# Interaction of Context Descriptor and Ontology for Semantic Service Discovery in Ubiquitous environment[1]

**JunWon Kwak**
Department of Computer Engineering,
Sungkyunkwan University,
300 Chunchun-dong,
Jangan-gu, Suwon,
Gryeonggi-do 440-746,
Republic of Korea
ssimle777@ece.skku.ac.kr

**UngMo kim**
Department of Computer Engineering,
Sungkyunkwan University,
300 Chunchun-dong,
Jangan-gu, Suwon,
Gryeonggi-do 440-746,
Republic of Korea
umkim@ece.skku.ac.kr

## Abstract

The ubiquitous computing vision is to make knowledge and service easily available in our everyday environment. A wide range of devices, applications and services can be interconnected to provide intelligent and automatic systems that make our lives and more enjoyable and our workplaces more efficient. Before services can be used, they first have to be found. Service discovery is a mechanism for finding services. Current mechanisms are syntactic methods (i.e. they match service requests and descriptions of service offerings based on keywords) which often lead to a poor quality result. This our work focuses on how the quality of the service discovery result can be improved. Service discovery quality denotes the extent to which a returned match is relevant for the user. Interaction of Ontology and context descriptor is to improve quality of result for service discovery interoperability of a between user and service provider.

## 1 Introduction

In the Ubiquitous Computing [8] (ubicomp) vision of the future, workplaces, homes and public environments will contain a wide range of networked devices intended to make workplaces more efficient, to increase quality of life, and empower users by providing information and services in an effortless way. Current networked environments are populated with a diverse set of devices, services and computational entities. Enabling these components to work together harmoniously and allowing users and applications to interact with them without considerable administrative and configuration overhead poses a number of logistical and technical challenges. As a result, there has recently been considerable research into service location and device interaction technologies, including Jini technology [5,10], SLP [4, 14], UPnP [1].

The key function of such service location and device interaction technologies is to allow users and applications to deploy, discover and interact with the services provided by devices and software components on the network. This interaction is required to occur without the users, the applications, or the service providers needing detailed knowledge of the local network configuration. While these technologies were originally developed for zero configuration networks (e.g. those aimed at installation in the home en-

[1] * This work was supported in part by Ubiquitous computing Technology Research Institute(UTRI) funded by the Korean Ministry of Information

vironment) researchers have recently begun to investigate how they can be used to support aspects of mobile and ubiquitous computing (e.g. enabling a user in an unfamiliar network to discover nearby printers from their laptop).

We believe that these future service environments will be characterised by a heterogeneous mix of services and technologies. Moreover, contextual information plays a crucial role in the service discovery process. In particular, devices, applications and users will need to interact with multiple, potentially specialised, service location and device interaction technologies. Based on an analysis of existing approaches to service discovery and interaction, we have identified a number of shortcomings that we believe will seriously impact on the scalability, efficiency, utility and usability of current techniques in emerging service environments.

In this paper we present a design for a new semantic service discovery that combines the strengths of service discovery and interaction of context descriptor and ontology. Semantic service discovery is new mechanism to reduce semantic gap of a between user and provider.

After a short introduction to service discovery technique (Section 2), Section 3 describe semantic service discovery propose in this paper. Section 4 describe difference for performance of a between proposed system and current existing service discovery. Finally, we present our summary and conclusions, in Section 5.

## 2 Service Discovery Technique

### 2.1 Jini Technology

Jini technology [5, 10] is designed to help people build systems that deal gracefully with a fundamental characteristic of any distributed system: change. This appendix presents a quick introduction to Jini technology, and shows how Jini technology helps manage change within an operational distributed system.

Jini technology is used to build systems that exhibit a service-oriented architecture. Functionality on the network is divided into, and provided by, discrete services. In Jini technology, each service is represented by an object called a Jini service *proxy*. If a client wants to use a service, the client obtains a proxy for the service. The client then invokes methods (or functions) on the service proxy, and the proxy takes care of fulfilling the promised service, including possibly talking across the network to a server or other entity.

