Proximity Based Semantic Service Discovery for Pervasive Business

David Bell
School of Information Systems, Computing and Mathematics
Brunel University, Uxbridge, United Kingdom.

Email: david.bell@brunel.ac.uk

Abstract

Ubiquitous information systems (UBIS) adapt current Information System thinking to explicitly differentiate technology between hardware devices and software components in relation to people and process. A dynamic mix of users, mobile devices and software services requires research into new approaches to service discovery (including that within a business domain). Central registries of services and brittle fixed connections to single application services do not provide the necessary mobility and flexibility for an adaptive mobile enterprise. A future Web containing a heterogeneous mix of software services and embedded, mobile devices offers many opportunities to more effectively interact with enterprise applications (extracting information for rendering on appropriate devices at optimal times in appropriate places). This is all premised on the ability to discover appropriate service in a natural way. Service discovery can be found in a number of guises: Hardware oriented discovery, Enterprise Service discovery and Proximity based discovery to name but three. This paper explores how enterprise and proximity based services can come together and be discovered using semantic models and a SPARQL based web service (both within the proximity of the mobile clients). More practically, novel techniques for service selection using proximity based ontology filtering are proposed. The ontology is derived from a number of enterprise applications and then used by the semantic search service to identify appropriate business services. Proximity characteristics are proposed as an optimised means to select the enterprise service and filter the ontology search space. The ontology engineering process explores how proximity can be integrated alongside domain concepts. In order to experiment with service discovery a number of banking applications are used as a test bed. Functionality available within the applications is used as example application services – described using OWL ontology and discovered using SPARQL based semantic search algorithms. The proof of concept discovery platform is presented along with performance metrics relating to a set of investment banking trading and risk services.

Keywords: m-Business, Semantic Web, Semantic Discovery

1.0 Introduction

The Ubiquitous computing (Ubicomp) goal of an enhanced computer that makes use of the many devices embedded within the physical environment - effectively invisible to the user - impacts all areas of computing, including hardware components, network protocols, interaction substrates (e.g. enterprise data on ambient screen or process enactment on mobile devices), applications, privacy, and computational methods [Weiser 1993]. Invisibility within the physical environment is a central theme in Ubicomp. John Seely Brown at PARC called it the periphery [Weiser, et al. 1996]. In order to de-couple the information that we associate with our current applications and move it into the periphery, a means to discover available content and services is required that supports the user in their current place. Large numbers of devices, variation in how information can be adapted and the large number of source systems combine to make architecting such system challenging. Software service discovery in such an environment is not adequately supported by the hardware oriented approaches of universal plug and play (UPnP) and Bluetooth or enterprise approaches such as UDDI with its syntactic structure and centralized approach.

This paper presents a flexible approach to service discovery in the mobile business (m-Business) environment – supporting access to current business applications using mobile and ubiquitous devices. Semantic Web techniques are used to both construct models of the environment (and the software service residing within) in such a way that the services can be discovered using proximity description, local interests and ontology. Section 2 covers some current discovery approaches utilised in the enterprise and the mobile environments. Section 3 presents ubiquitous information system (UBIS) architecture and a number of applications that are then used as experimental input. Section 4 describes the design research method deployed. Sections 5 and 6 presents the technical artefacts and query performance measures.

2.0 Service Discovery in the large and small

A future Web containing a heterogeneous mix of software services and embedded, mobile devices offers many opportunities to more effectively interact with enterprise applications (extracting information for rendering on appropriate devices at optimal times in appropriate places). This is all premised on the ability to discover appropriate service in a natural way. Service discovery can be found in a number of guises: Hardware oriented discovery, Enterprise Service discovery and Proximity based discovery to name but three.

