Ubiquitous Computing: Connecting Pervasive Computing through Semantic Web

SACHIN SINGH SUSHIL PURADKAR YUGYUNG LEE

School of Computing and Engineering

University of Missouri-Kansas City, MO 64110, USA

{sbs7vc,shpgzd,leeyu}@umkc.edu

Abstract

Ubiquitous computing refers to building a global computing environment where seamless and invisible access to computing resources is provided to the user. Pervasive computing deals with acquiring context knowledge from the environment and providing dynamic, proactive and context-aware services to the user. A Ubiquitous computing environment is created by sharing knowledge and information between pervasive computing environments. In this paper we propose a framework that uses the potential of the Semantic Web to weave pervasive computing environments into a Ubiquitous-computing environment. We discuss how the collaboration of these pervasive environments can create an effective Ubiquitous computing environment referred herein as the Integrated Global Pervasive Computing Framework (IGPF). We test the effectiveness of the Ubiquitous environment through a small scenario from a prototype system that we have implemented over this framework to handle medical emergency scenario.

Key Words: Ubiquitious Computing, Pervasive Computing, Semantic Web, Context Aware Services

Introduction

We believe that Ubiquitous computing is the next wave of computing after the Internet wave. Ubiquitous computing aims to revolutionize the current paradigm of human-computer interaction. Computers have been used in various aspects of human life, but in most cases human beings have to adapt their behavior to each system. Ubiquitous computing as envisioned by Weiser [22] is a computing environment computing systems weave themselves in the fabric of everyday life and become invisible. Invisibility is the most important aspect of Ubiquitous computing. The user is exposed to a few sets of services available to him/her and is oblivious to the complex system implementing those services [18]. This takes the human-computer interaction into a whole different dimension, where the user is surrounded by a complete smart environment with devices/sensors communicating with each other and aggregating their functionalities to provide a set of consolidated services.

The terms Ubiquitous computing and Pervasive computing are used interchangeably [17], but they are conceptually different [16]. Ubiquitous computing uses the advances in Mobile computing and Pervasive computing to present a global computing environment. Mobile computing is about elevating computing services and making them available on mobile devices using the wireless infrastructure. The focus here is to reduce the size of the computing devices so that they can be carried anywhere or by providing access to computing capacity through high-speed networks. But Mobile computing has some limitations. The computing model does not change considerably as we move since the computing devices cannot acquire the context information and adjust accordingly. Pervasive computing, on the other hand, is about acquiring context from the environment and dynamically building computing models dependent on context. Pervasive computing is invisible to human users and yet provides useful computing services. Ubiquitous computing aims to provide Pervasive computing environments to a human user as s/he moves from one location to another.

A Ubiquitous computing environment can be built in two ways. The traditional approach is to achieve it by using Mobile computing and Pervasive computing together, in which

the mobile devices remember the information about past environments they operated in and activate when we reenter into a known environment or proactively build up services as we walk into new environments [16]. We present an alternate approach and use Semantic Web technologies for Pervasive computing environments. This allows context information to be stored on the Web and then shared across Pervasive computing environments via the Web to provide context-aware services.

The paper is organized as follows. Section 2 describes our motivation for this approach. Section 3 focuses on related work. Section 4 describes the proposed solution. Section 5 explains the Integrated Global Computing Framework, referred as IGPF and explains its architecture. Section 6 explains a prototype implementation system over the IGPF and through a scenario from that implementation brings out the effectiveness of the ubiquitous computing. Section 7 concludes the paper.

Motivation

Existing methodologies for implementing a Ubiquitous computing environment use smart devices, which have some processing power and are specialized in a set of specific tasks. Usually the user needs to carry these devices with her/him as s/he moves either within or across pervasive environments. These devices are not readily available and are often difficult to build. The issues that limit fabrication of such personal devices are limitations like battery power for achieving higher computational tasks in conjunction with shape and weight, making practical use of such devices extremely difficult. The advantage of using smart devices is their ability to communicate with each other by building and storing contextual information which is used by the pervasive environment to offer services based on the stored information. In addition, current devices are costly and thus it is difficult to replace all current devices with smart devices to implement pervasive computing environments. Finally, smart devices need to have functionality beyond what they are expected to do because they are integral to the environments.

Our solution eliminates the need for smart devices by using the Semantic Web to build dynamic context models as a user moves from one environment to another. We can

achieve dynamic building of contexts by sharing knowledge and context information between local pervasive environments through the Semantic Web. The vision of Semantic Web proposed by Tim Berners-Lee, James Hendler, Ora Lassila [3] is "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." The Semantic Web can be seen as a huge repository of Web data, like database as a repository of structured data.

However, the challenges associated with the Web data are 1) it is not easy to process by machine since the Web is not structured like a database; 2) it is mainly useful in certain contexts, especially when humans participate, but not in a machine processing manner because there may exist different meanings in different contexts. Thus, the majority of data on the Web is difficult to use on a large scale; 3) Recent advance in XML offers a way to introduce new syntax but does not support semantics (i.e., machine accessible meaning) for being integrated or interoperable with other applications.

The Semantic Web is generally built on Resource Description Framework (RDF) which annotates concepts, properties and their relationships through triple representations and is associated with standard semantics. This enables a machine to ‘understand’ the semantics, processes the data and infers knowledge from it and allows data to be shared and reused across application and community boundaries. In consequence, the Semantic Web standards and tools are widely deployed and recently received growing attention from Pervasive computing community [15].

We believe that the Semantic Web is suited for realizing the Integrated Global Pervasive Computing Framework (IGPF) due to the following reasons: The IGPF applications use a context model generally defined by multiple devices, dynamic services and diverse users. Each entity in the IGPF is identified by a unique URI that is used to share context across different IGPF applications. This ensures interoperability within the IGPF applications without misinterpretation of the information. The Semantic Web standards and tools are suitable to represent the expressiveness of the context-aware knowledge in the IGPF and

specifically support interoperability and reusability of the knowledge across the IGPF applications.

Furthermore, by building a framework running on a powerful, highly available, reliable and static machine we shift the intelligence of context building, storage and decision making to the framework enabling far greater flexibility in implementation. In this approach we can utilize currently available resources, letting the devices do their basic tasks without saddling them with any pre-requisites to participate in Ubiquitous environments. Also we believe that this approach will help us quickly implement Ubiquitous computing since we can use currently available resources and do not need specialized devices.

Related work

Here we present the research being done in the field of Pervasive computing. The three major research projects considered in this section are Project CoBrA by the ebiquity group at UMBC[1], Project Oxygen[2] at MIT and Project Aura [3] at CMU.

The CoBrA project, one of the recent approaches in this field, is an Agent-based approach, where the pervasive computing is achieved using Agents that can model humans, devices and other concepts. The CoBrA project [7][8][9][10] started with the idea of providing an autonomous Agent called broker, which facilitates the interactions between these Agents. The idea is to provide a framework for different Agents to share contextual information and interact according to common understandings regarding these contexts. The work in [7] describes how Pervasive computing can be achieved by using Agent teamwork, and specifically considered the challenges associated in building Agent teamwork. Agent teamwork allows the sharing of context information; where context can be defined as a collection of information that characterizes the situation of a person or a computing entity [13][12]. Central to the CoBrA architecture is the broker Agent that

[1] http://ebiquity.umbc.edu
[2] http://oxygen.lcs.mit.edu/
[3] http://www-2.cs.cmu.edu/~aura/

maintains the shared model of the context for all the computing entities in the space and enforces the privacy policy defined by users and devices [8][9].

