UNIVERSITY OF SOUTHAMPTON

Semantic Description and Matching of Services for Pervasive Environments

by

Ayomi Bandara

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

July 2008

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

SEMANTIC DESCRIPTION AND MATCHING OF SERVICES FOR PERVASIVE ENVIRONMENTS

by Ayomi Bandara

With the evolution of the World Wide Web and the advancement of the electronic world, the diversity of available services is increasing rapidly.This raises new demands for the efficient discovery and location of heterogeneous services and resources in dynamically changing environments. The traditional approaches for service discovery such as UDDI, Salutation, SLP etc. characterise the services by using predefined service categories and fixed attribute value pairs and the matching techniques in these approaches are limited to syntactic comparisons based on attributes or interfaces. More recently with the popularity of Semantic Web technologies, there has been an increased interest in the application of reasoning mechanisms to support discovery and matching. These approaches provide important directions in overcoming the limitations present in the traditional approaches to service discovery. However, these still have limitations and have overlooked issues that need to be addressed; particularly these approaches do not have an effective ranking criterion to facilitate the ordering of the potential matches, according to their suitability to satisfy the request under concern.

This thesis presents a semantic matching framework to facilitate effective discovery of device based services in pervasive environments. This offers a ranking mechanism that will order the available services in the order of their suitability and also considers priorities placed on individual requirements in a request during the matching process. The proposed approach has been implemented in a pervasive scenario for matching device-based services. The Device Ontology which has been developed as part of this research, has been used to describe the devices and their services. The retrieval effectiveness of this semantic matching approach has been formally investigated through the use of human participant studies and the experimental results have indicated that the results correlate well with human perception. The performance of the solution has also been evaluated, to explore the effects of employing reasoning mechanisms on the efficiency of the matching process. Specifically the scalability of the solution has been investigated with respect to the request size and the number of advertisements involved in matching.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Ayomi Bandara, declare that the thesis entitled *Semantic Description and Matching of Services for Pervasive Environments* and the work presented in this thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published in a number of conferences and workshops (see Section 1.3 for a list).

Ayomi Bandara

Acknowledgements

I would like to extend my sincere thanks to my supervisors Dr. Terry Payne and Prof. David De Roure for all the support, guidance and encouragement I received throughout my PhD. I would also like to thank Dr. Nick Gibbins and Dr. Valentina Tamma for providing insightful reviews and feedback on my research. My appreciation goes to Jade Lodwidge, Nicky Harding and the rest of the administration staff, for all their help during my years at Southampton.

I am grateful for the stimulating conversations, motivation and encouragement I received from my colleagues at the University of Southampton, particularly Jing Zhou, Roxana Belecheanu, Martin Szomszor, Nishan Karunatillake, Sebastian Stein and Jamie Lawrence. My sincere appreciation also goes to Wasara Rodhetbhai and Aniza Othman for their good company and friendship.

I am thankful for the funding and support provided by the Telecommunications Research Laboratory of Toshiba Research Europe Ltd. and to my advisers at Toshiba, Dr. Gary Clemo and Dr. Tim Lewis for their support throughout my PhD. I would also like to acknowledge the partial funding received through the Semantic Media project: grant EP/C010078/1 from the UK Engineering and Physical Sciences Research Council.

Last but not least, I am extremely grateful to my parents, brothers and husband for their love, support and encouragement, without which I would not have been able to come this far.

To my parents, brothers Kapila & Sidath,
husband Samitha and
daughter Samindee

Chapter 1

Introduction

With the recent advances of the electronic world, the number and diversity of devices being used in home and office is increasing rapidly; these can vary, from headsets and printers to GPS devices, mobile phones, PDAs and laptops. Thus, hardware services and the platforms on which they run are becoming increasingly heterogeneous. Also, the advent of wireless technologies has simplified the task of physically connecting and communicating between devices, and facilitates the creation of personal area networks and mobile ad-hoc networks. This has resulted in the possibility of combining devices to yield new capabilities, such as sending text messages from a PDA through a mobile phone, or using an MP3 music player to store and browse photos directly from a camera. With these trends in technology, personal wireless devices are becoming increasingly popular and have resulted in the proliferation of these gadgets in everyday life. This has raised new demands for the efficient discovery of heterogeneous devices and services (to suit a user's preferences and context) in dynamically changing environments.

Further, with the evolution of the World Wide Web, thousands of web services of various forms and different complexities are becoming available and these originate from diverse sources. These may vary from a simple service that provides a weather forecast for a given location, to services that will arrange a complete holiday to suit user preferences. As it stands today, users/agents will have to browse through the available services in order to select the appropriate one that matches their needs. With the high volume of services available, a manual search for suitable services can become a cumbersome and infeasible task. Hence, a facility that will automatically discover services that have appropriate capabilities and properties to suit a user's request is increasingly becoming a necessity in the web services arena.

Whether the services are in the form of software programs or hardware devices, service discovery (i.e. the ability to find and subsequently utilise a service based on some published description of its functionality and operational parameters) is an essential component for assisting both humans or software in finding and selecting appropriate

services in both service-oriented computing and multi-agent systems alike. Various approaches have been proposed to facilitate the discovery and location of services, such as UDDI [129], UPnP [134], SLP[118] and Bluetooth[15]. However, such approaches traditionally assume a description that uses key-words, attributes and interfaces and therefore resource discovery depends on matching mechanisms based on these syntactic descriptions.Thus they can only support exact syntactic matches (i.e. where service descriptions use the same tokens as those that appear in the service request), due to the fact that attributes will either be equivalent or not. Although this approach has been used within various existing industry-driven solutions, where a set, or taxonomy of keywords or service classes can be agreed beforehand, it necessitates that all services and requests generated adhere to the agreed taxonomy.

However, the ability to satisfy inexact, approximate matches may be necessary in many scenarios, when a pragmatic solution is required in the absence of the ideal solution. For example, a user may want to find a computer that has Linux operating system, Pentium4 processor and has 100GB hard disk capacity. In the absence of an exact match to satisfy these requirements, one has to determine how the available resources can be matched and approximated against the request. Handling such queries requires a degree of interpretation, based on the context and available services, and cannot be effectively handled by simple key word searches. Thus, it will be necessary to discover services by comparing the semantic content of the service request and advertisements. This is one of the main challenges faced by semantic service discovery mechanisms.

Semantic service discovery has two main research issues, namely *Service Description* and *Service Matching*. Service description is important since, in order for the services to be located in response to a relevant request; those services should be sufficiently described, in a way that the matchmaking process will understand. Secondly, to find services that are appropriate to a service request, an effective matchmaking mechanism must exist.

A *Service Description* is a specification of the functional and non-functional capabilities and characteristics of a service. A flexible and expressive approach is necessary to effectively describe various aspects of service advertisements and service requests, including the functionality and capabilities, in such a way as to be interpreted and understood by service consumers. Since semantic service discovery needs to go beyond simple keyword search and attribute-value matching, the description of services must be rich enough to support this.

Service Matching is the process of comparing the service request against the available service advertisements and determining which service best satisfies the request. An efficient and scalable matching process should exist for effective discovery of services. The functionality and capabilities of different services should be reasoned against the context and preferences of a user/agent to come out with the most appropriate set of matches.

The adoption of Semantic Web languages and inference techniques to overcome the limitations of the traditional service description and discovery solutions, has gained considerable attention in the recent years. Several emerging frameworks for web service description (e.g. OWL-S [90], WSMO[38], SAWSDL [115]) follows a semantic web line of research whereby service descriptions can be defined in terms of concepts or instances with formally defined semantics. The advantage of such frameworks include the ability to extend and adapt the vocabulary used to describe services, to utilise existing concepts defined in alternate ontologies, and to harness the inferential benefits of logical reasoning over such descriptions. Such benefits are necessary within dynamic, evolving environments, where no assumptions can be made about the availability of any given service, nor can such services be expected to adhere to any given standard.

In the initial stages of this research, current approaches to device description and their ability to facilitate semantic discovery of devices in pervasive environments was investigated. It was observed that although several device description proposals have been published, no collective effort has emerged that supports the development of a formal framework for describing devices with the specific aims of facilitating semantic service discovery. Hence as a part of this research, the Device Ontology [10] has been developed that provides a general framework for describing devices and the services that they provide. In order to be able to perform reasoning over such descriptions, the ontology has been described using OWL-DL, for which several reasoning tools are currently available, such as Pellet [104] and Racer [64].

There have been several research efforts [25, 7, 82, 101, 57] that use Semantic Web technologies to improve service discovery mechanisms on top of what could be achieved with keyword and attribute based syntactic searches. These use ontologies for the description of services and description logics and other logical reasoning methods to support inferencing during the service matching phase. These efforts focus on discovery and matching of services in a variety of domains; such as web services, e-commerce, grid environments and pervasive environments. In general, such approaches provide important directions in overcoming the limitations present in the traditional discovery techniques, in order to come up with pragmatic solutions to meet the challenges and requirements in the service discovery arena.

However, the existing semantic matching solutions still have limitations and lack several desirable characteristics that should be present in a pragmatic approach for resource matching. Particularly these approaches lack an effective criterion to approximate the available services and to rank them in the order of their suitability to satisfy a request. Also, most existing semantic matchers place equal importance on all the constraints or requirements in a request during the matching process. However, in many practical scenarios certain requirements will be more important than others and hence assuming equal weighting on all the requirements will not reflect the context dependencies and subjective preferences in the match results produced.

By analysing the limitations present in the existing semantic matching efforts and considering the practical concerns in service discovery scenarios in general, we have devised a semantic matching approach to facilitate the effective discovery of resources. This thesis presents the proposed semantic matching solution, which provides a pragmatic approach to approximate and judge the suitability of an available resource, by semantically comparing its properties against the requirements of the request under concern. This approximating criterion provides a heuristic that can be used to rank the available resources according to their suitability to satisfy a resource request. A ranking facility in turn will aid the resource seekers to gain an understanding of the appropriateness and the order in which they should consider the available resources. The proposed approach also facilitates the description of priorities or weights on the individual requirements in a request and takes them into account during the matching process.

We have implemented the proposed semantic matching approach in a pervasive computing scenario to match device-based services. The Device Ontology, which has been developed earlier in this research, has been used to facilitate the description of the resources in the pervasive environment. This implementation has been used to systematically evaluate the semantic matcher in terms of its retrieval effectiveness and scalability and we present the empirical results obtained.

1.1 Thesis Contributions

This thesis presents a semantic framework to facilitate the effective description and discovery of resources in pervasive environments. The proposed approach addresses several limitations present in the existing semantic matching approaches. Specifically; it provides a ranking mechanism to order the available resources according to their suitability, it presents an effective methodology to approximate the available resources in relation to the request concerned and it incorporates priorities or weights on individual requirements of a request in the resource description and matching.

The core contributions of this research could be summarised as below:

- The Device Ontology: This provides a general framework that describes devices and their services in a flexible and expressive way, to aid the description of a variety of devices and to facilitate effective semantic discovery of services. It is important to note that the Device Ontology describes devices from a 'usage' point of view, where the characteristics of the device that are relevant to a user seeking to utilise the device are described. It does not describe the device from a 'product design' point of view, as in the case of the ontological framework provided in [76] where the internal dynamics of a device are modelled. The ontology has been formally evaluated by using the current best practice approaches for ontology evaluation

[55], specifically the OntoClean method [63] and ODEval[54]. The validation case studies provide an assessment of the usefulness and usability of the ontology, and have shown how the Device Ontology coupled with a DL reasoner can dramatically improve the results of service discovery and matching.

- The Semantic Matching Framework: This presents an approach to semantically compare the preferences of a request against the properties of available services and provides a ranked list of most suitable services. The rank of a service advertisement indicates the appropriateness of a service to satisfy a given request and thus can aid the users of the matching system, to identify the order in which they should consider the returned matches in the absence of an exact match. This matching approach also takes into account the priorities or weights (that indicate the relative importance) placed on the individual requirements of a request during the ranking process; therefore the matcher results produced are more suitable for the context involved, and are better able to capture the subjective preferences of users. The proposed approach has been implemented in a pervasive context to match device-based services and the Device Ontology has been used to describe the devices and their services in this implementation.

- Evaluation of the Semantic Matching Approach: An empirical evaluation of the proposed semantic matching approach has been carried out, to evaluate the solution both in terms of effectiveness and efficiency. The retrieval effectiveness of the solution has been investigated through the use of a human participant study, which allows us to compare the matcher results obtained with human judgement. The study includes a number of experiments, to evaluate the solution from several perspectives and to explore the benefits of the features provided by the matching solution. The performance of the proposed matching approach has been evaluated to investigate the effects of using logical reasoning mechanisms in the matching process on its efficiency; specifically the scalability of the matching solution has been investigated with respect to the size of the resource request involved and the number of advertisements matched.

1.2 Structure of the Thesis

The remaining of this document is organised as follows:

Chapter 2 introduces the background in the service discovery area; it explores the traditional approaches for service description and discovery, the Semantic Web technologies and how it has contributed to the area of service discovery. This also discusses the literature specific to service discovery in the pervasive computing domain and investigates the existing efforts on semantic matching and discusses their limitations.

Chapter 3 presents the the Device Ontology that provides a general framework for describing devices. This is intended to be used as a base ontology, to describe the features and functionalities of a range of devices typically used in pervasive environments. This also discusses the methodology adopted in developing the ontology and the process followed in evaluating the ontology to ensure its formal correctness.

Chapter 4 presents the semantic matching framework which is one of the core contributions of this research. It discusses the motivating factors behind the proposed matcher and the requirements for an effective semantic matching solution. The basis for the proposed approach for semantic matching is analysed and the detailed methodology behind the proposed matching framework is discussed.

Chapter 5 discusses the application of the proposed semantic matching framework in a pervasive computing scenario and presents the prototype implementation of the approach and an experimental analysis of the results.

Chapter 6 provides an empirical evaluation of the proposed semantic matching framework and discusses the results obtained. This explores how retrieval effectiveness of a semantic matching approach can be evaluated and presents the experiments carried out to evaluate this aspect and the conclusions drawn from them. It also discusses the evaluation of the performance of the proposed semantic matching approach with respect to its scalability and efficiency.

Chapter 7 contains the concluding remarks and discusses the possible future directions for this research.

1.3 Publications

During this research, the following work has been published.

- Bandara, A., Payne, T. R., De Roure, D. and Clemo, G. (2004) An Ontological Framework for Semantic Description of Devices (Poster). In *Proceedings of International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan.
- Bandara, A., Payne, T., De Roure, D. and Lewis, T. (2007) A Semantic Approach for Service Matching in Pervasive Environments. Technical Report, School of Electronics & Computer Science, University of Southampton.
- Bandara, A., Payne, T., De Roure, D. and Lewis, T. (2007) A Semantic Approach for Description and Ranked Matching of Services in Pervasive Environments. In *Proceedings of 2nd International Workshop on Applications of Semantic Technologies 2007*, Bremen, Germany.

- Bandara, A., Payne, T., De Roure, D. and Lewis, T. (2007) A Semantic Framework for Priority-based Service Matching in Pervasive Environments. In *Proceedings of 2nd International Workshop on Pervasive Systems*, Vilamoura, Algarve, Portugal.

- Bandara, A., Payne, T., De Roure, D., Gibbins N. and Lewis, T. (2008) A Pragmatic Approach for the Semantic Description and Matching of Pervasive Resources. In *Proceedings of 3rd International Conference on Grid and Pervasive Computing*, Kunming, China.

- Bandara, A., Payne, T., De Roure, D., Gibbins, N. and Lewis, T. (2008) Semantic Resource Matching for Pervasive Environments: The Approach and its Evaluation. Technical Report, School of Electronics & Computer Science, University of Southampton.

Chapter 2

Service Discovery

2.1 Introduction

The word 'Service' can be used in many contexts to refer to a variety of types of services. Priest [105] highlights several uses of the word 'Service' ranging from business services that 'provide a value in some domain' such as provision of an Internet connection, to software entities that 'provide something of value' - the usage that is commonly used in computer science to refer to web services. In pervasive computing environments, the word service can also be used to refer to accessing hardware resources and peripheral devices such as printers.

Service Discovery is the process through which appropriate service(s) are found to potentially satisfy a user/agent's request. There are three distinct, but related components involved in service discovery:

- The language or model used to describe the service (in terms of its functionality, operational parameters and properties) and a service request.
- The discovery architecture used for advertising and querying services.
- The matching or resolution mechanism that compares a service advertisement and a request and determines whether or not a service satisfies a request description.

The design of any of these components has an impact on the others; for example, the expressivity of the service description language determines how complex the matching algorithm can be [100].

The discovery architecture determines how the interactions and communications take place between the parties involved in service discovery; this involves the service provider and requester and may or may not involve a third party or middle agent that facilitates

the discovery. Decker et al. in [39] identifies nine ways in which a middle agent can assist in this communication flow of finding appropriate service requesters/ providers. For example a matchmaker/ yellow pages or directory agent is a middle agent that stores capability advertisements in a registry that can then be queried by requesters. The middle agent matches the request against the stored advertisements and returns the result, a subset of the stored advertisements. The requesters can then contact any provider they wish directly. In contrast a blackboard agent is a middle-agent that keeps track of requests. Requesters post their problems; providers can then query the blackboard agent for events they are capable of handling. In these two different scenarios the matching actually occurs at two different places; in the case of the matchmaker agent, the matching takes place at the middle agent, but when the middle agent is a black board agent the service provider does the matching.

Service matching is the process by which the available service advertisements are compared against the service request and judged for their suitability to satisfy the required properties. The matching mechanism used to compare service advertisement(s) is independent of the communication procedures and protocols used between service providers, middle agents (if applicable) and service requesters. However, the service description language (used to provide a high level description of the service capabilities and service requisites) is of key importance for effective matching to take place; the functionality of a service and all applicable properties and parameters must be expressed in a service description for a proper comparison and matching of advertisements and requests, and the service description language must facilitate this.

Since providers and requesters are often heterogeneous and are unable to understand each other, an appropriate language must be used to describe the service advertisements and requests effectively. Gonzalez-Castillo et al. in [57] identify the following as requirements that should be met by a language to express service descriptions in the context of a matchmaking service.

- ***High degree of flexibility and expressiveness***: This refers to the ability of the language to allow for the advertisements to be composed to varying degrees of complexity and completeness. Depending on the properties of a service that needs to be described, some properties may be expressed with simple attribute-value pairs, some others may need more structuring. Also the service providers may want to describe certain aspects of the service to a great detail but may want to leave certain properties unspecified either because they are not known, not applicable or because they need to be negotiated later.
- ***Support for types and subsumption:*** To allow the matchmaking to provide complex matches based on relationships rather than just string comparison, the language must allow for a type system with subsumption relationships[1].

[1]Subsumption refers to the subconcept/ superconcept relationship between the concepts in a given

- ***Support for data types:*** Attributes such as quantities, dates and prices can be part of a service description. Hence the description language must support data types to facilitate expression and matching of such attributes in the service descriptions.

- ***Express restrictions and constraints:*** Services and requests most often have to be described as conceptual definitions of acceptable instances rather than one single instance of a service. Since such cases are usually described by expressing constraints over certain parameters, the description language must facilitate this.

- ***Semantic level of agreement:*** For the matchmaker to understand and compare different service descriptions, they must share the same semantics. This refers to the requirement of using common ontologies (domain ontologies and reference ontologies) in service descriptions, to attain interoperability between diverse service providers and requesters.

There are various standards, specifications and research efforts that relates to the description and discovery of devices and services. Whilst several service discovery protocols currently exist for discovering devices in open, dynamic environments (e.g. UPnP [134], Bluetooth [15] and Jini[6]), characterisation of these services is limited to static data structures, and agreed, implicitly defined key-word based categorisations. Also matching of requests and advertisements is done at a syntactic level by using key word searches and string comparisons. Although interface description and workflow definition languages have been developed such as WSDL[29] and BPEL4WS[2] for service description, and open discovery mechanisms have been improved such as UDDI[2], these still lack extensibility and any form of declarative semantics.

Frameworks such as, OWL-S [90] and WSMO [38] have emerged which aims to formally describe various aspects of web services in order to facilitate a greater degree of automation in the selection, invocation and interoperation between services. These approaches follow a semantic web line of research and the languages adopted in these frameworks, describe resources and services in terms of objects and relations among them and also allow us to express constraints and restrictions between these objects. Since these languages facilitate the use of logical reasoning over such descriptions, such approaches are better able to meet the challenges of intelligent service selection. Also, several research efforts have produced improved matching techniques to offer more pragmatic, flexible solutions to service discovery, in comparison to traditional methods. More on these topics will be discussed in later sections.

The rest of this chapter is organised as follows. In Section 2.2, we discuss the web services paradigm and the current work and emergent technologies in the area. Specifically we discuss the current standards aimed at web service discovery and description and the

terminology [8].

concept of the Semantic Web. Also we discuss the Semantic Web Service vision which is a result of the diffusion of Semantic Web and web services domains, and some of the current semantic web approaches to web service description. Then, in Section 2.3, we discuss the existing standards and research on discovery and description of device oriented services. In Section 2.4 we discuss the existing approaches to service matching.

2.2 Web services and Semantics

Web Services are defined as self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Web [58]. The Web Services architecture can be seen as an evolution of object-oriented analysis and design, and components geared towards the architecture, design, implementation, and deployment of e-business solutions. Both approaches have been targeted to deal with the complexity of large systems.

The Web Services architecture [58] describes principles for creating dynamic, loosely coupled systems based on services where each component is regarded as a service, encapsulating behaviour and providing the behaviour through an API available for invocation by other services over a network. The suite of programs provided by www.amazon.com that collectively let users buy books is a typical example for a web service.

