Novel Ubiquitous Interoperable Context-aware Smart Environments through Web Services

R. Vinob chander

***Abstract*— Today's digital world is made up of a diverse collection of electronic devices like mobiles, net books, PDA's, digital cameras, navigation systems etc for varied uses like information retrieval, entertainment and so on. A typical smart environment is heterogeneous in nature where the various devices assist the user(s) seamlessly with their services. For such an intelligent environment, one of the major research challenge is interoperability among the services discovered and composed. In this paper, I present an architecture using web services (SOAP and/or REST based) as an effective way of addressing the interoperability issue. The proposed solution could be integrated into any smart environment framework that may cater for other requirements as well such as context-awareness, user preferences and more. A case study on a typical smart conference room is done to depict the process.**

***Index Terms*— Interoperability, Smart Environments, Web service, Middleware, SOAP**

I. Introduction

A smart environment is a context sensitive system based on ubiquitous computing, in which the environment interacts with its inhabitants through embedded dedicated devices. And the number of these devices are growing steadily. They range from television sets and other appliances to mobile handsets like tablets PCs, PDAs and phones. One of the compelling goal of any smart environment is interoperability among the services provided by these heterogeneous devices, in a seamless fashion. The interoperability factor is due to differences in operating systems, programming language and hardware for the sub-systems. While device interoperability in one issue, this paper focuses on syntactic interoperability. The ability to inter-operate with other services will permit a smart environment to automatically adapt/compose services to satisfy user preferences, or resolve conflicts when two or more users share information or services. The system will thus be able to provide the user automatically with what they wanted, in a way they wanted and at times they wanted. This paper proposes an architecture for an efficient interoperable mechanism for any smart environment. More specifically, in section 2, some related work is examined. In section 3, a brief case of a smart conference room that aims to highlight the interoperability scenario is described. Section 4 describes the middleware solutions for interoperability. Section 5 elaborates on the architectural proposal, ISE. Section 6 proposes a hypothetical environment to implement ISE. Finally, section 7 draws the conclusion of the paper.

II. Related Work

WHYRE [1] is a hands-free, sensory augmented, wearable computer designed to turn museums and archaeological sites into communicating machines. It offers a unified interface to multiple format contents, including interactive 3D, sensors driven QTVRs, and streamed animations. It is based on an IA32 mobile platform with a 3D graphics accelerator. Its operating system is Windows XP Embedded. Its high-level architecture is shown in Figure 1. WHYRE uses a Centralized multimedia content that are shared and distributed though the network. The architecture is for a specific use case. Context is only managed at the application. It is not shared and does not have any impact on the environment. (interoperability is thus reduced).

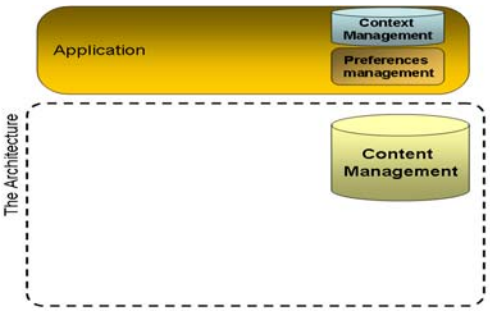

Figure 1. WHYRE architecture

Service-Oriented Context-Aware Middleware (SOCAM) [2] is an architecture for the building and rapid prototyping of context-aware services. It makes use of a context model based on OWL. SOCAM uses a centralized architecture and has no privacy protection. Another work, CoCA(Collaborative Context-aware) [3] platform for pervasive computing uses an efficient distributed architecture and an ontology model. However there is no privacy protection with CoCa. CoBrA [4] is a context broker architecture based on software agent system. The broker manages contextual information described by OWL and RDF/RDFS. Though it also allows users to define privacy policy, the architecture of CoBrA is a centralized one. ISE makes use of an approach where the problems with the above frameworks could be overcome.

Manuscript received Jan 11, 2012; revised Jan 22, 2012.
R. Vinob chander is with SSN College of Enginering, Chennai, INDIA (phone: 918754488284; e-mail: vinobchanderr@ssn.edu.in).

III. CASE STUDY: SMART CONFERENCE ROOM

A typical conference room (Fig. 2) will have a door, a projector, microphone, chairs, lights and more . Say each of the following devices has built in sensors/embedded devices equipped with a framework that makes use of a centralized architecture or a distributed one. Also outside the meeting rooms door is a bulb that glows whenever a meeting is on. With such a setup if the services are not able to interoperate in a timely, and efficient fashion, the bulb will not glow even if the meeting is on. At a higher lever success will be basically on the following condition (Fig. 3).

