# A Survey on Middleware in Pervasive Environments

C. Elangovan
Associate Professor, Department Of Information Technology, Priyadarshini Engineering College, Vaniyambadi, Tamilnadu.

## Abstract

*The vital challenge of pervasive computing is to develop technologies that allow smart devices to automatically adapt to changing environments and contexts, making the environment largely invisible to the user. Pervasive middleware can support the developer by supporting rapid development and deployment of applications by domain experts with minimal programming expertise. A survey was made with five different pervasive middleware Architecture behavior and concept. This paper contains a classification of these Pervasive middleware according to pervasive requirements parameters they are discoverability, adaptability, context awareness, heterogeneity and environment. The Pervasive middleware classification highlights what has been done and what remains to do in developing new Pervasive middleware.*

*Keywords: Middleware; Pervasive environment; Contexts; Discoverability; Heterogeneity*



## 1. Introduction

Middleware are enabling technologies for the development, execution and interaction of applications. These software layers are standing between the operating systems and applications. They have evolved from simple beginnings - hiding network details from applications - into sophisticated systems that handle much important functionality for distributed applications - providing support for distribution, heterogeneity and mobility. SOA middleware [1] is a programming paradigm that uses ``services" as the unit of computer work. Service-oriented computing enables the development of loosely coupled systems that are able to Communicate, compose and evolve in an open, dynamic and heterogeneous environment.

If middleware were designed to help manage the complexity and heterogeneity inherent in distributed systems, one can imagine the new role middleware has to play in order to respect to the evolution from distributed and mobile computing to pervasive one.

Hardly a day passes without some new evidence of the proliferation of portable computers in the marketplace, or of the growing demand for wireless communication. Support for mobility has been the focus of number of experimental systems, researches and commercial products, and that since several decades. The mission of mobile computing is to allow users to access any information using any device over any network at any time. When this access becomes to every information using every device and over every network at every time, one can say that mobile computing has evolved to what we now call pervasive computing [2].

### *1.1 About Middleware*

Any piece of software that glues together various other pieces of software can be labeled as middleware [4]-[5].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 separate the details of implementation from users of middleware in Fig.1. The typical users of middleware are application developers who build new applications to be deployed in the target environment. Other typical middleware services include message passing, transaction monitoring, directory lookup and object brokerage or other distributed computing environment 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 [6].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).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.

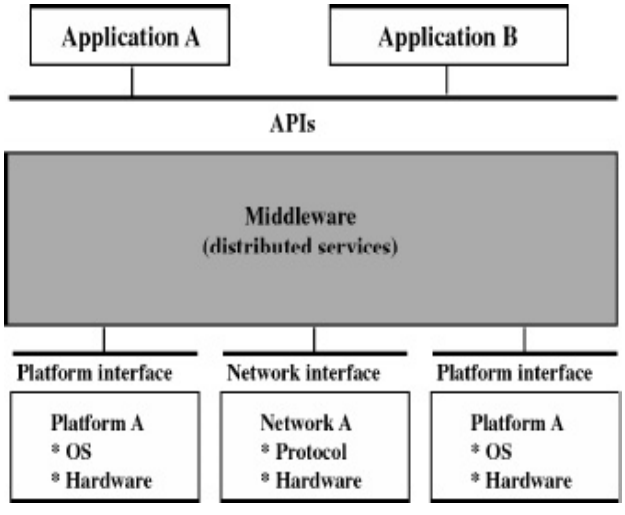


Fig 1: Architecture of Middleware

*1.2Pervasive computing technologies*

Pervasive computing involves three convergence areas of ICT [7] computing, communications and user interfaces.

1. *Devices*

PCS devices are likely to assume many different forms and sizes, from handheld units (similar to mobile phones) to near-invisible devices set into 'everyday' objects (like furniture and clothing). These will all be able to communicate with each other and act 'intelligently'. Such devices can be separated into three categories: sensor: input devices that detect environmental changes user behaviors, human commands etc; processor: electronic systems that interpret and analyze input-data; actuator: output devices that respond to processed information by altering the environment via electronic or mechanical means.

2. *Connectivity*

Pervasive computing systems will rely on the interlinking of independent electronic devices into broader networks. This can be achieved via both wired (such as Broadband (ADSL) or Ethernet) and wireless networking technologies (such as Wi-Fi or Bluetooth), with the devices themselves being capable of assessing the most effective form of connectivity in any given scenario. The effective development of pervasive computing systems depends on their degree of interoperability, as well as on the convergence of standards for wired and wireless technologies.