Two examples of lower level discovery approaches are UPnP and Bluetooth – contrasted by Richard [2000] and Helal [2002]. UPnP, an XML based discovery approach, uses Simple Service Discovery Protocol (SSDP) to undertake ID lookups. ID searches include service type, service name and location. Bluetooth's Service Discovery Protocol (SDP) supports searching by service class or associated attributes, as well as service browsing. Both lower level approaches are syntactic with few options for rich service description. The discovery of web services, often carried out using an UDDI registry, typically provides a yellow page style lookup of available services. The approach relies on common business and service categorizations having utility across a community [Bell, et al. 2007]. The semantic web has also added ontology to the web services stack; supported through the Web Ontology language (OWL). Semantic languages enable the relations between mobile and web resources (including sub-classing) to be made explicit. Work on device and sensor ontology has already started with examples such as OntoSensor, Gas, A3ME, MMI Device Ontology and CESN Ontology. The W3C has also started an incubator project looking at this area. The focus of this paper is not however to utilize or critique these approaches, but investigate the utility of semantic web techniques to ubiquitous software service discovery – determining whether the tools of the semantic web are appropriate.

Proximity based service discovery has been introduced to mobile ad-hoc network [Meier, et al. 2005] with service providers defining areas of service availability. The Proximity Discovery Service (PDS) allows clients to register an interest in a service - subsequently being informed when entering its proximity. PDS relies on a common vocabulary being used. Before describing our discovery approach in detail it is worth reflecting on the UBIS architecture in which it operates.

3.0 Ubiquitous Architecture

Working outwards (in Figure 1) the pervasive network is made up of a number of devices (sensors, actuators, user interfaces etc.) that are interconnected on one or more networks. Devices have been extended to include both local user interfaces and local services in order to support a range of devices from Smart-phones to Active RFIDs moving around the network. Processing capabilities on mobile and embedded devices are ever expanding and the extended service approach motivates design away from traditional client-server discovery approaches. These devices are then able to interface together in order to undertake specific tasks, for example peer to peer content distribution, or with access to remote applications and services. In order that mobile devices are able to access remote services or applications they must first utilize pervasive middleware. Original middleware definitions describing a middleware service as general purpose services that sit between platforms and applications [Bernstein 1996] appear too general (in light of the UBIS environment with application components sitting on many devices and applications) and requires additional clear and usable constructs encompassing more of the overall system in order to progress further.

In a more pragmatic vein the architecture in Figure 1 can be used to construct a high level architecture. Indentifying and discovering the applications, services and devices that are within scope of the design and mapping or choosing networks and middleware that is required to create connections between chosen components. Traditionally, these connections would be fixed at design time. Connections between specific applications and devices would be designed – choosing or developing appropriate middleware and network systems. Discovering applications, devices and the connections between them provides a dynamic alternative that requires services to be described up-front in such a way as to support discovery.

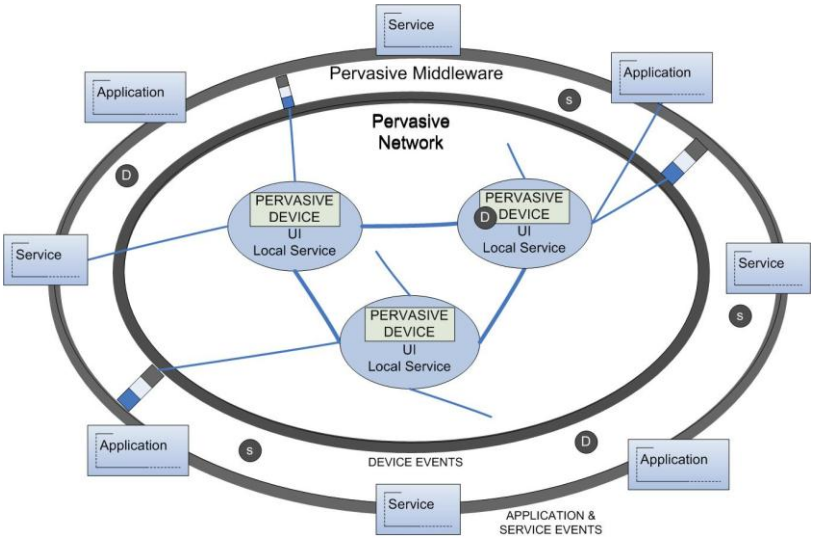

Figure 1: UBIS Architecture [Bell 2009] (extended from Mukhergee and Saha [2003])

The first extension to the architecture is the addition of events to the middleware layer. Two event channels are added to enable: (1) Events to be consumed and generate the Services and Applications (e.g. informing that a specific bond analytic has changed) and (2) Events to be consumed and generated by the pervasive devices

