Ontology-Based Resource Discovery in Pervasive Collaborative Environments

Kimberly Garcia, Manuele Kirsch Pinheiro, Sonia Mendoza, Dominique Decouchant

Ontology-Based Resource Discovery in Pervasive Collaborative Environments

Kimberly Garcia*, Manuele Kirsch-Pinheiro†, Sonia Mendoza*, Dominique Decouchant‡
*Department of Computer Science, CINVESTAV-IPN, México DF., México
kimberly@computacion.cs.cinvestav.mx, smendoza@cs.cinvestav.mx
†Centre de Recherche en Informatique Université Paris1-Panthéon Sorbonne, Paris, France
Manuele.Kirsch-Pinheiro@univ-paris1.fr
‡(1) C.N.R.S. Laboratoire St Martin d'Heres, France
(2) Department of Information Technologies UAM-Cuajimalpa, Mexico DF., Mexico
decouchant@correo.cua.uam.mx

***Abstract*—Most of the business, academic and even public environments (e.g., airports and malls) offer multiple services (e.g., flight and hotel reservation), hardware (e.g., printers, scanners and cameras) and software. However, it could be particularly difficult to take advantages of all these resources without a proper software support capable of finding resources that fulfill user's requirements. A user can be close to the resource he needs, but in a crowded and busy environment, it is highly probable that he would never find that resource without the help of a tool that discovers and locates it. To try to overcome this problem, multiple service discovery protocols (SDP) have been developed. Their main goal is to promote the use of network resources by offering some basic finding operations and by reducing configuration tasks. Unfortunately, most of these protocols are focused on providing support for software clients. They do not offer any suitable tools for describing resources for both publication and search. Consequently, users may waste their time making several queries with different parameters. In this paper, we tackle this issue by providing users means for semantically describing resources and their needs effortlessly. We propose an ontological approach to manage resources description. This proposal grants several advantages, such as organized storage of information and semantic-based knowledge retrieval through the use of reasoning tools. Finally, we also present a matchmaker proposal intended to find the best resource available for a user's request.**

I. Introduction

The ubiquitous computing [11] main objective is to incorporate technology into the users environment by making it so easy to use that it would become invisible to users. They will be able to focus on their goals instead of thinking on the tools the environment offers. This ubiquitous computing world, as dreamt by Weiser [11], is now becoming a reality with the increasing on the amount of devices and information available everywhere. People is more and more habituated to receive information according to their location and preferences. Not only multiple computer-based devices are getting more popular, but it is also more and more common to have enabled environments for providing users the ubiquitous experience. Our work wants to go one step further by adapting such ubiquitous environment to support collaborative work.

In collaborative ubiquitous environments, sharing resources (e.g., devices or files) among people in a controlled way (e.g., through the usage of restrictions and access rights) is a necessity. In this case, the correct management of resource information can make the difference between providing a good response when requesting a resource or not. We argue that creating a proper descriptive model of the shared pervasive environment, which involves describing resources and the environment in which they are shared, constitutes the first step to cope with the previously stated limitations. Service discovery protocols [14] can be used to try to solve the problems described above, but unfortunately most of them are not oriented to work with human clients. They work with software clients, so their support for describing resources are neither as flexible nor as rich as a human client would need. Besides, some protocols use a description approach based on static templates that requires users to specify particular service characteristics. These approaches restrict the information provided by the service owners and, when a user needs to find a service, he cannot specify more information than the one required by the template. Other protocols use a more flexible approach based on XML templates to describe services. Although this approach allows the addition/deletion of information to/from a service description, they are semantically poor.

It results that, without a proper support to describe resources, a system dedicated to manage resources sharing could not succeed because of the lack of suited information. This lack makes difficult the search for a resource that actually satisfies the users requirements. In this paper, we propose a suited description support for heterogeneous resources that can be shared among several collaborators. Being distributed among several buildings of an organization, these resources are governed by different usage restrictions (e.g., a schedule has to be followed or a limited amount of work that can be completed at once). We propose a semantic description that tackle this question through an ontological approach. Indeed, ontologies are particularly suited for knowledge sharing and for efficient reasoning. Thanks to ontologies, we are able to describe concepts representing resources (physical and virtual resources human users own) and properties representing relationships among these concepts. These relationships provide meaning to each concept involved in the whole domain we are modeling. Ontologies are also a central part of the matchmaking process we propose, which consist in determining the best available resource for a specific request. This process is not a trivial task since it must consider not only static resource information, but also dynamic information involving changes in the

environment. By using ontologies for describing resources, we can perform inference and reasoning processes to discover available resources that fulfill a request and consider dynamic status of those resources.

This paper is organized as follows. In section II we present the related work. Following, section III illustrates the multiple scenarios in which an architecture for resource management can be deployed. Here, we present a set of environments and we introduce our case of study to validate the proposed RAMS (Resource Availability Management Service) architecture. This architecture is describe in section IV. Then, section V details the set of ontologies proposed for describing shared resources. These ontologies are the foundations of the matchmaking algorithm presented in section VI. After, in section VII, we illustrate in a real scenario all the RAMS architecture components working together to provide resource sharing support. Finally, in section VIII we present our conclusions and the future path of our research.

II. Related Work

In this section we analyze and compare some outstanding works involving resource discovery. Since several works proposing discovery protocols focus on service discovery, we present in section II-A, some traditional Service Discovery Protocols (SDP). Then, a comparison of features of these protocols is made in section II-B. Finally, in section II-C a couple of frameworks that have been recently developed to provide context awareness capabilities to the service discovery task are discussed.

A. Service Discovery Protocols

One of the pioneers of service discovery is the Service Location Protocol (SLP) [13], which was developed by IETF in order to prevent users (which take the form of applications) from having to know the specific network location of the required services. SLP is composed of three agents: user, service and directory. A user agent (client) denotes a software entity that looks for a service. A service agent (server) is a process dedicated to announce a service, which consist of the location of a device driver. Finally, a directory agent denotes a software entity acting as a centralized repository of information about the services that are published. When a user agent needs a service, he makes a request specifying the characteristics he needs. In response, the user agent receives a URL, which specifies the location of the service that fulfills his requirements. This URL has to be used by the user agent to contact the service. In small networks, a user agent can make requests directly to service agents through multicast messages. The service agents allow to satisfy the request by sending back to the user agent unicast messages containing the service location. In bigger networks, a directory agent is used as a cache in which service agents are registered in order to announce their services. Announcements are refreshed periodically to confirm that the services still available. The user agent sends a unicast request to a directory agent rather than a multicast request to several service agents. In this way, SLP avoids network overloading resulting from one user agent request.