3. *User interfaces*

User interfaces represent the point of contact between ICT and human users. For example with a personal computer, the mouse and keyboard are used to input Information, while the monitor usually provides the output. With PCS, new user interfaces are being developed that will be capable of sensing and supplying more information about users, and the broader environment, to the computer for processing. With future user interfaces the input might be visual information –for example recognizing a person's face, or responding to gestures. It might also be based on sound, scent or touch recognition, or other sensory information like temperature. The output might also be in any of these formats. The technology could 'know' the user (for example through expressed preferences, attitudes, and behaviors) and tailor the physical environment to meet specific needs and demands.

## 2. Middleware in Pervasive Environment

*2.1 ubiSOAP:*

The ubiSOAP middleware enriches the Web service architecture with key features for services to become truly pervasive by taking full benefit of the rich capacities, including multi-radio interfaces, now embedded in wireless devices.

The ubiSOAP middleware empowers the service-oriented architecture with pervasive networking capabilities, in particular enabling adaptive lightweight services to be run on mobile nodes and access to services over multi-radio, multi-network links. ubiSOAP brings multi-radio, multi-network connectivity to services through a comprehensive layered architecture:

- The multi-radio device management and networking layers together abstract multi-radio connectivity, selecting the optimal communication link to/from nodes, according to quality parameters.
- The communication layer allows for communication in the pervasive networking environment according to SOAP. ubiSOAP in particular enriches traditional functionalities of a SOAP engine to allow for SOAP-based point-to-point and group-based interactions in the pervasive network. It further enables

access to services that may be in distinct networks thanks to multi-network routing.

- The middleware services layer brings advanced distributed resource management functionalities customized for the pervasive networking environment. From that layer, the Discovery Service enables the dynamic advertising and location of networked services, in particular accounting for extra-functional properties.

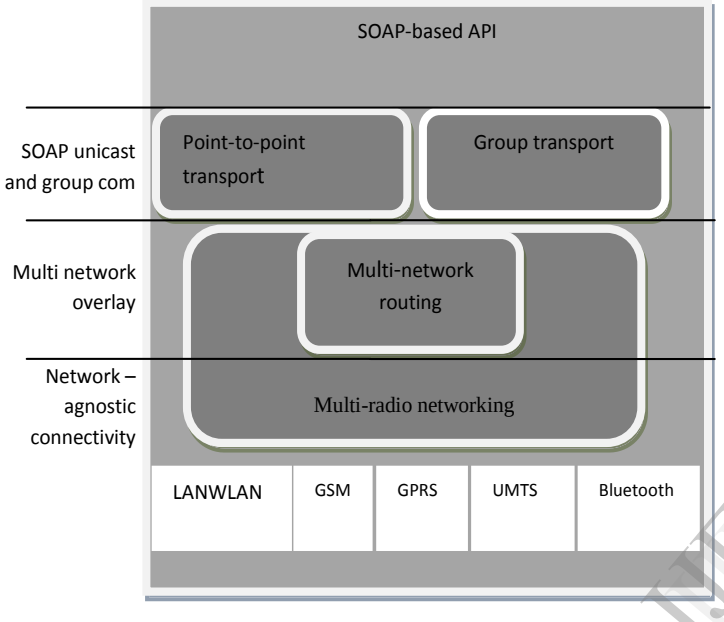


Fig 2: ubiSOAP Middleware Architecture

Last but not least, the ubiSOAP middleware is mobility-aware so that its functionalities adapt to the physical mobility of both clients and services, in particular exploiting the rich multi-radio, multi-network connectivity. The ubiSOAP middleware has been implemented using Java for both desktop (J2SE) and mobile (J2ME CDC) environments.

*2.2 CDTOM*

Context-Driven Task-Oriented Middleware (CDTOM) for Pervasive Homecare, which is established in such a fact that man is a creature of habit, and he will perform a certain activity at a particular situation as a routine. The CDTOM uses the abstraction of tasks in order to separate logical relations of relevant items from the services realizing and fulfilling the intended goals.

CDTOM provides a number of system-level services, such as context data acquisition, context storage, context-driven task reasoning, service discovery and task-oriented mapping, to facilitate the development and deployment of various pervasive homecare applications. The middleware architecture separates context management, context-driven task, and task related services in different layers that are accessible by the applications. This scheme, not only decouples the dependency of techniques used in individual layers, but also provides a greater flexibility for the selection and deployment of appropriate techniques in each layer by the specific system requirements. Similarly, the development effort and cost of Homecare system would be reduced through the task layer and its programming interfaces.

*2.2.1Overview of the CDTOM Architecture*

The overall architecture of CDTOM which is to support the context-driven tasks in pervasive homecare environment, and provide the assistance of daily activities and necessary healthcare to the elders. The CDTOM consists of the following four logical layers:

*Context Provider:* Real-world contexts often originate from diverse sources, leading to dissimilar approaches to generating context description. Context providers obtain raw context information from various sources such as hardware sensors and software programs and transform them into context mark-ups. Some context providers, including the location context providers, the environment context providers (which gather environmental information such as temperature, noise, and light from embedded sensors), work with the hardware sensors deployed in our prototypical smart space. Software-based context providers include the task context providers, which extracts schedule information from the inhabitant's.