(e.g. Trader X is at location Y and is interested in Bond Z). It is this additional stage that allows the designer to explore both the capabilities and functionality of the devices and applications (inputs, outputs, requirement – when and where etc.). Consequently, the discovery and construction of linkages between the applications, services and devices are in response to specific events. The shared spaces ("S") are also included in the middleware layer as repositories of space based information and logic – embedded within a specific location. The space can be viewed as a *data cache* with integrated logic that is able to react to the cache and external events. Within this research the space becomes an *ontology space* containing domain, service and sensor ontology (e.g. data being the list of services that are then used by the discovery service D). The event message (an XML document) is read by the intelligent middleware that then identifies appropriate connections (interested parties). The second extension to the architecture is the connection middleware itself – comprising service adaptors, transcoders and device adapters. Initial analysis of the source systems (and their information provision) allows specific groupings of information (e.g. screen parts) to be identified as useful and described accordingly. Service adaptors extract this information in XML format (using the previous description as tags). A similar process is undertaken when describing particular devices, identifying and describing what they are able to render. The device adapters render XML on specific devices. The selection and use of the above applications, services, devices and spaces depends on their discovery. Discovery services ("D") are able to reside in the middleware or the device.

In order to experiment with service discovery a number of banking applications are used as a test bed. Functionality available within the applications is used as example application services – described using OWL and RDF ontology and discovered using SPARQL based semantic search algorithms that are presented later. Business concepts are explored alongside the architectural concepts presented as part of the UBIS architecture.

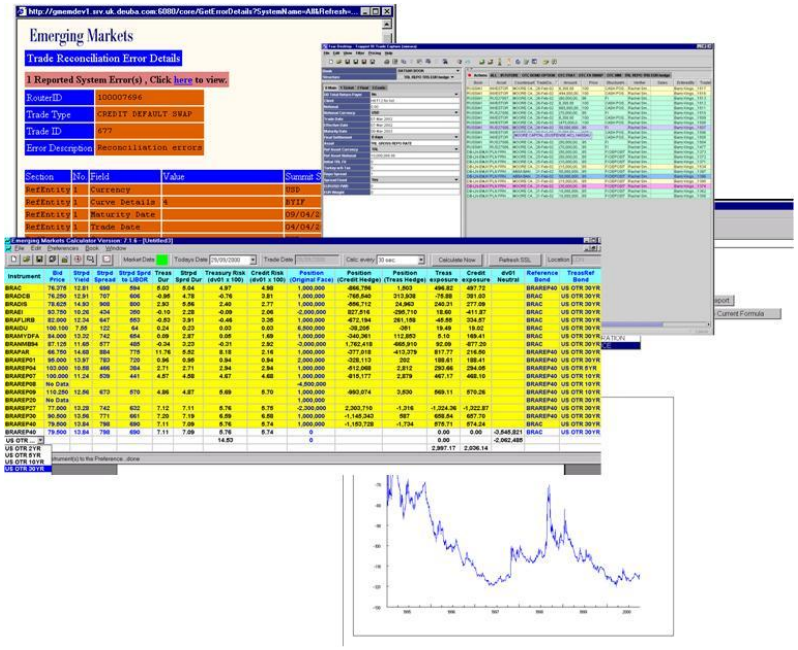

Figure 2: Banking Applications (Bond Calculator in the fore)

For example, consider Bond Calculator (shown in figure 2) that calculates a risk summary for each of the bonds that a trader has a position in. As underlying markets change, risk measures are recalculated and displayed in near real-time. Transporting this type of application into an UBIS environment, the intelligent middleware would react to sensor or environmental events; re-calculate risk measures and generate an output that is appropriate for the devices near to the trader. Practically, this would involve the reading of events (ΔBondZ's market, TraderX@Foyer, TraderX=iPhone) and constructing two pipelines that are able to convert the bond analytics into a format that can consumed by the device adaptors for the trader's iPhone and Ambient screen in the office foyer. The discovery of appropriate adapter and connector software requires a new type of flexible service discovery in order to provide invisible connectivity to enterprise application services. Proximity features support the trader within a certain distance of the ambient screen and iPhone as well as providing a means to support two traders in close proximity with common interests.