A fundamental kind of change in a distributed system happens whenever new participants join the system or others move around the network. If a client wants to use a service that just arrived or was recently relocated on the network, how does that client find the service? Jini technology addresses these issues by providing a lookup service where clients can find services they want to use. In addition, Jini technology provides a set of discovery protocols that enable Jini clients to find lookup services with no prior knowledge of their location. If a client desires to use a service for the first time, the client can find that service's proxy in the Jini lookup service. If that service later moves, the client can still find the service via the Jini lookup service. If the lookup service moves, the client can locate that lookup service via the discovery protocols. Once a client has obtained a proxy to a desired service via a Jini lookup service, the client uses the proxy to interact with the service directly, without further involvement from the lookup service.

### 2.2 Service Location Protocol (SLP)

The Service Location Protocol (SLP) is a product of the Service Location Protocol Working Group (SVRLOC) of the Internet Engineering Task Force (IETF) [4, 14]. It is a protocol for automatic resource discovery on Internet Protocol based networks. SLP is a language independent protocol. Thus the protocol specification can be implemented in any language. It bases its discovery mechanism on service attributes, which are essentially different ways of describing a service. It can cater to both hardware and software forms of services.

The SLP infrastructure consists of three types of agents:

1. User Agents
2. Service Agents
3. Directory Agents

UAs discover locations and settings needed by the potential user of the service; SAs advertise the availability of services; and DAs act as brokers, caching information about services. The system can operate in two modes—with or without DAs. When operating without DAs, the UA will send a multicast request for services,

and will receive unicast replies. When there are DAs present, SAs will attempt to register with a DA, and UAs will send all discovery requests to these brokers. Service descriptions in SLP are very basic 'Service URLs' that categorize service types.

SLP offers the following services:

1. Obtaining service handles for User Agents.
2. Maintaining the directory of advertised services.
3. Discovering available service attributes.
4. Discovering available Directory Agents.
5. Discovering the available types of Service Agents.

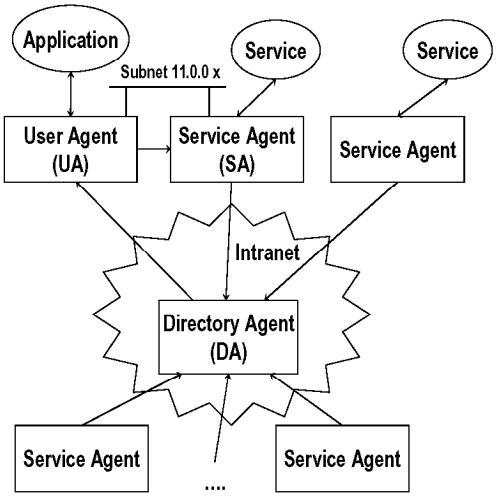



Figure 1: The basic transactions of the Service Location protocol

A service is described by configuration values of the attributes possible for that service. For instance, a service that allows users to download audio or video content can be described as a service that is a pay-per-use real-time service or a free-of-charge service. The SLP also supports a simple service registration leasing mechanism that handles the cases where service hardware is broken but the services continue to be advertised.

### 2.3 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP), pushed primarily by Microsoft, is an evolving architecture designed to extend the original Microsoft Plug and Play peripheral model to a highly dynamic world of many network devices supplied by many vendors [1]. UPnP works primarily at lower layer network protocols suites (i.e. TCP/IP), implementing standards at this level. UPnP attempts to ensure that all device manufacturers can quickly adhere to the proposed standard without major hassles. By providing a set of defined network protocols UPnP allows devices to build their own Application Programming Interfaces that implement these protocols - in whatever language or platform they choose.