This approach provides a better support for knowledge sharing and context reasoning using common ontology expressed in explicit semantics [9]. Furthermore it explores the use of Semantic Web technologies (i.e., languages, logic inferences, and programming tools) for supporting context-aware systems in smart spaces [10]. However, CoBrA's Agent-based approach necessitates major advancements in modeling and implementing Agents. Right now the lack of good modeling techniques put this pervasive computing model at a big disadvantage in its practical and large scale application development and deployment.

Another significant approach in this field is the Oxygen project carried out at MIT. Oxygen aims at providing user interaction through user technologies like speech and vision technologies and provides individualized knowledge access and collaboration technologies, which perform a wide variety of tasks. Oxygen is based on computational devices, called Enviro21s (E21s), embedded in homes, offices, and cars that sense and affect a user's immediate environment. Additionally, it uses Handheld devices, called Handy21s (H21s), which empower users to communicate and compute no matter where they are; dynamic, self-configuring networks (N21s) help user machines locate each other as well as the people, services, and resources that they want to reach. Software that adapts to changes in the environment or in user requirements (O2S) help users do what they want when they want to do it.

This approach depends on the availability of H21s handheld devices which are powered by battery or power outlets. The H21s are powerful machines which combine the functionality of a cell phone, PDA, camera, and television while performing a range of functions and supporting many communication protocols. There is still a far way to go before the H21s devices become widely available. This is the biggest flaw of Oxygen's overall approach.

Another approach is the Aura project at CMU. This approach [14][20] proposes an architectural framework that solves two very hard problems when developing software systems for Pervasive computing. First, it eliminates the problem of allowing a user to preserve continuity in his/her work when moving between different environments and the framework it employs has a key advantage over other approaches in that it allows the system to tailor the user's task to the resources in the environment. Second, it successfully solves the problem of adapting the on-going computation of a particular environment in the presence of dynamic resource variability. As resources come and go, the computations can adapt appropriately. The key ingredients of Aura's architectural framework are explicit representations of user tasks as collections of services, context observation that allows the task to be configured in a way that is appropriate to the environment, and environment management that assists with resource monitoring and adaptation. Each of these capabilities is encapsulated in a component of the architectural framework (the task manager, environment manager, and context observer, respectively). The services needed to support a user's task are carried out by a set of components termed *service suppliers.* Service suppliers typically are implemented as wrappers of more traditional applications and services. Finally, explicit connectors that hide details of distribution and heterogeneity of service suppliers carry out interactions between the parts.

To achieve true Ubiquitous environment, we need to facilitate knowledge sharing between individual Pervasive environments. Unless knowledge is explicitly represented in a standard way and shared between these environments, seamless transition of users between disparate environments cannot be achieved. In the Aura project there is no such standard mechanism. The framework that we present in this paper is based on Semantic Web technologies and sharing of knowledge between Pervasive environments that makes it possible to achieve a truly global Ubiquitous computing.

Proposed Solution

The idea of creating global smart environments doesn't need to expand a smart space to a global level. This is similar to the idea of the global Internet, where numerous small

physical networks connect together to form a large homogenous network [21]. Similarly a better approach here is to connect these smart environments together ‘Semantically’ so that they conglomerate into a virtual global smart environment similar to the Internet, creating a Ubiquitous environment.

In daily life an average human being is associated with many different environments and has different roles in each. For example, in Figure 1, Dr. Smith is associated with environments like hospital, bank, and pub and plays the role of doctor, client, and customer, respectively. A Ubiquitous system should truly reflect this real life scenario.

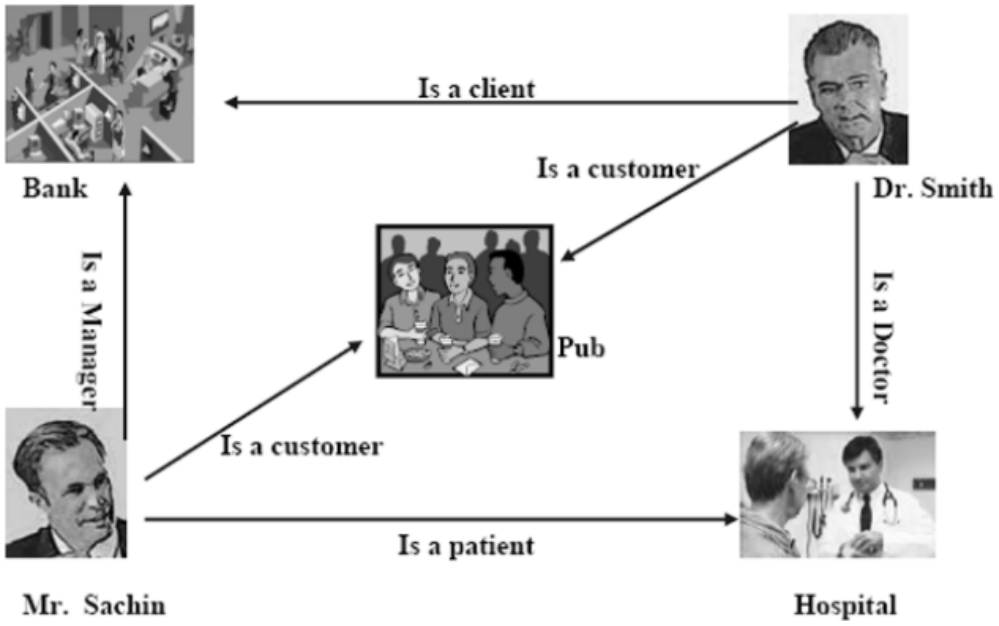

Figure 1. User Role Model in Ubiquitous Environment

Each of these environments may be an independent smart system since they belong to different domains/organizations they are disparate. What’s important is that as the user moves from one environment to another, the system transparently recognizes the user and associates the user’s specific roles with the domain the user enters. The system passes information required by the user (files/contact information etc) between domains, accessing/updating the user profile as needed, etc. which allows the user to receive

appropriate services. This true sense of Ubiquitousness can be achieved only if the two environments are semantically integrated.

Context-awareness is an integral part of Ubiquitous computing. The concept of context has often been interwoven and used in many different fields. When the information has to be conveyed from one environment to another we need to let the receiving environment know the reference of our discussion. Dey and Abowd [13] defined context as a piece of information that can be used to characterize the situation of a participant in an interaction. Similarly, Brown et al., [5][6] defined context as location, environment and/or identity of people and time. By sensing context information, context enabled applications can present context information to users, or modify their behavior according to changes in the environment. As a user moves from one environment to another the new domain should assess the context and present services to the users based on the context factors. In our case, the context model is comprised of the location of the user, the role s/he plays in the domain, time of the day and the current state of the environment. Current state of the environment means the services that are currently available and services that are not. The philosophy behind this is that the Ubiquitous system should provide the best possible service or composition of services which are currently on hand based on what role the user is playing, including his/her privileges and restrictions, his/her location and the current time of day.