4.0 Design Research Method

Recent activity around design research methodology has provided a number of frameworks for research artefact generation – with artefacts often categorised using March and Smith's [1995] terminology of concepts, models, methods and instantiations. Artefacts are wide ranging and examples include data sets, methodologies and digital media. A consequence of focusing on the iterative design and implementation of more effective products and processes are a catalogue of interlinked design artefacts each generated over the life of the research project. Subsequent analysis at both a holistic level and of each artefact's unique life-cycle is limited without detailed recording of artefact characteristics over time (including their origination). Expanding analysis to evaluate against existing artefacts (typically applications or databases that are external to the design exercise) has additional difficulties, typically the result of many external artefacts being documented at a high level or not at all. In summary, the effective use and reuse of design research artefacts is heavily reliant on comprehensive capture of artefact detail (the same knowledge capture limitations are compounded when re-using artefacts across research projects).
The research question being considered within ubiquitous information system architecture is how service discovery should be undertaken more effectively – bringing together hardware devices and software/application services. The research described in this paper initiates this process by using ontology and ontology query to describe and select software services. The research follows a design research approach, which is a search process to discover an effective solution to a problem [Hevner, et al. 2004 p.88]. The relevance of the problem for the research community must be demonstrated and the solution must be effective to a satisfactory level. The effective solution may not (and generally does not) coincide with the best or optimal solution however - generally the effectiveness of the solution must be demonstrable through an iterative evaluation of the designed artifact(s). The research process can be seen in Figure 2.

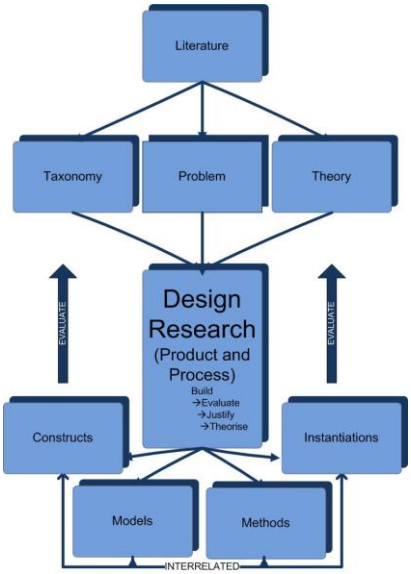

Figure 3. Design research framework (terminology from March and Smith 1995).

The design research process presented in this paper, and depicted in the diagram above, is methodologically based on and adapted from the approach described by Nunamaker et al. [1991] and the guidelines presented by Hevner et al. [2004]. The research outputs are also described according to March and Smith's [1995] terminology for design research. Systematic analysis of a number of design artifacts in unison provides additional rigor when investigating artificial, future environments (e.g. synthesising existing business applications alongside a number of developed artefacts). Artefacts developed in this research include the discovery software, the ontological models and experimental result sets. The can be considered in unison when evaluating the effectiveness of the approach.

The previous section highlighted an existing gaps and limitations in the current literature. The following section will address this gap, confirming relevance of the problem being investigated, both in the ontological modelling and software artefacts. The resulting artefacts are now presented.

5.0 Proximity Based Semantic Discovery

In order to explore how ontology can be utilised in the discovery of appropriate services, a web based discovery platform has been developed. Initial requirement for the platforms were: (1) An ability to process SPARQL queries, (2) A light web interface (allowing http connections from small devices) and (3) Support for proximity definition and querying. The service discovery is undertaken by a web service based system – see Figure 4 – that is built using Servlet technology running on a Jetty server (http://www.mortbay.org/jetty/). Emulators were used to test the system, calling the service (via a http request) and returning a list of services within close proximity of a specified object. It is assumed that the calling device recognises local objects via Bluetooth, WIFI triangulation, GPS or other. Proximity to an identified object is part of the search request. The system returns a list of available services within the desired proximity of the object. The benefits of object proximity (as opposed to location) in service discovery are to be obtained when dealing with moving objects (between moving and static or two moving objects).