Similar to SLP, UPnP [9] was proposed by the UPnP Forum in order to free users from the configuration task when a new hardware device is plugged into a computer for the first time. UPnP allows hardware devices to automatically find the services they need and to perform the required configuration operations. In this way, final users (i.e., a human) can employ hardware devices effortless. The UPnP architecture handles two types of devices: 1) controlled devices and 2) control points. A controlled device acts as a server, which responses to requests coming from several control points. In a complementary way, a control point acts as client, which looks for information about the server capabilities. When connected to the network, a controlled device publishes its services so that the network control points can know it. Similarly, UPnP allows a control point connected to the network to look for the controlled devices of its interest and to subscribe to them. In this way, the control point can get a specific device information from the URL provided by the device within the discovery message. Once the information about a device and its services has been obtained, the control point can ask these services to execute some commands and to send the results back to it. Moreover, the control point can listen to state changes of some devices. Finally, if the required devices have a URL to inform about the actions performed, the control point can display them on a browser, so that the user can control such devices or visualize their status.

Ninja Service Discovery Service (SDS) [2] is one of the academic proposals to service discovery. It is integrated by three types of entities: services, clients and SDS servers. A service is any application software with a well known interface capable of performing any task, from a small mathematical computing process to a task able to control a hardware device. Clients are entities that want to discover services executed in the network. Finally, SDS servers are secure, fault tolerant and scalable data repositories that hold information about shared services.

Ninja SDS uses predefined templates designed for each type of service. These templates are XML documents that include the capabilities and the Java RMI address of the described services. When an announcement of a new service is received by a SDS server, it uses hash functions over subsets of parameters from the service descriptions in order to define a bit vector. This vector acts as a bloom-filter that condenses the description of the service. Once the vector is created, it is sent to its SDS servers neighbors, which become aware of the existence of the new service. When a SDS server receives a request, it can apply two kinds of filters: 1) hierarchical and 2) by index. Hierarchical filter is based on parents, the SDS server verifies if there are local matches within its children. If no match is found, the request is rerouted upwards to try to find a match in another level of the hierarchical structure. In indexing filter, the SDS server verifies all neighbor filters and in case of match, it sends the request to the proper SDS server. Once a match is found, the correct SDS sends to the user a XML document with information about the service and how it can be contacted.

Jini [12] is one of the most popular service discovery protocol. It was introduced by Sun Microsystems to share any type of services among any type of client. Jini considers as a service any artifact (e.g., a coffee maker) that can be represented by a Java object. The Jini infrastructure is integrated by three protocols and three participants. The protocols are

discovery, search and union. Participants include a client, a service provider and a look up service. A client is the entity which looks for a service, a provider is the one offering services and a look up service is in charge of holding the services that are shared in the Jini system. The union and discovery protocols allow a new service to be added to the system. When a new service enters to the system, it looks for a look up service to register. This search is made through a multicast request. Once the look up service is found, the union protocol helps the service to register itself into the look up service. That registering process consists in making a copy of the java object that represents the new service into the look up service. When a client needs to find or invoke a service, he makes a request to the look up service, which looks among the services registered in it. In case a service satisfying the client request is found, the look up service transfers the java object representing the service to the client. In this way, the client would have enough information to contact the service.

Bonjour [10] is the Apple solution for service discovery. Its main goal is to achieve zero-configuration IP networking, which consists in creating an IP-based network without user intervention. This network will help users finding services easily by avoiding the need of knowing the name of the service, its location and the service configuration process. Bonjour is able to locate shared devices (e.g., printers and computers) and files (e.g., music). Since users of Bonjour look for services, not for IP addresses, this SDP uses Multicast DNS (mDNS), in which DNS format queries are sent using multicast. When a service sees a query with its name, it supplies with its own IP address. Service discovery allows applications to find out all available instances of a given type of service and to maintain a list of those services. This list is used to resolve service IP address given its name. When a service is needed, a mDNS query is sent specifying the type of service. Any service of that type may respond with their name, which results in a list of services where application (or user) can choose from. Service discovery is performed only when a new type of services is needed, so a list of service names can be gotten. Based on this list, a client application (or the user himself) can use mDNS requests in order to resolve service names into IP addresses, whenever a service is needed.

B. Comparison

Figure 1 compares some interesting features of the previously studied discovery protocols. First feature considers the discovery architecture, which refers to whether a protocol is based on directories or not. A directory-based architecture comprises a repository, which stores data about the registered services, and processes service requests. From the studied protocols, only UPnP does not follow a directory-based architecture, since its control point searches for a service on several controlled devices until finding one that satisfies its requirements.

The next feature, called supported services, refers to the types of services that can be discovered by a protocol. SLP allows to find networking software (e.g., a device controller) and it provides additional features allowing user agent to take into account both applications and people. Although these features can be implemented, SLP is not specifically oriented to people-services interaction. UPnP is intended for physical

SDP / Features	SLP	UPnP	Ninja SDS	Jini	Bonjour
Discovery Architecture	Based on directories or multicast messages	Not based on directories	Based on directory hierarchies	Based on directories	Based on directories
Supported Services	Device driver location	Hardware driver installation	Software and hardware usage	Hardware, software and artifact usage on Java	Hardware and software
Communication Model	Push and Pull	Push and Pull	Pull	Push and Pull	Push and Pull
Service Description	Static templates and template creator	XML documents	XML documents	Static templates	DNS resource records
Information Filtering	Scopes	Channels	SDS finding	Look up service finding	mDNS responder finding
Matchmaking Algorithm	Type of service	Type of device and service	Bloom-filtering	String comparison of type names	Type of service

Fig. 1. SDP Comparison Table

devices since its main goal is to prevent users from configuring hardware when it is plugged into a computer for the first time. Ninja SDS and Bonjour consider both hardware and software services. Jini is able to manage different types of services (e.g., hardware, software and physical artifacts) wheter can be represented by a Java object.