More recently, the principles of Semantic Web [13] have been brought into the web services domain, in an attempt to formally define web service capabilities and thus improving the quality of existing tasks such as service discovery, invocation, composition and interoperation.

2.2.1 Existing standards for Web Services

Several standards have emerged to support the web services architecture, which includes: protocols to facilitate message exchange; service description languages to describe services in terms of their interface and workflow; and standards to provide registries for description and discovery of web services. WSDL (Web Services Definition Language) which is now popularly used to model and describe web services and UDDI (Universal Description Discovery Integration) which is being used to publish and discover businesses and web services, are discussed below.

Web Services Description Language (WSDL)

WSDL [29] is an XML-based format for describing Web services and how to access them. This describes services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are

described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

As communication protocols and message formats are standardised in the web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

Thus WSDL provides a language to describe the interface of a web service; it specifies how to define the inputs and outputs of the service so that the sender and recipients understand the data being exchanged. When WSDL definitions are used in a UDDI registry searching or matching can be performed by using these WSDL definitions of interface.

Universal Description Discovery and Integration (UDDI)

UDDI [129] is an XML based standard for description and discovery of web services. The goal of UDDI is to provide an Internet wide registry of web services. The UDDI registry accepts information regarding a business including the web services it offers and allows interested parties to perform online searches and downloads of information.

UDDI information consists of three main categories of business information White pages, Yellow pages and Green pages. White pages contain business name and address, contact information, web site name and other classification and identification information. Yellow pages contain Type of business, location and products including various categorisation taxonomies for geographical location, industry type and business ID. Green pages specify technical information about business services such as how to interact with them and business process definitions. Classification information is useful when conducting searches about businesses and services. Businesses can add any number of classifications when registering with a UDDI registry. Classification information includes industry codes, product codes, and geographical codes by referring to classification taxonomies such as North American Industry Classification System -(NAICS) [23] and Universal Standard Products and Services Classification -(UNSPSC) [133].

UDDI supports keyword based search for businesses and services based on their names and the classification information provided. Also a search can be done for services by their type specification, for instance a search could be done for services that adhere to a particular WSDL specification.

2.2.2 Semantic Web

The Semantic Web [13] can be viewed as an extension to the existing Web that aims to make the descriptions of Web resources machine understandable. This will make the current human understandable Web, to one which is also machine processable. This is done by giving the Web resources a well defined meaning by describing and annotating them with a suitable language.

An appropriate language for encoding and describing Web content and resources is a key necessity in realising the Semantic Web. Such a language must have a well defined semantics, be sufficiently expressive to describe the complex relationships and constraints between Web objects, and be amenable to automated manipulation and reasoning. Several languages have built on XML which include RDF [107], RDF Schema [108], DAML+OIL [36], and OWL [140]. DAML+OIL and OWL are Web ontology languages based on description logics [8]. They provide a natural way to describe class and subclass relationships between Web vocabulary, as well as constraints on the relationships between classes and between class instances. A brief description on OWL and description logics will follow in this section.

Description Logics

Description Logics (DL) are a family of logic based knowledge representation formalisms suitable for representing and reasoning about knowledge [8]. This originated from semantic networks and frame-based systems, which represent knowledge in terms of categories of objects and relations among them.

In DL, knowledge is represented in terms of concepts (unary predicates or classes) and roles (binary relations). Complex concepts and roles can be constructed from atomic ones by using constructors provided in the language. A DL knowledge base consists of a TBox (Terminological Knowledge) and an ABox (Assertional Knowledge). TBox contains concept definitions (that introduce names for concepts) and axioms (that define how concepts and roles are related to each other). ABox contains concept assertions and role assertions that specify the instances of concepts and the relations between those instances respectively.

There are a number of description logic languages of varying expressivity; these languages are distinguished by the constructors they provide. For example the constructors available in the minimal (least expressive) description logics language - $\mathcal{AL}$ are: conjunction, disjunction, negation, existential restriction and value restriction. The more expressive languages have additional constructors on top of what is available in $\mathcal{AL}$. But when the expressivity of the language increases, the efficiency of reasoning decreases, hence there is a trade-off between quality of reasoning and the efficiency. Table 2.1 depicts the constructors available in description logic languages and the DL expressiveness that they provide [57]. The different DL variants are named by the expressiveness (in terms

of the constructors) they provide, for example $\mathcal{SHOIN}$ and $\mathcal{SHIQ}(\mathcal{D})$

DL Expressiveness	DL Syntax	Constructor
$\mathcal{ALC}$	A	Concept
	$\top$	Thing
	$\bot$	Nothing
	$(C \subseteq D)$	Subsumption
	$(C \equiv D)$	Equivalence
	R	Object Property
	$(C \sqcap D)$	Conjunction
	$(C \sqcup D)$	Disjunction
	$\neg C$	Negation
	$\forall R.C$	Universal Role Restriction
	$\exists R.C$	Existential Role Restriction
$\mathcal{N}$	$\leq nR.\top$	Non-Qualified Cardinality
	$\geq nR.\top$	
	$= nR.\top$	
$\mathcal{Q}$	$\leq nR.C$	Qualified Cardinality
	$\geq nR.C$	
	$= nR.C$	
$\mathcal{I}$	R^-	Inverse Roles
$\mathcal{H}$	$R \subseteq S$	Subsumption of Roles
	$R \equiv S$	Equivalence of Roles
$\mathcal{O}$	$\{o\}$	Nominals
	$\exists T.\{o\}$	Value Restrictions
$(\mathcal{D})$	T	Datatype Property
	$\forall T.d$	Universal Datatype Restriction
	$\exists T.d$	Existential Datatype Restriction

TABLE 2.1: Description Logic Constructors and Expressiveness

OWL

The Web Ontology Language OWL [11] is a semantic markup language for publishing and sharing ontologies on the World Wide Web. Ontologies are becoming popular and widely used in the semantic web arena to describe knowledge and resources, and provide vocabularies in order to understand shared information. An ontology can be viewed as "a formal, explicit specification of a shared conceptualisation" [61]. Describing resources using an ontology coupled with an appropriate language has an advantage over syntactic forms of describing information, because the former method provides a structure that makes it possible to reason about and derive knowledge from the given descriptions. By using an ontology, the relationships and constraints between entities can be more clearly expressed and it allows us to harness the inferential benefits of logical reasoning over such descriptions. Several ontology languages have been developed in the past decade [53] but the Web Ontology Language (OWL) is widely accepted as being suitable for

the semantic web since it is open, extensible and is compatible with the architecture of the World Wide Web [96].

OWL describes knowledge in terms of classes and properties (similar to concepts and roles in description logics). OWL is based on description logics and hence a DL reasoner can be used to semantically compare between descriptions written in OWL. There are three different flavours of OWL with varying expressivity: OWL Full, OWL DL and OWL Lite. The entire language is called OWL Full. OWL DL is a sublanguage of OWL Full which restricts the way the OWL constructors can be used in order to gain computational efficiency. OWL Lite is a further restriction on the use of the constructors. OWL DL corresponds to $\mathcal{SHOIN}(\mathcal{D})$ description logic variant and OWL Lite corresponds to $\mathcal{SHIF}(\mathcal{D})$. The choice of the sublanguage needs to be determined by considering the expressivity and reasoning support that is required. Reasoning support for ontologies is usually provided by mapping the ontology language to a known logical formalism and by using reasoners that already exist for those formalisms. Since OWL (partially) maps onto description logics, description logics reasoners can be used to reason on the ontologies specified in OWL.

The recently developed OWL 1.1 [102] extends the OWL language with a number of useful features. Specifically it provides: extra Description Logic expressive power (such as qualified cardinality restrictions and disjoint properties); additional syntactic constructs for ease of expressivity; user-defined datatypes which allows to express dataranges such as ≥ 18 [2] and metamodeling constructs. The expressivity of OWL 1.1 corresponds to the $\mathcal{SROIQ}$ description logic.

2.2.3 Semantic Web Services

Although several XML based standards exist to support the description and discovery of web services (such as WSDL and UDDI), they cannot properly represent what the service does (usually termed as the semantics of the service) and allow only simple service discovery based on business and service types. Therefore they offer only limited support in realising the challenges of automated discovery, invocation, interoperation and composition. To allow for the automatic discovery of available services and to facilitate interoperability between services that are not pre-designed to work together, the functionality and properties of services must be formally defined so that they can be unambiguously interpreted by computers. This is the founding principle behind the Semantic Web Services [91]. If the functionality and properties of a service is described appropriately, the service selection mechanism has all the necessary information to reason over and select the appropriate services accordingly.

[2] The lack of the ability to provide such expressivity was considered as a major limitation in the OWL language [99, 98]

The OWL-S and WSMO frameworks have emerged in support of the vision of Semantic Web Services, and have developed ontologies to formally describe web services with the aim of facilitating a greater degree of automation in the discovery and interoperation of services. We briefly discuss these frameworks in the sections that follow.

OWL-S

OWL-S [90] is an ontology based on OWL which aims to provide web service developers with a core set of mark-up language constructs to describe the properties and functionality of a service. The OWL-S ontology has three sub-ontologies, namely Service Profile, Service Model and Service Grounding. The Service Profile contains information about; the provider of the service, the function performed by the service and the characteristics of the service. The functional description of the service is expressed in terms of the transformation produced by the service. Specifically, it specifies the inputs required and the outputs generated; furthermore, since a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile describes the preconditions required by the service and the expected effects that result from the execution of the service. The characteristics of a service includes service parameters used to describe features and properties of the service; such as quality rating of the service and service category within a certain taxonomy (such as the United Nations Standard Products and Services Code - UNSPSC [133]).

The Service Model describes the process taking place when the service is executed. For services composed of several processes (composite services) the service model could be used to coordinate activities among the processes and to monitor the execution of the service. Service Grounding describes the details of how to access the service. Typically the grounding will specify a communication protocol, message formats and service-specific details such as port numbers used in contacting the service. From a service discovery perspective, what is important is the information given in the Service Profile, this acts as the service description in finding appropriate services to match a given request.

Although describing a service in terms of its inputs, outputs, preconditions and effects (IOPEs) provides a good approach to describe the functionality of a service, its effectiveness depends on how sufficiently the actual IOPEs are described by the writer of the service profile. Otherwise, the service functionality cannot be understood properly and may lead to improper matching of services. For example there could be two 'Room Booking' services, one that books any room in a hotel given the requirements, and another that books luxury rooms in a hotel given the requirements. Both these services can have the same IOPEs (Input credit card, Output booking confirmation, Precondition valid credit card, Effect credit card charged, room booked) thus leading to a misunderstanding that both these services provide the same functionality when they actually do not. It is possible to differentiate these two by changing the Effect of the second service to

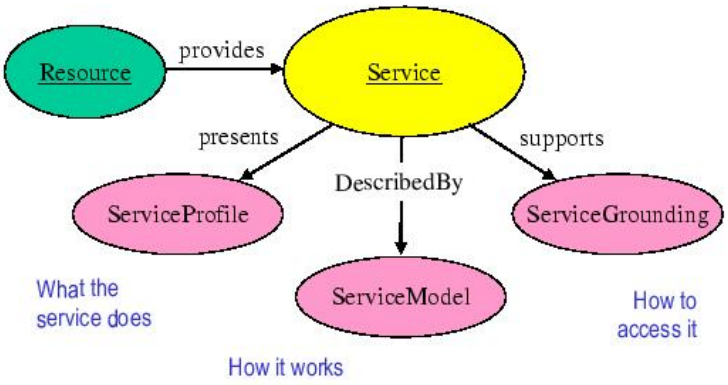

FIGURE 2.1: Top level of the OWL-S service ontology

'luxury room booked': but this means that the effectiveness of the description depends on how well the description of the IOPEs is formulated. Also the OWL-S approach does not have provisions to describe priorities of preferences in a service request or for the specification for mandatory/ optional parameters.

WSMF/WSMO

The Web Service Modeling Ontology (WSMO) [38] and WSMF (Web Service Modelling Framework)[44] provide an ontology and a conceptual model which aims to describe various aspects related to a Semantic Web Service. It is based on the principles of strong decoupling among the components involved in e-commerce applications and strong mediation services enabling anyone to communicate with everyone else in a scalable manner. The language used in WSMO is WSML[37] which is based on F-Logic [3].

WSMF consists of four different main elements for describing semantic Web Services:

- Ontologies: Ontologies to be used in the other elements of the WSMF specifications, thereby enabling reuse of terminology and interoperability between components referring to the same ontology.
- Goal repositories: Goal repositories specify objectives the client may have when he consults a web service. Goal specifications consists of preconditions (what a web service expects in order to provide the service) and post-conditions (what a web service returns in response to its input). These are separate from web service descriptions since there can be an n:m mapping between them. (Same service can provide different goals and different web services can satisfy the same goal.)
- Web service descriptions: WSMF distinguishes between the external visible description (interface) and the internal description of a web service. Web services are described as black boxes and essentially contain web service name, pre and post conditions, input and output data and error data.

- Mediators: Mediators are needed to deal with heterogeneities of data structures, business logics, message exchange protocols and service invocation.

As with OWL-S the functionality of a service is represented in terms of its inputs, outputs, preconditions and post-conditions. Hence the same problem of functionality representation lies here as well the correct and sufficient representation of functionality will depend on how these inputs, outputs, pre and post conditions are formulated in the service description.

2.3 Device Oriented Service Description and Discovery

There are several device description and discovery solutions that have been developed to date, that include mechanisms to support:

1. dynamic discovery of available devices and their services within some network, based on a given search criteria;
2. the provision of descriptions of available devices or services, which can then be browsed (and a suitable service subsequently chosen), or used by a third party to facilitate service requests.

The provision of service listings is typical of infrastructures where human users participate in the selection of services, by inspecting the description of each service and making a selection out of the available services. This mechanism is also used to inspect the services provided by some device (as in the case of UPnP [134] and the Bluetooth SDP [15]), so that a client can determine, select, and utilise the right service with the desired properties. In most cases, however, the discovery mechanisms used are typically based on a matchmaker or yellow pages model [103], where the services are advertised with a registry, and requests for service types are submitted and resolved using a keyword-based matching mechanism. They primarily differ in how services are described (in terms of vocabulary and structure), and how they are subsequently utilised (e.g. invoked).

Salutation [111] and the Service Location Protocol (SLP) [118] are examples of simple service discovery mechanisms that describe services using a service template or record format, which specifies the attributes of a service and their default values. For example, in Salutation, this format includes the service type such as “print” and attributes such as “colour”. Service requests are submitted to a directory service, which then matches these requests syntactically with the service descriptions, based on the type or attributes of the service as specified in the service template. The directory agent then returns a reference to the service, such as the service address. Salutation also returns a personality Profile, a description of the service and its interface to facilitate service invocation.

Jini [6] on the other hand uses a class or interface based service discovery mechanism, whereby a mobile thin-client is registered with a discovery service, and then used by the client to invoke the service (and therefore eliminating the need for any agreement on communication protocol). The capabilities, properties, and interface of a service are constrained by the global class definition of the type of service being implemented (e.g. an object representing a printing service on a printer, implements the printer interface using a printer class), but include instance specific details, such as device location. The class definitions and attributes of the service classes are defined by either the local administrator or the service designer. To locate a desired service, the client specifies the type and attributes of the required service in a service template, and submits this to the lookup service. The lookup service performs matching by comparing types and attributes of services and the query; the types are matched by comparing interface of the objects using standard Java typing rules and attribute matching is done with simple expressions that use exact (i.e. "string" or token) matching. The matching services are then returned to the client, which can subsequently invoke the service(s).

The Bluetooth Service Discovery Protocol (SDP) [15] is designed to identify services provided by a given device (by querying that device) and does not have a central broker or registry facility as in Salutation or SLP. A 24-bit Class of Device (CoD) field provides information about the service class [59], and major/minor class groupings. For discovery, the Bluetooth SDP uses a simple matching scheme based on search patterns defined by a 128-bit UUID (Universally Unique Identifiers). Clients attempting to discover service(s) must specify the correct service UUID(s) in the discovery requests. The Bluetooth SDP server on each device providing information about services, matches the UUID in incoming discovery requests against existing service UUIDs. SDP also allows service attribute UUIDs to be specified along with service UUIDs. Thus, the server attempts to match against attribute UUIDs if a match on the associated service UUID succeeds.

UPnP [134] has a similar architecture to that of Salutation and SLP. UPnP uses SSDP (Simple Service Discovery Protocol) to discover services and it can work without any central directory service. Services are advertised through local broadcast announcement, which include device-type token, and an IP address or URL, which refers to an XML document providing the description of the device. This description includes vendor-specific, manufacturer information including the model name and number, serial number, manufacturer name and URLs to vendor-specific Web sites. Other links provide information about any embedded devices or services, details about control, eventing, presentation, and other details necessary to facilitate interaction between a client and the device.

Ontologies for Devices and Device-Oriented Services

Although the above mentioned service discovery infrastructures use various means to describe the devices and services, all of these approaches are limited to describing service functionality at a syntactic level or object level, and hence lack the expressivity required

to describe heterogeneous devices and the flexibility required to represent dynamically changing conditions. Also, these description approaches do not have facilities to express constraints and relationships among resources (such as cardinality, transitivity, disjointness, domain and range restrictions) that in turn facilitate inferences to be made from the descriptions. Thus to facilitate inferencing and to make intelligent discovery possible, services must be described using concepts and objects with formally defined semantics. There have been several related efforts that attempt to describe devices using ontologies [25, 7, 49, 24, 27], but each uses different methods for structuring the knowledge, and different languages that vary in expressivity.

The FIPA Device Ontology [49] is a specification by FIPA (Foundation for Intelligent Physical Agents) for describing devices used by hosting agents. The objective of this ontology is to provide information about devices and their capabilities to communicating agents. This information is necessary for purposes such as content adaptation; e.g. when an agent wants to send an image to another agent residing on another device, the former agent may want to know the screen resolution of that device in order to transform the image into a suitable format that could be viewed on the device. The ontology is based on a frame-based structure to describe information about a device a set of classes to represent different properties related to the device description and relationships among them. The ontology contains descriptions of the device's hardware (e.g. CPU, screen and memory), software and connection details. As the FIPA ontology is intended specifically for agent communication, terminal capabilities are described in great detail. Although generic devices that can host agents can be described using this ontology (such as PCs, note books and PDAs), it is not suitable for effectively describing service-specific devices and peripherals, including printers, scanners, cameras and storage devices.

There are many similarities between the FIPA Device Ontology and the W3C's CC/PP (Composite Capabilities/Preferences Profiles) framework [24], an RDF-based framework for describing and managing software and hardware profiles. It includes information on a user agent's capabilities (physical and programmatic); the user's specified preferences within the user agent's set of options; and specific qualities about the user agent that can affect content processing and display, such as physical location. CC/PP is designed to work with a wide variety of web-enabled devices, from PDAs to desktop machines to laptops to WAP phones to phone browsers to web television units to specialised browsers for users with disabilities. CC/PP is developed with the specific intention of facilitating the decision making process of a server, on how to transfer web content to the client device in a suitable format. For example the web content that could be displayed on a WAP phone is different from the content that could be displayed on a note book computer. Therefore the server customises the web content sent to a client by looking at the capabilities and preferences described in its CC/PP profile.

SOUPA [27] is a set of shared ontologies designed to model and support pervasive computing applications. It consists of a set of ontologies that define vocabularies for express-

ing various concepts associated with a pervasive environment, such as Person, Space, Location, Time, Meeting and Device. The current SOUPA device ontology only describes a limited number of characteristics pertaining to a device, as is apparent from the publicly available documentation.

DReggie [25] and Avancha et.al. [7] have proposed frameworks to enhance the simple discovery mechanisms offered in Jini and Bluetooth, by using a Prolog logic reasoning system. These are discussed in more detail in the following section.

2.4 Existing Work on Semantic Service Matching

Service matching, as stated earlier, is the process by which the available services are compared against a given user request and judged for their suitability to satisfy the request. In the traditional service discovery approaches, simple matching techniques are used, where the comparison is done on a syntactic level based on keywords, attributes and types. The disadvantages of such methods include their inability to find approximate matches which deviate from the specified request, and the lack of flexibility on the service description approaches where the providers and requesters are expected to strictly adhere to the agreed standards and taxonomies. In semantic matching, the content of the available services and the request is considered and compared at a semantic level, in terms of the concepts involved, relations and constraints among them and computes the similarity between the request and advertisements, and thereby the ability of the advertisement to satisfy the request.

There are numerous scenarios when matching of available services and a given demand becomes necessary. These varies from e-commerce scenarios [26] where one would want to search for a certain entity that that meets certain preferences (for e.g. finding a hotel room); to web services domain [16] where one needs to find a web service to provide a particular functionality; to grid environments [74] where one looks for computer resources with certain capabilities and to pervasive environments[65] where one looks for a certain device to obtain a certain functionality.

There are a number of recent research efforts proposing various semantic service matching frameworks aimed at overcoming the limitations of syntactic matching. These are targeted at different domains such as web services, grid services and pervasive environments. The existing approaches for semantic service matching can be broadly categorised depending on the basis they use to find the semantic matches. Certain approaches use the subsumption relation[3] between concepts to find semantic matches and certain others define specific matching rules that define when an available service satisfies the request

[3]Subsumption in description logics refers to the subconcept/ superconcept relationship between the concepts in a given terminology [8].

concerned[4]. These approaches will be discussed in Section 2.4.1 and Section 2.4.2. A critical evaluation of these approaches is provided in Section 2.4.4.

