireless Middleware have been defined but what is exactly inside them is still an open research issue.

Since it is highly improbably that there will be a single dominant middleware platform there is a clear need for interoperability. The paper identifies two levels of interoperability: “between middleware platforms and between parts of an application running on different middleware platforms.”
5 Conclusions

This paper started with an introduction and discussion of pervasive computing and middleware and how they are connected to each other. The traditional middleware solutions however have been designed for a completely different operating environment than where pervasive devices of today and tomorrow will live so they are not suitable solutions without (radical) modifications.

Some important middleware categories and technologies suitable for today’s mobile middleware are reviewed in the third section. An interesting case study is discussed in the fourth section, which points out some functional requirements that haven’t been solved yet. There remains still many hard research problems related to mobile middleware and new problems will appear as a better understanding of all involved technologies is gathered.

The ultimate purpose of the middleware is to ease the development of the end user applications. Many middleware technologies fulfil that goal but usually that facilitation comes with a price ticket. Many middleware technologies are quite complex to use and maintain plus expensive to obtain. The already mentioned interoperability remains to be a problem, which is usually solved by writing application for just one platform, or pair of platforms that are a “natural fit” to each other.

For the application architect today the most important issue to solve during the design phase of a new application is how to connect the mobile device to back-end servers. There is no one correct solution to that question since no middleware solution cannot satisfy all of these three tough requirements: “very efficient, very adaptable and very scaleable” at the same time.

REFERENCES