The global pervasive framework splits the ‘world’ into smaller domains called Autonomous Systems (AS). Each AS represents either a smart environment or a part within that environment. Each AS is managed by the AS head, which is a resource-rich, powerful, highly reliable and stationary machine hosting the Integrated Global Pervasive Framework (IGPF) described in detail in Section 5. The IGPF is a generic framework over which domain specific components are built to provide a local pervasive environment. An inherent functionality of the IGPF is the ability to interact with other ASes sharing the same context information, passing user information as needed and working together creates a Ubiquitous environment.

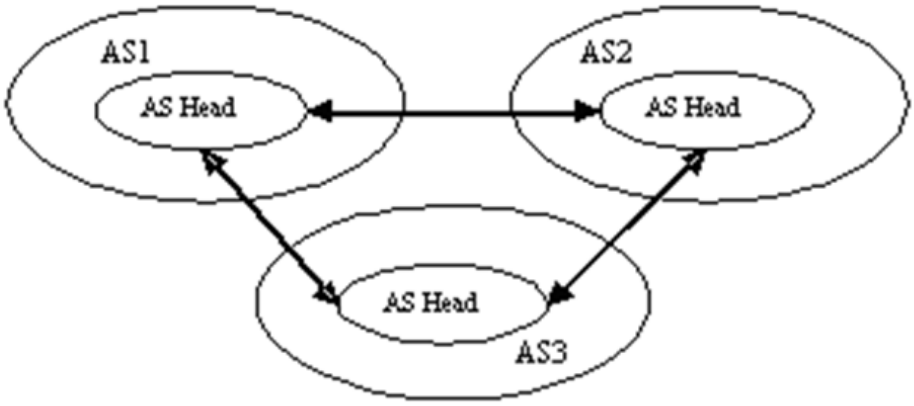

Figure 2. Autonomous Systems

The whole Ubiquitous computing environment is a peer-to-peer (between ASes) and hierarchical structure as shown in Figure 2. The AS layer is formed by the AS's heads which control whole pervasive computing environments such as the office, home or pub.

One of the central points of our approach is that all concepts, tangible or abstract, are represented by Unique Resource Identifiers (URI) [1], similarly to how concepts are represented in Semantic Web [3][2]. Each AS has its own URI; the ASes recognize, communicate, coordinate and also base their trust information about each other using the URI. Each device within an AS also has a Unique URI. Let's say the URI for an AS is [*igpf://umkc.edu/AS1*], the URI of a device within this AS is [*igpf://umkc.edu/AS1/D1*]. This device could be of any one of the types (and subtypes) of a Device type, say a Digital Camera. Each device Type will also have its own URI. For example a Device Type, say a Camera, will have its own unique URI; a specific subtype of camera, say a Digital camera, will also have its own unique URI [*igpf://device/samsung/camera/digitalcamera*]. Providing unique identities to concepts and physical entities creates the ability to represent devices and environments which are unrestricted to a physical domain, making it possible for ASes to interact unambiguously.

The framework mentioned is based on the Semantic Web; each concept URI points to its rich semantic description on the Semantic Web. This description includes device type (say a camera), functionalities (it captures images) and capabilities (it can capture

pictures at a resolution of X by Y), input and output formats of a device. One of related on-going efforts includes the Composite Capability/Preference Profiles[4] project (CC/PP) sponsored by W3C. In addition each of these defining metrics is also semantically expressed. This information can be processed by machine(s) capable of inferring what devices to use to achieve the desired functionality. In addition the AS is also semantically described, which helps other ASes to determine the type of AS (office, pub or hospital), its functionalities, and privacy and trust information.

The user is the most important entity in Ubiquitous computing. In our approach each user has a Ubiquitous User Identity and Profile. User profiles will represent user specific information like contact information, schedules, appointments, preferences, financial and medical information and the profiles determine the Ubiquitous User Identity. In addition, the profile will also model user likes/dislikes or interests, so that the system can proactively fetch or disseminate information on behalf of user. Each user profile will be hosted on a central location and will be addressed by a Unique Resource Locator (URL) [4] for each user. As soon as a user gets inside an AS, the associated user device (Cell Phone, PDA etc) will provide the user URL to the AS head. The AS head can then get the user profile and retrieve the user profile, extract the required information, construct the appropriate context models and offer the user services based on the constructed context model. The AS can also "learn" about user's preferences based on his/her usage patterns and update his/her profile accordingly.

Using Semantic Web as part of the solution introduces some open issues as listed below. The major issue of using Semantic Web in Ubiquitous computing is privacy and trust. Since we assume that user information can be made accessible over the Web, we need to provide security and privacy to the user information. This needs to setup a mechanism for protecting user information from unauthorized access. Also, it is imperative to trust that the information given and the source of information is correct.

---

[4] http://www.w3.org/Mobile/CCPP/

We are using URIs to identify any entity in the environment. There is no single authoritative body, which would assign and manage URI distribution to ensure that URIs are unique.
There is not much ontology present on the Web today and even those, which are available, don't have concrete mappings between them.
However, these problems are of the Semantic Web in general and not specific to our framework. We believe that as research progresses in the Semantic Web area, solutions would emerge which can be incorporated in our framework.

Architecture of IGPF

Following points were considered while designing Integrated Global Pervasive Computing framework (IGPF).

IGPF should be generic. That means we should be able to use IGPF to setup a pervasive computing environment for any domain. For example, the IGPF can be used to setup a pervasive computing environment for a hospital or for an office with little modification. The framework is generically designed and can be extended by domain specific components. The generic components are the core components of the framework and are present in every domain. For example, one of the core components, the KBHandler is used to query and inference on the knowledge base of the system. The domain specific components provide knowledge and services specific to the domain. If the IGPF is used to setup a pervasive computing environment in a hospital then the knowledge base will consist of domain specific ontologies like the drug ontology and patient ontology; examples of the services would be drug administration and patient care service. These domain ontologies would help to model the knowledge about the environment and domain while the drug administration service would handle the scheduling of nurses to administrate drug to the patient. One can add more domain ontologies to enhance the knowledge base and add more domain services to enhance the environmental functionality.

IGPF should be interoperable**.** The IGPF needs to incorporate exiting solutions used in the organization and interoperate with them. We achieve this by using service oriented design for IGPF. Existing solutions within an organization can be integrated with the IGPF by writing simple wrapper services around them. For example, a simple wrapper can be written around an existing E-Mail server to integrate it with the IGPF. This would provide enhanced functionality to the pervasive environment to communicate with a user through email, in addition to other existing communication channels.

IGPF should be dynamic. Existing functionality of the pervasive environment can be enhanced by adding new services. It should be easy to add/modify these services without bringing down the running system. We adopt event based architecture which provides more flexibility to incorporate these services at run time**.** The services are loosely coupled; a service doesn't need to know which service is providing a particular functionality for a given event. Thus, addition of new services at run time has no impact on the existing running system.