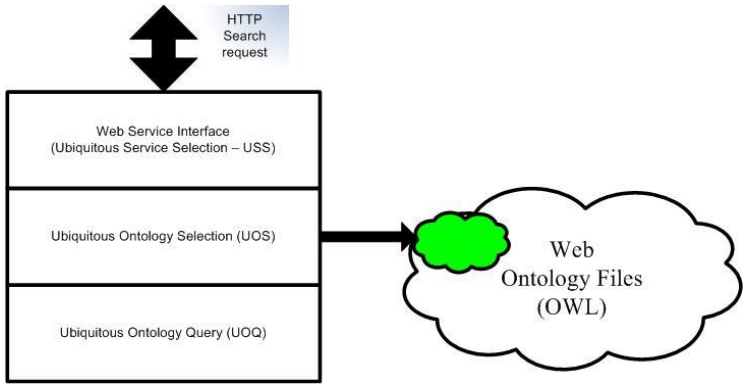

Figure 4: Service Discovery

Once a search request is received by the USS servlet, the discovery system identifies which ontology are to be searched. Ontology describing services (realised in Notation 3 http://www.w3.org/DesignIssues/Notation3). The use of N3 and OWL web ontology will be discussed later. Ontology selection is carried out by UOS. Once suitable ontology are chosen the requested SPARQL query is executed (by UOG) and utilises ARQ (http://jena.sourceforge.net/ARQ/) a query processor from Hewlett Packard's Semantic Web activity. Before execution, the query is expanded to include classes in a specified branch (using an extended syntax that includes "*expand(class)*" as part of the query). This functionality is not covered in this paper, but is used to overcome SPARQL's lack of reasoning functionality with regard to OWL. It should be noted that the same limitation however improves the performance of SPARQL.

The ontology (see Figure 4 and 5) are used by the semantic search (UOQ) to identify appropriate services and includes a proximity clause that is used as a means to services in the local proximity. Two ontology files being used are: (1) domain ontology (Figure 4) and (2) service ontology (Figure5). The domain ontology formed in OWL resulted from an ontology engineering process [Bell, et al. 2007] that interprets existing software components identifying classes and their properties. This process resulting in three distinct categories of concept: (1) *Domain concepts* that specify the language used in the banking system being interpreted, (2) *UBIS concepts* that describe the software service in terms of the UBIS architecture presented earlier and (3) *Proximity concepts* that provide a language for describing when a service has utility with respect to its proximity. The service ontology realised in N3 is a list of triples that describe the service (using concepts present in the domain ontology). Software and proximity specific concepts are added to the domain ontology (see Figure 5). When describing the service, the place it resides and its proximity to a specific object can be specified. It should be noted that the object (NearObject) could be a static or movable object (e.g. a PC or a trader). For example, if a software service exists and only has utility within 2 metres of a particular PC, the NearObject could have a value of "PC" and Proximity of 2. Similarly, the NearObject could be a moveable object "aTrader" or a specific object described in the domain ontology.

The ontology engineering process typically results in a number of class hierarchies (e.g. type hierarchies) as different software services are interpreted. This provides a mechanism for searching to move beyond the syntactic and expand branches of the ontology for comparison where appropriate. This expansion is possible for both

subject and object query terms. For example, a bond payment service may return a calculated coupon; the same service could be described using C*oupon* or the more specific *CashCoupon* (see Figure 5). Less specific queries would return both descriptions.