The communication model feature refers to the communication approach employed by a protocol to allow participants to make requests and to announce services. Ninja SDS relies on a pull approach to discover services. However, the others SDPs follow both the push and pull approaches. Indeed, in addition to allowing user entities require for services, these SDPs allows to the services or the directories to advertise themselves periodically in order to promote their existence.

The service description involves the way in which a protocol handles information about the services it manages. From the analyzed protocols, SLP and Jini use predefined templates to uniformly describe services and requests. Bonjour uses DNS resource records, which specify the type and the location of the published service. UPnP and Ninja SDS use XML documents, which constitute a more flexible mean for service description since they can be enriched by adding new attributes (labels) when needed.

The information filtering feature refers to the process of selecting messages for reception and processing. SLP groups services into categories called scopes. UPnP allows control points to listen to evented variables, which are specified by an initial event message during subscription. Ninja SDS, Jini and Bonjour do not choose the messages they receive prior to processing them. When a provider or a user wants to either announce his service or ask for one, he contacts an available directory storing services. However, this directory is not chosen amongst others considering a specific reason, but just the first one found.

The matchmaking algorithm characteristic refers to the way in which a protocol determines which service can satisfy a request. In SLP, either a directory agent or a service agent compares the URL of the announced services with the user

agents request to determine whether a service that fulfills this request exists. UPnP also makes comparisons between the device type and the offered configuration services in order to find the most adequate one. Meanwhile, Bonjour just compares type of services. Ninja SDS uses a mechanism called bloom filter, which consists in gathering each service attributes into sets and applying hash functions on these sets to obtain a vector that represents the service. Then, each service vector is compared with the request vector in order to determine whether a service satisfies the clients needs or not. Jini makes comparisons between the description and request templates to find matches.

From the comparison described above some important drawbacks of the already developed Service Discovery Protocols can be detected. Most of the SDPs were mainly designed for applications asking for services, so they provide minimal or null support for human users. The way the offered services and requests are described is also an improvable feature, because some of the studied SDPs use predefine templates that should be filled out to describe offered or required services. This description process limits users to express their needs. It is also important to notice that SDPs do not consider the environment nor the actual conditions of a service or of the user asking for a request.

C. Frameworks

The Adaptable Intelligent Discovery of context-Aware Services (AIDAS) framework [8] offers contextual service discovery. AIDAS is integrated by three type of entities: 1) services, which represent applications available on the network; 2) users, which correspond to either humans or applications looking for a service and 3) devices, which refer to the computers (mobile devices mainly) used to access the system. The AIDAS framework architecture is based on two major logical sets of modules: the discovery management and the configuration management sets. The discovery management set includes: a) a context manager in charge of considering the changes in the environment and b) the profile matching engine, which is home of the semantic matchmaking algorithm in charge of comparing each service capability to the request made. AIDAS considers non exact matches by incorporating subsumption operations; their results will affect the matching degree between the offered and the required services. Finally, c) the discovery manager is responsible for providing a list of visible/accessible services to users based on their context. This modules uses the profile matching engine and the context manager to achieve its duty. When a service provider wants to add a new service, he describes it by filling static templates, which are syntactically checked before being stored in a service registry allocated by the discovery management services set. When a user starts a discovery session, his and the device profiles are retrieved and assessed by the discovery management modules to generate a view of accessible services according to the user context (i.e., location). A user also has the possibility to make a specific request about the service he is looking for. Finally, the user gets as response a lists of services and a reference to contact them.

The DAIDALOS (Designin Advancednetwork Interfaces for Delivery and Administration of Location Independent, Optimized personal Services) project [6] proposes to add a semantic layer to a traditional service location protocol (e.g., SLP or Jini). This semantic layer consists of: 1) an ontology where punctual characteristics of services are expressed, and 2) a contextual manager, which holds pointers to contextual sources (e.g., sensors) of the entities participating in the protocol. These entities are: 1) a user, who can be either a person or an application starting a discovery process and will receive the list of found services, 2) the service, which represents an application or device a user is looking for and 3) the environment, referring to the pervasive entity in which the user is submerged. When a user needs a service, it makes a request to a service discovery server specifying the basic and semantic characteristics a service should have to fulfill its request. The user also provides the pointer to his context source. Then, a service discovery server processes the user request by using three types of filters: basic filter, semantic filter and context filter. The basic filter uses the traditional service discovery protocol filter engine to select a set of services of the correct type. This set is then directed to the semantic filter, where services that have the exact same characteristics the user is looking for are selected and then passed to the context filter. In this phase, the service context requirements are evaluated to determine if they fit in the user and environment conditions. Then, the filter compares the users contextual requirements with the user and environment conditions to obtain a selected set of services that may satisfy the users request in a better way.

In this paper, we present the RAMS (Resource Availability Management Services) architecture, which is mainly focused on facilitating the development of groupware applications that manage the availability and suitability of requested human, virtual (e.g., files and software) and physical resources (e.g., laptop, scanner and tablet). These resources are described in a semantic support consisting of a set of expressive ontologies. The environment in which all the considered resources are involved is also modeled into an ontology. This semantic description enables the matchmaking process to be more complex and accurate. Because, instead of just comparing keywords as traditional service discovery protocols do, the matchmaker algorithm will try to match offers and requests of services according to their actual meaning. As the RAMS architecture is designed to operate in pervasive collaborative environments, the preferences of human resources and the conditions on their changing environment are treated with high concern.

III. The RAMS architecture Scenarios

The RAMS architecture is being constructed to provide support for resource sharing in multiple environment of different sizes. From small places where there are different but quite limited number of resources to places full of a large quantity and variety of resources. To illustrate the versatility of the RAMS architecture, three types of institutions where resource sharing management is vital are briefly described.

Design Agency

In a design agency, a library of reusable virtual resources (e.g., photoshop elements, css files or JavaScripts) is essential. These elements can be just the foundation of a new and completely different creative work, but they will still consume time and work of a designer. So, to make a profit on projects

with a tight budget, a good management of these virtual resources is needed.

In most of the agencies, designers store their work in their own computers. In bigger and more organized design agencies, a shared repository is kept. Even though, this is a basic solution for virtual resource sharing, new designers or even designers from different teams could be clueless about the existing elements in the repository if a good description is not provided. Additionally, designers are in risk of waisting much time trying to find something they are not even sure it really exists. A variety of physical resources are also shared in these type of environments. From computational devices such as printers, digital cameras and design tables to meeting rooms used by designers to exchange their creative ideas. By using a computational support to manage resources, the time spent doing tedious and repetitive stuff could decrease, giving designers more time to produce excellent results for their clients.