IGPF should use shared knowledge. We use the Semantic Web based approach where the shared knowledge is stored on the Web and it can be accessed by different ASes. Currently we have defined user profile ontology stored on the Web and shared by IGPF environments to get information about the user.

Figure 3 depicts the architecture of IGPF, Integrated Global Pervasive Computing framework.

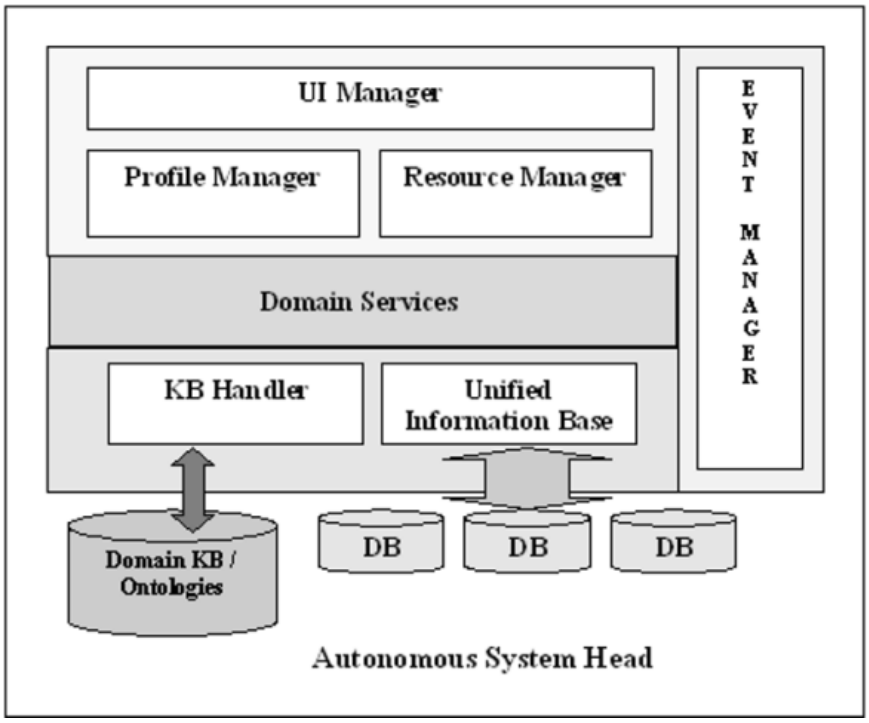

Figure 3. IGPF Architecture

Event Manager: Event Manager manages event based communication between components of the framework. Each component in the IGPF framework communicates with each other by throwing and listening events through the Event Manager. The Event Manager ensures that all services registered for an event receive that event and don't receive any duplicate events.

KBHandler: The KBHandler maintains the local Knowledge base. The KBHandler maintains a 'Knowledge Model' which reflects the current context. This context is acquired by listening to events and converting them to knowledge facts which are added or removed from the Knowledge base (KB). The Knowledge Model is explained in Section 6, when we explained the working of the system within a small scenario. We have implemented persistent KB in our framework so that we can recover the KB in case of system failure.

Resource Manager: The Resource Manager manages the resources available in the environment. It maintains the status of each resource and also takes care of scheduling

resources for different activities. In IGPF, a resource is anything that can be scheduled including the human actors and/or equipments. Each resource has a semantically annotated schedule, which describes when a particular resource is available. When there is a request for a certain resource, the system will look for its availability and then schedule that resource. An important feature is that requests can be made for roles rather than for persons. For example, requests could be for a 'Cardiologist' or a 'Sales Officer'. Since the systems' Knowledge base specifies the role of each person and its current schedule, the system finds the match, schedules the requested event and marks its schedule with this new booking. The user is then notified about the new task assigned to him/her through the user interface.

Profile Manager: The Profile Manager is responsible for fetching user profile information from the Web. We assume that the user profiles have a single access point where one can request for user profiles or certain portion of the user profile. The IGPF communicates this server to get the user profile or extract specific information about the user from his/her profile.

User Interface (UI) Manager: The framework is designed to provide a dynamic user interface, that is, the interface can be changed depending on the user role and the requirement of the service as well as the device used to communicate with the user. For example, the system generates different UI screens corresponding to desktop or PDA, or sends SMS over a cell phone, depending on what device the user is currently using. Thus, the framework provides pervasiveness in the sense that it uses an appropriate device to communicate with the user depending on the context.

Domain Services: The Domain Services provide functionalities specific to a given domain. These services use the framework to communicate with each other, and the external environment and also to access knowledge base of the system.

Unified Information Base: The Unified Information Base (UIB) integrates information from disparate data sources present in the environment and presents it at a more

conceptual level by linking it to a local domain ontology. For example, a database field 'BP' in a hospital setting can be linked to the concept "BloodPressure" of a standard medical ontology like the UMLS5. This creates an abstraction of a single homogenous data source for other services, which need to refer to the data in terms of domain concepts. The UIB thus allows linkage to a real data element and fetch or update of the same. The idea of a unified information base is an extension to our previous work [11].

Prototype Implementation

We have implemented the IGPF on J2SE[6] running on the RedHat[7] 9.0 Linux platform. The event based, service oriented architecture is implemented using Jini[8] and Javaspaces[9] Technology. The Profile Server is hosted as a Web Services[10] on a TOMCAT[11] application server running on a RedHat 9.0 Linux platform. The ontologies are encoded in standard RDF[12] and Owl[13] language and designed using Protégé ontology editor[14]. The KBHandler uses HP Jena[15] technology to query and inference on these ontologies. The KB implemented in our framework is persistent so that we can recover the KB in case of system failure.

To exemplify how IGPF would work in real world, we present a small scenario that we implemented over the IGPF as a proof of concept. We derived this scenario from our current implementation of the "ICareNet" system [19] on the IGPF. The ICareNet system is an emergency handling system which aims to weave different actors within the healthcare setup through a pervasive wireless environment, with context-aware intelligent

---

[5] http://www.nlm.nih.gov/research/umls/
[6] http://java.sun.com/j2se/index.jsp
[7] http://www.redhat.com/
[8] http://wwws.sun.com/software/jini/specs/index.html
[9] http://wwws.sun.com/software/jini/specs/index.html
[10] http://www.w3.org/2002/ws/
[11] http://jakarta.apache.org/tomcat/
[12] http://www.w3.org/rdf
[13] http://www.w3c.org/TR/owl-features/
[14] http://protege.stanford.edu/
[15] http://jena.sourceforge.net/

services available on mobile devices. The objective is to provide a communication model that will facilitate rather than intrude in the functioning of the healthcare providers.

This scenario implements a "Chest Pain" Scenario derived from the AHA/ACC guideline16 for chest pain triage, where a patient admitted to the hospital with an acute emergency chest pain goes through the typical clinical workflow and interfaces with different actors within the workflow. Since, the focus in this paper is to connect different pervasive systems together to form a truly global ubiquitous environment; we concentrate on the initial part of the scenario, the patient's arrival at the hospital. As described in previous sections, the key to providing a truly global ubiquitous environment, is to move the profile of a person as s/he moves from one pervasive environment to another. Each of these environments is providing services to the user based on the role that s/he plays in that environment. Figure 4 displays the graphical representation of the personal profile of the patient, which has many different broad level aspects like "FinancialProfile", "DemographicProfile", "MedicalProfile" etc.