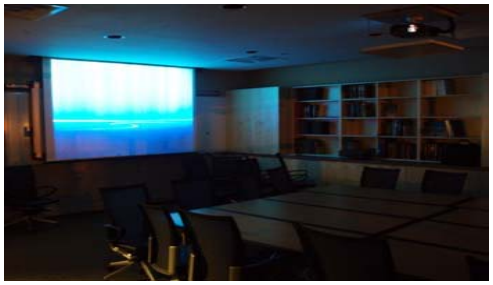

Figure 2. A smart conference room

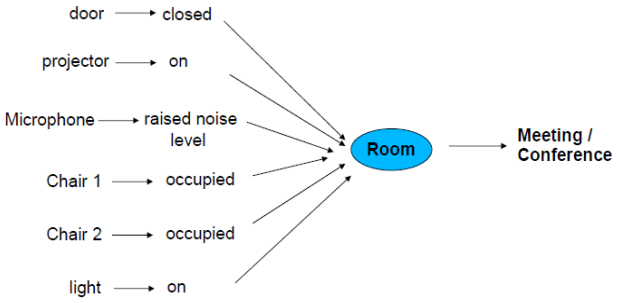

Figure 3. Success criteria in a smart conference room

IV. MIDDLEWARE SOLUTIONS FOR INTEROPERABILITY

There are several middleware technologies and languages that address interoperability issues in a smart environment. These middleware technologies are built to cater the interoperability tiers suiting the smart environment requirements. The common selected approaches that drive interoperability in any smart environment are Common Object Request Broker Architecture (CORBA), Microsoft Component Object Model (COM), .NET Framework, Sun's Java 2 Enterprise Edition (J2EE) and World Wide Web Consortium's (W3C) extensible Markup Language (XML) based Web Services. In the following section, we evaluate the interoperability approach using mentioned technologies and selected SOAP/REST based Web Services as our interoperability solution for a smart environment.

A. Common Object Request Broker Architecture (CORBA)

Common Object Request Broker Architecture (CORBA) is an architectural framework established by The Object Management Group (OMG) [5] as part of standard in Object Management Architecture (OMA). The OMA set of standards consists of Object Services, Object Request Broker (ORB) function, common facilities, application objects and domain interfaces. CORBA is structured to allow integration of wide range of object systems and provides mechanisms to find object implementation for a request, to prepare implementation to receive request and finally communicate with the data that making up the request. CORBA provides a mechanism to define interfaces between components, and specifies standard services such as persistent object services, directory and naming services and transaction services which describes the interoperability feature for CORBA compliant applications. Researchers from University of Texas at Arlington developed a smart home architecture called MavHome [6]. MavHome architecture was built using CORBA interface catering software components and power line control for managing systems in smart home environment. Although CORBA interface could resolve interoperability by providing interoperation feature for managing disparate systems, it also has some drawbacks which may not be ideal for implementation in a smart environment. Modification is needed to enable joint execution of tasks among heterogeneous sub-systems, especially if the systems are not in compliant with CORBA specifications. Any modification of legacy systems in smart home environment could be costly and time consuming. Therefore a framework that will enable interoperability in managing heterogeneous systems without requiring modification to the existing systems is highly needed.

B. Component Object Model (COM)

Component Object Model or widely known as COM was introduced by Microsoft which enables applications built from binary components defined by software vendors. COM's successors are Distributed COM (DCOM) and COM+. These technologies are aimed to provide generic mechanism in integrating the components on Windows based platforms. In terms of interoperability, Component Object Model (COM) technologies provide similar features as CORBA. The difference is that COM address interoperability among binary software components while CORBA tackles at the source code level. However, the drawback of COM in providing interoperability is that it requires information of the remote systems before functioning and eventually leads into modifying the legacy systems that are not complied with COM standards. Similar to CORBA, modification of legacy systems in a smart environment is not desirable

C. Dot NET Framework

.NET Framework was developed by Microsoft to provide a set of standard components and languages which is compliant to ECMA-335 Common Language Infrastructure (CLI) standard. Common Language Infrastructure (CLI) enables code reusability in single or multiple operating system platforms. The main advantage of the .NET Framework is the Common Language Runtime (CLR) mechanism that allows objects used in one language can be used in another language. CLR depends on Microsoft Intermediate Language (MSIL) in producing managed code. All language development tools could produce the same MSIL regardless of the language used to write particular codes. The MSIL code produced by language development tools are then compiled by a Just-in-time (JIT) compiler to

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

produce the actual machine code that executes for specific applications. .NET Framework supports interoperability by providing cross-language platform where classes and objects are interchangeable and reusable without using a specialized Interface Definition Language (IDL). .NET also enables integration of .NET programs and legacy codes. Interfacing legacy codes is supported by enabling applications that are part of managed code environment generated by CLR to access unmanaged Dynamic Link Library (DLL) functions.

