# Feature Based Generic Middleware

V. Subashini [1], R. Vivadha [2], J. Madhusudanan [3]

[1] *PG Student, Computer Science and Engineering, Sri Manakula Vinayagar Engineering College, Puducherry, India.*

[2] *PG Student, Networking, Sri Manakula Vinayagar Engineering College, Puducherry, India.*

[3] *Research Scholar, Pondicherry University, Puducherry, India.*

[1] suba89.v@gmail.com, [2] vivadha.tech@gmail.com

*Abstract* - **Now-a-days pervasive middleware systems tends to increased a lot and used in many application areas like home, health, retail shop, transport, mobile etc., to use the environment in a smarter way. In the existing system, middleware has been proposed for each and every application separately. To use different middleware for different applications, a generic middleware is proposed in this paper, so that this architecture framework can be used in different applications. To propose a new middleware architecture framework, a set of pervasive systems architectural features relevant to all the applications are collected and corresponding component diagrams are generated according to the feature selection. At last by the selection of features, new reference architecture is generated automatically that does not compromise the quality of the architecture designed by human architects.**

*Keywords* – **Pervasive Computing, Pervasive Middleware, Context Awareness, Component Diagram**

## I. INTRODUCTION

The thought at the back of pervasive computing is to surround ourselves with computers that are carefully tuned to put forward an unremarkable support as we navigate through our work and personal lives. [1] Pervasive Computing also referred to as Ubiquitous Computing abbreviated to “ubicomp” refers to a new sort of computing in which the computer completely involve into the life of the users. In pervasive computing, computers are useful but invisible force, supporting the user in meeting his or her needs without getting in the way.

Mark Weiser, the father of Ubiquitous Computing, described it as “The highest ideal is to make a computer so imbedded, fitting, so natural, that we use it without even thinking about it.” Additionally, this computing infrastructure is supposed to be able to sense the context in which particular situations takes place and adapt to them according to its location of use, the people and

The main component of this computing infrastructure is Context Awareness which describes a model of computing in which users intermingle with many different mobile and immobile computers and categorize a context-aware systems as one that can adapt according to its location of use, the group of nearby people and things, as well as the changes to those things over the course of the day. The assurance of context-awareness is that computers will be able to recognize enough of a user’s current situation to suggest services, resources, or information appropriate to the particular context. The features of context to a particular situation may vary and include the user’s current location, current role, past movement, and affective state. Further than the above the feature include the current date and time, and other things and people in the surroundings.

The most important component in pervasive architecture is the pervasive middleware. Middleware can be considered as software which provides a set of enabling services that exist in between applications and the basic operating systems, network protocol stacks and hardware.[2] It allows multiple processes running on one or more hosts to interact visibly across a network and can also allow and shorten the integration of heterogeneous software and hardware components.

The main concept of this paper is to create a generic middleware architecture that fits for all type of pervasive applications in a much easier and efficient way. This generic architecture uses reusability concept from Software Engineering approach called Software Product Line (SPL). It aims the growth of software components that share a common and managed set of features. The basics of the Software Product Line approach are divided between Domain Engineering, Application Engineering and variability and commonality management. Domain Engineering is the advancement of core assets to be used in the product line, while Application Engineering is disturbed with building the final products on top of the product line infrastructure. They are loosely coupled and are coordinated by platform releases. Domain engineering addresses development for reuse while application engineering addresses development with reuse. Variability and commonality management is for configuring the Software Product Line, adding new core assets, or enhancing existing ones.

In coming sections we are going to discuss about the basics of middleware, features of middleware systems, the proposed work and how it is working with a scenario.

## II. MIDDLEWARE

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is a type of software layer that provides services to software applications ahead of those available from the operating system. It is present in between the operating system the application program on both the sides of the distributed computing in a network which provides a common programming abstraction across a distributed system.[3] Middleware acts as an abstraction layer that hides detail about the software used in an application or the hardware devices used. A middleware layer is used for bridging the back end systems and the user.

Generally middleware should be easy to use and should provide language transparency, location transparency and message integrity.[10] It makes things visible to the end user, provides consistency, security, privacy, and capabilities.

Traditional middleware has some common requirements like network communication, coordination, heterogeneity, reliability, scalability. To overcome some of the limitations in existing middleware solutions and to increase the range of applicability of middleware, next-generation middleware gratifies one or more of the following requirements: dynamic reconfiguration, adaptivity, context-awareness, asynchronous communication and lightweight design.