2.4.1 Description Logic or Subsumption-based Approaches

The LARKS (Language for Advertisement and Request for Knowledge Sharing) matchmaker is an approach for service matchmaking which was proposed by Sycara et. al. [126, 127]. It provides a combination of syntactic and semantic matching through the use of five different 'filters' of matching. The filters progress from simple text analysis for relevance by using word occurrences, to more sophisticated semantic matching which relies on subtype inference rules and concept subsumption. A combination of different filters can be used to obtain different degrees of partial matching, depending on the accuracy and efficiency requirements of the user/agent. This work distinguishes between exact, plug-in and relaxed match. LARKS is a service description language which describes capabilities and requirements of service advertisers and requesters in terms of the services' inputs, outputs and constraints on them. It also allows us to specify domain knowledge by using a local ontology to describe terms used in LARKS descriptions.

Although this work overcomes the limitations of exact matching to some extent, no ranking is presented other than distinguishing between a relaxed, plug-in and exact match. Though the LARKS description language provides a good base for agent/service descriptions by allowing the description of inputs, outputs and constraints on them, it is not expressive enough to express non-functional properties of a service or priorities among them. Thus, the matchmaking process defined does not make use of non-functional properties or their priorities when matching requests and services.

The OWL-S matchmaker (which is built on the LARKS framework) by Paolucci et. al. [101, 100], introduced a matching algorithm to match service advertisements and requests that are described using OWL-S [90] ontology. It matches outputs and inputs of the service advertisement against the outputs and inputs of the service request. The objective is to provide a flexible semantic match between advertisements and requests. The advertisements are classified into one of four categories (exact, plug-in, subsumes and fail), depending on the degree of similarity between the advertisement and the request. The classes indicate the "degree of match" that are organised along a discreet scale indicating the level of match. Subsumption and sub class relationships among the outputs (inputs) of the advertisement and outputs (inputs) of the request are considered here in order to classify the advertisements into one of these classes. Exact matches are matches with equivalent outputs (inputs); Plug-in matches are matches with outputs (inputs) that subsume the request's outputs (inputs); Subsumes matches have output

[4]There are advantages and disadvantages in both these approaches which will be discussed later in Section 2.4.4

(inputs) that are subsumed by the request's outputs (inputs) and Failed matches are when there is no subsumption relation between the outputs and inputs.

The matching process in this work takes into account only the inputs and outputs of service advertisements and requests and does not consider any other properties/preferences of the advertisement/request. In certain situations it is important to consider the specified property requirements/attributes of a service; for example if a user requests a "web service that gives stock quotes for free", then it is vital that the price attribute is considered when looking for appropriate services. No ranking is performed in the matching process, although the matches are classified into four classes to indicate the 'degree of match' which gives an indication of how 'good' a match is.

The approach developed by Jaeger et.al. [70] uses a four stage matching process to match service descriptions described using DAML-S. First, the inputs of the request are compared against those of the advertisements and a rank (or score) is assigned depending on the subsumption relation between the inputs. Then, the outputs are compared and a rank is assigned, depending on the subsumption relation (as in the first stage). This is followed by a comparison of the service category of the request and advertisement to determine the existence of any subsumption relation with respect to some service-classifying ontology. The final stage allows for any user-defined criteria (for example quality rating) to be checked in the service advertisement. The final rank will be determined by aggregating the ranks/results of the individual four stages. The ranking provided in this work is a useful addition to what is offered by previous matching frameworks for DAML-S in [101] and [82].

The use of DAML+OIL for service descriptions was explored by Gonzalez-Castillo et.al. [57], and the applicability of the description logic reasoners, FaCT [69] and Racer [64] for matching has been investigated. A service description ontology was used in which the functionality of services are characterised by using different classes of services (for example Delivery Service class and Sale Service class) and by specifying restrictions over the properties. Service matching is done by considering the classification hierarchy of the service advertisements. When a request R is considered; the equivalent concepts to R, the super-concepts and sub-concepts of R and sub-concepts of direct super-concepts of R for which the intersection is satisfiable, are returned as matches. Whilst the approach has addressed the satisfiability of queries matched to service descriptions, no form of ranking or classifying of matches is provided here in the matchmaking process.

A focus on DL based reasoning techniques was also conducted by Li et.al. [82, 83], who introduced a framework for service matchmaking which uses a DAML-S based ontology and used reasoning to compare ontology based service descriptions. Racer [64] has been used as the DL reasoner to compute the matches for a given request. The matches are classified into one of its five "degrees of match"; namely Exact, PlugIn, Subsume, Intersection and Disjoint by computing the subsumption relationship of the request

description w.r.t. all the advertisement descriptions.

No ranking of advertisements (according to their suitability to satisfy a request) is available in this work, although some understanding of the appropriateness of the advertisement could be gained by the "degree of match" class that it has been classified into. Also, there is no distinction between mandatory/optional parameters and no allowance for priority assignments to the preferences of the request. Where there are numerical values involved in the attributes, crisp or exact boundaries are considered for reasoning and fuzzy boundaries are not considered.

2.4.2 Logic Programming and Rule-based Approaches

InfoSleuth [95] is an agent-based system for information gathering and analysis. This utilises Broker Agents, which perform the syntactic and semantic matchmaking. The broker agent matches agents that require services with other agents that can provide those services. By maintaining a repository containing information about the operational agents and their services, the broker enables the querying agent to locate all available agents that provide appropriate services.

The system makes use of ontologies to represent semantic concepts and terms familiar to the users in a particular domain. The "InfoSleuth ontology" is used by all the agents to specify advertisements and queries. Concepts represented within the InfoSleuth ontology include the capabilities, properties and performance details of the agent.

InfoSleuth assumes two types of brokering to match agent services: syntactic and semantic brokering. *Syntactic brokering* is the process of matching requests to agents on the basis of the syntax of the incoming messages which wrap the requests. *Semantic brokering* is the process of matching requests to agents on the basis of the requested agent/ service capabilities and advertised agent/ service capabilities, which are described in the common shared ontology of attributes and constraints. This single domain-specific ontology is a shared vocabulary that all agents can use to specify advertisements and requests to the broker. In InfoSleuth, the service capability information is written in LDL++ [5], a logical deduction language. Agents use a set of LDL++ deductive rules to support inferences about whether an expression of requirements matches a set of advertised capabilities through the use of a deductive database system.

An approach for enhancing the service discovery protocol (SDP) in Bluetooth is presented by Avancha et. al. [7], which uses semantic service matching. An ontology has been defined using RDF which is used to describe the service advertisements and requests as constraints over a number of properties, one of which is the device type. The semantic information associated with services is described using this ontology, including the priorities and expected values of attributes. An XSB-based knowledge base and reasoning engine [142, 113] is used for service registration and semantic matching in the

enhanced Bluetooth SDP prototype. Semantic matching process begins when a SDP client send a query to the SDP server. The query is an RDF description of the required service and the client and server both use the same ontology. When the query is received by the server, it parses and validates the RDF query description and creates a list of all specified attributes and their priorities (if stated). The list of attributes is sorted and matched against all the available service instances. If no exact match exists the closest values of the corresponding attribute are returned; but it is not clear from the literature how the 'closeness' of the attributes are judged. No ranking of matches is performed during this matching process.

Chakraborty et. al have presented DReggie [25], which is an effort to enhance the Jini service discovery infrastructure by enhancing the service description approach and by using reasoning mechanisms in matching. In DReggie the services are described using the DARPA Agent Markup Language (DAML). A DAML ontology has been created to describe services in terms of their properties and attributes. The service advertisements and requests both must use the same ontology. The Jini Lookup Service is enhanced with a Prolog-based reasoning engine that is capable of complex matching based on DAML service descriptions. However the criteria used for judging the closeness between the service advertisements and the request is not clear from the literature.

To facilitate resource matching on the grid, Tangmunarunkit et. al. [128, 66] have proposed a matchmaking framework, where the requests and advertisements are described ontologically using RDFS. Rules are used to describe additional domain knowledge and also matching rules. These matching rules define the matching constraints between a request and advertisement; they explicitly specify what criteria must be satisfied by an advertisement for it to be determined as a match for the given request. For example if the request is for a particular computing resource, the advertisement matches the request if it satisfies the minimum memory requirement, disk space requirement, operating system requirement and so on. This is a rather specific approach since rules must be defined specifically for each type of resource, as opposed to other approaches [101, 82, 40], where the classification approach can be used generally for any type of service. Also, this approach does not facilitate approximate matching; therefore advertisements with properties that deviate from the request specification or that are incomplete in their specification are not included in the matchmaking output.

2.4.3 Other Approaches

T. Noia. et.al. [40, 31] have presented a semantic matching approach, which allows for match ranking and categorisation; this aims to overcome the limitations of a match classification approach provided by using subsumption reasoning alone. The matches are classified into three categories of exact, potential and partial. Exact matches are service advertisements which perfectly match a given request; i.e. all properties specified in the

request is present in the advertisement. Potential matches are advertisements in which some of the properties specified in the request are not specified in the advertisement (and further action needs to be taken to inquire on those properties). When some of the properties in the advertisement contradict with those specified in the request, it is considered as a partial match (and further action could be taken to negotiate over these properties). This matchmaking approach also allows for ranking within the categories of potential and partial matches.

The matching engine is implemented by adapting the CLASSIC subsumption algorithm [18] to support match categorisation (into exact, potential and partial) and ranking. Ranking within the potential and partial categories is done by assigning a positive number to each advertisement; a higher number indicates a lower ranking. The number corresponding to the rank is assigned by taking into account the number of unspecified properties in the case of potential matches, and the number of contradictory properties in case of partial matches. However, the matching and ranking procedure does not consider subsumption relations within the property constraints of service descriptions. Suggestions for modifying the algorithm to consider weights of concepts has been proposed, but it is not clear how a user/ agent can specify weights or priorities in the description. Also, the matching algorithm does not consider approximate matching where properties with numerical values or ranges are involved.

An approach for web service matching is presented by Skoutas et. al. [117], whereby the available services are ranked according to the notion of suitability. The services are classified into the five classes of *Exact*, *Plug-in*, *Subsumes*, *Intersection* and *Disjoint* (similar to the approach in [82]) according to the 'degree of match'. This approach goes a step further and it uses the *precision* and *recall* metrics [5], to rank the services within these classes. Thus, the services are ranked on a continuous scale rather than on a discreet scale as in [82]. The inputs, outputs, preconditions and effects of a service are considered and the similarity between these parameters of a request and an advertisement are judged by using precision and recall metrics. The services are then ranked according to these similarity values.

2.4.4 Discussion

A number of recent research efforts on service matching were discussed in the previous section; these provide important directions for the semantic matching of services in overcoming the limitations of traditional syntactic approaches to service discovery. However, each of the approaches discussed in the previous sections possess weaknesses, or fail to address issues in certain areas, such as the ranking of matches, considering priorities or preferences on individual attributes of service requests, and considering approximate

[5] In information retrieval, Recall is the proportion of relevant resources actually retrieved in answer to a search request. Precision is the proportion of retrieved resources that is actually relevant.

matches when certain types of requirements are present. These issues are discussed in the following sections.

2.4.4.1 Approximate or Flexible Matching

One of the main objectives of semantic matchers are to provide flexible or approximate matches, as discussed in [101, 25]. The subsumption-based approaches for semantic matching uses the sub-class or taxonomic relations between concepts to find the approximate matches. For example, if we assume a "real-world" ontology, a 'Car' can be considered as an approximate match for a request that asks for a 'Vehicle' since the concept Car is subsumed by the concept Vehicle.

However, there are situations where using the subsumption or taxonomic relation alone, the matcher will be unable to identify services that can be intuitively considered as approximate matches; i.e. there can be disjoint concepts that have "similarities" with each other. For example, if we consider different *Processor* types in the device domain (such as *Pentium4*, *Pentium3*, *Pentium2* and *Pentium1*), although these concepts can be considered as disjoint concepts, a *Pentium3* can be thought of as being 'closer' to a *Pentium4* than a *Pentium2*. That is, a *Pentium3* will be considered as a better match to satisfy a request for a *Pentium4*, than a *Pentium2*. When properties of advertisements and requests involve such concepts, subsumption relation alone will not be effective in finding approximate matches and thus some other method is required to judge the semantic similarity between them.

Another issue is that of how the similarity is judged when service attributes with numerical values and ranges exist within service/request descriptions. For example, when a user/agent specifies the request as price ≤ 500 (the price attribute of the service must be less than or equal to 500), a service that has a price of 510 pounds will be counted as a contradiction to the requirement when using subsumption reasoning alone, irrespective of how good the other properties of the service may be. But if a human user matches the services, such cases will be considered with respect to the violation/deviation of the specified constraint and may well be considered as an acceptable approximate match. Thus, they should be ranked lower but should not be excluded from the set of matches.[6] Therefore, an appropriate criterion must be adopted to find approximate matches when numerical properties are involved in the service requirements.

[6]There are exceptions to this situation when a user has strict constraints on a property, for example "the price must be less than or equal to 500 since I only have 500 in my hand". In such cases, service description must facilitate the description of 'mandatory' attributes and hence approximate matching must not be carried out with respect to such attributes. This aspect will be discussed in more detail later in Section 4.2.4.

2.4.4.2 Ranking of Potential Matches

Ranking or classifying is the ordering of the possible matching advertisements in the order of their suitability to satisfy the given request. Ranking helps the service seekers to gain an understanding of the suitability of a service and the order in which they should consider the potential matches. In [82, 83, 101], the available service advertisements are classified into a number of classes on a discrete scale, by taking into account the subsumption relation between the request and advertisement. This is helpful in gaining an understanding of the closeness of the request and advertisement; however when a large number of potential matches fall into the same class of match, the service seekers will find this classification less helpful.

The current matching frameworks use various approaches to judge the similarity between the request and a service advertisement; this same criterion ultimately helps to find approximate matches and to rank them. The basis used to judge similarity between advertisements and requests is a key factor in determining the effectiveness of a matching approach. In recent matching algorithms [82, 83, 101, 57], subsumption is used as a basis for classifying matches. Although no ranking is provided, this is intended to give an indication of how close a match is with respect to a given demand. In the approach proposed by Li et.al. [82], the matches are classified into five categories of Exact, Plug-In, Subsumes, Intersection and Disjoint, by considering the subsumption relation between the request and the advertisement. A similar approach has been adopted by Paolucci et.al. [101, 100].

However, in certain cases, using subsumption relationship between the request and the advertisement for match classification or ranking, may not be adequate in indicating the appropriateness or suitability of a service advertisement in relation to the request. For example consider the request (R) which is looking for a *HotelRoom* with TV and *Fridge*. Suppose there are two advertisements, $A1$ and $A2$ which could match this request; $A1$ advertises a *HotelRoom* with TV and *Fridge* and $A2$ advertises *HotelRoom* with TV, *Fridge* and *HairDryer*.

$R \equiv HotelRoom \sqcap TV \sqcap Fridge$
$A1 \equiv HotelRoom \sqcap TV \sqcap Fridge$
$A2 \equiv HotelRoom \sqcap TV \sqcap Fridge \sqcap HairDryer$
$A3 \equiv HotelRoom \sqcap TV$
$A4 \equiv HotelRoom$

As far as the request is concerned, both advertisements $A1$ and $A2$ completely satisfies the request, hence they should be ranked equally. But if we consider the subsumption relationship between the request and advertisement, as done in the matchmaker proposed in [82], $A1$ is equivalent to R, and hence would be classified as an Exact match. $A2$ is more specific than R, hence it will be classified as a Subsumes match giving the

indication that $A2$ is more distant to R. This is misleading, and demonstrates how subsumption alone cannot necessarily be relied on providing a good measure of the semantic similarity. Also when the advertisements $A3$ and $A4$ are considered, both will be classified as Plug-In. Intuitively, $A4$ is more distant from the request since it does not specify two of the required properties, but this is not reflected in the above mentioned classification scheme. Hence an appropriate ranking criterion is necessary to rank the available services according to their suitability to satisfy the given request.

The matchmaker proposed in [40] ranks the advertisements within the categories of potential and partial matches. However, as already mentioned, this ranking mechanism does not consider subsumption relations or approximate matching within the property constraints of service descriptions.

The work proposed in [70] presents a matchmaker for DAML-S [4] and defines a ranking mechanism based on the subsumption relation between inputs/outputs in service advertisements and requests. Skoutas et. al. [117] have also presented an approach for ranking within the the five classes of *Exact*, *Plug-in*, *Subsumes*, *Intersection* and *Disjoint*. Whilst both of these approaches are specifically targeted for matching web services, neither will be able to detect approximate matches, other than a subsumption relation between the individual constraints or requirements in the service descriptions.

2.4.4.3 Considering Priorities on Individual Requirements

A request typically specifies a number of constraints or requirements that should be ideally satisfied by a potential match. In many practical situations, these requirements will have varying priorities or importance placed on them, depending on the context and subjective preferences of the service seekers. For example, in a computer purchasing scenario, a user looking for a computer with a particular processor and a certain amount of memory may have higher importance placed on the processor attribute over the memory size attribute. Hence, the service description should support the specification of such priorities in the service request and the matcher should consider these priorities in the matching process, if the match results are to agree with the perception of the service seekers. This aspect is overlooked in the current semantic matching approaches (except in [7]).

2.4.4.4 General Application of the Semantic Matching Approaches

The semantic matching approaches discussed above have been developed to facilitate service discovery in a variety of domains. Certain approaches for matching are domain-specific; i.e. they can be used for service matching only in the domain they were specifically intended for, and cannot be easily extended to be applied for other domains.

Certain other approaches provide general semantic matching solutions, and therefore need not be limited to a particular domain.

In rule-based approaches, such as those described in [128, 66], the matching rules specify when exactly an available service can satisfy a request. These rules have to be specified for the particular domain that the service matching is intended for, and the rules that are applicable to one domain may not apply to other domains. Therefore, the rule-based approaches are domain-specific, and cannot be easily generalised to facilitate service discovery in other domains. However, since the matching rules are defined specifically for the domain concerned, an advantage of using such a rule-based approach for matching is that, the approximate matching criterion can be tailored so that it ideally suits the context. The approaches that do not rely on matching rules for semantic matching (such as the approaches proposed in [40], [82], [57]), can typically be applied to facilitate service discovery in a wide variety of domains.

Chapter 3

Device Ontology

3.1 Introduction

The effective description of services is a key factor in facilitating the semantic discovery of services, as emphasised in the previous chapter. In order to overcome the limitations of syntactic representations of services, a flexible and expressive approach is required that describes the services at a conceptual level. Hence, we have developed an upper-level ontology [10] to describe devices and the services they provide; this will facilitate the use of logical reasoning over such descriptions which in turn will aid the semantic matching and discovery of device based services.

The fundamental purpose of this ontology is to answer the following questions:

- What is a device?
- What types of "services" are offered by a device?
- What are the features describing a device?

The Device Ontology is designed to be a shallow, but heavy-weight ontology[1]: the hierarchy is shallow, however the concepts are described by properties with restrictions, following the principle that *"a little semantics goes a long way"* [68].

In order to be able to perform reasoning over such descriptions, the ontology has been described using OWL-DL, for which several reasoning tools are currently available, such as Pellet [104, 60] and Racer [64]. The ontology has been evaluated using the current best practice approaches and validated and tested through the use of several case studies,

[1] A heavy-weight ontology refers to an ontology simply based on a hierarchy of concepts and a hierarchy of relations, enriched with restrictions and axioms used to fix the semantic interpretation of concepts and relations [50]

three of which are presented in this chapter. The validation experiments show how an ontological representation of device descriptions, coupled with the use of existing reasoning engines, can improve dramatically the automatic discovery of devices.

The rest of this chapter is organised as follows. In section 3.1 we discuss the motivating scenarios and justifications for the development of the Device Ontology. Then in section 3.2, the process involved in the development of the ontology is described and related to existing ontology design methodologies. Section 3.3 includes an overview of the device ontology. Then the findings of the investigation on how the proposed Device Ontology could be evaluated, and the results of the evaluation are presented in section 3.4. Case studies to validate the Device Ontology are presented in section 3.5 followed by concluding remarks in section 3.6.

3.2 Motivation and Requirements

Considering device-oriented service discovery, several discovery mechanisms currently exist such as Salutation [111], Service Location Protocol [118] and Jini [6], which were discussed in chapter 2. In these approaches, the services are characterised either by using predefined service categories and fixed attribute value pairs (as in SLP and Salutation) or using interfaces (as in Jini). Such descriptions are inflexible and difficult to be extended to new concepts and characteristics, and since these descriptions do not describe devices or services at a conceptual level, no form of inferencing can be carried out on them [25]. Therefore the matching techniques in these service discovery approaches (where a service request is matched with the available services to judge their suitability to satisfy the request), are limited to syntactic comparisons based on attributes or interfaces. Due to these reasons, the above mentioned discovery approaches cannot provide effective discovery of devices and their services in a dynamic environment, since they will fail to identify equivalent concepts (service requests and advertisements) which are syntactically different or approximate matches that deviate from the service request in certain aspects.

To illustrate this fact, consider the case where a user wants to seek a computer with *Unix* operating system and the devices are advertised as having either *SunOS* or *Linux*. Although both *Linux* and *SunOS* are types of Unix operating systems, the traditional service description and discovery approaches fail to realise this. Hence a device described as having a *Linux* operating system, will not be returned as a match for a request looking for a device with *Unix* operating system. In order to discover such services using the current approaches, the requester will have to look exhaustively for all possible combinations of *Unix*, which becomes very unwieldy in the case where there are a large number of possibilities. This problem is highlighted by Tangmunarunkit et. al. in their work in [128].