Legacy codes support are one of the contributing factor in promoting interoperability for smart home environment. .NET Framework could become an ideal solution for smart environment technologies especially with its language and platform independence features for application development. In addition, recent development of Mono Framework, enable porting the .NET features into broad broad-based interoperability solution by supporting open source based operating systems [7].

D. Java Middleware Technologies

Java Middleware technologies support interoperability by providing distributed protocols and APIs that can be used to create an interoperable system. In Java based platform, remote invocation or messaging is the key to achieve interoperability. Java middleware offers Remote Method Invocation (RMI) mechanism that is similar to CORBA-like object oriented middleware layer as distribution protocol. RMI enables objects to be called remotely from other applications in a heterogeneous environment. This feature also extends for interoperation execution between systems and information exchange. One of the implementation of Java Middleware in a smart environment is the OSGi framework [8]. The OSGi Alliance introduced the Open Service Gateway Initiative (OSGI) specification defines a standardized, component oriented, computing environment for networked services. Work by Diaz Redondo et.al [9] focused on service composition using OSGi framework for home environment. Another solution proposed by A.R Al-Ali et.al [10] demonstrated the potential of Java Server Pages in managing home appliances over heterogeneous environment. However, all the proposed design requires installation of Java Virtual Machine (JVM) in the remote systems. Java Middleware presents a competing approach to heterogeneous systems management similar to the one offered by CORBA and COM family. The core advantage of using Java Middleware technologies include its support for interoperability in terms of interoperation execution and information exchange, and full support for modification of existing systems. However, Java Middleware can only be implemented with the presence and requirement of Java Virtual Machine (JVM) in remote and local component of the system involved.

E. Web Services

Web Services are collected set of XML based protocols that provides fundamental blocks for creating distributed applications [11]. The functionality of Web Services are based on publish, discover and invoke that describes standardized concept of function invocation relying on web protocols, independent of any platform (operating system, application server, programming language, database and component model). Web Services consist of three entities [12]: a) Service Provider – Create Web Services and publish to the external environment by registering through Service Registry b) Service Registry- Registers and categorizes published services

c) Service Requester - uses Service Registry to find a needed service and bind them accordingly to Service Provider. Figure 4 below shows the three entities of Web Services.

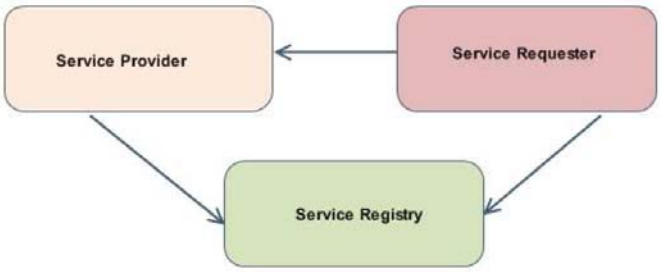

Figure 4. The core entities of web service

These entities of Web Services are founded upon three major standard technologies: Simple Object Access Protocol SOAP), Web Services Description Language (WDSL) and Universal Description Discovery Integration (UDDI). All these standards are based on XML as defined mechanism for data definition, initiated by the World Wide Web Consortium (W3C). Web Services Description Language (WDSL) provides a model along with XML based format in describing the Web services. A WSDL description is done with two levels of stages. One is the abstract stages consist the messages that it sends and receives. On the concrete stage, a binding determines the transport and wire format details for one or more interfaces. Ports or known as EndPoint combine the interface bindings information with a network address. Finally a service groups all the Endpoints that implement a common interface. Universal Description Discovery and Integration (UDDI) is the last element needed in providing Web Services implementation. The main goal of UDDI is the specification of a framework for describing and discovering Web Services. UDDI defines data structures and APIs for publishing service descriptions in the registry and querying the registry to look for published descriptions. UDDI is expected to be a service repository of business organization near future towards extending their business information and value added service for smart home environment. In a smart environment context, Web Services are identified as potential solution for solving interoperability dimension in managing disparate systems. It is also worth highlighting about organization like Open Building Information Exchange Group (OBIX), working towards developing comprehensive standards using XML and Web Services to cater information exchanges between heterogeneous systems in home and buildings [13].