*Context Server:* The main objective of this layer is to enable the effective and efficient context data aggregator, storage and query. Its functions are deployed in the system server. Briefly, the context aggregator component discovers context providers and gathers context mark-ups from them. The context knowledge Base (CKB) provides persistent context knowledge storage. Contexts in smart spaces display very high change rates, so the aggregator must regularly update the CKB with fresh contexts.

*Task Processor:* This is the most important layer in CDTOM; there are three key components in it: Task

Reasoner, Task Scheduler, and the Rule Engine. The task reasoned is to infer the inhabitant's current task according to the real-time context data and the historical context information. The Task Scheduler enables the high-priority task, e.g. fire alarm, to be executed in a priority-scheduling algorithm when there are several concurrent tasks in a given context.

*Service Manager:* This logical layer of CDTOM manages context-aware services to enable the orchestration of homecare applications, and also it is the coordinator between the inferred tasks and diverse services enabling the task to be executed automatically and adaptively. All the services in smart home will be registered and updated as a bundle in the service manager located in the OSGi service gateway. It is implemented by a mapping scheme for abstract and specific service management.

*2.3 MIPA*

Pervasive computing creates environments that embed computation and communication in a way that organically interacts with humans to enhance or ease their daily behavior. Contexts refer to the pieces of information that capture the characteristics of computing environments, and context-awareness allows applications to dynamically adapt to the pervasive computing environment.

PDPC (Property Detection for Pervasive Context) is one of the primary approaches to achieving context-awareness. Specifically, context-aware applications need to detect whether contexts bear user-specified properties, in order to adapt their behavior accordingly. Predicate detection is a promising approach to detecting contextual properties in asynchronous environments. Predicate detection basically relies on the classical Lamport's definition of the happen-before relationship resulting from message causality, and it's "on the fly" coding given by logical vector clocks. The Middleware Infrastructure for Predicate detection in Asynchronous environments (MIPA), to support PDPC in pervasive computing environments. MIPA aims at supporting the development and deployment of various predicate detection-based contextual property detection schemes for different pervasive computing scenarios.

*2.4 IPAC*

The IPAC aims at delivering a middleware and service creation environment for developing embedded, intelligent, collaborative, context-aware services in mobile nodes. IPAC relies on short range communications for the ad hoc realization of dialogs among collaborating nodes. Advanced sensing Components leverage the context-awareness attributes of IPAC, thus rendering it capable of delivering highly innovative applications for mobile and pervasive computing. IPAC networking capabilities are based on rumor spreading techniques, a stateless and resilient approach, and information dissemination among embedded nodes. Spreading of information is subject to certain rules (e.g., space, time, and price). IPAC nodes may receive, store, assess and possibly relay the incoming content to other nodes. The same distribution channel is followed for the dissemination of new applications and application components that "join the IPAC world". IPAC aims at providing all the communication functionality, relying on flexible components: the Sensing Elements Component, the Short Range Communication Component, and the IPAC core middleware itself. An important feature of IPAC is the embedded intelligence which relies on emerging knowledge representation and reasoning schemes, allowing behavior self-adjustment, seamless interoperation at the messaging level and software integration.

*2.5 PANOPLY*

Panoply is a middleware that aims to enable secure and scalable interactions among devices that participate in a ubiquitous computing environment. This research builds on earlier work done in *active or intelligent spaces* and deals with the concepts of group formation and change, event management and policy management, which have not been dealt with in a comprehensive manner. Current and past ubiquitous computing research has concentrated on building more intelligence into physical spaces or designing better applications. Though largely effective, existing systems lack a common representational model for device communities, semantics for the formation and interaction of such communities, and ways to handle the vast permutations of context and policy disagreements that might occur in a global-scale ubiquitous system. Security and access control solutions are domain-specific and neither extensible nor scalable. The heterogeneity of devices, software and networks we are seeing today is only going to increase in the future, and ad hoc design of active spaces cannot provide a scalable or a secure solution for the systems of the future, where interoperation is going to be all-important.

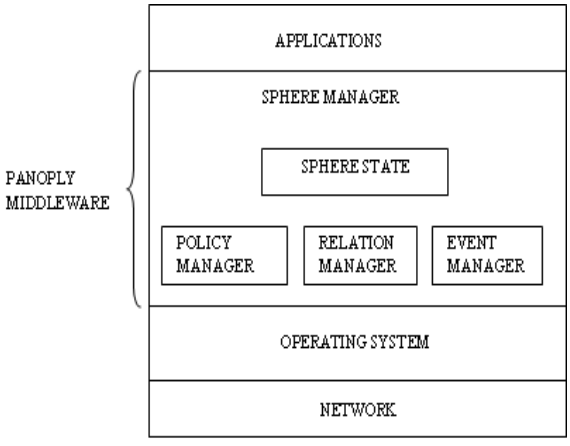

Fig 3: The Panoply Architecture

## 3. Classification of the Pervasive Middleware Requirements

Pervasive computing brought new challenges to distributed and mobile computing. We identify the following five fundamental requirements for service composition in pervasive environments: discoverability, adaptability, context awareness, heterogeneity, and environment.