UPnP uses the Simple Service Discovery Protocol (SSDP) to discover services on Internet Protocol based networks. SSDP can be operated with or without a lookup or directory service in the network. SSDP operates on the top of the existing open standard protocols, using the Hypertext Transfer Protocol over both unicast User Datagram Protocol and multicast User Datagram Protocol. The registration process sends and receives data in hypertext format, but has some special semantics.

### 2.4 OSGi platform service discovery

The Open Service Gateway Initiative (OSGi) [7, 12] Alliance is an open forum in which major companies participate (e.g. Samsung, Siemens). Their mission is to specify, create, advance, and promote an open Service Platform for the delivery and management of multiple applications and services to all types of networked devices in home, vehicle, mobile and other environments. The alliance was founded in 1999 and at this moment version three of the specification is released.

The mission of the OSGi alliance is to create an open specification for the network delivery of managed services to local networks and devices. The efforts of the alliance resulted in the OSGi platform specification. This specification describes a framework for an open and common architecture for service providers, developers, software vendors, gateway operators and equipment vendors to develop, deploy and manage services in a coordinated fashion The OSGi specification describes much more than service discovery alone. The complete platform specification is too extensive to describe completely in this section. Furthermore, parts of the framework are out of the scope of this research. Therefore, we will focus solely on the service discovery mechanisms of the OSGi platform in this section. The OSGi platform defines a java platform for service provisioning. The function-

ality of services is implemented by *bundles*. Bundles are a collection of java classes that can provide functionality to end users or other services. Services are described by their java *service interfaces*. The interface and the implementation classes together are used as a *service reference*. Furthermore, pairs of key/value can be used to specify some property of the service. The service is registered by the service provider in a central service registry, meaning that a centralized service discovery model is applied (see Figure.2).

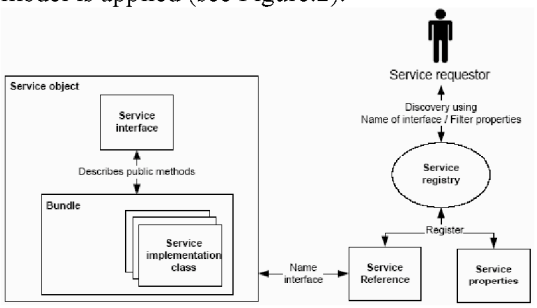


Figure 2: OSGi service discovery overview

After the services are registered in the service registry a service requestor can discovery the services by requesting service references (can be used to invoke the service). The request can be based on the name of the interface (syntactical matchmaking) or based on the defined properties using *filters*. By using retrieval methods on the service reference, properties of the service can be read and the requestor can determine which service it wants to use.

### 2.5 Ontology based Service Discovery

Historically, ontology is a part of metaphysics. Metaphysics is the branch of philosophy concerned with the nature of ultimate reality. It is customarily divided into ontology, which deals with the question of which fundamentally distinct sorts of entities compose the universe, and metaphysics proper, which is concerned with describing the most general traits of reality. The word 'ontology' originated in the Greek language, meaning "the study of being".

Currently, ontologies have also been used in computer science. In the beginning, they where developed in Artificial Intelligence to facilitate knowledge sharing and reuse. More recently, they have also been used in information retrieval, electronic commerce and knowledge management. They have been developed to provide machine-processable semantics of information sources that can be communicated between entities.

**Ontologies** can be defined as: "*a formal, explicit definition of a shared conceptualization*" [3]. A 'conceptualization' refers to an abstract model of some phenomenon in the world that identifies the relevant concepts of that phenomenon. 'Explicit' means that the used concepts and their constraints should be explicitly described and 'formal' means that the ontology should be machine-readable.

## 3 Semantic Service Discovery

Service discovery considers multiple parties in its functionality. The service requestor needs a particular service. The service provider offers a certain service and the context provider delivers contextual information about services and service requestors. These parties are distributed and have different knowledge on the service.

### 3.1 Ontology Technology

This section presents the major enabling technologies (i.e. ontology languages) for ontology-based applications. The discussed technologies are DAML+OIL.