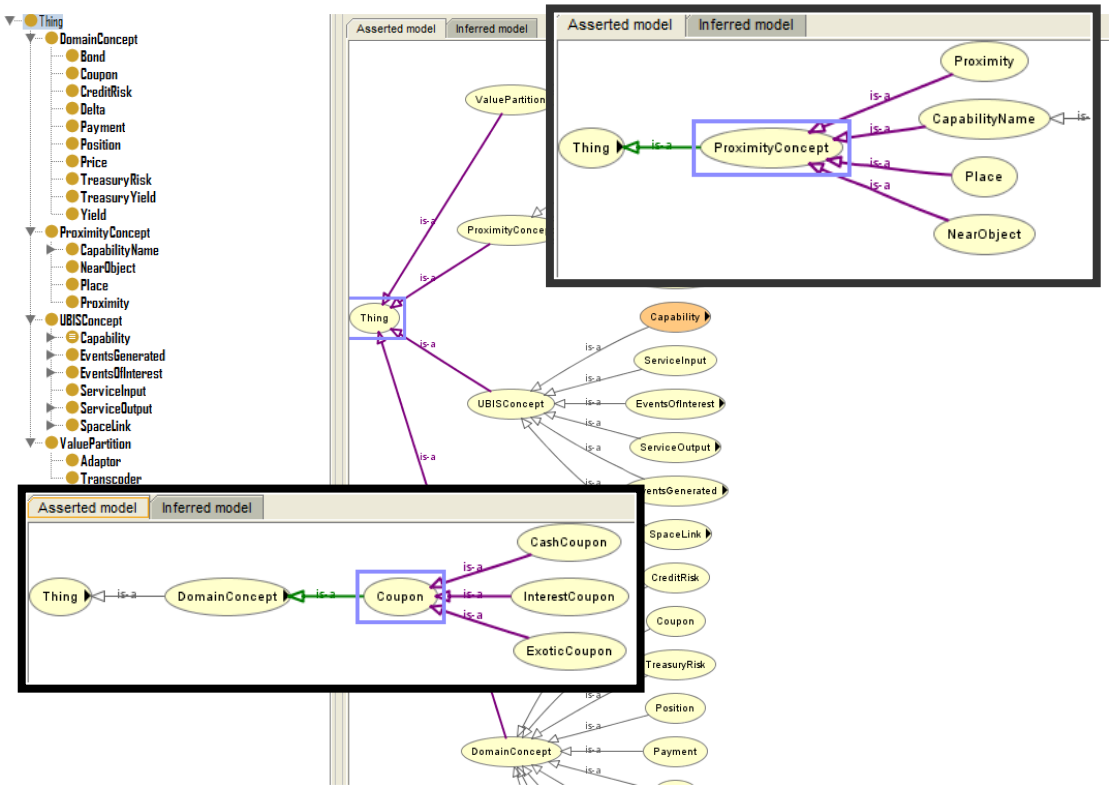

Figure 5: Proximity Enhanced Domain Ontology (OWL)

With a computer usable description of the domain (that includes language for specifying physical object and proximity) it is now time to describe the software services themselves. Initially, the description was carried out using OWL-S (a service description language that uses OWL http://www.w3.org/Submission/OWL-S/). This approach superseded in later iterations for two reasons: (1) heterogeneity of the environment, devices and software make a homogeneous description limiting and (2) natural description of software service by the user (or developer) is more likely with some descriptive freedom that can be achieved using relatively simplistic approaches. Consequently, Notation 3 (N3) triples were chosen and used as a more effective means to describe services. This allowed the description to have a grounding in the domain language directly and not indirectly via a service ontology.

The description (summarised in Figure 6) simply involves the specification of subject, predicate and object triples. As seen in the snippet, the subject (the service name) is described once before a predicate-object list. Although only proximity related triples are shown any description can be added – using verbs or objects defined an external ontology (ubi in this case). Application capability concepts could also replace the string based ubi:CapabilityName etc.

```
@prefix ubi:    <http://www.fluidity.org.uk/ontology/ontology/UBIS.owl#> .
@prefix ns:     <http://example.org/ns#> .
@prefix :       <http://example.org/services/> .

:getCashSummary
        ubi:CapabilityName      "getCashSummary" ;
        ubi:NearObject          "MiddleOffice" ;
        ubi:Proximity           30 ;
        ubi:Place               "Bank1" .
:getRate
        ubi:CapabilityName      "getRate" ;
        ubi:NearObject          "TradingFloor" ;
        ubi:Proximity           70 ;
        ubi:Place               "Bank1" .
:InsertTrade
        ubi:CapabilityName      "InsertTrade" ;
        ubi:NearObject          "RatesTradingDesk" ;
        ubi:Proximity           5 ;
        ubi:Place               "Bank1" .
:MirrorTrade
        ubi:CapabilityName      "MirrorTrade" ;
        ubi:NearObject          "RatesTradingDesk" ;
        ubi:Proximity           5 ;
        ubi:Place               "Bank1" .
```

Figure 6: Service Ontology (N3)

With a usable description of the domain and available services, we are now able to experiment to determine its effectiveness alongside SPARQL.