## III. FEATURES OF PERVASIVE MIDDLEWARE SYSTEMS

The optimization of quality is critical for pervasive systems as they involve invisible operation that causes them to be miniature in size and work with limited memory. In order to have a pervasive computing environment, it is essential to have the following features like ubiquitous access, context awareness, intelligent interaction and natural interaction. Let us discuss in detail about the above features.

### *A. Ubiquitous Access*

Ubiquitous access is the sensors and the actuators that transfer input and output between the real world and the virtual world based on wireless communication infrastructures.[4] Owing to the mixture of hardware and software capabilities, a communication infrastructure is necessary for preserving knowledge about device characteristics and organizing consistent device interactions.

### *B. Context Awareness*

Context awareness refers to the ability of the system to recognize and localize objects as well as people and their intentions. It also includes tracking other objects and organizing the activities with respect to and relative to other objects.[4]

### *C. Intelligent Interaction*

Intelligent interaction refers to the ability of the technology-rich environment in the pervasive systems to adapt to people dealing with it.[5]

### *D. Natural Interaction*

Natural interaction refers to the interaction between the humans and the surrounding environment and how the surrounding environment receives inputs from the user and acts upon it.[6]

## IV. PROBLEM STATEMENT

This paper mainly focuses on generating an automatic pervasive middleware architecture that can be used for any type of application by selecting features according to the prescribed environment. In the existing system, for each and every application, the user has to choose different middleware architecture to be used in a smarter way. By using Software Product Line approach from software engineering, we move to generic architecture instead of using different architecture for different environment. [9] Product lines have mainly been specified for application families that are distinguished by their multi-layer systems. Analysis is made for existing pervasive middleware architectures and a suitable enough reference architecture has been created that can be used as a guide for building any type of systems.

Commonality and variability approach is used to collect set of features from different applications. By displaying all the features, the designers may select the needed features according to the environment. By selecting features, the pervasive architecture is generated automatically to be used. These automatically generated architectures are compared with human designed architecture.

## V. PROPOSED APPROACH

The process of generating pervasive middleware architecture automatically is splitted into three modules for implementation purpose. In the first module, the pervasive middleware has to be defined and main features of pervasiveness have to be achieved. The features of pervasiveness are discussed in above section. The second module focuses on grouping different features from various middleware architectures and categorizes them according to the environment. The third module focuses on generating component based architectures that best matches the selected features. At last the comparison is done by automatically generated architecture with human designed architecture.

## VI. IMPLEMENTATION PROCESS

Initially, the features are categorized according to the environment presented. The needed features are selected by the designers which have been displayed in a tree view form. Any type of features can be added or removed by checking availability the criteria.

Secondly, the component diagrams are generated automatically according to feature selection. These selected features and the component diagrams are exported in the form of XML documents. The cause behind using XML throughout the architecture generation process is that XML is easier and improved way for normalizing among the tools used in our approach.

Thirdly, using Visual Studio 2008, programs have been developed that maps each selected features to corresponding set of components in XML format.

A tool called RA generator for architecture generation is designed in order to remove the unnecessary connections and to glue the unconnected components that come from different categories according to a pre-defined lookup table.

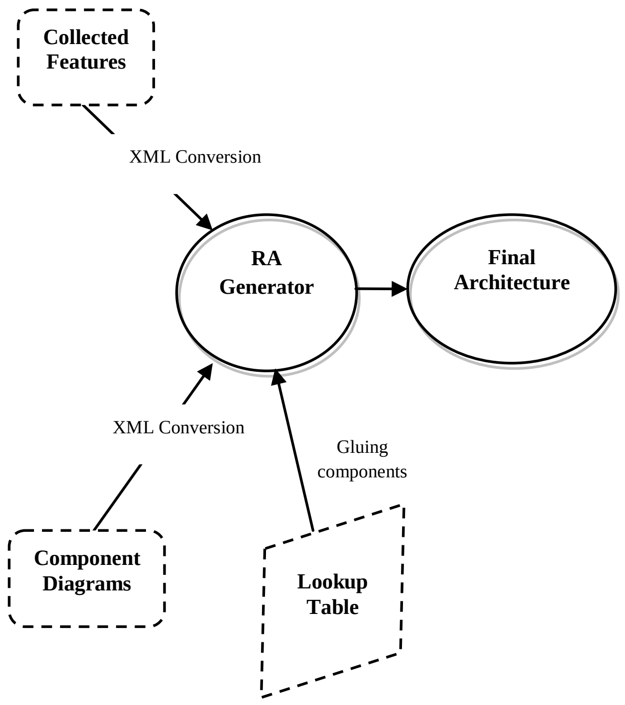


**Figure 1: Implementation Process**

The lookup table is developed by collecting the similar components together from the different categories and checking if there could be any relation among them.