***DAML + OIL*** [2, 11] is a combination of the DARPA Agent Markup Language (DAML) and the Ontology Interface Layer (OIL). Both initiatives recognized their common goal and in 2000 they merged in the combined initiative DAML+OIL. DAML+OIL extends existing standards (i.e. XML, RDF) to describe the structure of a particular domain in terms of classes and properties. A DAML+OIL description consists of a set of axioms that asserts relationships between classes or properties (i.e. intersectionOf, unionOf, complementOf, etc). In this way the semantics of the document is captured. (See Figure.3)

***OWL*** "The OWL Web Ontology Language is designed for use with applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics."[6, 13]

```
1 <daml:Class
        rdf:about="file:C:/ActiveSpaces/Semantics/MyOntology/file:C:\ActiveSpaces\Semantics\MyOntology\ActiveSpace.daml#
        Temperature">
2  <rdfs:label>TemperatureInformation</rdfs:label>
3  <rdfs:comment><![CDATA[]]></rdfs:comment>
4  <oiled:creationDate><![CDATA[2002-10-06T19:18:06Z]]></oiled:creationDate>
5  <oiled:creator><![CDATA[angarai]]></oiled:creator>
6  <rdfs:subClassOf>
7   <daml:Class rdf:about="file:C:/ActiveSpaces/Semantics/MyOntology/ActiveSpace.daml#WeatherInformation"/>
8  </rdfs:subClassOf>
9  <rdfs:subClassOf>
10  <daml:Restriction daml:cardinalityQ="1">
11   <daml:onProperty rdf:resource="file:C:/ActiveSpaces/Semantics/MyOntology/ActiveSpace.daml#subject"/>
12   <daml:hasClassQ>
13    <daml:Class>
14     <daml:unionOf>
15      <daml:List>
16       <daml:first>
17        <daml:Class rdf:about="file:C:/ActiveSpaces/Semantics/MyOntology/ActiveSpace.daml#PhysicalPlace"/>
18       </daml:first>
19       <daml:rest>
20        <daml:List>
21         <daml:first>
22          <daml:Class rdf:about="file:C:/ActiveSpaces/Semantics/MyOntology/ActiveSpace.daml#Person"/>
23         </daml:first>
24         <daml:rest>
25          <daml:nil/>
26         </daml:rest>
27        </daml:List>
28       </daml:rest>
29      </daml:List>
30     </daml:unionOf>
31    </daml:Class>
32   </daml:hasClassQ>
33  </daml:Restriction>
34 </rdfs:subClassOf>
```

Figure 3: DAML+OIL document

As an example, we have included a part of the description of an MP3 Server. This entity maintains a set of songs in MP3 format in its database. It allows other entities to search this set of songs using various parameters like name of artist, type of song, etc. It can also be sent commands for playing songs – other entities can either request a particular song to be played or a random song to be played. In addition, there is a human-understandable description about the entity. This is specifically meant for the average user who wants to know more about the entity in a simple language.

### 3.2 Context Descriptor

A key feature of applications in pervasive computing environments is that they are context-aware, i.e. they are able to obtain the current context and adapt their behavior to different situations. For example, a music player application in a smart room may automatically play a different song depending on who is in the room, in addition, it may decide on the song volume, depending on the time of day.
Context descriptor [15] has rules that describe what actions should be taken in different contexts. An example of a rule is :
*IF Location (Alice, Entering, Room 3231) AND Time (morning) THEN play a classic song.*

A rule consists of a condition, which if satisfied, leads to a certain action being performed. The condition is a Boolean expression consisting of predicates based on context information.

### 3.3 System Architecture

The system architecture of a semantic service discovery is equal to the Figure 4.