Another example is where a service seeker requests to utilise a *widescreen* display, when the service advertisements for the devices describes only the *aspect ratio* of the display. Although the aspect ratio is sufficient to infer whether the particular device is a widescreen or not (the so called *necessary and sufficient* conditions), such knowledge cannot be specified in the syntactic approaches for service description; and inferencing between such properties and concepts cannot be done in the conventional matching mechanisms.

Therefore, to allow inferences to be made on the service descriptions and to enable intelligent discovery possible, a flexible and extensible approach is essential to describe the devices and their services at a conceptual level [25, 128].

3.2.1 Related Work

There are several specifications and research efforts (which were discussed in section 2.2.) that have attempted to semantically describe devices in a flexible and expressive manner. In this section, we revisit these approaches to review on their ability to provide a general framework for device descriptions. These efforts are:

FIPA Device Ontology [49], which has been developed specifically to support agent communication, provides information about device capabilities to communicating agents. The capability information will be used for purposes such as content adaptation during the communication process. Since this has been developed to fulfil a specific purpose, it is only intended to describe certain types of devices and cannot be extended to describe devices such as printers and cameras. Also, it is not comprehensive enough for describing general capabilities of devices; for example it cannot describe devices in terms of the communication methods that the device can support (such as Bluetooth and Infrared) or the kinds of storage media that the device is capable of reading (such as floppy disk, CD and Compact Flash card). Therefore FIPA ontology cannot be used as a general framework for describing device capabilities and properties.

CC/PP [24], provides a framework for describing device capabilities and user preferences, that will facilitate the content adaptation and transfer of web content to clients in an appropriate manner. As with FIPA Device Ontology, the CC/PP framework is targeted at a specific purpose; of supporting content adaptation during web content transfer. Hence this suffers from the same drawbacks as with the FIPA Device Ontology, and therefore cannot be used as a general framework for device description.

The work described in [25] and [7] have used ontologies to describe devices, to facilitate the proposed device discovery frameworks. But since publicly available descriptions of these ontologies are sketchy, it is difficult to compare with this work and to properly establish the variety of devices they can be used to model. SOUPA [27] is a set of

ontologies designed to support pervasive computing applications, which also includes a device ontology for the description of devices. But this device ontology in its current state is not comprehensive, and supports the description of only a limited number of properties and characteristics.

An ontology and taxonomy for services in a service-oriented architecture (SOA) is described by Cohen in [30]. This is intended to provide a common language for software architects and engineers and to facilitate better communication within and across different disciplines and organisations. The presented taxonomy categorises the services based on their common characteristics. For example *Bus Services* are services that are infrastructural in nature and provide common facilities that would not be considered as part of a particular software application. Examples of *Bus Services* include services that provide memory management and I/O handling and facilities such as C Runtime Library (RTL) [33] and the Java Platform [71]. *Application Services* on the other hand are those that are part of a software application and provide the application's building blocks. The categorisation provided by this service taxonomy, supports composability by clarifying the roles of the different components, thus helping to reason about component interrelationships and also assists with the discoverability of services (for example, searching for existing services by using a service repository). However the taxonomy presented here focus on software services rather than on hardware resources such as computers, printers and display devices.

GOM [78, 56] is a more recent effort that has provided a set of tools and ontologies intended to facilitate the management of semantic meta-data in Grid systems. It manages various kinds of data stored in the form of OWL ontologies. The Resources Ontology, which is part of this set of ontologies, provides a generic description of the hardware resources (e.g. Server, ClusterNode, StorageNode) and software resources (e.g. Operating System, Software Library, Software Application) that can be available in a Grid environment. The main goal of this ontology is to provide means for the description of physical resources that are part of a grid deployment. Its main focus is on computing resources (such as Cluster, CPU, StorageResource and OperatingSystem) that are strictly part of the computing domain and thus can be considered as GridResources. The ontology also allows the addition of concepts specific to a particular application and not necessarily regarded as a computing resource, but still related to the grid environment. Service-specific devices fall into this category (e.g. SurveillanceCamera for Traffic Monitoring application). In the current version of the ontology they are considered as RealWorldResources. Since the Resources Ontology is intended to provide a generic description of grid components, it does not cover the possible sub-concepts of RealWorldResource as they are domain specific.

3.2.2 Creating a New Device Ontology

As we have highlighted earlier in Section 3.2, a semantic description of device capabilities is necessary in order to support effective service discovery. Therefore the Device Ontology was developed to provide a formal ontological framework to describe devices to facilitate the effective semantic discovery of device based services. Although several device description proposals have been published prior to the development of this Device Ontology [10] (discussed in Section 3.2.1), none of them fulfilled the requirement for a general framework, which provides a description of device capabilities and functionality to facilitate semantic service discovery. However, the Resources Ontology (that has been developed as part of Grid Organisational Memory Ontologies [78, 56]) which has been developed subsequently, also provides a framework for the description of generic computing resources. Although in its current state it does not handle the description of service-specific devices such as printers and cameras (which are termed as RealWorldResources in the ontology), the ontology can be extended to facilitate the description of such devices (by creating the appropriate sub-concepts, slots and restrictions as necessary).

Whilst the primary intention of the Device Ontology is to facilitate service discovery, it should also be able to support any application that needs to query about the capabilities and resources of a device, such as content adaptation applications described in [49, 24].

Service discovery is the process where devices and services are sought to provide a certain (required) functionality. Considering the typical queries in a service discovery scenario, there can be mainly two types of queries:

- *Queries for a specific device with certain properties*
  This is where the requester needs to utilise a certain type of device with a number of specified properties and constraints. For example, requests such as "I need a Colour, Inkjet Printer that can print on A3-Size paper", "I need a Widescreen, LCD display" or "Computer that has Unix operating system and FireWire connectivity", fall into this category.
- *Queries for a device that can provide a certain service or functionality*
  These are requests where a certain kind of service is sought to fulfil a particular task. For example, use cases such as "I need to show an image on this Secure-Digital Card to my colleague, what device can I use for this purpose" or "I need to play an audio file on a DVD" fall into this category.

The Device Ontology needs to capture and model the necessary knowledge relevant to the devices and the services they provide, in order to be able to answer such queries and to support semantic service discovery.

As emphasised before, our Device Ontology is intended to provide a general framework for device description that can be used as a base ontology to describe a range of devices typically used in pervasive environments. For this reason, the ontology is not bound to represent concepts that are task dependent. As pointed out by van Heist et. al in [136], a key principle in organising knowledge and ontology libraries is to modularise the ontologies; for this they suggest to first single out basic concepts in the domain to develop widely usable base ontologies; to specialise these concepts with respect to various relevant subdomains; and then add task-oriented extensions. This concept of modularisation promotes re-usability of knowledge bases and ontologies. Our Device Ontology agrees with this concept of modularisation, since it contains the base knowledge to represent device capabilities, which can be used to create specialised ontologies to describe specific devices and then to create task dependent ontologies. This aspect will be discussed in more detail in section 3.3.

3.3 Ontology Development Methodology

In this section the process followed in designing and implementing the Device Ontology is described and related to existing ontology design methodologies. A methodology is a "comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed" [119]. Since ontological engineering is a relatively immature discipline, there is no widely accepted methodology for ontology development and each work group employs its own methodology [47]. However, certain research groups have worked towards formalising the process involved in the building of ontologies, and have come up with several ontology design methodologies to aid the ontology development process; these are briefly discussed in the following section.

3.3.1 Existing Ontology Design Methodologies

Comprehensive surveys of the existing methodologies are provided in [72, 47, 48]. Fernandez et. al in [48] have identified seven ontology design methodologies that have been proposed to date, that aims to guide the process of developing ontologies from scratch. These are:

Uschold and King's Method: This method [135] is based on the experiences of building the Enterprise Ontology[2], an ontology for enterprise modelling processes. The method provides guidelines for ontology development within the phases of: Purpose identification (intended use of the ontology), ontology capture (identifying concepts and

[2]http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html

relations), coding (explicitly representing the knowledge captured in the previous stage), integrating existing ontologies and evaluation.

Cyc Method: Cyc method [81] has been defined from the experiences gathered in building the Cyc Ontology[3]. This specifies the phases of manually extracting "common sense knowledge" from knowledge sources and codifying the knowledge aided by tools. However it fails to specify the processes of requirements identification or concept design. So far this method has only been used to develop Cyc knowledge base since the processes defined are quite specific to the building of the Cyc ontology [48].

Gruninger and Fox's Methodology: This methodology [62] is based on the development experiences of the TOVE project ontology[4] within the domain of business processes and involves building a logical model of the knowledge to be specified in the form of an ontology. This follows a stage based approach where an informal model of the knowledge specification is made first and the formalised at a later stage. The steps proposed are: capture of motivating scenarios (example problems which are not adequately addressed by the existing ontologies), formulation of informal competency questions (which are considered as expressiveness requirements that are in the form of questions), specification of the terminology of the ontology within a formal language (which is done by first extracting the informal ontology from the informal competency questions and then representing them in a formal language), formulation of the formal competency questions using the terminology of the ontology, formal specification of axioms and term definitions in the ontology, establishing conditions for characterising completeness of the ontology.

Approach by Bernaras et. al: The approach proposed by Bernaras et. al.[12] is based on the process followed in the KACTUS[5] project which involves knowledge reuse in complex technical systems. This approach is tightly coupled with the development process of the application for which the ontology is built for. The development phases defined are: specification of the application (which provides application context and the components the application needs to model), preliminary design based on the top-level ontological categories (this involves searching ontologies developed for other applications, which are refined and extended for use in the new application) and ontology refinement and structuring. In this approach, since the building of the ontology is based on the building of a particular application; the ontology developed will be quite specific to the application under concern and therefore can be viewed as an application-dependent strategy [47].

Methontology: This development methodology [46, 85] consists of the development phases of *specification* (identifies the intended uses of the ontology), *conceptualisation* (consists of identifying concepts and building a conceptual model), *formalisation and*

[3]http://www.cyc.com/
[4]http://www.eil.utoronto.ca/enterprise-modelling/tove/
[5]http://hcs.science.uva.nl/projects/NewKACTUS/home.html

implementation (which transforms the conceptual model into a formal model and representing this in a formal ontology language) and *maintenance* (consists of updates and corrections to the ontology when necessary). The techniques and guidelines to be followed in each of the development activities are specified by the methodology in detail. It also identifies project management activities (planning, control, quality assurance) and support activities (knowledge acquisition, integration, evaluation, documentation and configuration management).

On-To-Knowledge: This methodology [45] was developed based on the On-To- Knowledge project[6] which introduces and maintains ontology based knowledge management applications in enterprises. The stages specified in the ontology development process are: *kick-off* (which includes the capture of requirements and analysis of knowledge sources), *refinement* (knowledge extraction and formalisation), *evaluation* (which includes the technology focused and user focused evaluation of the ontology) and *application and evolution* (applying the ontology for the intended use and maintenance). Again this methodology provides detailed guidelines for the activities to be carried out in each stage and focuses on ontology use in industrial contexts.

SENSUS-based Method: This method [125] is intended to be used when developing ontologies to be linked to the SENSUS ontology[7], which is an ontology for use in natural language processing to provide conceptual structure for machine translators. Hence this method cannot be used for ontology development in general. This involves a series of steps to be followed in ontology development which includes identifying seed terms in the domain to be modelled, manually linking them to the terms in the SENSUS ontology, and extracting the concepts in the path from the seed term to the root of the SENSUS ontology to be included in the new ontology.

Jones et. al [72] and Fernandez et. al [48] have also identified methods and approaches that address specific aspects of ontology development such as; ONIONS for integrating heterogeneous sources of information and OntoClean [63] that specifies formal guidelines for constructing taxonomical relations in ontologies. However these are not comprehensive enough to be classified as formal ontology design methods.

Summarising over the available ontology development methodologies and approaches, the following observations can be made:

- The more comprehensive development methods typically contain the following stages in the development process: a specification stage where the requirements and/ or intended uses of the ontology are identified; a conceptualisation stage where the knowledge sources are investigated to identify concepts, relations and

[6] http://www.ontoknowledge.org/
[7] http://www.isi.edu/natural-language/projects/ONTOLOGIES.html

attributes and constraints among the concepts, and an informal intermediate representation is made of the ontology; an implementation stage where the conceptual ontology is coded in a formal ontology language.

- The Cyc method and SENSUS based methods are specifically intended for use with the core ontologies (of Cyc and SENSUS respectively) and therefore cannot be adopted as development methodologies for general ontology building.

- In the analysis provided in [72, 48], the methodologies are distinguished by the criteria of *application dependence*; which indicates whether the strategy for building ontologies is dependent, or coupled with, the specific application for which the ontology is built for. According to this criteria; Gruninger and Fox's Methodology, approach by Bernaras et. al, On-To-Knowledge and SENSUS-based method are not application independent. Therefore, out of the general ontology development methodologies, the application independent approaches are: Methontology and Uscold and King's Method.

- The methodologies can also be distinguished by the criteria of the *recommended life cycle* it proposes to use [72, 47]: stage-based approaches and evolving prototype based approaches. Stage based approaches are where one stage or phase in the development process is always completed before going into the next stage. Evolving prototype based approaches are where the activities in the development phases are revisited throughout the development of the ontology. Out of the general ontology development methodologies; Gruninger and Fox's Methodology and Uschold and Kings Methodology are stage based approaches. Methontology, On-To-Knowledge and the method proposed by Bernaras et. al are evolving prototype based approaches. As analysed by Jones et. al in [72], stage based approaches are better suited when the ontology developed is for a specific purpose or application, in which case the requirements are clear and well defined at the inception stage. Evolving prototype based approaches are better when the ontology is not targeted at a specific application, and hence the requirements are not very well defined or fixed.

Although the primary purpose of the Device Ontology is to facilitate service discovery, it is intended to be a general framework to describe the capabilities of devices and hence to facilitate any application that needs to reason about device capabilities and resources. Hence, a methodology with an application independent approach is better suited for the development of this ontology. Further, for the same reason, stage based approaches are not suitable since the Device Ontology is not targeted at a single application. Thus, the Methontology approach is the most suitable approach for this case and also, is the closest to the approach followed in developing the Device Ontology since this was built in an iterative fashion. Methontology is also the most mature since this it has been

used by different groups for the development of ontologies in diverse domains and is recommended by FIPA for ontology development.

3.3.2 Design Process

In previous section, the current ontology development methodologies were discussed and after a detailed analysis, Methontology [46] was found to be the most appropriate and applicable development methodology that can be related to the development process of our Device Ontology. In this section we discuss the activities and tasks that were involved in the process of building the Device Ontology, in relation to each of the development stages in Methontology.

Specification

This is the initial stage in the development of an ontology, where the reasons for building the ontology and its intended uses are identified. As discussed in the previous sections the motivation and the main reason for the development of the Device Ontology was to facilitate semantic discovery of devices and their services. It is also intended to provide a general description framework, to facilitate any application that needs to understand about device capabilities and resources; such as in agent communication scenarios discussed in [49] or content adaptation discussed in [24]. However it is important to note that our Device Ontology describes devices from a 'usage' point of view, where the characteristics of the device that are relevant to a user seeking to utilise the device are described. It does not describe the device from a 'product design' point of view, as in the case of the ontological framework provided in [76] where the internal dynamics of a device are modelled.

Conceptualisation

This refers to the stage where the domain to be modelled is analysed using the available knowledge sources[8], and a semi-formal specification is constructed based on tabular and graphical representations that can be understood by the personnel involved in the ontology development. This involves: identification of the relevant terms (and their meanings) in the domain to be modelled, and thereby identifying the concepts in the ontology and building the concept taxonomies; identification of properties and relations among concepts; analysis of any knowledge in the domain that needs to be modelled as axioms and defining the formal axioms. Knowledge sources in the case of the Device Ontology development were: technical specifications of devices (as in product catalogues); device and service description methods in existing service discovery protocols such as service templates in Service Location Protocol (SLP) [118] and service records in Bluetooth

[8]Sources which are used to gain an understanding about the concepts and relations that should be modelled in the domain concerned.

[15] and other approaches for device description such as the FIPA device ontology[49], CC/PP[24] and UAProf [132]. Technical specifications of a variety of devices were studied, which helped to identify the attributes and properties of devices which characterises their capabilities and functionality. The existing approaches for representing device capabilities, such as SLP service templates and FIPA device ontology were also helpful in determining the important properties and characteristics that should be represented in a device description. By using these sources, a conceptual structure was built consisting of the concepts to be modelled in the ontology, their taxonomic relations, their attributes and properties and axiomatic knowledge. Additional technical sources were referred, to obtain additional domain knowledge and to clarify certain technical details (for example to clarify the relation between the concepts of *transmissive*, *reflective* and *trans-reflective*, and to be certain that *back-lit* referred to exactly the same concept as *transmissive*).

An important step in the conceptualisation stage was to determine, which concepts, properties and relations in the domain are general enough to be included in the top-level Device Ontology. Any concepts and properties that were specific to certain types of devices were left out from the top-level Device Ontology since they could be included in the specific ontology for that device (specific types of devices are described as specialisations of the generic class Device in the Device Ontology, this aspect will be discussed in more detail in section 3.4). For example Communication Method (such as Ethernet and Infrared) and Storage Media (such as optical disk and flash memory) can be considered as general concepts that will be required to describe the characteristics of a range of devices from computers, cameras, printers and mobile phones and hence will be included in the Device Ontology. Whereas the concept Printing Technology (such as Inkjet, Laser and Dot-Matrix) will be useful only in describing the characteristics of a printer, and hence this will be left out from the top-level Device Ontology (to be included in the Printer Ontology).

Implementation

This stage involves formalising the conceptual model built in the previous stage and representing this in a formal ontology language. The ontology was implemented using the Protégé ontology editor [94], and translated into OWL-DL [140] through the Protégé OWL plugin. Pellet DL reasoner [104] was used in the background along with Protégé editor throughout the implementation stage, to check the consistency of the ontology being developed.

Maintenance:

This involves the activities of updating and correcting the ontology as and when necessary. Methontology recommends a life cycle based on evolving prototypes for developing ontologies, because it allows for additions and modifications to the conceptual structure in each new version of the ontology [48]. The Device Ontology was in fact developed in

an iterative fashion, by iterating through conceptualisation and implementation stages several times before arriving at the final perceived ontology.

3.4 The Proposed Device Ontology

In this section we provide an overview of our Device Ontology in terms of the key concepts, properties associated with these concepts, the relations between the concepts and the associated axioms. The ontology has been implemented using the Protégé ontology editor [94], and translated into OWL-DL [140] through the Protégé OWL plug-in. The Device Ontology as maintained in Protégé is illustrated in Figure 3.1.

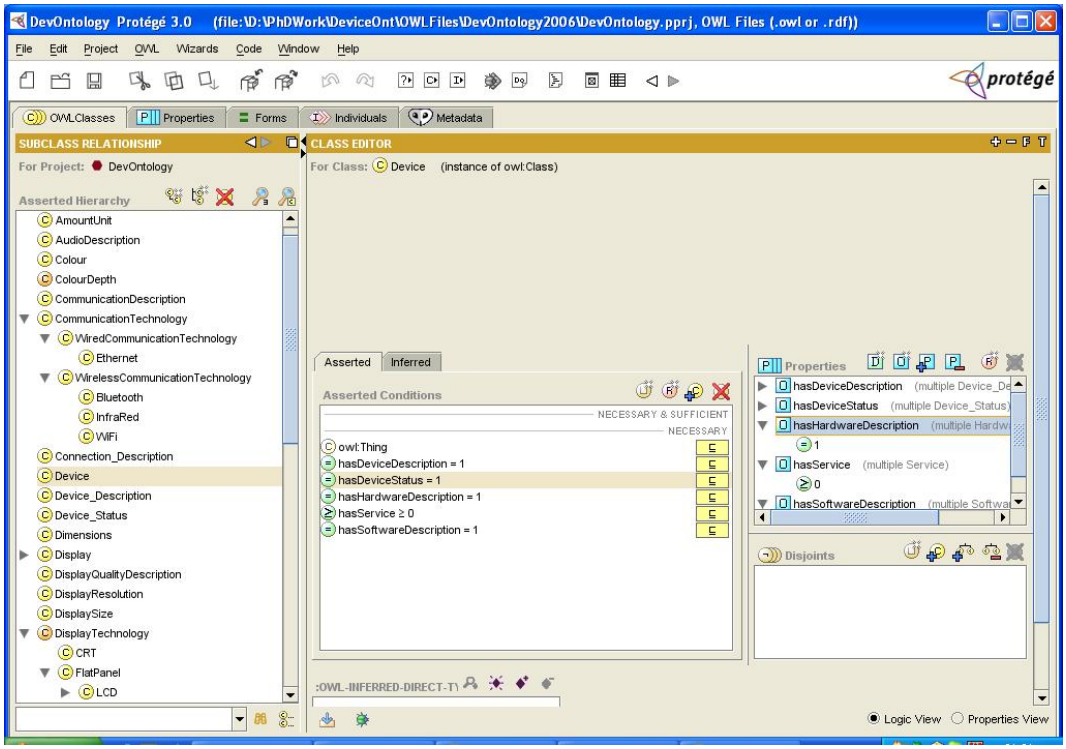

FIGURE 3.1: A view of the Device Ontology using Protégé.

The Device Ontology is organised in five main, top-level classes; which are, *Device Description*, *Hardware Description*, *Software Description*, *Device Status* and *Service*. These concepts are described below, with the associated properties and restrictions. The root level concept in the ontology is the *Device* concept.