Hospital

Everyday doctors discuss patients cases in meeting rooms enabled with computer equipment. These meetings can be already scheduled, such as when nurses switch shifts and give report to other nurses about the health of patients, or there can be urgently needed meetings where various specialists need to participate to give a diagnosis or to discuss a treatment. Any of these meetings commonly leads to the scheduling of test and therapies. This situation involves the management of human (i.e., nurses and physicians), physical (e.g., meeting room and medical/computer equipment) and virtual (e.g., patients medical records) resources. Another evident situation of management resource sharing is present in large hospital systems with multiple medical campuses, where the equipment used is often taken with patients who are transferred between hospitals. These transfers lead to the loosing or misplacement of vital equipment which along with the time people spend in locating available equipment cause a big financial impact.

Institutions as big and dynamic as hospitals arise many challenges in resources management, because of the mobility of not just human but also physical resources. However, the current RAMS architecture can be enhanced to fulfill all the requirements of this type of institutions by developing additional components (e.g., a physical resource tracker).

University and Research Center

A multidisciplinary researcher center full of many heterogenous resources is the case of study we have chosen to prove the RAMS architecture. This environment is of our interest, because in our institution we have encountered situations where a shared-resource support will be of great utility. The bigger the institution is, the smaller the probabilities of knowing all the existing resources in it are. A computational support for resource sharing also allows to share resources among unknown colleagues from the same institution but different departments. Thus human relationships are also promoted. Following, a scenario is presented to give a better idea of the situations the RAMS architecture deployed in a university or research center is targeted to solve.

Miss Andrew requires a portable interactive whiteboard to give a Ubiquitous Computing conference in a nearby auditorium. She has prepared some interactive material, as she likes to provide interesting and fun presentations. However, in the auditorium where she is going to give the talk, there is not an interactive whiteboard installed. As Miss Andrew considers really important to keep the audiences attention, she decides to look for a portable interactive whiteboard among her colleagues. In section VII the details of how this request is solved by the implemented application based in the RAMS architecture are presented.

IV. The RAMS architecture

To accomplish our goal of providing a semantic support for finding the best resource available for a collaborators request in a pervasive environment, we propose a set of ontologies and a Matchmaking Service. The set of ontologies store and manage static (e.g., capabilities and technical characteristics, usage policies and access rights granted to colleagues) and dynamic information (e.g., human resource goes to a different office). The Matchmaking Service components rely on these both types of information to select the most suitable and available resources capable of satisfying a consumer request. To obtain these static and dynamic information some other components of the RAMS architecture are used. Following, we give an overview of the proposed architecture.

The RAMS architecture is based on the asynchronous publish/subscribe model [5], which was chosen above others message models (e.g., message passing) because most of its principles can be directly applied to end-users of RAMS-based applications. In this way, collaborators can play the roles of producers and/or consumers of events related to the state of shared resources (e.g., presence, location and availability). However, unlike the publish/subscribe model, where producers do not know the consumers of their messages and vice versa, users of groupware applications should identify their colleagues in order to define filters that control the scope of event production and consumption.

The RAMS architecture defines two types of roles that collaborators might play: 1) producers, who publish resources to share them with their colleagues and generate events to change their state; and 2) consumers, who subscribe to the system to find resources they need to use and receive events about the state of resources they are allowed and interested in.

The RAMS architecture also defines actions that collaborators playing these roles can accomplish through RAMS-based applications. A producer can perform the following actions:

1) Publishing a resource: when a producer wants to share a resource with his colleagues, he can publish it using functionalities that allow him: a) to describe the resource in terms of its capabilities and technical characteristics; b) to grant his colleagues access rights on the resource; and c) to define policies that regulate the resource usage.
2) Producing application events: a producer can change the state of his resources at any moment, e.g., to temporarily set a resource as unavailable or to activate the do not disturb mode.
3) Modifying resource information: a producer can modify information about his published resources when needed. Modifiable information concerns resource

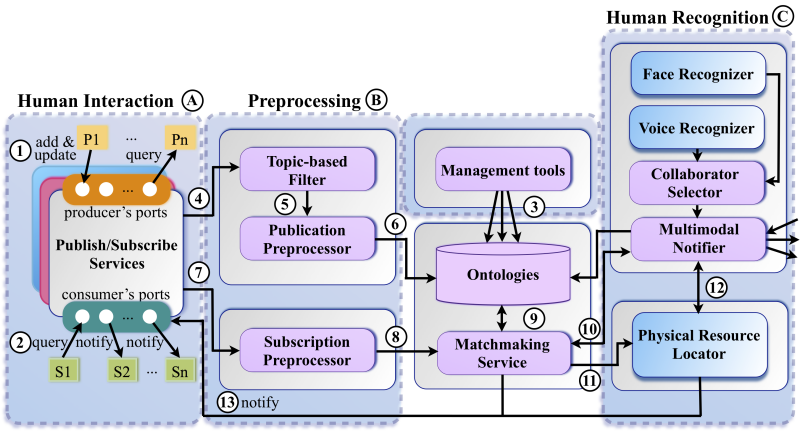

Fig. 2. RAMS architecture

capabilities and technical characteristics, access rights and usage policies.

4) Canceling a publication: a producer can stop sharing one or more of his published resources whenever he wants.

A consumer can perform the following actions:

1) Subscribing to resource state information: a consumer interested in receiving state information about some published resources can subscribe to them. In addition, he can activate information filters to only receive notifications about the resources he is interested in at a given moment.
2) Searching for a resource: if a consumer needs to use some resources, he can use functionalities that allow him to describe the technical characteristics he is looking for.
3) Canceling a subscription: at anytime, a consumer can stop receiving state information about some resources he is subscribed to.

By granting access rights and specifying usage policies, a producer will be certain that his resource will be reached just by people he relies on and that the resource is going to be treated properly.

The components of the RAMS architecture are classified according to the type of service they provide into: human interaction (see Fig. IV-A), preprocessing of data (see Fig. IV-B) and human recognition (see fig IV-C).