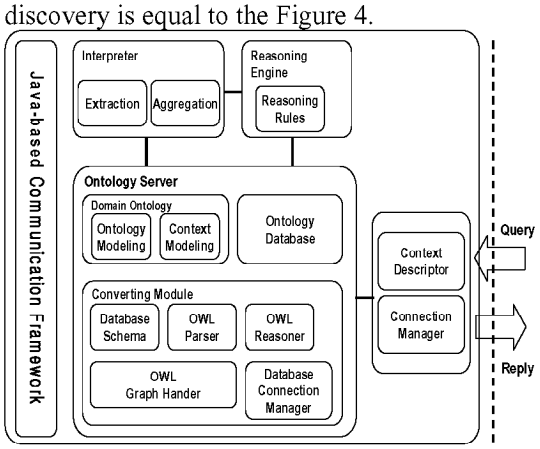



Figure 4: System Architecture

1. The interpreter module extracts and aggregates data from sensors. The reasoning engine produces new situation information based on the aggregated data.
2. The ontology server includes an ontology database, which stores the filtered context data from a reasoning engine
3. The converting module consists of a database schema, owl parser, and owl reasoner. The design of the ontology schema is shown in section 3.4.
4. Ontology modeling is divided into two steps: input/output parameters modeling and services modeling. First, we collect services input/output parameters, represent them with classes in ontology, and transform services to properties in ontology, defining relations between classes and properties.
5. The quality of the query result can be enhanced as uses of the context descriptor also provide specifications. The rule is as mentioned above.

### 3.4 The design of Ontology Schema

Ontology schema has basic two's table such the program code.

**Schema**

```
CREATE TABLE Onto_sys_stmt (
  Subj          VARCHAR(250) NOT NULL,
                /* subject */
  Prop          VARCHAR(250) NOT NULL,
                /* predicate */
```

```
 Ojb          VARCHAR(250) NOT NULL,
              /* Object */
 GraphID      INTEGER
              /* ID of graph (an owl file) */
);
CREATE        TABLE Onto_graph (
 ID           INTEGER NOT NULL
              AUTO_INCREMENT
              PROMARY KEY,
              /* ID of graph */
 NAME         TINYBLOB
              /* name of the ontology */
);
```

## 4 Implementation and evaluation

The main goal of this research was to improve service discovery by creating a new service discovery [13] approach. To be able to conclude that our system is an improvement or deterioration, it has to be evaluated. This chapter presents the result of this evaluation of the developed our system service discovery approach. To be able to evaluate our system, we developed a prototype. Our prototype should demonstrate the improvement of the quality of the discovery result of our approach compared to discovery results of current approaches (syntactic methods).

### 4.1 Reasoning Engine

Description Logic [16] allows specifying a terminological hierarchy using a restricted set of first-order formulas. The equivalence of OWL and description logic allows OWL to exploit the considerable existing body of DL reasoning fulfill important logical requirements. These requirements include concept satisfiability, class subsumption, class consistency, and instance checking.

Table 1 shows a sub-set of reasoning rules that support OWL-Lite entailed semantics.

**Table 1**. Parts of OWL ontology reasoning rules

| | |
|---|---|
| Transitive-Property | (?P rdftype owl:TransitiveProperty) ∧ (?A ?P ?B) ∧ (?B ?P ?C) ⇒ (?A ?P ?C) |
| subClassOf | (?a rdfs:subClassOf ?b) ∧ (?b rdfs:subClassOf ?c) ⇒ (?a rdfs:subClassOf ?c) |
| subProperty-Of | (?a rdfs:subPropertyOf ?b) ∧ (?b rdfs:subPropertyOf ?c) → (?a rdfs:subPropertyOf ?c) |
| disjointWith | (?C owl:disjointWith ?D) ∧ (?X rdf:type ?C) ∧ (?Y rdf:type ?D) ⇒ (?X owl:differentFrom ?Y) |
| inverseOf | (?P owl:inverseOf ?Q) ∧ (?X ?P ?Y) ⇒ (?Y ?Q ?X) |