F. The Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is an inter-application communication mechanism targeted for exchanging structured information in a distributed environment. SOAP exchanges information using messages. In the specification developed by World Wide Web

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Consortium (W3C), it is also included a method for encapsulating Remote Procedure Calls (RPCs) within SOAP messages. Ideally, SOAP is created to support loosely-coupled application that could exchange one-way asynchronous messages. SOAP comprises the following elements: an envelope describing the content of the message and the way to process it, a set of encoding rules to express instances, application defined data types and a convention for the representation of remote procedure calls and responses. Figure 5 shows the structure of a SOAP message.

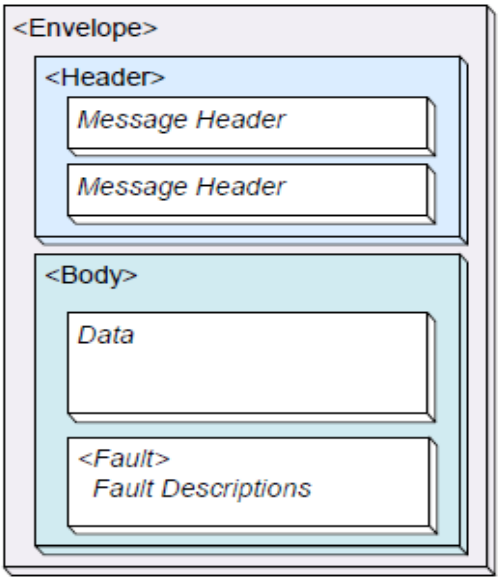

Figure 5. Structure of a SOAP message

Each envelope consists of a header and a body. The information intended to be transported will reside in the body of the message. Any additional information or value added services will be included in the header. SOAP protocol can be used in two different style called document-style and RPC-style. In document style, interaction happens between two applications agreeing upon the structure of documents exchanged among them. While in RPC-style, a SOAP message encapsulate request while another message encapsulate the response. In this paper, we will demonstrate the ability of SOAP in providing generic interoperability mechanism.

V. ISE (Interoperable Smart Environment) Architecture

Sub-systems in a smart environment comprise a number of tasks that are associated with the sequential use of different systems and applications. The need for interoperability in managing these sub-systems has led towards a transition of vendor independence and open systems, taking into account of middleware and Internet technologies. In this section we propose an architecture that builds upon the general trend towards interoperability for a smart environment taking an example of a smart conference room. Figure 6 shows the proposed system architecture.

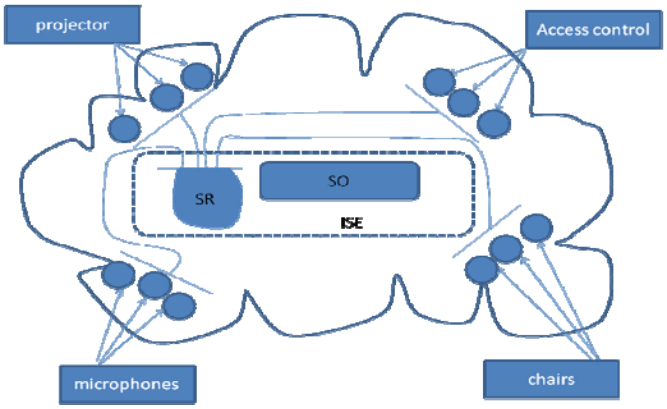

Figure 6. ISE architecture

The architectural diagram is shown with respect to a smart conference room. Each of the devices expose their services to a universal service repository(SR). The devices can exchange information among themselves through an application gateway called as an service orchestrator(SO). The actual servant could be written in any language and resides with the individual devices. The devices then expose the services by registering themselves in a public service repository. The service is identified by their corresponding WSDL(Web service definition language) files, if SOAP based or with their uri's, if REST based. The message exchange is then XML-SOAP based or HTTP- based. Depending on the kind of smart environment we can opt for either SOAP based implementation or REST based one. Although the service, if implemented as a SOAP based one is heavy, it comes with features such as security, scalability and all others that are part of the container upon which they are deployed. So, with this proposed style interoperability issues could be solved.