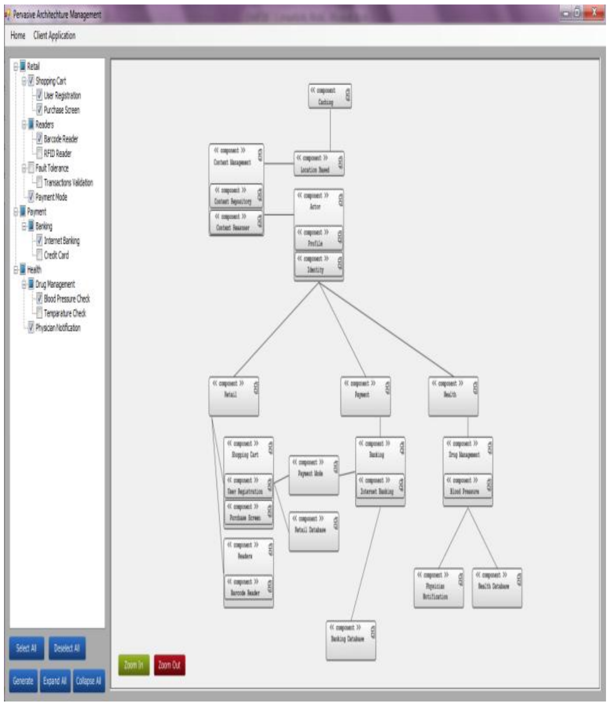


**Figure 2: Architecture generation according to feature selection**

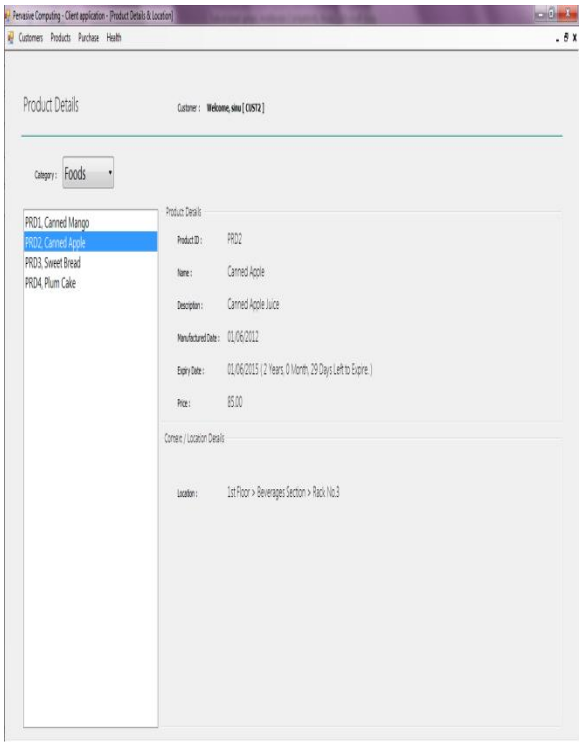
At last, a final architecture is generated according to the designer's need. By using this middleware architecture, the designer can use the living environment in a smarter way.

## VII. THE CASE STUDY

A scenario has been worked out for generating architecture relating to retail shop, banking and healthcare, in a smarter way.

### A. *Retail Scenario*

The customer enters the retail shop and selects a shopping cart outfitted with Radio Frequency IDentification (RFID) readers and Personal Digital Assistant (PDA). He/ She recognizes to the system using her RFID enabled loyalty card where the customer ID is read into the PDA and transmitted to the authentication server on the cart and gains right of entry by entering her personal identification number (PIN). [7] The system logs him/ her in, act in response with a welcome message and then proceeds to present a shopping list.