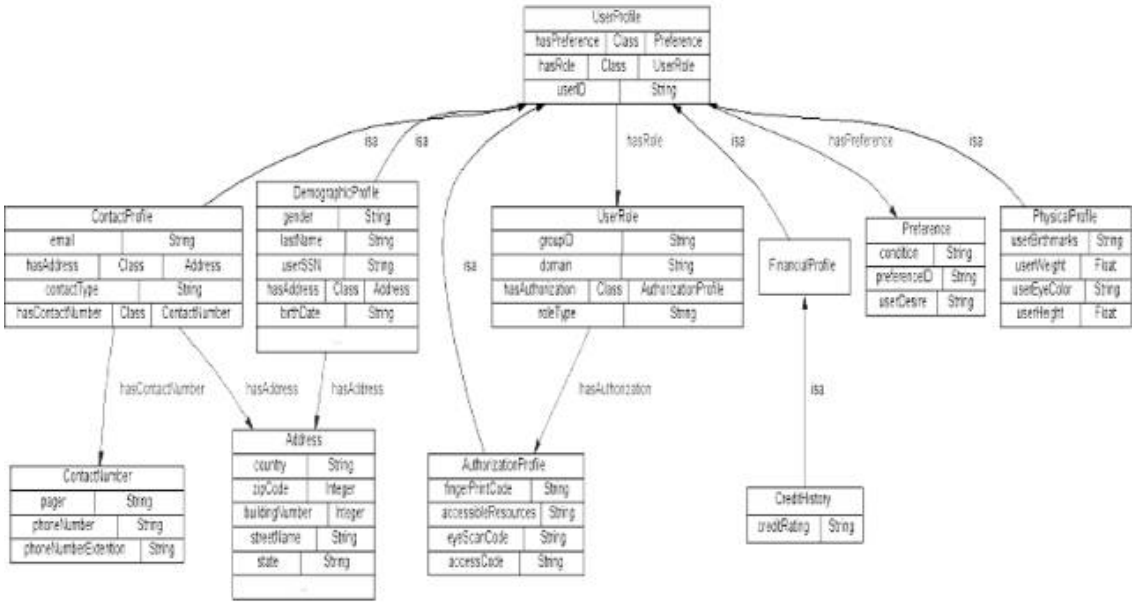

Figure 4. Snippet of Personal Profile Schema

[16] http://www.acc.org

When a patient is brought in with an acute chest pain, it is imperative that as much medical information about the patient is presented to the doctor for accurate diagnosis and subsequent treatment. In emergency cases, a patient may be unconscious or unable to communicate his/her information; in other cases an average patient may be unaware of the nuances of his/her own medical history, also the information that s/he has may not be medically accurate. The solution to this problem is to have his/her medical history as part of the patient's personal profile. Figure 5 shows a snippet of the patient's profile as a subset of the overall profile encoded in the RDF notation.

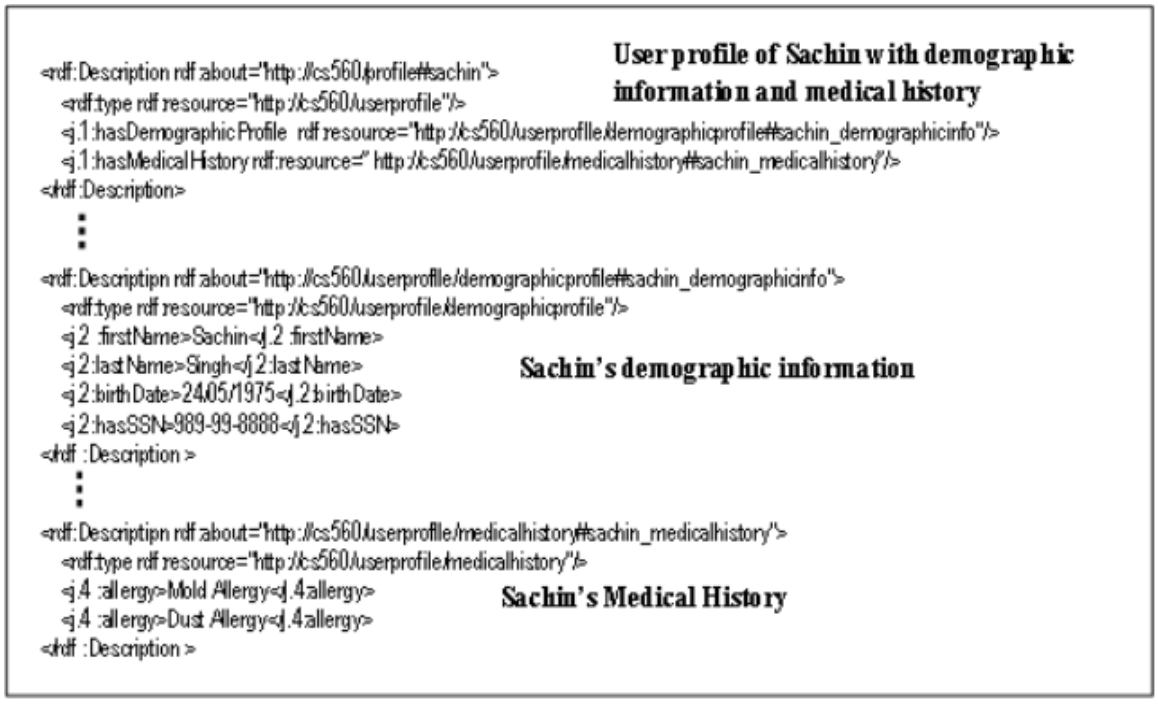

Figure 5. Profile Detail - RDF Notation

When a person is brought into an Autonomous System (AS), for example a Hospital AS, the AS is able to get the person's URI like 'http://cs560/userprofile#sachin'. In ICareNet we have used Bluetooth based mobile devices to pass this information. When a new Bluetooth device is discovered by a Bluetooth sensor a set of device information is passed to the discovering sensor. In our implementation the user URI is passed along with the device information. Once the URI is obtained the profile Web Services is invoked and the relevant part of the user profile is downloaded to the local Knowledge

Base (KB). Being a hospital, this AS has access only to the medical history part of the profile. The AS system by default detects the user in the "Patient" role, and once the system is triggered for a Chest Pain scenario, it then accesses the local KB for relevant information regarding the case. The scenario is triggered when the receptionist inputs details about the patient and specifies that the patient is suffering from chest pain. The rules specified by the chest pain service says that if chest pain is a new complaint of the patient, then schedule a nurse to take initial inputs from the patient. The sample Jena rules are given as shown in Figure 6. The details of the Jena syntax can be found at the Jena ToolKit site[17].