The class *Device* describes the general features of a device. It is described by the following slots:

- *hasDeviceDescription*: A device has descriptions that can be used to advertise the device. Only one description can be associated to a device;

- *hasDeviceStatus*: A device has status, a device can be in only one state at a given time;
- *hasHardwareDescription*: A class that is used to describe the hardware resources of the device, only one hardware description can be associated to device description;
- *hasSoftwareDescription*: A class that is used to describe the software resources of the device, only one software description can be associated to a device description;
- *hasService*: The service provided by the device. A device can provide more than one service, but also no service at all.

The *Device Description* class specifies basic information that should be listed in a device description. It has the following slots:

- *deviceName*: A string that specifies the name used to identify the device on the local network, for example "Level Four Printer" or "Mary Computer";
- *deviceURI*: URI for the device which could be used to uniquely identify the device and is of type string;
- *hasDeviceManufacturer*: A class that is used to refer to an instance of the class Device_Manufacturer, which specifies the details of the manufacturer of the device;
- *deviceModel*: Specifies the model of the device and is of string data type;
- *hasPhysicalCharacteristics*: A class that is used to describe the physical characteristics of the device such as dimensions and colour;

In addition, cardinality constraints were defined on these properties, to make sure that only one value is associated with each property.

The *Hardware Description* class is used to describe the details about the hardware of the device and in terms of eight components: Audio characteristics, communication description, display description, physical interface description, power description, slot-drive description, storage and connection description. The Hardware Description is described by the following slots:

- *hasAudioDescription*: This relates the Hardware Description to the audio description of the device. The device can have only one audio description.
- *hasCommunicationDescription*: This relates the Hardware Description to the communication description of the device. A device can have only one communication description[9].

[9]The CommunicationDescription class can in turn be related to zero or more Communication Technology classes as described later in this section, since a device may be able to communicate using zero or more communication methods

- *hasDisplay*: This relates the device class to a display class. The device can have zero or more displays.
- *hasPhysicalInterfaceDescription*: This relates the Hardware Description to the physical interface description of the device, (which describes the interfaces the device has such as USB, serial port and parallel port). There can be only one Physical Interface description for a particular device.
- *hasPowerDescription*: This relates the Hardware Description to the power description of the device.
- *hasSlotDriveDescription*: This relates the Hardware description to the slot-drive description of the device, (which describes the slots or drives the device has in order to support reading from and writing to media). There will be exactly one slot-drive description.
- *hasStorage*: This relates the Hardware Description to the description of storage. A device can have zero or more storages.
- *hasConnection*: This slot associates the Hardware Description to the description of a connection, described by the class Connection;

The *AudioDescription* class is used to describe the audio characteristics of the device. It has the following slots:

- *hasMicrophone*: This is a Boolean data type property to indicate if the device has an in-built microphone.
- *hasSpeakers*: This is a Boolean data type property to indicate if the device has in-built speakers.
- *numberofSpeakers*: This is a data type property to indicate the number of in-built speakers of the device.

The *CommunicationDescription* class is used to describe the types of communications the device is capable of, such as Wi-Fi and Infrared. It has the slot:

- *hasCommunicationTechnology*: This relates the CommunicationDescription class to a subclass of Communication Technology, which can take the values of Ethernet, WiFi, Bluetooth or Infrared. The CommunicationDescription class can relate to zero or more Communication Technology classes (since a device may be able to communicate using zero or more communication methods).

The *Connection* class describes the type of connection the device has to the local network or the internet and has two slots:

- *connectionType*: is an enumerated class which can take one of the values from WirelessLAN, Ethernet or Bluetooth or from Dialup or Broadband;
- *connectionSpeed*: is an integer specifying the speed of the connection in kbps;

Other properties may be defined to augment this definition for particular subclasses of devices, such as "connectionCost" or "connectionPowerDrain", which might be relevant for mobile, wireless devices.

The *Display* class is used to describe the characteristics of the display if the device has one. It has the slots:

- *aspectRatio*: This indicates the aspect ratio of the display and is expressed using the class Ratio.
- *colourDisplay*: This is a data type property of type Boolean to indicate if the display supports colour;
- *hasDisplayQuality*: This relates the Display class to the DisplayQualityDescription, which is used to describe the quality features of the display.
- *hasDisplayResolution*: This relates the Display class to the DisplayResolution class;
- *hasDisplaySize*: This relates the Display class to the DisplaySize class, which in turn describe the size of the display in terms of the diagonal length, width and height;
- *hasDisplayTechnology*: This relates the Display class to the DisplayTechnology class, which indicates the type of display technology of the display; DisplayTechnology has subclasses of LCD, Plasma, CRT.

In addition to these properties related to the *Display*, there are axioms relating the low level properties such as *aspect_ratio* to more abstract concepts such as *WidescreenDisplay*. A *WidescreenDisplay* is defined as a *Display* with an *aspect_ratio* of 16:10, 16:9 or 15:9.

The *DisplayQualityDescription* class is used to describe the features indicating the quality of the display and has the following slots:

- *hasBrightness*: This indicates the measurement of the light level on the display screen;
- *hasColourDepth*: This indicates the colour depth of the display, in 'bits per pixel', This an enumerated type and can take values from 2, 4, 6, 8, 15, 16, 24, 32;

- *hasContrastRatio*: This indicates the measurement of the difference in light intensity between the brightest white and the darkest black and is expressed using the class Ratio.
- *hasDotPitch*: this is a measurement that indicates the diagonal distance between cells of the same colour on a display screen;
- *hasRefreshRate*: This indicates how frequently the display is refreshed expressed in Hertz;
- *hasResponseRate*: This is a measurement indicating how quickly a pixel can change colours, expressed in milliseconds;

The *DisplayResolution* class is used to describe the parameters related to the resolution of the display and has the following slots:

- *hasResolutionType*: This is an enumerated type which can take values from VGA, SVGA, SXGA, XGA, WSXGA, WUXGA, WXGA, QSXGA, QUXGA, QVGA, QXGA; corresponding to the standardised resolution sizes used by display devices.
- *hasMaximumResolution*: This indicates maximum pixel resolution of the display;

The *PhysicalInterfaceDescription* class is used to describe the types of physical interfaces a device has (such as USB connection, Serial port and Parallel port) and has the following slot:

- *hasPhysicalInterface*: This relates the *PhysicalInterfaceDescription* class to the *PhysicalInterface* class; the *PhysicalInterface* has the following subclasses: *FireWire*, *RJ45*, *RJ11*, *PowerIn*, *Headphone*, *Microphone*, *USB*, *VGAIn*, *Serial*, *Parallel*;

The *PowerDescription* class is used to describe the sources of power the device has and has the following slot:

- *hasPowerSource*: This is related to the class *PowerSource* which has the subclasses of *MainsPower* or *Battery*, which in turn has the subclasses *Rechargeable* (*Lithium-Iron* or *Nickel-Cadmium*) or *Non-rechargeable*;

The *Slot-DriveDescription* class is used to describe the types of slots or drives the device has in order to identify the kinds of media the device could access, it has the following slot:

- *hasSlotDrive*: This is related to the class Slot_Drive which has a number of subclasses such as *OpticalDrive* (*CDDrive*, *DVDDrive*), *FloppyDrive* and *PCCardSlot*;

The *Storage* class is used to describe the properties of the storage of the device, and has the following slots:

- *hasStorageCapacity*: This is a data type property indicating the capacity of the storage;
- *hasStorageMedia*: This related to the class StorageMedium and indicates the type of storage. The StorageMedium has subclasses such as HardDisk, OpticalDisk and FlashMemory;
- *removableStorage*: This is a Boolean data type property and indicates if the storage is removable or not;

The class *Software Description* is used to describe the software resources of the device. Details of any operating system or any other software platforms and applications will be described here.

The *Device Status* class is used to describe the volatile information pertaining to the device. This contains the following slots:

- *isLocatedAt*: used to specify the details of the location of the device. (An external ontology that models location will have to be used for describing location).
- *power*: This is a slot that describes the details of how the device is powered. Its range is the class Power_Status;

The *Power_Status* class in turn has two slots:

- *methodOfPower*: an enumerated type that can take the values from one of battery or mains.
- *remainingPower*: is an integer specifying the level of remaining power in the device as a percentage (when it is battery powered).

Both these slots have cardinality 1. The slots related to power supply and power level become important when it is necessary to determine the resource capability of a device. Location details are required when the device selection process needs to consider the location of the device in choosing the right device or service.

The *Service* class provides the information about the service(s) hosted on the device concerned. Approaches such as OWL-S [90] and SOA Service Taxonomy [30] could be potentially used to facilitate the description of these individual services; however the service description is outside the scope of this thesis and will be the subject of future research. A device will provide at least one service.

Specialisation for Specific Devices

As discussed previously, the Device Ontology is intended to provide a top-level ontology that describes a common vocabulary to allow the inter-operation between device advertisements and request description. The specific types of devices are described as specialisation of the generic class Device, through subsumption. For example, the Computer class is defined as a subclass of the Device class which was specialised by defining additional concepts and properties (such as the CPU properties, operating System properties and methods for user input) that are necessary in order to effectively describe computers. Likewise the Printer class is defined as a subclass of the Device class, and specialised by defining additional properties (such as printer resolution, supported media types and printing speed). Wherever a device does not fall into any of the available categories, or when it is not clear to which category it belongs, it can be specified as an instance of the Device class itself, thereby avoiding the use of the hierarchical classification. We have developed the specific device ontologies for Computer (including desktop, laptop and handheld computer), Printer, Camera, Storage devices and Display devices. In this section, we give a brief overview of the Computer ontology, in order to illustrate how a specific device ontology can be created. Figure 3.2 illustrates the Protégé ontology where several instances of computers are modelled using the Computer Ontology.

Computer Ontology

The class *Computer* is defined as a subclass of the *Device* class.

In addition to the classes and properties already defined in the Device Ontology, the *HardwareDescription* class in the Computer ontology will have three additional slots, which are:

- *hasCPUDescription*: This relates the HardwareDescription class to the CPUDescription class, which is used to describe the information related to the CPU(s) of the Computer. The Computer can only have one CPUDescription;
- *hasSystemMemory*: This relates the HardwareDescription class to the MemoryDescription class, which is used to describe Computer's system memory. The Computer can only have one MemoryDescription;.
- *hasUserInterfaceDescription*: This relates to the class *UserInterfaceDescription* and is used to describe the user interfaces of the computer, such as keyboard, touch-screen and pointing devices;

CPUDescription class in the Computer ontology will have the following slots:

- *hasCPU*: This relates the CPUDescription class to the CPU class which specify the type of CPU on the Computer;

- *busSpeed*: This is a data type property indicating the bus speed of the CPU;
- *clockSpeed*: This is a data type property indicating the clock speed of the CPU;
- *level1Cache*: This indicates the amount of level 1 cache in the CPU;
- *level2Cache*: This indicates the amount of level 2 cache in the CPU;
- *level3Cache*: This indicates the amount of level 3 cache in the CPU;
- *numberOfCPU*: This indicates the number of CPUs in the Computer, since a Computer can have more than one CPU in the case of servers;

MemoryDescription class in the Computer ontology will have the following slots:

- *hasMemoryTechnology*: This relates to the class MemoryType and indicates the type of memory the Computer has. The MemoryType has subclasses such as RAM, DRAM and SDRAM.
- *memoryCapacity*: This indicates how much system memory the Computer has;

UserInterfaceDescription class will be used to describe the user interfaces of the Computer and has the following slot:

- *hasUserInput*: This relates the class UserInterfaceDescription to the UserInput class. UserInterfaceDescription can be related to more than one UserInput classes since a Computer can have multiple user interfaces; UserInput class is used to identify different types of user inputs, and has subclasses such as Keyboard, TouchScreen and PointingDevice.

SoftwareDescription class in the Computer ontology will have one additional slot, which is:

- *hasOSDescription*: This relates the *SoftwareDescription* class to the *OSDescription* class, which is used to describe the information related to the Operating Systems of the Computer. The Computer can only have one *OSDescription*;

3.5 Evaluation

Currently, ontologies are increasingly being used in a variety of domains and applications varying from knowledge management, natural language processing, e-commerce and information retrieval. As with any other resource used in software applications, the

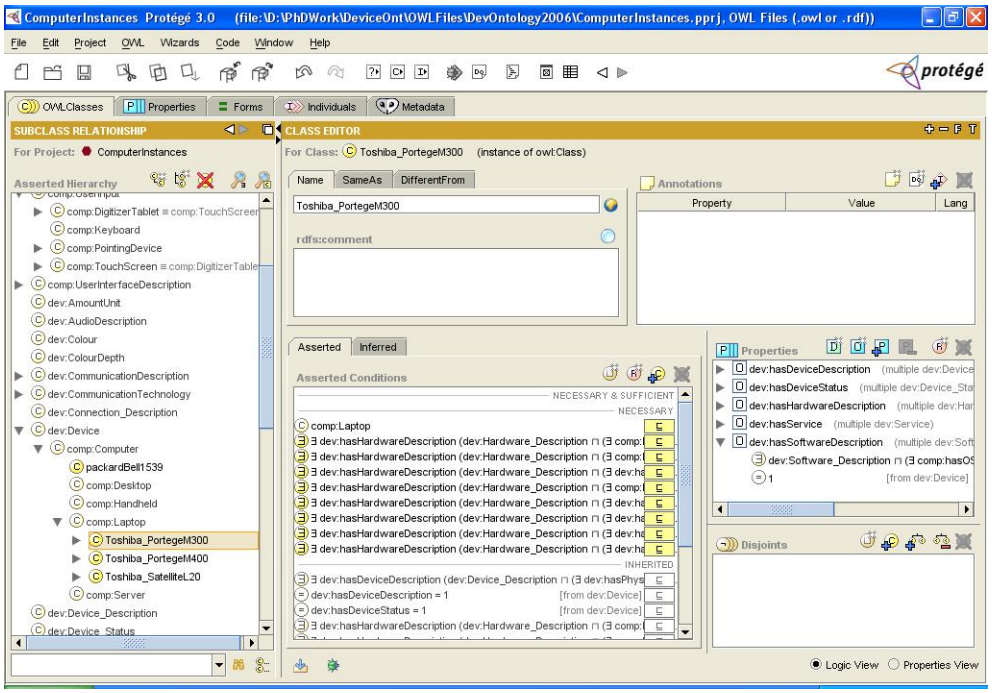

FIGURE 3.2: Protégé Ontology where several instances of Computers are modelled using the Computer Ontology.

content of ontologies should be evaluated before using it in any other ontologies or applications to ensure they are effective and fit for purpose [55]. Thus in this section we present the findings of the investigation on how the proposed Device Ontology could be evaluated, and the results of the evaluation by using current best practice approaches.

Ontology evaluation is an emergent field and is gathering attention with the increased use of ontologies in research and industry. Several ontology evaluation methods and tools have been proposed to date: Hartmann et. al in [67] and Brank et. al. in [20] present surveys of currently available ontology evaluation methods which are applicable in different situations. For example, OntoMetric [86] is a method that can be used when knowledge engineers have to choose an appropriate ontology (among several candidate ontologies) to be used in an application or project. OntoMetric proposes the decision criteria to be considered, and the process that should be followed in order to obtain a valuation of the suitability of each candidate ontology that will help in assessing the suitability of the ontologies. On the other hand EvaLexon [109] is a method that can be applied on the results of automatic ontology mining techniques (where ontologies are created from text documents in the application domain). EvalLexon provides a rough reference to determine whether or not the results of ontology mining, capture most of the notions of the input text by using a number of metrics (such as coverage and accuracy [109]). The survey also outlines Natural Language Application metrics which helps in evaluating the content of ontologies with respect to natural language applications (applications that involve populating an ontology of concepts with instances drawn from textual data). These include metrics such as Precision and Recall [28], and the Tennis

measure [21] (which evaluates the extent to which items in the same cluster are closer together in the ontology than those in different clusters).

However, none of the above mentioned evaluation methods or metrics can be used to evaluate the content of the proposed Device Ontology, since they are applicable in different situations: OntoMetric is intended to help ontology selection, when one has to choose a single ontology among several alternatives, to be used in a certain application. EvalLexon helps evaluate the quality of an ontology, in cases where ontologies are constructed automatically from input text documents. Natural Language Metrics help evaluate ontologies, in cases where ontologies are automatically populated (with instances) from input text documents.

Out of the available ontology evaluation methods, there are two principal methods that are relevant to ontology developers in general (as pointed out in [67]), and are applicable for the evaluation of the proposed Device Ontology. These are; the method proposed by Gomez-Perez in [55] and the OntoClean method[63]. These were investigated in detail and the principles behind the evaluation methods are outlined below.

The method proposed by Gomez-Perez in [55] presents several criteria which can be used to evaluate the taxonomic content of ontologies and points out several types of errors that can be made when developing taxonomies, which are:

- Inconsistency: This refers to circular errors (which occurs when a concept is defined as a specialisation or generalisation of itself, forming cycles in the taxonomy); partition errors (when a concept is defined to be a subconcept of two or more disjoint concepts) and semantic errors (incorrect semantic classification of concepts).
- Incompleteness: This refers the lack of completeness of the ontology with respect to the concept hierarchy, domain and range of relations and omission of disjoint knowledge.
- Redundancy: Redundancy errors occur when expressions of the ontology are redefined when they have been already defined explicitly or when they can be inferred from other definitions.

ODEval[10] is the publicly available tool that provides automatic evaluation of ontologies supporting the method outlined above. This uses a set of algorithms based on graph theory to detect possible problems in ontology concept taxonomies. For OWL ontologies, this detects circularity problems, partition errors and redundancy problems.

As stated by the authors of [55], any ontology should be checked for the presence of these fundamental problems to ensure correctness and usability of the ontology in other ontologies and applications. The proposed Device Ontology was evaluated using ODEval

[10]http://minsky.dia.fi.upm.es/odeval

and as indicated by the tool, none of the above problems were present in the ontology. Figure 3.3 illustrates a screenshot of the results obtained from evaluation of the ontology using ODEval.

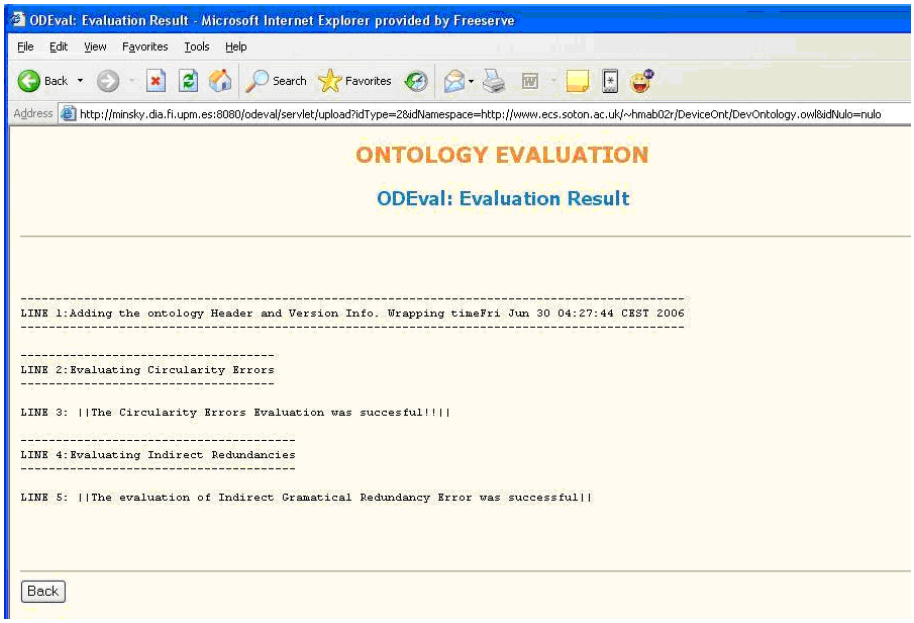

FIGURE 3.3: The results page produced by ODEval during the evaluation process.

OntoClean [63] which is the other main evaluation method applicable to ontology developers, is based on philosophical notions for a formal evaluation of taxonomic structures. In other words this method helps to remove wrong taxonomic relations in ontologies based on philosophical notions of *rigidity*, *unity* and *identity*. OntoClean method consists of (1) a set of axioms that formalise definitions and constraints specified in the methodology and (2) a "meta-ontology" or a "taxonomy of properties" that provides a frame of reference for evaluations. The philosophical notions of *rigidity*, *unity* and *identity*, on which the OntoClean method is based on, are briefly described below.

Rigidity: A property is rigid if it is essential to all its possible instances; an instance of a *rigid* property cannot stop being an instance of that property in a different world. For example *being a human* is considered to be a rigid property since no instance of a human can stop being a human. On the other hand *being a student* is considered as an anti-rigid property since any instance of a student can stop being a student at any point in time. Rigidity and its variants (anti-rigid and non-rigid) are considered as important meta-properties in OntoClean since they impose fundamental constraints on the subsumption relations which are used to check the formal correctness of taxonomic links. For example *being a student* cannot subsume *being a human* when the former is anti-rigid and the latter is rigid.

Unity: Certain properties pertain to 'wholes'[11] , that is, all their instances are wholes,

[11]The meaning of 'whole' to be interpreted as 'an assemblage of parts that can be regarded as a single entity'

and others do not. For example, being (an amount of) water does not have wholes as instances, since each amount can be arbitrarily scattered or confused with other amounts. In other words, knowing an entity is an amount of water does not tell us anything about its parts, or how to recognise it as a single entity. On the other hand, being an ocean is a property that picks up whole objects, as its instances, such as "the Atlantic Ocean," which are recognisable as single entities. As with rigidity, the meta-property of unity (and anti-unity) is used to construct constraints on subsumption relations of an ontology. Hence "Ocean" cannot be a subclass of "Water" as the former carries unity and the latter carries anti-unity (oceans are *composed of* water not a "kind of water").