In addition, ontology reasoning is also useful in other aspects of context aware computing. For example, in the example context ontology described in previous section, we define the relation '*locatedIn*' between a '*ContextEntity*' and a '*Location*' as an '*owl:TransitiveProperty*' relation, and the relation '*contains*' as the '*inverse property*' of '*locatedIn*'. Therefore, we can make use of the rules entailed by OWL to reason with physical location. An example result is shown in Table 2. Explicit context is acquired from context sources directly, while implicit context is the additional information deduced from explicit context. For example, knowing the user '*Jun*' is currently '*locatedIn*' the room '*Bedroom*', which in turn is a part of the '*Home*' building, description logic can be used to conclude that '*Jun*' is located in '*Home*' building as the spatial relation '*locatedIn*' is transitive.

**Table 2**. Reasoning about location using ontology

| | | |
|---|---|---|
| IN PUT | DL Reasoning Rules | (?P rdf:type owl: TransitiveProperty) ∧ (?A ?P ?B) ∧ (?B ?P ?C) ⇒ (?A ?P ?C) |
| | | (?P owl:inverseOf ?Q) ∧ (?X ?P ?Y) ⇒ (?Y ?Q ?X) |
| | Explicit Context | <owl:ObjectProperty rdf:ID = "locateIn"> <rdf:type = "owlTransitiveProperty"/> <owl:inverseOf rdf:resource = "#contains"/> </owl:ObjectProperty> <Person rdf:ID = "Jun"> <locatedIn rdf:resource = "#Bedroom"/> |

| | | |
|---|---|---|
| | | </Person><br><Room rdf:ID = "Bed-room"><br><locatedIn rdf:resource = "#Home"/><br></Room> |
| OUT PUT | Implicit Context | <Person rdf:ID = "Jun"><br><locatedIn rdf:resource = "#Home"/><br></Person><br><Building rdf:ID = "Home"><br><contains rdf:resource = "#Bedroom"/><br><contains rdf:resource = "#Jun"/><br></Building><br><Room rdf:ID = "Bed-room"><br><contains rdf:resource = "#Jun"/><br></Room> |

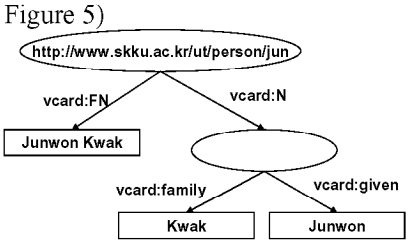
***User-Defined Reasoning*** A more flexible reasoning mechanism is user-defined reasoning. Through the creation of user-defined reasoning rules within the entailment of first-order logic, a wide range of higher-level, conceptual context such as "what the user is doing" can be deduced from relevant low-level context. Table 3 shows the user-defined context reasoning rules that are employed to derive user's situation in the smart phone scenario.

**Table 3**. User-defined context reasoning rules

| Situation | Reasoning Rules |
|---|---|
| Sleeping | (?u locatedIn Bedroom) ∧<br>(Bedroom lightLevel LOW) ∧<br>(Bedroom drapestatus CLOSED)<br>⇒ (?u situation SLEEPING) |
| Showering | (?u locatedIn Bathroom) ∧<br>(WaterHeater locatedIn Bathroom)<br>∧<br>(Bathroom doorStatus CLOSED)<br>∧<br>(WaterHeater status ON)<br>⇒ (?u situation SHOWERING) |
| Cooking | (?u locatedIn Kitchen) ∧<br>(ElectricOven locatedIn Kitchen)<br>∧<br>(ElectricOven status ON)<br>⇒ (?u situation COOKING) |
| Watching-TV | (?u locatedIn LivingRoom) ∧<br>(TVSet locatedIn LivingRoom) ∧<br>(TVSet status ON)<br>⇒ (?u situation WATCHINGTV) |
| Having Dinner | (?u locatedIn DiningRoom) ∧<br>(?v locatedIn DiningRoom) ∧<br>(?u owl:differentFrom ?v)<br>⇒ (?u situation HAVINGDINNER) |

## 4.2 Converting Module

Converting module is achieved two steps. First step is converting a ontology to statements.(see Figure 5)




Converting to first-order logic statements

Figure 5: Converting a ontology to statements

Next step is converting statements to an owl file.