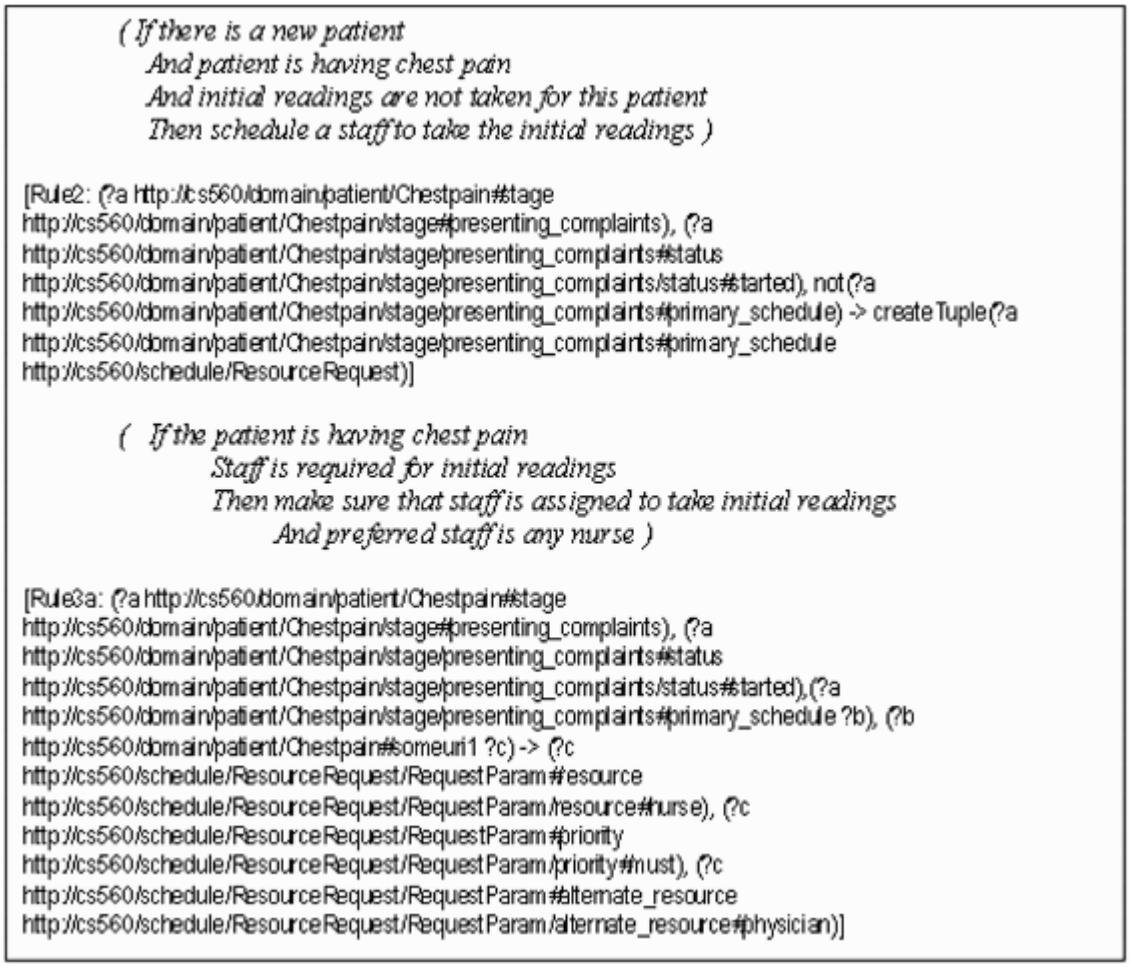

Figure 6. Sample Jena Rules

[17] http://jena.sourceforge.net

These rules add new facts in the knowledge base saying that a nurse is required to take initial input from the patient. The KBHandler maintains an internal knowledge model which would appear as depicted in Figure 7.

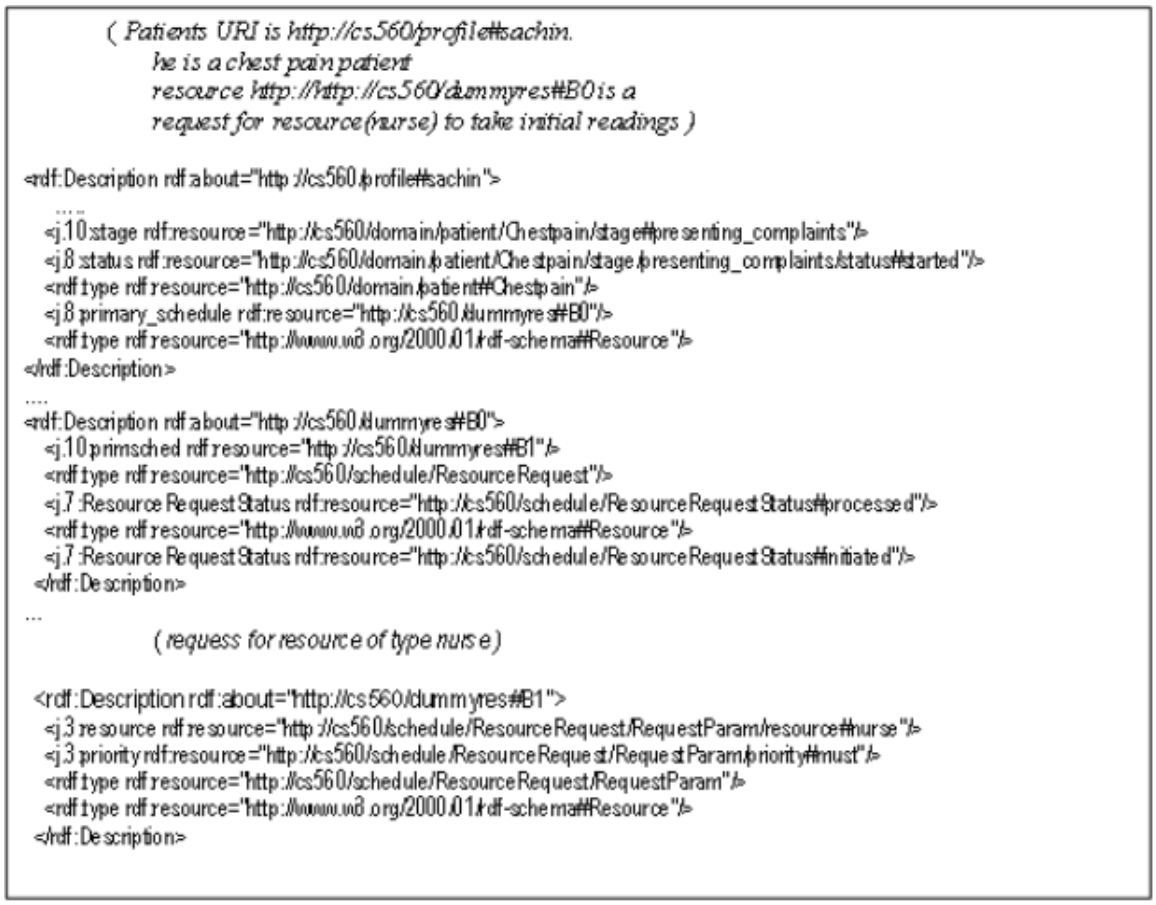

Figure 7. Internal Knowledge Model – Resource Request

The Chest Pain service reads information from the knowledge base and generates an event to request a nurse for this patient. The Resource Manager captures this event and schedules a nurse depending on the nurse's availability and patient's location. The Resource Manager then updates knowledge base with related facts.