Identity: Identity refers to the problem of being able to recognise individual entities in the world as being the same (or different). Identity criteria (used to 'identify' a certain individual as being that individual) are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions). For instance consider the example of statue and the clay: is the statue identical to the clay it is made of? Considering the essential properties: having (more or less) a certain shape is essential for the statue, but not essential for the clay. Therefore, they are different: it can be said that they have different identity criteria, even without knowing exactly what these criteria are. A property carries identity, if it has common identity criteria to identify the instances of that property.

As indicated in the examples, the OntoClean method uses these meta-properties and the constraints on the classes carrying these meta properties, to identify taxonomic relations that are fundamentally wrong and to "clean" the ontology.

As pointed out in [67], although OntoClean provides useful insights in to semantic models, these insights are more structural and formally driven and do not allow to infer anything about the usability of the analysed ontology; further the OntoClean evaluation process is quite time consuming due to the fact that each concept in the ontology to be evaluated, has to be tagged with appropriate meta-properties set out in the method. However, this is useful in cases where formal correctness is required such as in foundation ontologies where very abstract concepts needs to be modelled in describing the world and the hierarchical relations between concepts are not always clear. In the Device Ontology the taxonomical relations are relatively more straightforward; however the OntoClean method was applied to check for any potential problems outlined in the method.

To evaluate the ontology using the OntoClean method, Protégé ontology editor [94] was used in conjunction with the OntoClean Protégé ontology[12] and the Protégé PAL (Protégé Axiom Language) constraints tab[13]. The OntoClean Protégé Ontology contains the meta-ontology and the definitions and constraints set out in the OntoClean method. To check for formal correctness as set out in OntoClean, each concept in the ontology

[12] http://protege.stanford.edu/ontologies/ontoClean/ontoclean.owl
[13] http://protege.stanford.edu/plugins/paltabs/PAL_tabs.html

was tagged with the appropriate meta-properties (rigid, anti-rigid, unity, anti-unity and identity) and then evaluated against the constraints. No violations of the OntoClean constraints were present in the ontology.

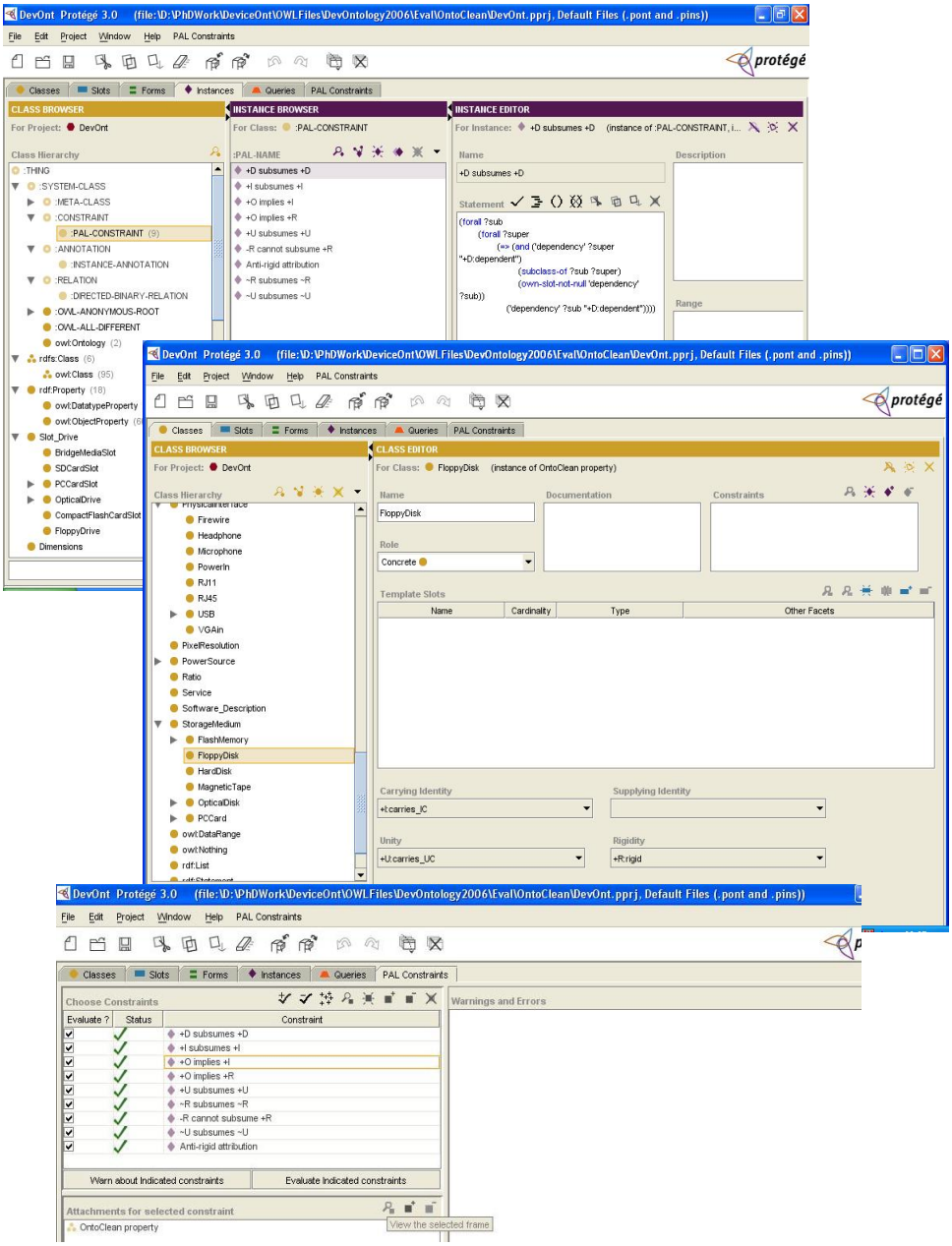

FIGURE 3.4: Ontology evaluation process using the Protégé and OntoClean.

Figure 3.4 illustrates the screenshots obtained during the evaluation process using Protégé environment. Each of the screenshots in the figure illustrates the major steps in the evaluation process, which are:

1. The conditions to be checked to ensure the formal correctness of the ontology are specified as PAL constraints.
2. The concepts in the Device Ontology are tagged with the appropriate meta-properties (in terms of rigidity, unity, identity).
3. The evaluation will check whether the ontology violates any of the constraints as set out in OntoClean and indicate the status. Warning and errors will be indicated if any violations are detected.

The evaluation approaches listed above, check for fundamental errors in the ontology (in terms of consistencies and redundancy) and the formal correctness of the organisation of concepts in the ontology. However such methods cannot asses the usefulness and appropriateness of the ontology for the purpose intended. This has to be assessed by means of experiments, to judge whether the ontology can answer the questions that it is intended to answer and if the ontology can deliver the results expected from it. The assessment of this aspect and the experiments conducted in this regard are discussed in more detail in the next section.

3.6 Case Study

In this section we discuss the experiments carried out in order to demonstrate the utility of the Device Ontology and to verify whether it fulfils the purpose it was intended for (as described in the motivating scenarios in Section 3.2). These experiments will also show, how the Device Ontology coupled with description logic reasoning, help to improve the results of service discovery, over and above what is provided by syntactic service discovery approaches.

We consider the examples set out in Section 3.2, where there is a query for a certain service or device, and a number of devices are available. By classifying the requests and the available devices using a DL reasoner, and by checking the subsumption relation between the query and the available devices, we can judge which devices satisfy the query (the devices subsumed by the query). The case study ontologies were developed using the Protégé ontology editor [94] and classified using the Pellet DL reasoner [104].

Example 1: ***Seeking a Wide Screen, Flat Panel display unit that has in-built speakers***

In this example, the request is for a *Widescreen* display, that uses *Flat Panel* technology and has integrated *Speakers*. The query expressed in DL notation is[14]:

$CS3 : Query \equiv dev : Device \sqcap$
$(\exists dev : hasDisplay.\ dev : WidescreenDisplay) \sqcap$
$(\exists dev : hasDisplayTechnology.\ dev : FlatPanel) \sqcap$
$(\exists dev : hasSpeakers.\ true)$

In this case we have the axiomatic knowledge that helps to determine if a display is *widescreen* or not by considering its *aspect ratio*. Thus although a device is not explicitly specified as a *widescreen*, this fact could be inferred from the aspect ratio property. This knowledge is specified by using *necessary and sufficient* conditions (equivalent classes) in OWL.

$WidescreenDisplay \equiv dev : Display \sqcap$
$\exists aspectRatio(as15\_9 \sqcup as16\_10 \sqcup as16\_9)$

The available devices are listed below:

- The display HP_VS19X is a TFT display, with 4:3 aspect ratio which does not have integrated speakers.

  $CS3 : HP\_VS19X \sqsubseteq \exists dev : hasDisplayTechnology.\ dev : TFT \sqcap$
  $\exists dev : aspectRatio.\ dev : as4\_3 \sqcap$
  $\exists dev : hasSpeakers.\ false$

- The display LG_RU23LZ50C is a LCD display, with 15:9 aspect ratio which has integrated speakers.

  $CS3 : LG\_RU23LZ50C \sqsubseteq$
  $\exists dev : hasDisplayTechnology.\ dev : LCD \sqcap$
  $\exists dev : aspectRatio.\ dev : as15\_9 \sqcap$
  $\exists dev : hasSpeakers.\ true$

[14]Please note here, that using our Device Ontology the query is expressed as:

$Device \sqcap$
$\exists dev : hasHardwareDescription(dev : Hardware_Description \sqcap$
$(\exists dev : hasDisplay(dev : WidescreenDisplay \sqcap$
$(\exists dev : hasDisplayTechnology.\ dev : FlatPanel))) \sqcap$
$(\exists dev : hasAudioDescription(dev : AudioDescription \sqcap$
$(dev : hasSpeakers.\ true))))$

due to the manner in which the classes are organised. But for the sake of readability and brevity of this discussion, we do not show the entire nested form, but only show the relevant property restrictions in the documentation (since the entire query/ advertisement in its original nested form will be too lengthy and confusing to follow). We will be using this simplification through out the discussion in this section.

- The display PackardBell_Maestro is a TFT display, with 16:9 aspect ratio which has integrated speakers.

  $CS3 : PackardBell\_Maestro \sqsubseteq$
  $\exists dev : hasDisplayTechnology.\ dev : TFT \sqcap$
  $\exists dev : aspectRatio.\ dev : as16\_9 \sqcap$
  $\exists dev : hasSpeakers.\ true$

- The display Philips_107E66_BlackCRT is a CRT display, with 4:3 aspect ratio which does not have integrated speakers.

  $CS3 : Philips\_107E66\_BlackCRT \sqsubseteq$
  $\exists dev : hasDisplayTechnology.\ dev : CRT \sqcap$
  $\exists dev : aspectRatio.\ dev : as4\_3 \sqcap$
  $\exists dev : hasSpeakers.\ false$

- The display Philips_32PW6520 is a CRT display, with 16:9 aspect ratio which has integrated speakers.

  $CS3 : Philips\_32PW6520 \sqsubseteq$
  $\exists dev : hasDisplayTechnology.\ dev : CRT \sqcap$
  $\exists dev : aspectRatio.\ dev : as16\_9 \sqcap$
  $\exists dev : hasSpeakers.\ true$

The classification results obtained when this example ontology was classified using Pellet is shown in Figure 3.5.

FIGURE 3.5: Classification results obtained from Pellet for Example 1.

According to the classification hierarchy produced by Pellet, the displays PackardBell_Maestro and LG_RU23LZ50C are subsumed by and hence satisfy the query. After inspecting the properties of the devices, we could observe that this is intuitively correct.

This use case shows an important advantage of using ontological descriptions coupled with reasoning engines for service discovery. That is, the ability to use inferencing to

reason between different representations of the same concept; (in this example it was inferred that displays with certain *aspect ratios* are *widescreen* displays).

It is often the case, that the service providers usually describe devices in terms of lower-level properties, and the service seekers or clients usually prefer to describe service requests using more abstract or higher level concepts. This also agrees with the principle set out by Gonzalez-Castillo et. al in [57], where they state that a requirement of a service description approach, is to allow the flexibility for the description to be more general or more specific. This fact is apparent in the above example, where the service request is expressed as a *widescreen* display, whereas the devices are described using the (lower-level) property of *aspect ratio*. The OWL language allows us to express *necessary and sufficient* conditions in the specifications, and we have exploited this ability of the language in the Device Ontology, and have specified that displays with certain *aspect ratios* correspond to *widescreen* displays. This knowledge in turn can be used to infer that a display with such *aspect ratios* 'must' be a *widescreen* display.

Such benefits cannot be gained from the traditional syntactic service discovery approaches where flat structures or interface based descriptions are used or from object-oriented programming approaches. This example clearly shows the flexibility provided by an ontological approach for device description, and the ability of a reasoning engine (coupled with such an ontological approach) to infer between such different representations of services thus providing effective semantic discovery of services.

Example 2: ***Seeking a device that is capable of displaying an image stored on a Secure Digital Card***

This is a kind of query that can be raised from a scenario where a user has an image stored on a SD Card which he wants to show to his colleague. In this type of request, the service seeker is in need of a certain service or functionality. There are two requirements expressed in this service query, firstly, there should be a device capable of reading from SD Card media, secondly, it should be capable of rendering an image.

Using description logic notation the query can be expressed as follows:

$$CS1 : Query \equiv dev : Device \sqcap$$
$$(dev : Slot\_Drive \sqcap (\exists dev : hasReadableMedium.\ dev : SecureDigitalCard)) \sqcap$$
$$(\exists dev : hasService.\ CS1 : ImageRendering)$$

The latter requirement of having a Image Rendering Service can be explicitly stated in the device description (for example as in the case of MP3 players, where the ability or inability to render images are inherent to the device and thus can be explicitly stated) or it can be judged or inferred by the fact that, the device has a display and software that is capable of rendering images.

To explicitly state this knowledge (that the device is capable of rendering an image, if it has a Image Rendering service OR if it has a Display and software capable of rendering images) we use two sets of necessary and sufficient conditions (equivalent classes in OWL) as:

$ImageDisplayDevice \equiv \exists dev : hasService.\ CS1 : ImageRendering$

$ImageDisplayDevice \equiv (\exists dev : hasDisplay.\ dev : Display) \sqcap$
$(\exists CS1 : hasApplication.\ CS1 : ImageViewingApplication)$

Now let us consider the available devices; below we provide the listing of the devices and their relevant properties:

- JacksLaptop which has an SD card slot capable of reading a SD card, and WindowsPictureandFaxViewer that is capable of rendering images

  $CS1 : JacksLaptop \sqsubseteq$
  $\exists dev : hasSlot\_Drive(dev : SDCardSlot \sqcap \exists dev : hasReadableMedium$
  $dev : SecureDigitalCard) \sqcap$
  $(\exists CS1 : hasApplication.\ CS1 : WindowsPictureandFaxViewer))$

- MarysOfficeLaptop which has an BridgeMedia slot capable of reading a SD card, and GhostViewer that is capable of rendering images

  $CS1 : MarysOfficeLaptop \sqsubseteq$
  $\exists dev : hasSlot\_Drive(dev : BridgeMediaSlot \sqcap \exists dev : hasReadableMedium$
  $dev : SecureDigitalCard) \sqcap$
  $(\exists CS1 : hasApplication.\ CS1 : GhostViewer))$

- Danny which does not have any slot capable of reading a SD card, but has WindowsPictureandFaxViewer that is capable of rendering images

  $CS1 : Danny \sqsubseteq$
  $\exists CS1 : hasApplication.\ CS1 : WindowsPictureandFaxViewer$

- The printer PhotoSmart2575 which has a slot capable of reading a SD card, and provides a image rendering service.

  $CS1 : PhotoSmart2575 \sqsubseteq$
  $\exists dev : hasSlot\_Drive(dev : Slot\_Drive \sqcap \exists dev : hasReadableMedium.$
  $dev : SecureDigitalCard) \sqcap$
  $\exists dev : hasService.\ CS1 : ImageRendering$

- The mp3 player I-FriendPES2052 which has a slot capable of reading a SD card, and provides a image rendering service.

  $CS1 : I - FriendPES2052 \sqsubseteq$
  $\exists dev : hasSlot\_Drive(dev : Slot\_Drive \sqcap \exists dev : hasReadableMedium$

$dev : SecureDigitalCard) \sqcap$
$\exists dev : hasService.\ CS1 : ImageRendering$

- The mp3 player ProportaMP3SDPlayer which has a slot capable of reading a SD card, and does not have an image rendering service.

  $CS1 : ProportaMP3SDPlayer \sqsubseteq$
  $\exists dev : hasSlot\_Drive(dev : Slot\_Drive \sqcap \exists dev : hasReadableMedium$
  $dev : SecureDigitalCard)$

Figure 3.6 shows the section of classification results obtained, when these concepts (query and the available devices) were classified using Pellet.

FIGURE 3.6: Classification results obtained from Pellet for Example 2.

The results show that the devices: *MarysOfficeLaptop*, *I-FriendPES2052*, *JacksLaptop*, and *PhotoSmart2575* are subsumed by and therefore satisfy the query.

In addition to illustrating how the Device Ontology has helped answer this service discovery request, this shows how an ontological approach can infer the functionality of a device although they are not explicitly stated. In this example, although the device does not explicitly state that it provides an *Image Rendering* Service, it could be inferred from the fact that the device has a display and that it has a software application that is capable of rendering images. This is an important contribution of ontological approaches to service discovery that the conventional syntactic service discovery approaches are not capable of.

Example 3: ***Seeking a portable Computer, with Unix compatible operating system and FireWire connectivity***

In this example, the query can be expressed in DL notation as:

$CS4 : Query \equiv dev : Device \sqcap$
$(\exists comp : hasOS.\ comp : Unix) \sqcap$
$(\exists dev : hasPhysicalInterface.\ dev : Firewire) \sqcap$
$(\exists dev : portability.\ true)$

In this case we have the taxonomical relations between certain Operating Systems; Specifically, that SunOS and Linux are subclasses of Unix.

$Linux \sqsubseteq Unix$
$SunOS \sqsubseteq Unix$

The available devices are listed below:

- The Desktop Computer FredsPC has Linux operating system and has FireWire connectivity, and since this is a desktop computer it has the inherited condition that it is not portable.

  $CS4 : FredsPC \sqsubseteq$
  $(\exists comp : hasOS.\ comp : Linux) \sqcap$
  $(\exists dev : hasPhysicalInterface.\ dev : Firewire) \sqcap$
  $(\exists dev : portability.\ false)$

- The Laptop Computer JacksLaptop has Linux operating system and has FireWire connectivity, and since this is a laptop computer it has the inherited condition that it is portable.

  $CS4 : JacksLaptop \sqsubseteq$
  $(\exists comp : hasOS.\ comp : Linux) \sqcap$
  $(\exists dev : hasPhysicalInterface.\ dev : Firewire) \sqcap$
  $(\exists dev : portability.\ true)$

- The Laptop Computer MarysOfficeLaptop has SunOS operating system and has FireWire connectivity, and since this is a laptop computer it has the inherited condition that it is portable.

  $CS4 : MarysOfficeLaptop \sqsubseteq$
  $(\exists comp : hasOS.\ comp : SunOS) \sqcap$
  $(\exists dev : hasPhysicalInterface.\ dev : Firewire) \sqcap$
  $(\exists dev : portability.\ true)$

- The Laptop Computer Danny has WindowsXP operating system and has no FireWire connectivity, since this is a laptop computer it has the inherited condition that it is portable.

  $CS4 : Danny \sqsubseteq$
  $(\exists comp : hasOS.\ comp : WindowsXP) \sqcap$
  $(\exists dev : portability.\ true)$

- The Handheld Computer JohnsHandheld has PalmOS operating system and has no FireWire connectivity, and since this is a handheld computer it has the inherited condition that it is portable.

  $CS4 : JohnsHandheld \sqsubseteq$
  $(\exists comp : hasOS.\ comp : palmOS) \sqcap$
  $(\exists dev : portability.\ true)$

The classification results obtained when this example ontology was classified using Pellet is shown in Figure 3.7. According to the classification hierarchy produced, the computers MarysOfficeLaptop and JacksLaptop are subsumed by the query (as would be expected) and hence satisfy the request.

Here the request seeks for a computer with Unix and the devices returned have specified their OS as Linux and SunOS respectively. The point to be noted here is that the Ontology exploits the ability to specify taxonomic relations within the ontology language and by using reasoners this knowledge can be used for effective discovery of devices and services.

This again shows the flexibility provided by ontological approaches; the service description can be made more general (as to specify that the required OS is Unix) or specific (so as to say that the OS should be Linux or SunOS) as appropriate for the circumstances; and since inferencing engines are used in finding the matches to satisfy a given request (as opposed to syntactic comparison) the relevant matches can be found, even though they are represented differently. This flexibility in the description language or effectiveness in service matching is not present in traditional service discovery approaches as already discussed in previous sections and pointed out by Chakraborty et. al. in [25] and Tangmunarunkit et. al in [128].