6.0 Preliminary Experimentation and Results

In order to test the effectiveness of the approach a number a service description data sets (Table 1) and queries (Figures 7 and 8) were produced. Service description data sets of various sizes were produced. Initially this involved the description of services from three banks (78 services in total). Larger data sets were produced by replicating data by factors of 3, 10 and 100.

Table 1: Experimental Data Sets

DataSet Name	#Services	#Lines	Description
Small – S	78	395	N3 List of 78 Services
Medium – M	234	1,175	S replicated 3 times (service name changed)
Large – L	780	3,905	S replicated 10 times (service name changed)
Extra Large – XL	7,800	39,005	S replicated 100 times (service name changed)

Two queries were used to experiment with the OWL domain ontology and the test service ontology in N3. The first focused on proximity selection and the second on wildcard searching. Wildcard searching was chosen as a natural approach that a user will likely adopt (e.g. services about *"risk").*

```
PREFIX ubi:     <http://www.fluidity.org.uk/ontology/ontology/UBIS.owl#>
SELECT ?CapabilityName ?Place
FROM <http://www.fluidity.org.uk/ontology/ontology/SERVICES10.n3>
WHERE
{
   ?loc   ubi:Place  ::location .
   ?loc   ubi:NearObject ::object .
   ?loc   ubi:Place  ?Place .
   ?loc   ubi:CapabilityName ?CapabilityName .
   ?loc   ubi:Proximity ?Proximity .
   FILTER (?Proximity < ::proximity) .
}
```

Figure 7: SPARQL Query 1 – Proximity (Q1)

The ontology are referenced both as a PREFIX (informing that this language will be used) and then in the FROM clause (identifying which service ontology will be queried). This service list was fixed in this experiment and not created on the fly as specified earlier.

```
PREFIX ubi:      <http://www.fluidity.org.uk/ontology/ontology/UBIS.owl#>
SELECT ?CapabilityName ?Place
FROM <http://www.fluidity.org.uk/ontology/ontology/SERVICES.n3>
WHERE
{
   ?loc   ubi:Place ::location .
   ?loc   ubi:NearObject ::object .
   ?loc   ubi:Place  ?Place .
   ?loc   ubi:CapabilityName ?CapabilityName .
   ?loc   ubi:Proximity ?Proximity .
   FILTER regex(str(?CapabilityName), ::wildcard)

}
```

Figure 8: SPARQL Query 2 – Wildcard (Q2)

The query execution times are shown in Figure 9. Each query was executed against each data set 10 times and the average was taken (although little deviation was observed). Times were logged prior to executing the SPARQL in ARQ and The ontology was placed on a remote web server and was accessed by the discovery service over a 2mbps wide area network. The results are promising, with sub three seconds being viable in a business environment. The XL dataset (~39 thousand triples describing services) is nearing the limit. In a real world context, one of the banks studied had an inventory of 3,000 application which would likely result in a service description list far exceeding the performance limits (XL being 300 services). This strongly supports the approach taken where service lists (N3 ontology) are generated for only those services that are available near to a specific object. Caching of proximity filtered service lists will further enhance performance. The wildcard query performed equally well – with faster performance likely due to a smaller result set being returned.

An ontological approach that focuses on domain ontology has a number of advantages over existing syntactic approaches allowing more search expressiveness. Users are able to use different terms (e.g. bond and hedge) and still receive the same results, due to the sub-class relations in the domain ontology. This functionality was an extension to the SPARQL language. The use of Web Ontology and SPARQL is also light weight – accessible from a simple http request (http://hostname/UBISServet?query=”sparql query”).