(Response of the resource request)

```
<rdf:Description rdf:about="http://cs560/dummyres#X0">
   <j.11:responseto rdf:resource="http://cs560/dummyres#B0"/>
   <rdf:type rdf:resource="http://cs560/schedule#ResourceResponse"/>
   <j.11:responseparameter rdf:resource="http://cs560/dummyres#X1"/>
   <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
 </rdf:Description>
```

(Nurse Tina is allocated for this request)

```
<rdf:Description rdf:about="http://cs560/dummyres#X1">
   <j.12:requestParam rdf:resource="http://cs560/dummyres#B1"/>
   <rdf:type rdf:resource="http://cs560/schedule/ResourceResponse#ResponseParam"/>
   <j.12:allocatedResource rdf:resource="http://cs560/ws04/Resources/Nurse#Nurse_Tina"/>
   <j.12:status rdf:resource="http://cs560/schedule/ResourceResponse/ResponseParam/status#given"/>
   <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
 </rdf:Description>
```

Figure 8. Resource Allocation

When the nurse is assigned for the patient, the Chest Pain service generates a message for the nurse, informing her about the new patient. It is important to know that during the process of diagnosis, medical personnel need different medical information about the patient, like past major surgical procedures, whether the patient is diabetic, is taking any drugs, has allergies etc. Thus depending on different contexts, the system would extract different information from the patient's profile. As mentioned in the domain rules, in the current context the nurse needs to be presented with the general allergies for the patient. It retrieves patient's allergy information from the knowledge base and passes it to the nurse so it is relayed to the physician later in the visit process. The RDQL[18] used to extract information from the knowledge base are as shown in Figure 9.

[18] http://jena.sourceforge.net/tutorial/RDQL/index.html

(Extract the URI of the nurse allocated for the patient
This query http://cs560/ws04/Resources/Nurse#Nurse_Tina, which is an URI of a nurse)

```
select  ?d where (<patient URI>, <http://cs560//domain/Chestpain/presenting_complaints#primary_schedule>,
?a),
(?a, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, <http://cs560/schedule#ResourceRequest>), (?b,
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, <http://cs560/schedule#ResourceResponse),
(?b,<http://cs560/schedule/ResourceResponse#responsero,?a),(?b,<http://cs560/schedule/ResourceResponse#re
sponseparameter>,?c), (?c, <http://cs560/schedule/ResourceResponse/ResponseParam#allocatedResource>, ?d)
```

(Extract patient's allergy information
This query returns Mold Allergy and Dust Allergy)

```
select ?c (<patient URI>, <http://cs560/userprofile#hasDemographicProfile>, ?b),(?b,
<http://cs560/userprofile/demographicprofile#allergy>,?c)
```

Figure 9. Example of RDQL Queries

When the nurse accesses the patient's URI, patient information such as allergies and previous patient history is readily available. The User Interface (UI) Manager then generates a dynamic user interface for the nurse depending on whether she is carrying a PDA or using a desktop computer. Figure 10 shows the Web interface of ICareNet generated for the nurse.

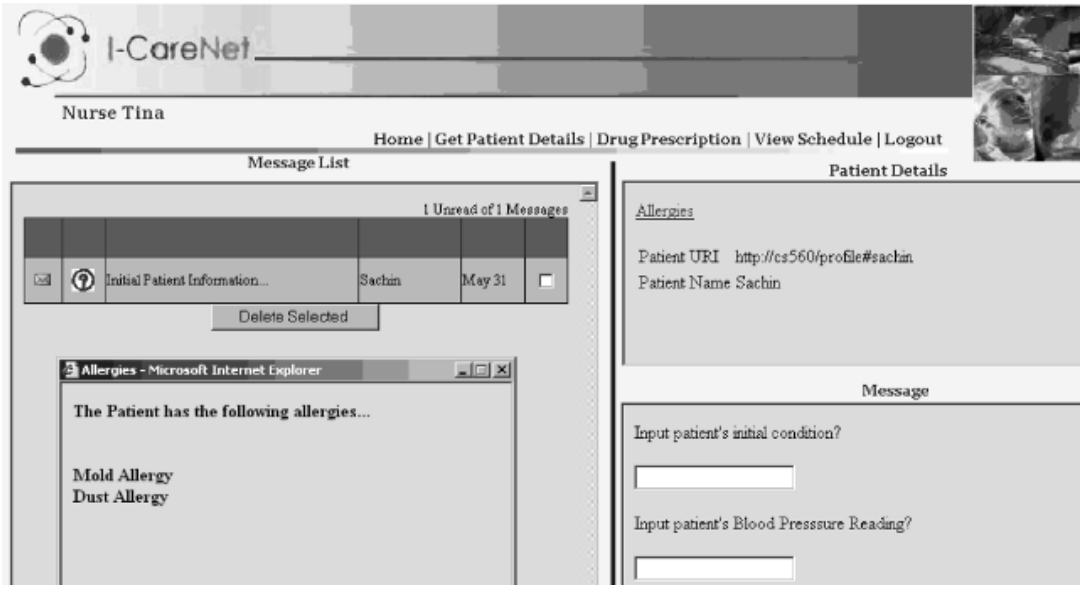

Figure 10. Interface for Nurse Showing Allergy Information from Patient Profile

Figure 11 shows the screen the patient's doctor would access during the advanced phase of diagnosis. Note that the patient medical information the doctor is presented is based on its current relevance to the patient's symptoms.