The human interaction category consist of a Broker that provides the application developer with services for implementing an interaction support between collaborators and RAMS-based applications. Particularly, the Publication Service (see Fig. IV step #1) allows collaborators playing the producer role to describe their resources in terms of technical characteristics, to define usage policies and to give access rights to colleagues they want to share their resources with. The Publication Service sends that resource-related information to the Topic-based Filter (see Fig. IV step #4), which classifies it into the right ontology according to the type of resource that is being published (e.g., human or virtual resource). The Publication Preprocessor structures the classified information received from the Topic-based Filter (see Fig. IV step #5) to make it comprehensible for the RAMS Ontologies (see Fig. IV step #6), which are in charge of storing and managing it.

On the other hand, collaborators playing the consumer role can interact with RAMS-based applications by means of the Subscription Service (see Fig. IV step #2), which allows them to describe the type of resources or the specific resource they are interested in. The Subscription Preprocessor structures the resource descriptions obtained from the Subscription Service (see Fig. IV step #7) to make them understandable for the Matchmaking Service (see Fig. IV step #8).

From relevant information retrieved from the RAMS Ontologies (see Fig. IV step #9), the Matchmaking Service (cf. see Section VI) can select authorized resources, whose attributes correspond to the technical descriptions provided by collaborators when subscribing to receive information about a resource or when emitting specific requests. As a result of this matchmaking process, a set of resources that potentially satisfies the consumers request is obtained. However, these resources cannot be considered the best match for the request if they are not available for the consumer at a given moment. To verify a resource availability, the Matchmaking Service takes into account dynamic information of these resources provided by the Multimodal Notifier (see Fig. IV step #10) and the Management Tools (see Fig. IV step #3). The Multimodal Notifier in charge of communicating the decision of the Collaborator Selector. It determines the presence and location of a human resource by considering information coming from a Face Recognizer and a Voice Recognizer. These components will ensure an accurate response about the identity and location of a collaborator. The Management Tools allow producers to modify their availability or the one of their published resources at anytime.

When a consumer is looking for a physical resource, the set of suitable and available resources selected by the Matchmaking Service is transmitted to the Physical Resource Locator (see Fig. IV step #11), which asks the Human Face Recognizer for the consumers current location (see Fig. IV step #12) in order to determine the closest resource to him and the path he should follow to reach it. The results produced by either the Physical Resource Locator (when looking for a physical resource) or the Matchmaking Service (when searching for a

human or virtual resource) are finally delivered to the consumer (see Fig. IV step #13).

V. The RAMS Architecture Ontologies

To present a proper explanation of the ontologies [1] designed for the RAMS architecture in section V-A some basic ontology concepts are introduced. Following, section V-B presents the taxonomy of each one of the ontologies of the RAMS architecture. Later, section V-C introduces the group of object properties proposed to create relationships among individuals belonging to the same or different classes. Finally, in this same section the utility of the same object properties characteristics and restrictions is explained.

A. Ontology Vocabulary and Representation

To make this paper self contained some ontology vocabulary and the corresponding graphical representation are introduced:

- Individual: It is an object of the interest domain. Individuals are also known as instances of classes and are represented by a diamond.
- Property: It is a binary relationship between two individuals or an individual and a value. There are three types: 1) data properties (represented by a green rectangle) relate an individual from a class of the interest domain to a XML Schema Datatype value or a RDF literal (i.e., individual-value relationship); 2) object properties (represented by a blue rectangle) link individuals belonging to the same or different classes; and 3) annotation properties (represented by a yellow rectangle) allow to provide additional information about individuals, properties, and classes.
- Class: It is a concrete representation of a concept. Thus, classes represent individuals that are similar among them in some way or another and are presented by a circle.
- Quantifier restrictions: They are defined over object properties and can be either existential or universal. By defining an existential restriction, an individual can belong to a class just if it is related to at least one individual from such a class. Universal restrictions define the type of individuals an individual can be related to through the object property with the restriction attached to it.
- Closure axiom: It is defined over an object property by creating a universal restriction, which is in charge of limiting the types of individuals that can be related through that property to the ones considered in the existential restrictions.

B. RAMS Architecture Ontology Taxonomy

At initial states, the resource description modeling for the RAMS architecture included a single ontology. However, that ontology was growing really fast as the proposal became more ambitious. So, it was decided to split the single ontology into four. This set of ontologies was carefully designed to cover the entities description needs for different resource sharing environments. Four our case of study, which is a research center, we added a fifth ontology that describes the research areas and fields of study present in the organization we are validating the RAMS architecture, these elements are pretty specific of our research center.

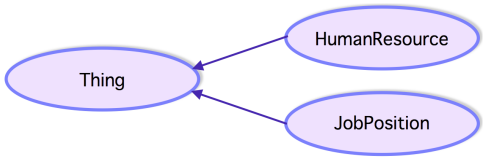

Fig. 3. HumanResource Ontology Hierarchy

All the classes in the RAMS ontologies are disjoint, because each class describes entities with unique characteristics, these classes can be part of a hierarchy but each one is different. None of the individuals of the RAMS system should belong to more than one class.

The Protégé tool [3] was used to create the taxonomy of the proposed set of ontologies. This tool uses the OWL-DL language which creates ontologies in a hierarchical class structure, where each class extends from the Thing class. Following, we give a brief summary of each one of the ontologies of the RAMS architecture.

Human Resource Ontology

This ontology (see Figure 3) includes classes related to individuals representing people that are part of an environment where a RAMS-based application is deployed. The classes included in the first level of the ontology hierarchy are:

`HumanResource`. Each individual belonging to this class represents a collaborator involved in the resource-sharing environment. A collaborator shares some of his personal information by either explicitly providing it or by agreeing to be observed by tools that obtain this information in an automatic or semiautomatic way.

`JobPosition`. Individuals belonging to this class represent actual job positions, which are linked to individuals representing collaborators. The job position information of a collaborator facilitates the granting of access rights and the definition of usage restrictions by classifying collaborators according to the job they perform.

Physical Resource Ontology

From the Thing class extends the PhysicalResource class, which is intended to group any kind of physical resources shared in an organization. The second level of the ontology hierarchy includes the following classes:

`Building`. The purpose of this class is to represent the physical environment where the resources are shared. Classes for the different type of areas (e.g., class room and meeting room) group individuals that represent an actual place in the building.

`Hardware`. The subclasses extending from this class categorize individuals representing actual devices shared by colleagues (e.g., computer and projector).