Discoverability is a major issue for ubiquity and composition as devices and services need to be located and accessed before being composed. One of the fundamental challenges of distributed and highly dynamic environments is how the applications can discover the surrounding entities and, conversely, how the applications can be discovered by the other entities in the system. In a pervasive system, the execution environment of applications can be logically considered as a single container including all applications, other components, and resources. Moreover, the idea in distributed environments is that the resources can be accessed without any knowledge of where the resources or the users are physically located.

Adaptability is the ability of a software entity to adapt to the changing environment. Changes in applications' and users' requirements or changes within the network, may require the presence of adaptation mechanisms within the middleware. Moreover, adaptation is necessary when a significant mismatch occurs between the supply and demand of a resource. As the application's execution environment changes due to the user's mobility, the vital resources need to be substituted by corresponding resources in the new environment in order to ensure continuous operation. The requirement for adaptation is present on many different layers of a computing system.

Context awareness is the ability of pervasive middleware to be aware in terms of devices coming and leaving; functionalities offered and retrieved quality of service changing, etc. They need to be aware of all these changes, in order to offer the best functionalities to applications regardless the context around. When considering context-aware systems in general, some common functionalities that are present in almost every system can be identified: context sensing and processing, context information representation, and the applications that utilize the context information. In general, the context information can be divided into low and high-level context information. Low-level context information can be collected using sensors in the system. Low-level context information sources can be combined or processed further to higher level context information.

Able to operate across different homogeneous environments, seamless integration of devices and environments, taking on new contexts when new resources become available

Environments, where connectivity is very variable. A pervasive middleware need to take the non functional parameters of applications and devices into consideration in order to provide viable and flexible composition plans. QoS parameters for environments concern not only the services but also the devices where the execution is taking place. The composition execution needs to rely on this parameter in order to take place in the best conditions. Not only the QoS of different services need to be compatible, but also the devices performing the composition need to respect certain characteristics and constraints.

## 4. Comparison of Pervasive Middleware Architecture

Fig 4: Comparison of pervasive middleware with requirements.

| Middleware | Adaptability | Context-awareness | Heterogeneity | Discoverability | Environment |
|---|---|---|---|---|---|
| UBISOAP | ✓ | | | ✓ | ✓ |
| PANOPLY | | | | ✓ | ✓ |
| MIPA | ✓ | ✓ | | | ✓ |
| IPAC | ✓ | | | | |
| CDTOM | | ✓ | ✓ | | ✓ |

## 5. Conclusion

In this paper, a survey of five middleware for pervasive environment, located in the middleware layer, UBISOAP, PANOPLY, MIPA, IPAC, and CDTOM. Here, the classification of these middleware under several requirements related to the ubiquity of the environments. If some requirements such as discoverability and context awareness are well verified, adaptability still being explored.

## 6. References

[1]T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, 2005.

[2]M. Satyanarayanan "Pervasive Computing: Vision and Challenges", IEEE Personal Communication, 2001.

[3]D.Saha and A.Mukherjee, Pervasive Computing: A paradigm for the21st Century. IEEE Computer Magazine, Mar 2003

[4]Qusay H.Mahmoud, Middleware for communication, John Wiley&Sons, 2001

[5]BernsteinPhilipA, Middleware, Communication of the ACM, vol. 39, no 2, pp86–98, Feb1996

[6]Guruduth Banavar and Abraham Bernstein "Software infrastructure and design challenges for ubiquitous computing applications", Commun.ACM, Vol. 45, pp. 92-96, December 2002

[7]Mark Weiser, Ubiquitous computing, IEEE Computer, 1993

[8]Panoplywebsite:http://www.lasr.cs.ucla.edu/panoply/panoply.html

[9] Caporuscio Mauro; Raverdy Pierre-Guillaume; Issarny Valérie*IEEE Transactions on Services Computing* (2010)

[10] IPAC website: http://pcomp.di.uoa.gr/projects.jsp

[11]SEEMPubSwebsite: http://www.fit.fraunhofer.de/projects/mobiles-wissen/seempubs_en.html

[12]ubiSOAP:website:https://wwwroc.inria.fr/arles/index.php/software/84-ubisoap-a-service-oriented middleware-for-seamless-networking-featuring multi-radio-networking-b3gsoap-and-multi-network-service-discovery-developed-as-part-of-the-plastic-middleware.html

[13]MIPA website: http://code.google.com/p/mipa/