### Program Code

```
//some definitions
String personURI =
"http://www.skku.ac.kr/person/
jun";
String givenName = "Junwon";
String familyName = "Kwak";
String fullName = givenName +
" " + familyName;
//create an empty model
Model model = ModelFac-
tory.createDefaultModel();

//create the resource
//    and add the properties
cascading style
Resource johnsmith =
model.createResource(personURI
)       .addProperty(VCARD.FN,
fullname)
        .addProperty(VCARD.N,

model.createResource()
       .addProperty(VCARD.Given,
givenName)
       .addProperty(VCARD.Famil
y, familyName));
//now write the model in xml
form to a file
model.write(Sytem.out);
```

**Converting statements to an owl file**

```
<rdf:RDF
   xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:vcard = "http://www.w3.org/2001/vcard-rdf/3.0#">
  <rdf:Description rdf:about = "http//www.skku.ac.kr/person/jun">
    <vcard:N rdf:nodeID = "A0"/>
    <vcard:FN>Junwon Kwak</vcard:FN>
  </rdf:Description>
  <rdf:Description rdf:nodeID = "A0">
    <vcard:Family>Kwak</vcard:Family>
    <vcard:Given>Junwon</vcard:Given>
  </rdf:Description>
</rdf:RDF>
```

### 4.3 Evaluation

This section evaluates our system approach by calculating precision and recall rates. Consider a set of relevant services (R) within a big set of advertised services (A) ($R \subseteq A$). We define:

**Recall**: The number of relevant items retrieved, divided by the total number of relevant items in the collection. The highest value of recall is achieved when **all** relevant items are retrieved.

**Precision**: The number of relevant items retrieved, divided by the total number of items retrieved. The highest value of precision is achieved when **only** relevant items are retrieved. With the result, we calculated the recall and precision rates. We compared these calculations with queries using current syntactic mechanisms (i.e. keyword and table based approaches).

**Query 1: "Discover all services that sell Music"**

This means that in our system all service types are selected and that "Music" is selected as output.

**Table 4**. Result of Query 1

| | Relevant services | Keyword-based mechanisms | Table-based mechanisms | Our system |
|---|---|---|---|---|
| Discovery result (services) | 19 | 6 relevant 2 irrelevant | 6 relevant | 19 relevant |
| Precision (%) | - | 75 % | 100 % | 100 % |
| Recall (%) | - | 32 % | 32 % | 100 % |

In the set of advertised services, there are 19 relevant services that sell Music. The keyword-based services retrieve only the services that have the word "Music" in their advertisements (8). Two of these services have "Music" as their input and not as output so these matches are irrelevant. The table-based mechanisms do not select the services with input "Music" and therefore does not have irrelevant matches. Our system selects only the services with output "Music" and the subclasses of "Music" like services that sell CD, DVD and LP. Therefore, it has full precision and full recall.

**Query 2: "Discover all shops that sell Music and that are nearby (250m)"**

This means that in our system "Sellingshop" is selected as service type and that "Music" is selected as output. Furthermore, "nearby" is selected as service property.

**Table 5**. Result of Query 2

| | Relevant services | Keyword-based mechanisms | Table-based mechanisms | Our system |
|---|---|---|---|---|
| Discovery result (services) | 5 | 1 relevant 7 irrelevant | 1 relevant 7 irrelevant | 5 relevant |
| Precision (%) | - | 15 % | 15 % | 100 % |
| Recall (%) | - | 20 % | 20 % | 100 % |

Both keyword-based and table-based mechanism cannot handle context in their discovery process. Therefore, nearby is ignored and many irrelevant services are returned. Our system evaluates all matches on the nearby rule and orders them by this property.