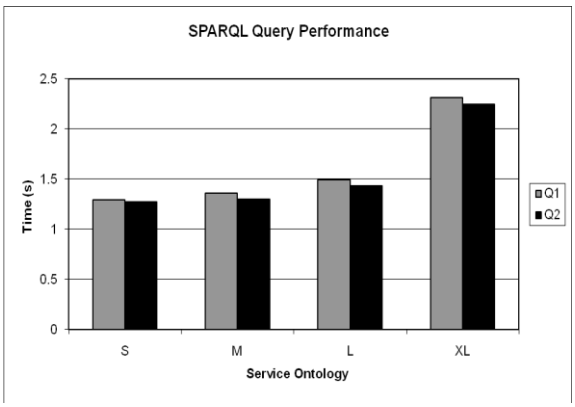

Figure 9: Query Performance

Although the results are promising, additional research is required in the following areas:

- Investigating variation in the size and shape of the domain OWL ontology.
- Overcoming limitations in SPARQL, specifically the lack of reasoning functionality (although a simple workaround has be adopted in the UBIS discovery system).
- Investgating how additional service characteristics can be integrated into the ontology (even originating from devices in use).
- Exploring more complex knowledge topologies – with domain, service and sensor ontologies across the Web (including some of the ontology efforts mentioned earlier).
- Adding richer semantics to proxity.
- Searching for, ranking and constructing piplelines of services in response to a particular requirement (e.g. US risk measures on a IP Phone display).

7.0 Conclusion

This paper proposes a novel approach to semantic search in a ubiquitous business environment, built in part to investigate the utility and effectiveness of semantic web tools in this area. Central to this approach is the use of OWL and N3/RDF ontology as a means to describe a domain and the services residing within it – focusing on describing services that have utility when within proximity of a particular object. Design research methods have been adopted that resulted in discovery software, ontology and experimental results. A web based discovery platform was developed that filters appropriate service ontology before using SPARQL to query and select appropriate services. An experiment is presented that supports the approach, executing queries against a range of service ontology of different sizes, with reasonable performance achieved for the largest service ontology. Semantic web tools are found to be effective in a ubiquitous information systems environment (with its flexible expression) and that the proximity based approach that filters ontology before searching with SPARQL is a necessity for organisations with a large number of dynamics services, devices and users.

References

Bell, D. 2009. Emergent Application Integration of Ubiquitous Information Systems (UBIS). ACM ICPS 2009 – Workshop on Services in Pervasive Environments, 2009.

Bell, D., Cesare, S., Lycett, M., Iacovelli, N. and Merico, A. 2007. A Framework for Deriving Semantic Web Services. *Information System Frontiers* 9, 69-84.

Bernstein, P.A. 1996, "Middleware: a model for distributed system services", *Communications of the ACM*, vol. 39, no. 2, pp. 86-98.

Brown, J.S. and Duguid, P. 1991. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization science* 40-57.

Helal, S. 2002. Standards for service discovery and delivery. *Pervasive Computing, IEEE* 1, 95-100.

Hevner, A., March, S., Park, J. & Ram, S. 2004, "Design Science in Information Systems Research", *MIS Quarterly*, vol. 28, no. 1.

March, S. & Smith, G. 1995, "Design and Natural Science Research on Information Technology", *Decision Support Systems,* vol. 15, pp. 251-266.

Meier, R., Cahill, V., Nedos, A. and Clarke, S. 2005. Proximity-based service discovery in mobile ad hoc networks. In *DAIS*, Anonymous Springer, , 115–129.

Nunamaker, J., Chen, M. & Purdin, T. 1991, "System Development in Information Systems Research", *Journal of Managemennt Information Systems,* vol. 7, no. 3, pp. 89-106.

Richard, G.G. 2000. Service advertisement and discovery: enabling universal device cooperation. *IEEE Internet Computing* 18-26.

Saha, D. & Mukherjee, A. 2003, "Pervasive computing: a paradigm for the 21st century", *Computer,* vol. 36, no. 3, pp. 25-31.

Weiser, M. 1993, "Hot topics-ubiquitous computing", *Computer,* vol. 26, no. 10, pp. 71-72.

Weiser, M. 1991, "The Computer for the 21th Century", *Scientific American,* vol. 265, no. 3, pp. 94-101.

Weisner, M. and Brown, J.S. 1996. Designing Calm Technology. *PowerGrid Journal* 1, 94-100.