Virtual Resource Ontology

From the Thing class extends the VirtualResource class, which groups all individuals representing virtual resources according to their type. The individuals can belong to the classes: File, DriverPlugin, Database or Software classes.

Context Ontology

One of the main characteristics of our proposal is the interest in performing resource discovery considering the environment where collaborator asking for resources are located. So, variables that describe the changes in this environment have to be evaluated to determine the availability degree of a resource. These variables are categorized in the following subclasses of the context ontology (see Figure 4):

AvailabilityMode. It is an enumerated class that groups some particular individuals, such as the doNotDisturb and okToDisturb individuals. Individuals from the HumanResource class are related to one of the individuals from this class through the hasAvailabilityStatus object property to know whether a human resource has manually activated the do not disturb mode or not, which means he agrees to be bothered.

Company. This is another enumerated class holding the accompanied and alone individuals. This class is used to know whether a person is alone or accompanied by somebody. A human resource is related to one of the two individuals through the hasCompanyStatus object property.

Restriction. This class holds individuals representing a usage restrictions defined by a producer. To specify the usage restriction, each individual of this class is related to data properties that define its metric and allowed value. So, an individual from the HumanResource class is related to an individual of the Restriction class by the object property hasToSatisfy and an individual from the Restriction class is related to an individual from the PhysicalResource or VirtualResource class by the isAssociatedTo object property

Task. This enumerated class holds the move, repair and use individuals. A human resource is related to one or some individuals from this class to denote the type of actions he can perform over a virtual or physical resource.

UsageStatus. This class holds the Free and InUse subclasses. An individual from the PhysicalResource or VirtualResource classes is related to an individual from one of these subclasses to denote whether someone is using it or not. This is not an enumerated class because their individuals contain information about the amount of time the resource will be free or in use. Thus, each individual should be unique.

Institution Information Ontology

This ontology is open to group all the organization-specialized classes needed to complete the model.

Meta-Ontology

The function of this ontology is to relate all five ontologies described above by defining object properties (see Figure 5). It holds the HumanResource, PhysicalResource and VirtualResource classes located in the ontologies with

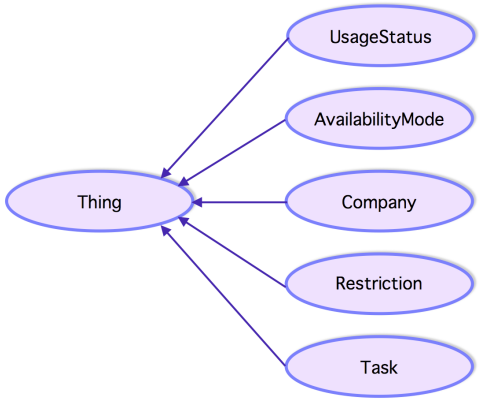

Fig. 4. Context Ontology Hierarchy

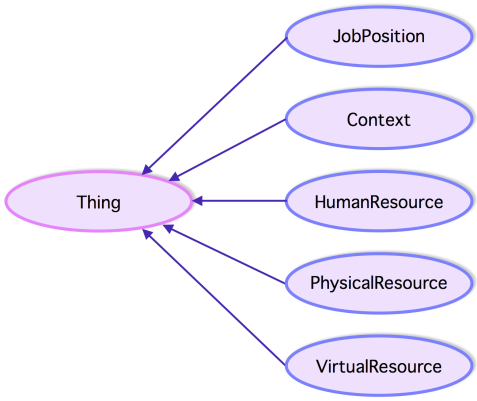

Fig. 5. Meta-Ontology Hierarchy

the same name. The Position class is part of the Human Resource ontology and the Context class groups all the classes modeled in the Context Ontology.

C. Object Properties

As mentioned earlier object properties are useful to give meaning to an ontology as they relate instances of classes that otherwise are independent. Nine groups of object properties have been specified to replicate in our semantic model the interaction that different individuals have in the real world. These groups are classified in:

1) humanResourceProperty. This group of properties can be customized according to the content of the institution information ontology.
2) locationProperty. This set of properties relate individuals from the PhysicalResource, HumanResource and VirtualResource classes to individuals representing physical or virtual places.
3) resourceRelationProperty. Properties grouped by this property specify the owner or responsible of a virtual or physical resource by relating individuals from the HumanResource class to individuals from the PhysicalResource or VirtualResource class.
4) constraintProperty. Properties categorized in this group create a three-individual-relationship be-

tween individuals from the HumanResource class to individuals from the Restriction class and from this last class to individuals from the PhysicalResource or VirtualResource class. The purpose is to define which usage restrictions has to be satisfied by a human resource when using a virtual or physical resource.

5) taskProperty. Some of the object properties of this group also define three- individual-relationships. These properties define who is authorized to perform a task over a virtual or physical resource. Thus, a collaborator who is sharing a resource can specify the privileges he wants to grant to the collaborators he shares his resource with.
6) usageStatusProperty. The properties grouped in here allow to identify whether a physical resource is being used at a given moment or whether it is free.
7) companyProperty: This group holds object properties that define if an individual from the HumanResource class is alone or accompanied.
8) jobPositionProperty. Object properties of this category are useful to create relationships between individuals from the HumanResource class and the JobPosition class to determine the job they performed inside the organization.
9) workRelationProperty. The object properties integrating this group specify working and hierarchical relationships among individuals from the HumanResource class.

Object Properties Characteristics and Closure Axioms

The meaning of the object properties can be of more value if their characteristics are determined. For that reason, each object property of the proposed model was analyzed and their suitable characteristics were determined. Following we present the most relevant characteristics considered along with an object property of the actual ontology of the RAMS architecture.

Inverse properties. They define relationships in both directions (i.e., from individual a to individual b and from b to a). As the isLocatedAt object property is the inverse of the allocates property. So, the following relationship can be established:

Building - *allocates* - Hardware
Hardware - *isLocatedAt* - Building

Functional properties. An object property with this characteristic can relate at most to one individual. An individual belonging to the Hardware class (i.e., a device) can be located at just one place at a time, so the relationship is functional:

Hardware - *isLocatedAt* - Building

Transitive properties. This characteristic defines that an object property relating an individual a to individuals b and c is capable of relating individual a to individual c. The isColleagueOf object property has this characteristic, making possible that different individuals from the HumanResource class relate to each other.