VI. Prototype Implementation

Considering the interoperability, privacy and scalability features of Java EE Container, Java language together with SOAP service classes are used to operate, manage and control the system operation of sub-systems in a smart conference environment. The Container used for the implementation is GlassFish Server V 3.1.1. The engine which manages the flow and orchestration of services using SOAP, is written in java and is a modified version of the BPEL engine, that is specific to a conference room. The engine is additionally modified so that it could be plugged in to NetBeans7.1. This is because NetBeans 7.1 does not out of the box have support for BPEL.

The XML SOAP web services is developed using Java Programming language in NetBeans 7.1 Integrated Development Environment. The sub-element details are stored in Java DB database and retrieved as JPA(Java Persistence API) entities. The sub-systems in smart conference room must be secure in terms of interoperation and reliability while changing states between multiple applications. Realizing the importance of security, GlassFish provides these features out of the box and the security level required with a service could be defined using a declarative approach. Services were developed for the elements projector, access control microphones and chairs

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

and were deployed into the Glassfish Container. These services were tested in a laptop within a virtual room environment created using VRML. These services communicated effectively by discovering each other through the registry and finally switched on the virtual bulb. In future with sufficient funds available this will be tested on sensor devices with embedded CPUs deployed with the implemented ISA.

VII. Conclusions and Future Work

The work presented in this paper deploys the interoperability requirement for a smart conference room namely, the Simple Object Access Protocol (SOAP) with Web Services ability in providing interoperation and scalability for managing sub-systems. In implementing interoperability among sub-systems, data representation must be independent regardless of operating platform. Our work indicates that SOAP protocol maximizes the interoperability and performance of sub-systems. Future research holds a lot of promises especially in extending the interoperability dimension towards semantic and business tiers. Also device's physical level and network level interoperability is of concern. As an extension to this work, efficient context-awareness and user preferences algorithms could be built and deployed on to ISA to truly make a self improving smart environment

REFERENCES

[1] Tullio Salmon Cinotti, Raviprakash Nagaraj, Giuseppe Mincolelli, Giuseppe Raffa, Luca Roffia, Fabio Sforza, "WHYRE: A Context-Aware Wearable Computer for Museums and Archaeological Sites," Eighth IEEE International Symposium on Wearable Computers (ISWC'04), 2004, pp.174-175.

[2] Tao Gua,b, Hung Keng Punga, Da Qing Zhangb, "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications 28 (2005),pp. 1–18 .

[3] Simon schubiger-Banz, Beat Hirsbrunner, "A model for software configuration in ubiquitous computing environments," first international Conference, Pervasive 2002, Springer, pp. 181-194.

[4] Akio Sashima, Noriaki Izumi, Koichi kurumatani, "Agents that coordinate web services in ubiquitous computing," Ubiquitous computing systems,Second International Sysmposium, UCS 2004, Springer, pp. 131-145

[5] Michi Henning and Steve Vinoski, Advanced CORBA Programming with C++, Addison-Wesley, 1999

[6] D. J. Cook, M. Youngblood, E. O. Heierman, III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: an agent-based smart home," in Pervasive Computing and Communications, 2003,(PerCom 2003), Proceedings of the First IEEE International Conference on, 2003, pp. 521-524.

[7] Mono-Project, http://www.mono-project.com/Main_Page

[8] OSGi Alliance, http://www.osgi.org

[9] R. P. Diaz Redondo, A. F. Vilas, M. R. Cabrer, J. J. Pazos Arias, and L. Marta Rey, "Enhancing Residential Gateways: OSGi Service Composition," Consumer Electronics, IEEE Transactions, vol. 53, University Science, 1989, pp. 87-95

[10] A. R. Al-Ali and M. Al-Rousan, "Java-based home automation system,"
Consumer Electronics, IEEE Transactions, vol. 50, 2004, pp. 498-504

[11] T. Perumal, A. R. Ramli, and C. Y. Leong, "Design and implementation
of SOAP-based residential management for smart home systems,"
Consumer Electronics, IEEE Transactions on, vol. 54, 2008, pp. 453-459

[12] Gisutavo Alonso, Fabio Casati, Harumi Kuno and Vijay Machiraju, Web Services: Concepts, Architectures and Applications, Springer-Verlag Berlin Heidelberg, 2004.

[13] oBIX (Open Building Information Xchange), http://www.obix.org

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)