3.7 Discussion

This chapter presented the Device Ontology which provides a general framework that describes devices and their services in a flexible and expressive way, to aid the description of a variety of devices and to facilitate effective semantic discovery of services. The ontology was developed by following a systematic design process, going through the activities and tasks applicable to each design stage, as suggested in the Methontology [46, 85] ontology design methodology. The ontology has been formally evaluated by using the current best practice approaches for ontology evaluation [55], specifically the OntoClean method [63] and ODEval[54]. The validation case studies provide an assessment of the usefulness and usability of the ontology, and have shown how the Device Ontology coupled with a DL reasoner can help to improve the results of service discovery and matching, over and above of what could be achieved with the syntactic approaches;

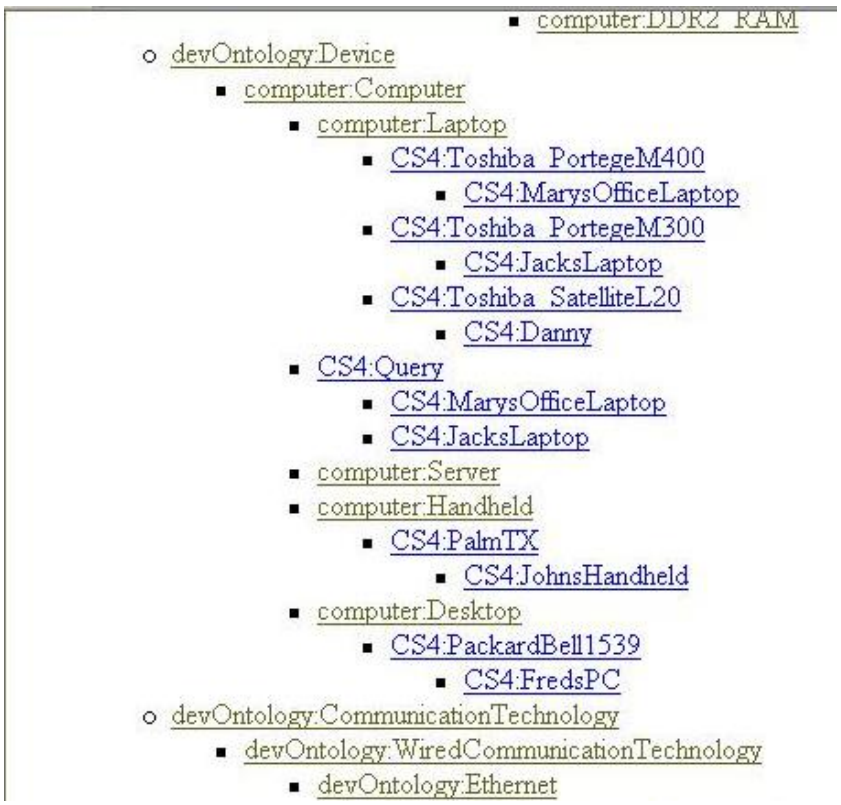

FIGURE 3.7: Classification results obtained from Pellet for Example 3.

i.e. they have demonstrated the benefits of semantic matching as opposed to syntactic matching.

Although an ontological description of services coupled with DL reasoning can provide semantic matching of service descriptions and thus improves on syntactic matching, there are several desirable properties of a semantic matching solution that cannot be provided through the use of an ontological description alone. These properties are:

- Approximate matching: In most practical scenarios, the resource seekers (in the absence of an exact match) will be willing to consider advertisements that deviate from the request in certain respects. Therefore, such advertisements that can 'approximately' match a request must also be considered during the matching process, and returned as potential matches as appropriate.

- Ranking of matches: If there are approximate matches identified during the matching process, these matches must ideally be ranked or ordered to reflect the suitability of the advertisements to satisfy the request.

- Consideration of priorities on individual requirements: A resource request can have a number of requirements or constraints specified in the request that must be met by a potential advertisement. In practical situations, the resource seeker will consider certain requirements to be more important than others, and thus will

have different priorities or weights placed on them. These priorities must be taken into account during the service discovery, for the match results to be suitable for the context.

As demonstrated in the validation case studies in Section 3.6, an ontological description of resources coupled with DL reasoning alone, will only identify resources (by semantically matching the conceptual content of the advertisements and requests) that can completely satisfy the request concerned; it does not provide any approximate matching or ranking. Also, it does not facilitate priority consideration in the matching process. In order to satisfy these requirements, a matching mechanism is needed, which will extend the semantic matching (provided by ontological descriptions and DL reasoning), to facilitate approximate matching, match ranking and priority consideration during the matching process.

Chapter 4 discusses these desirable properties and requirements of a semantic matching solution in more detail along with motivating scenarios, and presents the semantic matching mechanism that has been developed to address these requirements. The Device Ontology is used to facilitate the description of resource advertisements and requests, in the implementation of this proposed semantic matching solution (discussed in Chapter 5).

As with any ontology, the Device Ontology needs to be refined and evolved through use and application [55]. Also, a potential area for further development in this ontology, is to investigate how the *connectivity* of devices can be modelled in the ontology. That is when two devices are connected, how can this 'connectedness' be indicated in the ontology and how are their capabilities affected? For instance when a mp3 player is connected with a media card reader, the additional functionality obtained through this combination (such as the ability to read from the additional media types) will have to be modelled for the effective use of the Device Ontology in such situations.

Chapter 4

The Semantic Matching Framework

4.1 Introduction

Resource matching refers to the process of evaluating the available resource advertisements for their suitability to satisfy a given resource request. A resource seeker will specify the requirements that should be met by a potential resource in the request description and the resource providers will specify the capabilities of the provided resource in the advertisement description. The description of resource advertisements and requests will need to adhere to an agreed description format in the domain concerned. The outcome of the matching process will be a set of resource advertisements, that are deemed by the matching system as suitable for satisfying the request. The set of advertisements returned may or may not be categorised/ ranked depending on their suitability/ similarity to the given request. In resource rich, dynamic environments, where there is a large number of resources and where resources continually keep entering and leaving the environment, automatic resource matching is essential in facilitating potential resource seekers to find the resources to fulfil their requirements.

There are a number of mechanisms that have been designed to facilitate resource matching and discovery in a variety of domains, such as e-commerce, web services, grid computing and pervasive environments. These include approaches such as UDDI [129] which facilitates description and discovery of business services, MDS [34] which facilitates resource discovery on the Grid and Jini [6] and Salutation [111] which facilitates discovery of devices and their services within pervasive environments. Such conventional approaches to resource discovery and matchmaking typically use keyword based or attribute based structures to describe resources and the matching between resource requests and advertisements is done at a syntactic level based on the comparison of key-

words/ attributes or interfaces. The drawbacks of such approaches have been discussed previously in Chapter 2.

In semantic matching, the conceptual content of the advertisements and requests is considered in searching for compatibility rather than doing a string comparison or attribute-value based matching found in the traditional syntactic approaches. There are several recent research efforts in the area of resource matching that have provided important directions and contributions toward semantic service discovery; these research efforts, their limitations and unsolved problems have been discussed in section 2.3.

In this chapter we present the proposed semantic matching framework, which overcomes the limitations present in matching approaches based on subsumption reasoning alone (such as the approaches presented in [57] and [82]), and allows for match ranking depending on the suitability of the available advertisements to satisfy a given request. Section 4.2 presents the desirable features of a resource matching solution along with the motivating reasons behind them and highlights the intended contributions of our service matching framework. Then Section 4.3 presents a discussion that clarifies the position taken in comparing resource advertisements and requests which aids subsequent explanation of the semantic matching approach. Finally, Section 4.4.1 discusses the methodology behind the proposed semantic matching framework; it discusses the description of resource advertisements and requests and explains the matching mechanism in detail.

4.2 Motivation & Requirements

There are several characteristics that are desirable in a matchmaking solution which are discussed in this section along with the contributions of the proposed work.

4.2.1 Semantic Description and Matching Vs Syntactic Approaches

Considering device-oriented service discovery, several discovery mechanisms currently exist such as Jini [6] and Salutation [111], which were discussed in Section 2.3. In these approaches, the services are characterised either by using predefined service categories and fixed attribute value pairs (as in SLP and Salutation) or using interfaces (as in Jini). Such descriptions are inflexible and difficult to extend to new concepts and characteristics, and since these descriptions do not describe devices or services at a conceptual level, no form of inferencing can be carried out on them as pointed out in [25]. Therefore the matching techniques in these service discovery approaches (where a service request is matched with the available services to judge their suitability to satisfy the request), are limited to syntactic comparisons based on attributes or interfaces. Due to these reasons, the above mentioned discovery approaches cannot provide effective discovery of

devices and their services in a dynamic environment since they will fail to identify equivalent concepts (service requests and advertisements) which are syntactically different, or approximate matches that deviate from the service request in certain aspects.

An ontological approach for the description of services coupled with reasoning mechanisms to support service discovery and matching enables logical inferencing over the these descriptions and therefore offers several benefits over the traditional syntactic approaches, which includes:

Flexibility in the Description of Service Advertisements and Requests:

It is often the case, that the service providers usually describe devices in terms of lower-level properties, and the service seekers or clients usually prefer to describe service requests using more abstract or higher level concepts. This also agrees with the principle set out in [57], where they state that a requirement of a service description approach, is to allow the flexibility for the description to be more general or more specific.

To illustrate this fact, consider the case where a user seeks a computer with *Unix* operating system and the devices are advertised as having either *SunOS* or *Linux*. Although both *Linux* and *SunOS* are types of Unix operating systems, the traditional service description and discovery approaches fail to realise this. Hence a device described as having a *Linux* operating system will not be returned as a match for a request looking for a device with *Unix* operating system. In order to discover such services using the current approaches, the requester will have to look exhaustively for all possible combinations of *Unix*, which becomes very unwieldy in the case where there are a large number of possibilities; as highlighted in [128]. In an ontological approach where the specification of taxonomical knowledge is allowed, it could be specified that both *Linux* and *SunOS* are subconcepts of *Unix*; therefore a service request looking for a computer with a *Unix* operating system will be matched with any advertisements specifying the operating system as *Linux* or *SunOS*.

Another example is where a service seeker may wish to utilise a *widescreen* display, when the service advertisements for the devices describes only the (lower-level property of) *aspect ratio* for the display. The use of an ontology language (such as OWL) allows us to express *necessary and sufficient* conditions in the specifications so that it could be specified that displays with certain *aspect ratios* correspond to *widescreen* displays. This knowledge in turn can be used to infer (with the help of a logical reasoning mechanism) that a display with such *aspect ratios* 'must' be a *widescreen* display. Such knowledge cannot be specified in the syntactic approaches for service description (where flat structures or interface based descriptions are used or from object-oriented programming approaches); and inferencing between such properties and concepts cannot be done in the conventional matching mechanisms.

These examples shows the flexibility provided by an ontological approach for device

description, and the ability of a reasoning engine (coupled with such an ontological approach) to infer between such different representations of services thus providing effective discovery of services.

Automatic Classification of Devices:

As mentioned earlier services can be described in different levels of specificity. In pervasive contexts, devices can usually be categorised into a device type; a device can be either a printer, a display device, a computer and so on. However with devices offering more and more heterogeneous services, it can be difficult to assign a device to a particular category since they will be able to provide a variety of functionality.

Consider the case, where a service seeker, wishing to browse through the image files stored in a *SecureDigital Card* may raise a request for a *Display* device capable of reading *SecureDigital cards*. Although a *Photo Printer* with a *SecureDigital card* slot and a *Colour LCD display* may not necessarily be advertised as a *Display* device, it could provide the requested functionality. Hence by specifying the *necessary and sufficient* conditions that should be satisfied by a device to fall into a particular category, a discovery process supported by local reasoning can provide automatic classification of the devices thus facilitating the effective discovery of devices and their services.

Consistency Checking of Advertisement and Request Descriptions:

When services are advertised and requests raised, the descriptions of these advertisements and requests can be checked for consistency (against the reference ontology used to describe the services) with the help of logical reasoning mechanisms in the discovery procedure. This helps avoid any inconsistencies in the the service descriptions.

As emphasised in the above discussion, semantic approaches to service discovery can clearly provide many benefits over syntactic approaches. However, we have to bear in mind the fact that certain resources in pervasive environments (small mobile devices such as mobile phones and PDAs), are constrained in terms of computing power and memory. The standard semantic web tools and technologies on the other hand are memory intensive. Hence a feasible architecture has to be chosen for the discovery process, while facilitating the use of semantic descriptions and reasoning mechanisms to provide effective description and matching of services. For example, the matching process could always run centrally on the network and the devices could communicate through the network as appropriate.

4.2.2 Approximate Matching

Several related research efforts (For example [101] and [25]) in the past have identified the utility of approximate or flexible matching. In approximate matching, the matching

mechanism will recognise the resource advertisements that are not equivalent to the resource request, and thus may not be able to fully satisfy all the requirements of the request. An approximate or flexible matching process will not exclude such matches when returning the potential matches, but may include them depending on the degree of similarity to the resource request concerned. In the absence of available resources that exactly match the requirement, the resource seeker may be willing to consider such approximate matches, depending on the context involved. For example a resource seeker looking to print a certain document who requests for a Laser printer, may be satisfied with an Inkjet printer in the absence of a Laser printer; a resource seeker looking for a laptop with a 15 inch screen, may be satisfied with a 14 inch screen size. If such matches are excluded from the set of matches returned by the matching process, and if no exact matches are present, the requester will have to either; keep modifying the resource request in order to find resources that are suitable enough to meet his needs or, exhaustively list all the acceptable values for the properties concerned in the request.

It can often be the case that in some environments, resources that completely satisfy all the required properties of a request may be absent; but the resource seekers may be satisfied with a resource that is sufficiently close to the requirements. Hence the ability to find approximate matches is an important property in any matching mechanism.

There have been several semantic matching approaches that make use of description logics reasoning to provide flexible matches based on subsumption reasoning on taxonomies of concepts. However we argue that subsumption reasoning alone is not sufficient in providing approximate matches in certain cases; i.e. in certain situations subsumption reasoning is not effective in delivering appropriate approximate matches. For example, assume that we have the following request and the three advertisements:

Request: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasDiskSpace 100GB

Ad1: Computer with Processor Pentium3,
hasOperatingSystem WinXP,
hasDiskSpace 100GB

Ad2: Computer with Processor Pentium2,
hasOperatingSystem WinXP,
hasDiskSpace 90GB

Ad3: Computer with Processor Pentium2,
hasOperatingSystem WinXP,
hasDiskSpace 50GB

Intuitively Ad1 can be seen as the best match, Ad2 the next best and Ad3 the worst match. However if subsumption reasoning alone was used to approximate matches, all these three advertisements will be seen as *failed* matches[1] and it will be unable to distinguish between the suitability among these three advertisements. Thus to provide approximate matches effectively, a matchmaking approach that goes beyond subsumption reasoning is required. This aspect will be further discussed in the scenarios presented in Section 4.3.2.

4.2.3 Ranking of Potential Matches

Ranking is the ordering of the possible matching advertisements in the order of their suitability to satisfy the given request. An ordered list of services provides an important heuristic for the resource seeking agent to autonomously choose the best service possible [70].

When several potential services are available, a user may like to determine the 'appropriateness' of a service to suit his requirements. In order to do this the matchmaking engine will have to judge the 'closeness' or 'similarity' of the service advertisement and the service request and order the potential matches according to the closeness. Thus some form of ranking or ordering of the potential matches will be useful possibly with an assignment of a 'similarity score' to indicate the appropriateness of the service with respect to a given request. This is important since, in the absence of an exact match (an advertisement which satisfies a given request one hundred percent), the requester might be willing to consider other advertisements that are closer to the request and thus a measurement of the closeness of the advertisement and request will be extremely useful in gaining an understanding of the appropriateness of the advertisement to satisfy the request.

To illustrate the utility of ranking, consider the following example. Assume we have the following request for a certain computer and the three advertisements of available computers (in this context, USB2.0 and USB1.0 are both considered as sub classes of USB).

Request: Computer with Processor Pentium4,
hasOperatingSystem WinXP-Prof,
hasPhysicalPort USB2.0

[1]Taking into account the fact that Pentium2, Pentium3 and Pentium4 are disjoint concepts

Ad1: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasPhysicalPort USB2.0

Ad2: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasPhysicalPort USB

Ad3: Computer with Processor Pentium4,
hasOperatingSystem WinXP

Using the semantic matching approach proposed by Gonzalez-Castillo et. al. in [57], all three advertisements will be returned as matches, since they do not provide any match ranking or classification. In the approach proposed in [82], all three of the resource advertisements will be classified as Plug-In matches in their match classification scheme[2]. Hence, when the resource seeker gets the results of matchmaker, he will be unable to distinguish between the suitability of these three available resources; hence will have to look into the description of each advertisement in order to judge which one is best out of the three. Hence the availability/unavailability of a ranking mechanism, seriously affects the utility and usefulness of a matchmaker service. Ranking thus provides an important aid for the resource seeker, in gaining an understanding of the order in which he should consider the returned matching resources, so that he can start communicating/ negotiating with the relevant resource providers with a view to ultimately utilising the resource.

Most existing matchmaking solutions lack such a ranking facility as discussed in section 2.3. The proposed matchmaking framework provides a ranking mechanism so that the matches can be ordered or ranked by their suitability to satisfy a given demand or request. When ranking the available resource advertisements in relation to a given request, the matching mechanism will have adopt some methodology to measure the deviation between the resource request and the resource advertisement. Section 4.3 will discuss the issues that needs to be considered in the ranking process.

4.2.4 Considering Priorities on Individual Requirements

The current matchmaking research efforts do not consider any priorities or preferences that a user/agent may be having with respect to various aspects and properties of a service (except in [6]). In many practical scenarios certain requirements/ attributes in

[2]This approach classifies the matches into one of the classes of: Exact, Plug-in, Subsumes, Intersection or Disjoint and all the advertisements that are more general than the description provided in request, will be classified as Plug-In matches

a request will be more important than others, either due to the context involved or the subjective preferences of the user. In such cases, facilitating priority-handling in the matching process will produce match results that are more relevant and suitable for the context involved. For example consider two users looking for a printer; considering the time to service and quality properties of the printer, both may want to take the printouts as 'quick as possible' and with the 'highest quality possible'. But a user who wants to rush off to a meeting in the next five minutes will definitely be more concerned about the time factor and be willing to compromise on quality. But a user, who is working at leisure, will not mind waiting in order to obtain a more quality print. Thus in cases like this it is vital to consider the importance placed on the properties of the service by a user, by taking into account the priorities of the attributes.

Mandatory requirements or strict matching requirements have to be considered when, the resource seekers requires a certain individual property requirement in a request, to be strictly met by any potential resource advertisement; i.e. they will not want to consider any advertisements that will have even a minor deviation, with respect to that property. For example consider the case where a resource seeker needs to utilise a computer to run an application which will only run on the operating system WindowsXP, he will specify the operating system requirement in the request along with the other desirable characteristics. In the context involved the operating system property is a mandatory requirement and hence the resource seeker will not need to consider any available computers which deviates with respect to the operating system requirement (no matter how good it is with respect to other attributes). Hence this needs to be taken into account in the matching process and the available resources that deviate from this strict requirement must not be included in the result set (or ranked as the worst matches).

Priority matching is applicable when a resource seeker has varying importance placed on the individual property requirements of the request. Strict matching can in fact be considered as a specific case of priority matching.

This factor will be taken into account in the proposed work by giving a service requester the option of placing priorities/ weights on the specified attributes of the service request. These weights will be considered in the matching process during the ranking of advertisements.

4.2.5 Handling Unspecified Properties in Resource Advertisements

When the available services and service requests are described, the providers and requesters must have the flexibility to compose the service descriptions at varying degrees of completeness. Certain properties of a service advertisement may be left unspecified either because they are not known or because they are open for negotiation at a later

stage. Therefore service description languages and the matching process will have to deal with descriptions of varying specificity. The ability to handle such incomplete descriptions of advertisements, is pointed out as a requirement that should be present in a matchmaker by Noia et. al in [40] and they have taken this into account in their proposed matchmaker. When the potential matches (where some of the properties are left unspecified) are ranked; the ranking assigned depends on the number of unspecified properties in the service advertisement; the higher the number of unspecified properties, the lower the rank. This fact is not taken into account in the other matchmaking work.

In the proposed framework, if an advertisement fails to specify certain attribute(s) specified in the request, the score assigned to the particular advertisement will be reduced accordingly, but it will not be excluded from the set of matches.

4.3 The Basis for Matching Resource Advertisements and Requests

During service discovery, the resource seeker will provide a description of the requirements and characteristics that should be met by the ideal resource in the *Resource Request*. The resource provider will provide a description of the characteristics of the available resource in the *Resource Advertisement*[3]. During the matching process, the matching engine will have to judge the extent to which an *Advertisement* can satisfy a given *Resource Request*. It will have to adopt an appropriate criterion to judge suitability of advertisements when they do not exactly match the request under concern.

In this section we present an analysis of the abstract approach used to judge the suitability of an advertisement in relation to a given request. This analysis will aid the subsequent justification and explanation of the methodology behind the proposed matching framework.

4.3.1 Comparison of Advertisements and Requests

In this section we will discuss and clarify the position taken in comparing a resource request and an advertisement. As will be discussed in Section 4.4.1 in detail, a resource request and an advertisement will be described as a conjunction of the characteristics or the properties involved. We will use the following example scenarios in this discussion.

Case 1:

Request: Computer with,
MemorySize > 2GB

[3]Resource description will be discussed in more detail in Section 4.4.1

Advertisement:
Computer with,
OperatingSystem Linux,
MemorySize > 2GB

The given request can be seen from two different perspectives:

- View Point 1: *"I want **computers** with Memory Size > 2GB."* In this case the requester is seeking 'the set of all computers that has Memory Size > 2GB', and the Advertisement is giving 'Linux computers that has Memory Size > 2GB', which can be viewed as a subset of what is wanted.[4]
- View Point 2: *"I want **a computer** with Memory Size > 2GB."* In this case the requester is seeking 'a computer that has Memory Size > 2GB' and he/ she does not care what the other properties of the computer are. The Advertisement is giving 'Linux computers that has Memory Size > 2GB', which can be viewed as fully satisfying the requirement (although it is a more specific concept than the request).