Symmetric properties. An object property with this characteristic acts as its own inverse property. So, the property is able to relate individual a to individual b and individual a to individual b. So, The following relations are possible thanks to this characteristic:

Restriction - *isAssociatedTo* - Hardware
Hardware - *isAssociatedTo* - Restriction

Irreflexive properties. An irreflexive object property relates individuals that are different from each other. In the RAMS ontology set, all of the object properties are irreflexive.

Closure Axioms. Some of these axioms are proposed to complete our ontology model. As shown above, the isLocatedAt object property relates individuals from the VirtualResource class to individuals from the Computer class. Over this relationship an universal and a existential restriction are defined. So, the axiom states that a virtual resource has to be located at least in one computer and that it can only be located at a computer.

VirtualResource - *isLocatedAt only* - Computer
VirtualResource - *isLocatedAt some* - Computer

Data Properties

It was necessary to identify the data properties (e.g., storage capability of an external hard drive) that would create the best description for the virtual and physical resources that the RAMS architecture proposes as generic for any environment. Thus, a survey was designed and applied to 100 potential users. They chose from many characteristics the ones they considered were more useful to know when looking for a resource. It is important to remember that the RAMS architecture can be enhanced or customized to fit any resources and their characteristics needed for deployment in a specific environment.

VI. RAMS Architecture Matchmaker

The RAMS architecture matchmaker exploits the power of the ontologies by not just treating them like repositories but by taking advantages of their semantic nature. In fact, the actual implementation of the ontologies separates de information about resources (i.e., population) from the formal model of the domain of study. Those, the model remains intact every time a new individual is created and every time a new relationship among individuals is defined.

To accomplish the goal of considering the resources and the environment where they are shared, a matchmaker service consisting of two phases is proposed. In the first phase the resources able to fulfill the technical characteristics and/or capabilities of a request are categorized into different classes that denotes their availability degree. The second phase is activated by an action that triggers a rule that causes changes in the availability degree of a resource.

The description of an individual representing a resource is treated as a policy stored in an owl file. This description includes the technical characteristics or capabilities of the resource, the relationships that determine the collaborators access rights and the defined usage restrictions over this permissions. A collaborator request is also treated as a policy, which should be matched against a policy describing a published resource. In the first phase of the matchmaking process, the individuals representing shared resources are queried using SPARQL [7] to get a list of resources that satisfy a collaborator request in terms of capabilities and/or characteristics. Then, each resource that a consumer is allowed to use will be classified according to a set of rules defined in SWRL[4] language. These variations of availability differ for each type of resource. So, a physical

resource is treated distinctly from a virtual or human resource. For a physical resource, a resource that a consumer is allowed to use can be:

- free. In case no one is using the resource
- job can be completed. The usage restrictions specified by the producer of a resource allows the consumer to achieve the resources he needs
- reachable. For a resource inside a restricted place (e.g., an office) it is necessary that some able to give access to it is inside the same place where the resource is located. A resource classified as reachable does not state that the owner or responsible inside the office is available. Thus, the next category is presented
- owner is available. This category applies also to resources inside restricted places, meaning that a collaborator allowed to give access to a resource agrees to be disturbed at a given moment.

The first step is to know if a consumer is allowed to perform a task (e.g., use) over a resource. If the consumer is allowed, then, the resource will be classified in one or more of the categories presented above. The SWRL rule presented in (1) classifies a physical resource as allowed to a consumer if an individual *?consumer* that belongs to the `HumanResource` class has a relationship through the `canPerform` object property to an individual *?task* from the `Task` class and the individual *?resource* (representing the assessed resource) from the `Hardware` class has a `isPerformedOn` relationship with the *?task* individual.

$$\begin{aligned} & HumanResource(?provider) \\ & \wedge Hardware(?resource) \\ & \wedge Task(?task) \\ & \wedge canPerform(?consumer, ?task) \\ & \wedge isPerformedOn(?task, ?resource) \\ & \rightarrow Allowed(?consumer, ?resource) \end{aligned} \tag{1}$$

To provide a real pervasive environment where changes in the environment are reflected in the availability of a resource, the second phase of the matchmaking process was planned. This phase consists of another set of SWRL [4] rules that are triggered if a context situation occurs. The rule presented in 2 is activated if a producer gets busy and decides to change his availability status to do not disturb through the management tools included in the RAMS architecture. This rule will classify a resource that was in the owner is available category to reachable if the *?provider* who is an individual from the `HumanResource` class, has a relationship with the *?doNotDistrub* individual from the `AvailabilityMode` class through the `hasAvailabilityMode` object property and the individual *?resource* from the `Hardware` class as long as the `?provider` have a `isLocatedAt` relationship to an `?office`, which is an individual from the `Building` class.

$$\begin{aligned} & HumanResource(?provider) \\ & \wedge AvailabilityMode(?doNotDisturb) \\ & \wedge Hardware(?resource) \\ & \wedge Building(?office) \\ & \wedge hasAvailabilityMode(?provider, ?doNotDisturb) \\ & \wedge isLocatedAt(?resource, ?office) \\ & \wedge isLocatedAt(?provider, ?office) \\ & \rightarrow Reachable(?resource) \end{aligned} \tag{2}$$

Thus, rules are defined for each considered situation and they differ from one type of resource to another.

VII. RAMS Architecture Validation Test

Miss Andrew (from section III) is interested in using a UBoard, which is a portable interactive whiteboard. She prefers this device because she is familiar with it and the required MINT software is already installed in her laptop.

By using an application based in the RAMS architecture, Miss Andrew expresses that the type of resource she needs is a portable interactive whiteboard; she also makes known that she prefers a UBoard, but she is open to other device from the same type. Thus, the request will be treated by the components of the RAMS architecture. The Matchmaker will evaluate dynamic and static information of the already published resources. From the static information four possible matches are found:

1) A UBoard belonging to Mr. Fowler;
2) An ONfinityCM2 portable white board owned by Miss Park;
3) A public UBoard guarded by the academic secretary; and
4) An eBeam interactive whiteboard property of Mr. Thomas

The static information also reveals that Miss Andrew will have permissions assigned to use these four devices if he satisfies the following restrictions:

1) Mr. Fowler is willing to share his UBoard with Miss Andrew whenever it is available;
2) Miss Park shares his ONfinity CM2 with Miss Andrew from Wednesday to Friday afternoon;
3) The public UBoard can be used by any collaborator at any moment; and
4) Mr. Thomas does not share his whiteboard with Miss Andrew on Tuesday because he gives a lecture in another university; therefore, he is not present to give access to his whiteboard.