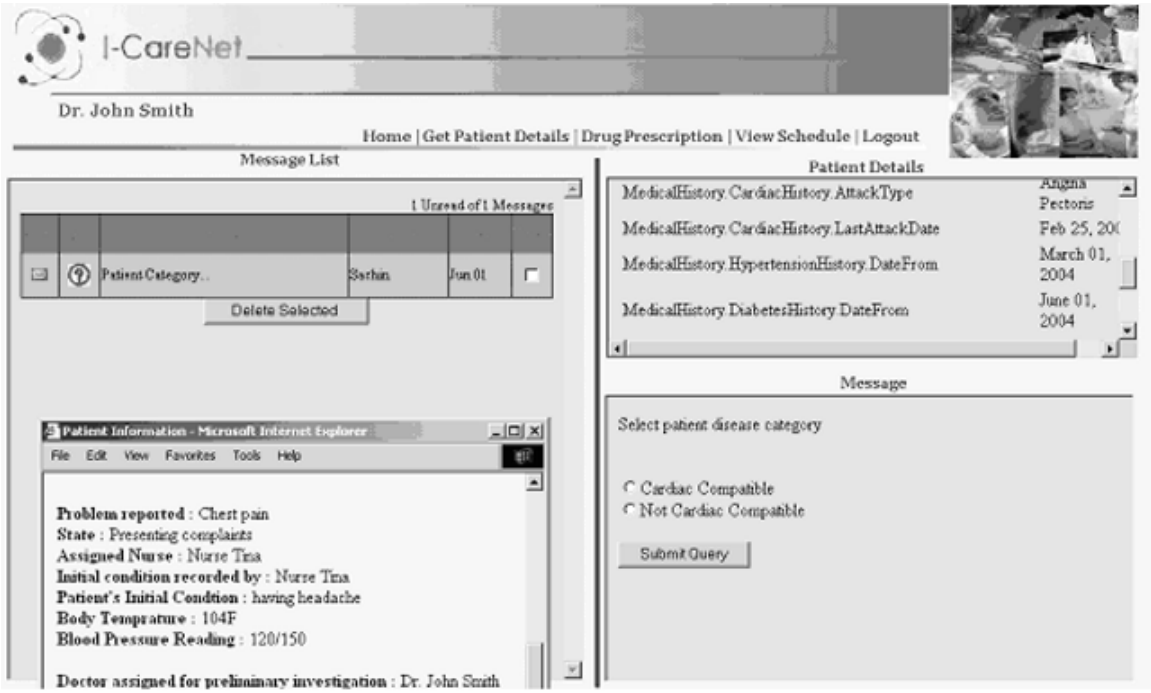

Figure 11. Interface for Doctor Showing Patient Status from the KB

An important point worth mentioning is that the profile server doesn't necessarily have all the information locally, esp., medical information. It has pointers to the information stores residing in different hospitals. Thus when an AS needs some information, it contacts other AS over the Semantic Web to query for the related information. These AS systems would form a peer-peer network and each AS would have its own markup specifying its "type" and its "capabilities" using the OWL-S markup language[19]. Based on these descriptions the AS would discover another peer AS that matches its requirements and then request it for required information or service as described in the AS. The details of this AS-to-AS discovery and communication are beyond the scope of discussion of this paper. The local AS employs the Unified Information Base (UIB) to

[19] , http://www.w3.org/2004/OWL/

aggregate the disparate information and present it in a unified manner. There are issues of privacy and trust when dealing with personal profile access which is a direction for our future work.

We highlight the following points illustrated by this scenario:

Role based services: User plays a specific roles within an AS. Here, the person brought in is playing the 'Patient' role while the person treating him/her is playing the 'Doctor' or 'Nurse' role within the same AS. The AS is providing different services based on roles defined in the profile.

Global Profile: This is the most crucial element of the idea presented here. As a person moves in from one AS to another and his/her role changes, different parts of his/her profile are accessed to get the corresponding information. Thus as a user moves from one AS to another, s/he is 'recognized' in a specific role, and presented with a set of services. The key is that all the ASes share the same profile ontology over the Semantic Web and hence the same knowledge which makes it possible to share and transfer information based on this shared understanding.

Conclusion

The approach presented in this paper introduces a whole new way of implementing Ubiquitous computing by semantically connecting different and diverse pervasive environments. Connecting these environments through the Semantic Web enables sharing of knowledge and information across pervasive environments, thus allowing environments to dynamically create context models for a user as s/he walks from one environment to another. The impact of the designed framework is that it creates the generic base for achieving these basic functionalities along with offering the user a set of abstract services. Furthermore, the environments can be 'programmed' to achieve specialized applications that use the services within a pervasive environment and also share and utilize the services offered through access to other environments. We believe that the impact of such a framework will speed up the process of creating local pervasive

environments and build true Ubiquitous environments based on them. As a whole, our work aims to use computing resources to create Ubiquitous environments to improve the way we utilize services that pervasive computing environments provide and in general, benefit our daily lives.

References

[1] Berners-Lee, T. (1994). *Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web.* Retrieved December 3, 2004, from Junehttp://www.w3.org/Addressing/rfc1630.txt

[2] Berners-Lee, T. (2003). *Primer:Getting into RDF & Semantic Web using N3.* Retrieved December 3, 2004, from http://www.w3.org/2000/10/swap/Primer

[3] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.

[4] Berners-Lee, T., Masinter, L., & McCahill, M. (1994). Uniform Resource Locators (URL). (RFC 1738) CERN, Xerox Corporation,University of Minnesota.

[5] Brown, P. J. (1996). The Stick-e Document: a Framework for Creating Context-Aware Applications. *Electronic Publishing 96*, pp. 259-272.

[6] Brown, P.J., Bovey, J.D., Chen, X. (1997). Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications, 4(5),* pp. 58-64.

[7] Chen, H. (2002). An Intelligent Broker Architecture for Context-Aware Systems (*Doctoral Dissertation,University of Maryland at Baltimore).*

[8] Chen, H *et al*. (2002). Beyond Distributed AI, Agent Teamwork in Ubiquitous Computing. *Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, AAMAS-2002.*

[9] Chen, H *et al*. (2003). An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review.*

[10] Chen, H *et al*. (2003). Semantic Web in a Pervasive Context-Aware Architecture. *Artificial Intelligence in Mobile System,* October 2003.

[11] Chong, Q., Marwadi, A., Supekar, K., & Lee, Y. (2003). Ontology Based Metadata Management in Medical Domains. *Journal of Research and Practice in Information Technology* (JRPIT), 35(2), pp. 139 - 154.

[12] Dey, A.K. (2000). Providing Architectural Support for building Context-Aware applications (Doctoral Dissertation, Georgia Institute of Technology)

[13] Dey, A.K., Abowd, G.D. (1999). Towards a better understanding of Context and Context-Awareness. *GVU Technical Report GITGVU-99-22, College of Computing, Georgia Institute of Technolog.y*

[14] Garlan, D., Siewiorek, D., Smailagic, A., & Steenkiste, P. (2002). Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*

[15] Lassila, O., Adler, M. (2003). Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web. *Spinning the Semantic Web: Bring the World Wide Web to Its Full Potential, MIT Press*, pp. 363-376.

[16] Lyytinen, K., Yoo, Y. (2002). Issues and Challenges in Ubiquitous Computing. *Communications of the ACM, Vol. 45, No.12*.

[17] Saha, D., Mukherjee, A. (2003). Pervasive Computing: A Paradigm for the 21st Century. *IEEE Computer, vol. 36,* pp. 25-31.

[18] Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications, vol. 8,* pp. 10-17.

[19] Singh, S., Puradkar, S., & Lee, Y. (2004). *I-CareNet: Facilitating Communication and Information Flow in Medical Domain.* University of Missouri – Kansas City, Technical Report. Retrieved December 3, 2004, from www.sce.umkc.edu/~sbs7vc/icarenet

[20] Sousa, J.P., Garlan, D. (2002). Software Architecture: System Design, Development, and Maintenance. *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture,* pp. 29-43.

[21] Tyson, J. (2004). How Internet Infrastructure Works. Retrieved December 3, 2004 from http://computer.howstuffworks.com/internet-infrastructure.htm

[22] Weiser, M. (2003). The Computer for the 21st Century. *Scientific American.*, pp 94-104; *reprinted in IEEE Pervasive Computing,* pp. 19-25