## 5 Conclusion

We have proposed a semantic service discovery approach, improving the quality of the service discovery result. Efforts towards this approach included (i) the identification of limiting factors for the quality of service discovery in current approaches (ii) the identification of types of service discovery matches and the definition of the degree of matching these (iii) the development of a novel service discovery approach, improving the service discovery result and finally (iv) the evaluation of the novel approach.

Some challenging future research areas must be addressed. Scalability and performance must be considered. Continuous researches regarding ontology maintenance and ontology relations for enhanced semantic reasoning are also required.

### References


1. Brent A. Miller, Toby Nixon, Charlie Tai, Mark D. Wood, "*Home networking with Universal Plug and Play*", Communications Magazine, IEEE Volume 39, Issue 12, Dec. 2001 Page(s):104 – 109
2. Deborah L. McGuinness, James Hendler, Lynn Andrea Stein, "*DAML+OIL: an ontology language for the Semantic Web*", Intelligent Systems, IEEE Volume 17, Sep/Oct 2002 Page(s):72 – 80
3. D. Fensel, *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce,* Springer-Verlag, Berlin, 2001
4. Erik Guttman, "*Service location protocol: automatic discovery of IP network services*", Internet Computing, IEEE Volume 3, Issue 4, July-Aug. 1999 Page(s):71-80
5. Kasper Hallenborg, Bent Bruun Kristensen, "*Jini Supporting Ubiquitous and Pervasive Computing*", CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 1110 – 1132.
6. Mark H. Burstein, "*Dynamic invocation of semantic Web services that use unfamiliar ontologies*", Intelligent Systems, IEEE Volume 19, Issue 4, Jul-Aug 2004 Page(s):67 – 73
7. Marples Dave, Kriens Peter, "*The Open Service Gateway Initiative: An Introductory Overview*", Communication Magazine, IEEE Volume 39, Issue 12, Dec. 2001 Page(s):110 – 114
8. M. Weiser, "The *Computer for the Twenty-First Century*," Scientific American, pp. 94–100, September 1991.
9. Phillip Lord, Pinar Alper, Chris Wroe, Carole Goble, "*Feta: A Light-Weight Architecture for User Oriented Sema Discovery*", ESWC 2005, LNCS 3532, pp. 17 – 31.
10. Rahul Gupta, Sumeet Talwar, Dharma P. Agrawal, "*Jini home networking: a step toward pervasive computing*", Computer Volume 35, Issue 8, Aug. 2002 Page(s):34 – 40
11. R. Scott Cost, Tim Finin, Anupam Joshi,Yun Peng, Charles Nicholas, Ian Soboroff, Harry Chen, Lalana Kagal, Filip Perich,Youyong Zou, and Sovrin Tolia, "*ITtalks: a case study in the Semantic Web and DAML+OIL*", Intelligent Systems, IEEE Volume 17, Jan/Feb 2002 Page(s):40 – 47
12. Tao Gu, Hung Keng Pung, Da Qing Zhang, "*Toward an OSGi-Based Infrastructure for Context-Aware Applications*", Pervasive Computing, IEEE Volume 3, Issue 4, Oct-Dec 2004 Page(s):66 – 74
13. Tao Gu, Hung Keng Pung, Da Qing Zhang, "*Toward an OSGi-based infrastructure for context-aware applications*", Pervasive Computing, IEEE Volume 3, Oct.-Dec. 2004 Page(s):66 – 74
14. The Salutation Consortium Inc. Salutation Architecture Specification (Part 1), version 2.1 edition, 1999
15. Vipul Kashyap, Amit Sheth, "*Semantic and schematic similarities between database objects: a context-based approach*", The VLDB Journal – The International Journal on Very Large Data Bases, December 1996
16. Vitaliy L. Khizder, Grant E. Weddell, "*Reasoning about uniqueness constraints in object relational databases*", Knowledge and Data Engineering, IEEE Transactions on Volume 15, Issue 5, S