According to the restrictions determined by the static information of these resources, Miss Andrew is allowed to use whichever of these four resources, because she is making his request on a Thursday at 2 p.m. In this way, restrictions in all four resources are satisfied.

The next step on selecting the best resource is to evaluate their dynamic information to compute their availability. The first resource evaluated is the public UBoard. Public resources are always evaluated firstly because it is faster to compute their

availability. Due to the public resource independent character, there is no need to obtain the owners or responsibles location and availability as in the case of private resources. From the evaluation of dynamic information, it was deduced that:

1) Mr. Fowlers UBoard is free, reachable and the owner is available. The RAMS components determine no one is using the UBoard and that Mr. Fowler is inside his office and management tools reveal he is available;
2) Miss Parks ONfinitu CM2 is also free, but the RAMS Face Recognizer locates her in the lunchroom. Consequently, she cannot give access to the whiteboard that is inside her office;
3) The public UBoard is being used by another professor and
4) Mr. Thomas eBeam whiteboard is equally free, but the RAMS Face Recognizer determines he is not inside his office, he is working with a colleague in a contiguous office. However, Miss. Dubois, Mr. Thomas PhD student, is inside her advisors office and she is available and allowed to give access to her advisors resources.

Considering this information, the RAMS Matchmaker determines that Mr. Fowlers UBoard and Mr. Thomas eBeam are suitable in terms of technical characteristics. They are free, reachable and the owner is available, so they will satisfy Miss Andrews requirements. So, the Physical Resource Locator is used to compute the closest resources to Miss Andrew. The Physical Resource Locator relies on the RAMS Face Recognizer to obtain Miss Andrews location and finds out that Mr. Fowlers UBoard, is the best option available for her.

VIII. Conclusion and Future Work

The objective of our work is to provide a computer support for finding available resources by creating a pervasive collaborative environment. To achieve this objective, an ontology based matchmaking approach is provided. This semantic approach consists of a customizable model expressed in a set of ontologies that suits many different type of organizations. A semantic approach is really useful because the resources and the environment itself can be accurately represented as they are and behave in the real world. This paper explains in detail how this set of ontologies was designed and constructed. Through the creation of class taxonomies more information can be given about the nature of the participants; data properties are helpful to characterize each instance of the classes, whereas object properties are needed to relate these instances. Properties restrictions are defined to emphasize the allowed and necessary relationships between individuals. This model is the foundation of the matchmaker, which treats individuals representing resources and requests from collaborators as policies that need to be matched. The matchmaker is able to give an accurate and up to date response when a request of a collaborator is made by querying the ontologies about resources already shared and responding to changes in the environment through the activation of defined rules.

The proposed ontology model and the matchmaker use the components of the RAMS architecture to accomplish their tasks. The RAMS provides an architecture integrated by components that can be grouped according to the task they perform in: 1) human interaction, which involves the publish and subscribe services in charge of interacting with collaborators interested in either sharing resources or using resources they do not own; 2) preprocessing, containing components capable of transforming the information coming from collaborators to information that the ontologies can understand and 3) human recognition, which are a set of components in charge of identifying and locating collaborators.

Many traditional service discovery protocols describe services in a limited way and use keyword searches. This has been hardly criticized due to the lack of expressiveness in the descriptions and the tendency to get wrong or null results because exact syntactic matches have to be found. Thus, a semantic approach works better for a resource sharing pervasive environment because even if the information about a user request is reduced, more information can be discovered when performing the request matching process.

The current state of our research lead us to keep working in the implementation of our matchmaking algorithm to consider more contextual situations. We will also evaluate its performance by putting in to work in our case of study (our institution) along with all the components of the RAMS architecture. The voice recognizer is also still in development. So, some more work has to be completed to bring all the components of the RAMS architecture together.

References

[1] A. Dean; J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Elsevier, April, 2008

[2] S. Herborn, Y. Lopez and A. Seneviratne, *A Distributed Scheme for Autonomous Service Composition*, In Proceedings of the First ACM International Workshop on Multimedia Service Composition, pp. 21-30, Singapore, November, 2005.

[3] M. Horridge et. al., *A Practical Guide To Building OWL Ontologies Using Protg 4 and CO-ODE Tools*, Edition 1.3, University of Manchester, March, 2011

[4] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.

[5] G. Muehl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems*, First Edition, Springer-Verlag, Heidelberg, Germany, 2010.

[6] V. Suraci, S. Mignanti and A. Aiuto, *A Context-aware Semantic Service Discovery*, In Proceedings of the 16th IST Mobile and Wireless Communication Summit, IEEE Computer Society, Budapest, Hungary, July, 2007.

[7] *SPARQL Query Language for RDF*, E. Prud'hommeaux, A. Seaborne, Editors, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/ . Latest version available at http://www.w3.org/TR/rdf-sparql-query/.

[8] A. Toninelli, A. Corradi and R. Montanari, *Semanitc-based Discovery to Support Mobile Context-aware Service Access*, Computer Communications, vol.31-5, pp. 935-949, March, 2008.

[9] UPnP Forum *UPnP Device Architecture 1.0*, Contributing Members of the UPnP Forum, October, 2008.

[10] K. White, *Apple Training Series Mac OS x Support Essentials*, Peachpit Press, Berkeley, CA ,USA, 2007.

[11] M. Weiser, *The Computer for the 21st Century*, SIGMOBILE Mob. Comput. Commun. Rev., ACM, pp., 3-11, New York, NY, USA, 1999.

[12] R. Zhao, J. Y. Zhi, G. D. Liu; , *Research and implementation of distributed testing system based on Jini technology*, Antennas, Propagation & EM Theory (ISAPE), 2012 10th International Symposium on, pp.1260-1263, October, 2012.

[13] W. Zhao and H. Schulzrinne, *Enhancing Service Location Protocol for Efficiency, Scalability and Advanced Discovery*, Journal of Systems and Software, vol. 75, num. 1-2, pp. 193-204, February, 2005.

[14] F. Zhu, M. Mutka, and L. Ni, *Service Discovery in Pervasive Computing Environments*, IEEE Pervasive Computing, vol. 4, pp. 80-90, October-December, 2005.