Pervasive Computing - Client application - [Product Details & Location]
Customers
Products
Purchase
Health

Product Details

Customer : **Welcome, sinu [ CUST2 ]**

Category : Foods

PRD1, Canned Mango
PRD2, Canned Apple
PRD3, Sweet Bread
PRD4, Plum Cake

Product Details

Product ID : PRD2
Name : Canned Apple
Description : Canned Apple Juice
Manufactured Date : 01/06/2012
Expiry Date : 01/06/2015 ( 2 Years, 0 Month, 29 Days Left to Expire. )
Price : 85.00

Context / Location Details

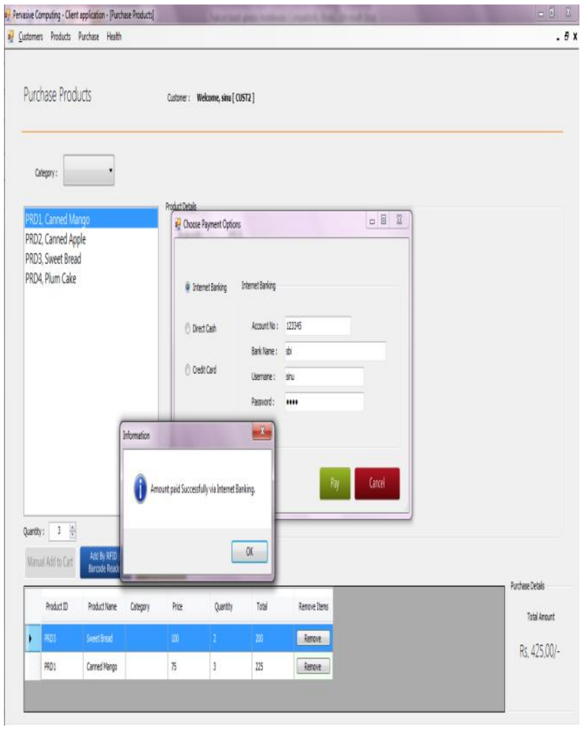
Location : 1st Floor > Beverages Section > Rack No.3

**Figure 3: Product details and location**

The shopping cart consists of a screen to display items in the shop as well as items present in the cart. When the customer chooses an item, details about the product location is also displayed. While the customer adds items in the cart, he/ she can see the item description, manufacturer date, expiry date, and price of the item. It also shows the total cost of all the items present inside the cart. If the customer wants to remove an item from the cart, the system rescans and the total amount is calculated again.

### B. *Banking Scenario*

After all the items are selected and placed in the cart, the total amount of the cart is displayed in the screen and displays options to pay the amount through direct cash or through internet banking or via credit card. The customer checks the needed option and pays the amount. If he/ she selects internet banking or credit card means, the customer logs into his/ her personal shopping account to pay the amount accordingly.


Pervasive Computing - Client application - [Purchase Products]
Customers Products Purchase Health
Choose Payment Options
Information
Pay
Cancel
OK
Manual Add to Cart
Add By RFID Barcode Reader

Purchase Products

Customer : Welcome, sinu [ CUST2 ]

Category :

PRD1, Canned Mango
PRD2, Canned Apple
PRD3, Sweet Bread
PRD4, Plum Cake

Product Details

(●) Internet Banking ( ) Direct Cash ( ) Credit Card

Internet Banking

Account No : 123345
Bank Name : sbi
Username : sinu
Password : ••••

Amount paid Successfully via Internet Banking.

Quantity : 3

| Product ID | Product Name | Category | Price | Quantity | Total | Remove Items |
|---|---|---|---|---|---|---|
| PRD3 | Sweet Bread | | 100 | 2 | 200 | Remove |
| PRD1 | Canned Mango | | 75 | 3 | 225 | Remove |

Purchase Details

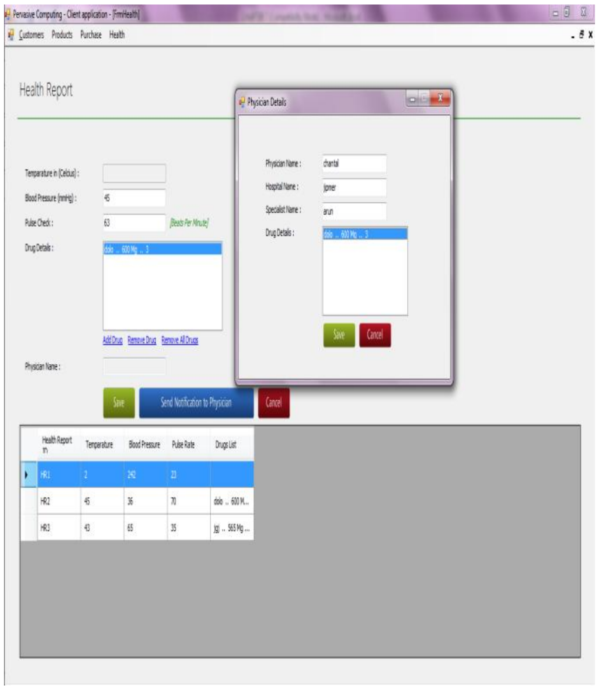
Total Amount

Rs. 425.00/-

**Figure 4: Payment options to pay the cart amount**

*C. Healthcare Scenario*

While walking in the shop, a need might happen to check for health condition.[8] If a customer suddenly faces with a certain health condition, he/ she should be reported to physician in nearby hospital according to the circumstances.


Pervasive Computing - Client application - [FrmHealth]
Customers Products Purchase Health
Physician Details
Add Drug
Remove Drug
Remove All Drugs
Save
Send Notification to Physician
Cancel

Health Report

Temparature in (Celcius) :
Blood Pressure (mmHg) : 45
Pulse Check : 63 [Beats Per Minute]
Drug Details : dolo .. 600 Mg .. 3
Physician Name :

Physician Name : chantal
Hospital Name : jipmer
Specialist Name : arun
Drug Details : dolo .. 600 Mg .. 3

| Health Report ID | Temparature | Blood Pressure | Pulse Rate | Drugs List |
|---|---|---|---|---|
| HR1 | 2 | 242 | 23 | |
| HR2 | 45 | 36 | 70 | dolo .. 600 M... |
| HR3 | 43 | 65 | 35 | jgj .. 565 Mg ... |

**Figure 5: Notifying the health status to physician**

The sensor senses the patient’s blood pressure, temperature and pulse and suggest for drugs reported by the physician.

## VIII. CONCLUSION

We finally conclude that new automatically generating middleware architecture has been proposed by feature selection process that meets all the user requirements in a smarter way. This approach will not compromise the quality of the middleware architecture. We have discussed a case study of retail shop with the requirements, and indicated how this approach can be used to generate a pervasive architecture.

This work can be extended by adding new features and removing the current unnecessary features for different applications. It should be extended by including an automated mechanism in order to sense if there are any inconsistency or redundancy between the selected features exist. And at last evaluation is done by comparing the automatically generated architecture with human designed architecture with some evaluation metrics that should not compromise with one another.

## REFERENCES

[1] R. Jason Weiss, “Ubiquitous Computing”, Development Dimensions International, J. Philip Craiger, University of Nebraska–Omaha,April 2002 Volume 39 Number 4.

[2] “Middleware Issues”, Nato Otan, RTO-TR-IST-030, 2008.

[3] David E. Bakken, “Middleware”, Washington State University, 2003.

[4] Ferscha. A, “Coordination in pervasive computing environments”, Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE’03). Washington, DC: IEEE Computer Society, June 2003. Pages: 3 - 9.

[5] Sherif G. Aly, Sarah Nadi, Karim Hamdan, “A Java-Based Programming Language Support of Location Management in Pervasive Systems”, International Journal of Computer Science and Network Security (IJCSNS). Vol. 8 No. 6 pp. 329-336, June 2008.

[6] Yared, Rami and Défago, Xavier, “Software architecture for pervasive systems”, In Journées Scientifiques Francophones (JSF), Tōkyō, Japan, November 2003.

[7] Panos Kourouthanasis, George Roussos, “Consumers and Pervasive Retail”, ELTRUN-Athens University of Economics and Business, Birkbeck College, 2003.

[8] D. Vassis, P. Belsis, C. Skourlas, G. Pantziou, “A pervasive architectural framework for providing remote medical treatment”, Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments, July 16-18, 2008.

[9] Javier Munioz, Vicente Pelechano, Carlos Cetina, “Software Engineering for Pervasive Systems- Applying Models, Frameworks and Transformations”, IEEE International Conference on Pervasive Services, 2007.

[10]Vaskar Raychoudhury, Jiannong Cao, Mohan Kumar, Daqiang Zhang, “Middleware for Pervasive Computing: A Survey”, 2011.