Case 2:

Request: Computer with,
OperatingSystem Linux,
MemorySize > 2GB

Advertisement:
Computer with,
MemorySize > 2GB

In this case the two view points pertaining to the given request are:

- View Point 1: *"I want **computers** with Linux operating system and Memory Size > 2GB."* In this case the requester is seeking 'the set of all computers that has Linux and Memory Size > 2GB', and the Advertisement is giving 'computers that has Memory Size > 2GB'; which can be viewed as giving all that is needed and more - a superset of what is wanted[5].

[4]See Figure 4.1 which is an example illustration of the two sets corresponding to Advertisement and Request. As the figure shows, the set corresponding to $(> 2GB)$ contains a, b, c, d, e and the set $(Linux, > 2GB)$ contains d, e which is a subset of the former.

[5]Again, the Figure 4.1 which gives an example illustration of the two sets corresponding to Advertisement and Request, will help to clarify this view.

- View Point 2: *"I want **a computer** with Linux operating system and Memory Size > 2GB."* In this case the requester is seeking 'a computer that has Linux and Memory Size > 2GB'. He or she is concerned that the available advertisement satisfies both these characteristics. The Advertisement is giving 'computers that has Memory Size > 2GB'. We do not know if the advertisement has the Linux operating system or not, since it is not specified. Therefore the advertisement can be viewed as only partially satisfying the requirement (it is a more general concept than the request).

In our semantic matching approach, the position we take corresponds to *View Point 2*. It is important to understand that in this case a resource requester is looking for a single resource (described by an advertisement) that can satisfy the specified characteristics (as described by the request); and the advertisement is an abstract description of a single individual or a number of individuals that share the same characteristics. This contrasts with the view that can be taken by a web service matcher for example as is evident from the discussions provided in [82, 117]. For instance, a service seeker may want 'a Vendor that sells computers with the Linux operating system' and an advertisement of a seller that sells 'computers with the Linux operating system and Memory Size > 2GB' actually gives a subset of what is wanted. This corresponds to View Point 1.

The same argument applies when the advertisement and request refers to atomic concepts. For example, similarly to the scenario in Case 1: we can have $Request = Windows$, $Advertisement = WindowsXP$; where $WindowsXP$ and $Windows$ are both $OperatingSystems$ and $WindowsXP$ is a subconcept of $Windows$. i.e. $WindowsXP \sqsubseteq Windows \sqsubseteq OperatingSystem$. In this case, considering View Point 2: advertisement fully satisfies the request since $WindowsXP$ is a type of $Windows$.

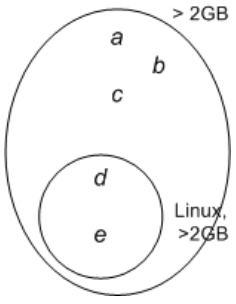

FIGURE 4.1: Extension of the Advertisement and Request Concepts

4.3.1.1 Extension and Intension of Concepts

We can relate the above discussion and the position taken in comparing a request and an advertisement, to the extension and the intension [106, 51] of the request and advertisement concepts. The set of formal objects of a formal concept is called its "extension";

the set of formal attributes is called its "intension" [106]; i.e. the extension of a concept refers to the actual instances or individuals of the concept and the intension refers to the attributes or the properties that the concept represents.

Let us assume that the individuals belonging to the request and advertisement concepts discussed in the previous two cases, are as illustrated in Figure 4.1; i.e. the individuals a, b, c, d and e belong to the concept *"Computers with Memory Size greater than 2GB"* and the individuals d and e belong to the concept *"Computers with Linux and Memory Size greater than 2GB"*. In Case 1: the extension of the Advertisement (d, e) is only a subset of the extension of the Request (a, b, c, d, e), thus it can be viewed as the advertisement only partially satisfies the request. In Case 2: the extension of Advertisement (a, b, c, d, e) is a superset of the Request (d, e), and hence one can view as the advertisement fully satisfies the request. Thus viewing in relation to the extension of the concepts, one ends up with view point 1. View point 2 (which we take in the proposed semantic matching approach) on the other hand, can be seen as viewing along the lines of the intention of the concepts.

4.3.1.2 Precision and Recall of Concepts

We can also relate the discussion regarding the comparison of a request and an advertisement to *Precision* and *Recall*, which are two widely used metrics for performance evaluation in Information Retrieval systems [9]. *Precision* is the proportion of objects retrieved that are actually relevant to the user's information need (relevant and retrieved/ retrieved). *Recall* is the proportion of objects that are relevant to the query that are successfully retrieved (relevant and retrieved/ relevant).

In Case 1: the Advertisement description covers only 'relevant' individuals with respect to the Request, thus the $precision = 1$; but it does not cover all the 'relevant' individuals and therefore $recall < 1$. In Case 2: the Advertisement description covers all the 'relevant' individuals with respect to the Request, therefore $recall = 1$; but since some of the individuals it covers are irrelevant, $precision < 1$.

However as emphasised earlier, in the proposed semantic matching framework, the Advertisement represents a single available resource or a number of resources that share exactly the same properties; it is a description of the characteristics of the resource(s). The Request, is a description of the required characteristics of the resource being sought. Therefore as per View Point 2: for Case 1 - the Advertisement fully satisfies the Request; thus the match score for the advertisement = 1 (the precision is also = 1 in this case). For Case 2 - the Advertisement only partially satisfies the Request and thus the match score < 1 (the precision is also < 1). Hence the proposed semantic matching framework can be seen as an approach that recognises *precision* rather than *recall*.

Note here that in computing the match score, we want to find out the likelihood of

satisfying the Request, given that what we have is the Advertisement. This can be seen as the conditional probability: given the fact that what we have 'belongs' to the Advertisement (this is what is retrieved), what is the probability that it also 'belongs' to the Request (this is what is relevant). As pointed out by van Rijsbergen et. al. in [137], Precision is an estimate of the conditional probability that an item will be relevant, given that it is retrieved. Therefore, the match score as per View Point 2 in fact represents Precision.

4.3.2 Judging Similarity within Attributes

The resource seeker will specify in the request, a number of properties that should be met by an advertisement for the requirement to be satisfied. The advertisement may not have the same properties exactly as specified in the request. Thus the matching process will have to determine the semantic distance between the advertisement and a request, depending on how much the advertisement deviates from the request.

As will be discussed in Section 4.4.1, a resource request will typically consist of several individual requirements to be satisfied. The description of an individual requirement will include the property or attribute the requesters are interested in and the ideal value desired. The resource provider will specify all the relevant characteristics of the available resource in the resource advertisement. To determine the distance between the request and the advertisement, the matching engine will check how similar the advertisement is with respect to each individual requirement specified in the request. Similarity judgement within each requirement will depend on the type of property or attribute it involves. In analysing the different types of attributes occurring in the resource descriptions, they can be categorised broadly into two types: Symbolic attributes and Numeric attributes. For example, considering a description of a computer, the *disk space* and *memory size* are attributes that will have numerical values (integers and decimals); thus these will be classified as *Numeric* attributes. On the other hand, *Operating System* and *Processor Type* are attributes which will have concepts as their values; and hence these will be classified as *Symbolic* attributes.

In most semantic matching approaches [82, 101, 100], the similarity between these concepts have been determined using subsumption reasoning based on the taxonomic relation between the concepts. However we argue that subsumption reasoning alone is not sufficient in determining similarity for the purpose of resource matching. Depending on the concept involved, reasoning based on the taxonomy alone, will not accurately reflect the similarity between concepts. We show this by example in the following discussion; we give separate examples for the symbolic attributes and numeric attributes.

4.3.2.1 Symbolic Concepts

Consider the concept *Processor*. The part of the taxonomy for processor is illustrated in Figure 4.2. Assume there is a request for a computer that has processor as *Pentium4* and that there are three advertisements of computers with processors *Pentium3*, *Pentium1* and *AthalonXP*. When matching using subsumption reasoning alone, all these three advertisements will be classified as *Failed* matches since *Pentium3*, *Pentium1* and *AthalonXP* are all disjoint with *Pentium4*. However if a domain user is to do the matching, the computer with *AthalonXP* will be viewed as the best match [6], computer with *Pentium3* will be viewed as the next best and the computer with *Pentium1* will be viewed as the worst match[7]. The same observation can be made with respect to *Display Technology* concept as well (see Figure 4.3); although CRT, LCD and Plasma technologies are all disjoint concepts, Plasma and LCD concepts are more similar to each other than the CRT display. Hence a request for a LCD display will prefer the Plasma display to a CRT display.

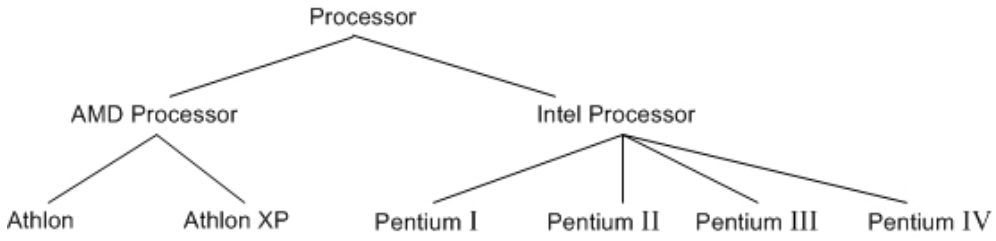

FIGURE 4.2: The Processor Taxonomy

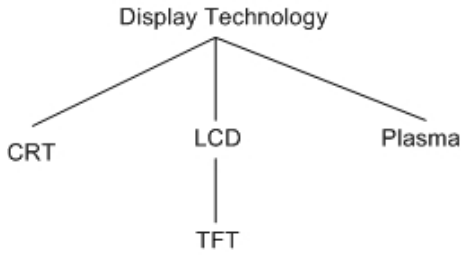

FIGURE 4.3: The Display Technology Taxonomy

Therefore this shows that, although the necessity of approximate matching is emphasised in semantic matching, the use of subsumption reasoning is not sufficient in providing effective approximate matches in certain cases. Hence some other measure of semantic similarity has to be adopted to aid the matching process, when such concepts are involved.

Semantic similarity can refer to a variety of meanings; for example semantic similarity measurements referring to linguistic similarity [112, 138], are used in information re-

[6]Since *AthalonXP* has a comparable performance to *Pentium4*

[7]These observations agree with human perception as shown in the studies carried out in the evaluation of this research, discussed in Chapter 6.

trieval tasks [9]. There are other perspectives of similarity measurement as described in [52]. But in the case of resource matching we are interested in similarity between concepts, from the perspective of the utility that they can provide. For example, when the processor is considered; *Pentium4* is viewed as being more similar to *Pentium3* than *Pentium1*, because *Pentium3* is closer in performance level to *Pentium4*, due to the features of Clock Speed and Cache Size. Hence a resource seeker requesting a computer with *Pentium4* is more likely to accept a computer with *Pentium3* (than a computer with *Pentium1*) due to the relative utility it can provide. Therefore we have to choose a similarity measure to judge similarity between such concepts, based on the features they have.

Several proposals for measuring concept similarity exist; Goldstone and Son in [52] and Borgida et. al. in [17] provides a survey of some of the existing proposals. For example Tversky et. al. in [131] has proposed a feature-based metric of similarity, in which common features tend to increase the perceived similarity of two concepts, and where feature differences tend to diminish perceived similarity. For instance, Tomato and Cherry are similar by virtue of their common features Round, Fruit, Red and Succulent. Likewise, there are dissimilar by virtue of their differences, namely Size (Large versus Small) and Seed (Stone versus No Stone). Hence in our work, if we wanted to find similarity between different Processor Types for example, the features/ properties of the Processors such as Clock Speed and Manufactured-By will have to be used in measuring the similarity.

However there are concepts where subsumption reasoning based on the taxonomy is sufficient for the matchmaking task. For example consider the *Storage Medium* for which the taxonomy is illustrated in Figure 4.4. If there is a request for a computer that can read a *Memory Stick* and there are three available computers that can read a *SD Card*, *USB Stick* and *Floppy Disk*: all the three available computers will be ranked accordingly in this case. The fact than the *SD Card* can be more similar to the *Memory Stick* than a *Floppy Disk* does not have any bearing in this situation. Therefore the domain expert defining the ontology will have to decide, whether a taxonomic relation is sufficient for reasoning with the attribute concerned or if another similarity measure has to be used in the matching process, depending on the concept involved.

4.3.2.2 Numeric Concepts

Consider the attribute of *Disk Space* in the description of a computer which will have a numeric value. Assume there is a resource request for a computer with *Disk Space* at least *100GB* and three available computers with *Disk Space* of *100GB*, *90GB* and *40GB* respectively. When using subsumption reasoning alone, computer with *Disk Space* 100GB will be returned as an *exact* match and both computers with *90GB* and *40GB* disk space will be returned as *failed* matches. i.e. when using subsumption reasoning

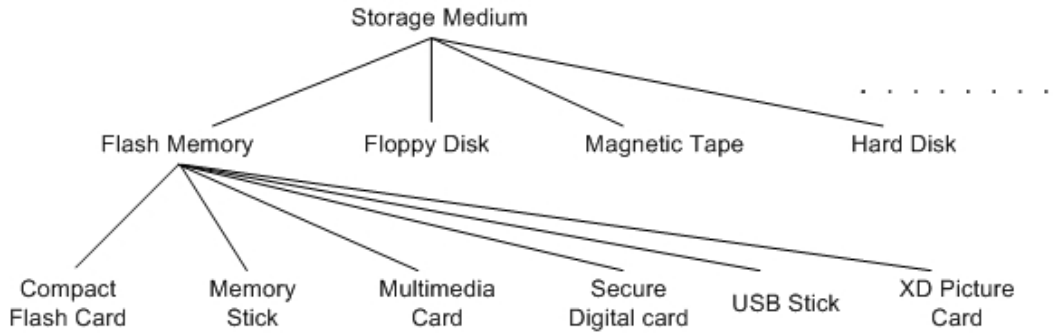

FIGURE 4.4: The Storage Media Taxonomy

alone, it is unable to distinguish between the computers with *Disk Space* of *90GB* and *40GB*. However in the absence of an exact match, a domain user is more likely to accept the computer with *Disk Space* of *90GB* because it is closer to the requested value of *Disk Space*. Hence when using subsumption reasoning alone, it is unable to approximate the numeric attributes effectively, to provide matches that agrees well with human perception [8].

When available resources fail to meet requested characteristics with respect to numeric attributes, it could be observed that domain users tend to evaluate the suitability of the available resources in proportion of the violation of the requested numeric constraint. This is similar to the principle behind fuzzy logic [143]. Thus to provide approximate matches that agree with human perception, an approach of reasoning along this principle will have to be used. Thus numeric constraints specified will be considered as fuzzy boundaries and the deviation with respect to the specified constraint can be evaluated with the use of a fuzzy membership function. The methodology followed in finding approximate matches in the presence of Numeric attributes will be discussed in more detail in Section 4.4.2.3.

Considering the above discussion; for the purpose of approximating and judging similarity, the attributes or properties in a resource description are categorised into three types. Namely:

- Type 1: Properties involving symbolic concepts for which judging similarity using the taxonomic relation is sufficient. For example the *Storage Media* concept discussed above fall into this category. In this case the matching engine will make use of a reasoner to judge the similarity by subsumption relation.
- Type 2: Properties involving symbolic concepts for which judging similarity using the taxonomic relation is not sufficient. For example *Processor* and *Display Technology* concepts discussed above fall into this category. In this case another

[8] Again, these observations are shown to agree with human perception in the studies carried out in the evaluation of the proposed matching work, discussed in 6.

method other than subsumption have to be used to judge similarity between these concepts.

- Type 3: Properties which are numerical. In this case if the attribute values in the request and advertisement do not match exactly, the similarity will be judged based on the level of deviation between the requested and specified values.

The detailed approach followed in judging similarity between concepts (in each of the above three categories) when finding approximate matches, is discussed in more detail in Section 4.4.2.

4.4 Methodology

In this section we discuss the methodology behind the proposed semantic matching framework; we discuss the description of the resource advertisements and requests, the process involved in match ranking and how similarity within attributes are judged.

4.4.1 Description of Advertisements and Requests

For the matching process to succeed resource advertisements and requests must be described in an appropriate manner. The expressivity of the description language affects how the service seekers and providers can describe the resource requests and advertisements, which in turn affects how the requests and advertisements are matched. The relevant properties, capabilities and functionality of the resource, relevant to the domain should be described adequately in a suitable description language. The desirable features of a resource description language and the drawbacks of conventional syntactic approaches for resource description have been identified in Chapter 2. To reap the benefits of semantic matching, the resources must be described in a language that facilitate the use of logical reasoning, such as OWL [140] and FLogic [3]. In our framework, we consider resources that are described in the Web Ontology Language (OWL). OWL has a wide range of tool support for editing, parsing and reasoning; and is widely accepted as being suitable for the semantic web since it is open, extensible and is compatible with the architecture of the World Wide Web [96].

We assume that the same ontology is being used to describe both the resource advertisement and the resource request. Although there are several solutions for cross ontology mapping such as the approaches proposed in [73, 92], this is still the subject of much research and there is no widely accepted solution. Therefore an OWL ontology that describes the resources in the domain concerned will be used to describe both the requests and the advertisements. In this research, we apply matching in pervasive environments, where the resources concerned are devices and the services offered by them. Hence we

use the Device Ontology discussed in chapter 3 for the description of resource requests and advertisements.

A request will typically consist of several sub requirements to be satisfied. Each individual requirement will specify:

- The description of the requirement, which is the resource characteristic the resource seekers expect in a resource, for the their needs to be satisfied.
- The priority or weight of that individual requirement, which will be a decimal value that indicates the relative importance of the particular requirement.

The priority value can also be used to indicate if the requirement considered is a mandatory requirement; i.e. if the requirement should be strictly satisfied in an advertisement for the requester to consider it as a potential match. The description of an individual requirement will include the property or attribute the requesters are interested in and the ideal value desired.

The request will take the form of:

$$Request \equiv (Req_1) \sqcap (Req_2) \sqcap \cdots \sqcap (Req_n)$$

where Req_i is an individual requirement. The requirement in turn can take the form of:

$$Req \sqsubseteq (= 1 hasDescription.RD) \sqcap (= 1 hasPriority.PriorityValue)$$

where RD is the requirement description, which can be either a named concept or an existential restriction of the form, $\exists p.C$ where p is a role and C is a named concept or a complex concept. For describing each RD, an ontology that describes the services in the domain concerned can be used. The $PriorityValue$ indicates the relative importance of the individual requirement in the request. This is a decimal value defined between 0 and 1. In addition, to indicate that the requirement is a mandatory requirement that must be strictly met in any potential match, the $PriorityValue$ is defined as 2. The resource seeker must pick the appropriate $PriorityValue$ (according to these pre-defined values) for each individual requirement, to indicate its relative importance.

For example, assume there is a request for a Printer that has Print Technology as *Laser* and supports Paper Size *A2*. Let us say that the Paper Size requirement has a higher priority (with assigned priority value = 0.8) over the Print Technology requirement (with assigned priority value = 0.2). Since the request is for a *Printer*, this is stated as a mandatory requirement with priority value 2.0. Then the request can be expressed in description logic notation as:

$$
\begin{aligned}
Request \sqsubseteq \exists\ hasRequirement\ (Requirement \sqcap \\
&\exists hasPriority. = 2.0\ \sqcap \exists hasRequirementDescription.RD1) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \\
&\exists hasPriority. = 0.6\ \sqcap \exists hasRequirementDescription.RD2) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \\
&\exists hasPriority. = 0.2\ \sqcap \exists hasRequirementDescription.RD3) \sqcap
\end{aligned}
$$

$RD1 \equiv Printer$

$RD2 \equiv \exists\ hasPaperSize\ .\ A2$

$RD3 \equiv \exists\ hasPrintTechnology\ .\ Laser$

The resource provider will specify all the relevant characteristics of the available resource in the resource advertisement. The advertisement can take the form of:

$$Advertisement \equiv (r_1) \sqcap (r_2) \sqcap \cdots \sqcap (r_n)$$

where r_i is either a named concept or an existential restriction (of the form $\exists p.C$ where p is a role and C is a named concept or a complex concept) describing a characteristic of the resource.

For example, an advertisement for a printer that supports paper size *A2*, has print technology *Laser* and has printing colour *Black & White* can be described as:

$$
\begin{aligned}
Advert1 \sqsubseteq\ & Printer \sqcap \\
&\exists hasPaperSize\ .\ A2\ \sqcap \\
&\exists hasPrintTechnology\ .\ Laser \sqcap \\
&\exists hasPrintingColour\ .\ BW
\end{aligned}
$$

4.4.2 Judging Semantic Similarity

As discussed in 4.3.2, we distinguish between three types of concepts or properties occurring in the individual requirements of a resource description for the purpose of semantic matching. These are:

- Type 1: Named concepts for which judging similarity using the taxonomic relation is sufficient. In this case the matching engine will make use of a reasoner to judge the similarity by subsumption relation.

- Type 2: Named concepts for which judging similarity using the taxonomic relation is not sufficient (disjoint concepts). In this case another method other than subsumption may be used to judge similarity between these concepts.
- Type 3: Constraints on Datatypes. In this case if the attribute values in the request and advertisement do not match exactly, the similarity will be judged based on the level of deviation between the requested and specified values.

In this section, we discuss in detail how similarity is determined within each of these types during the matching process.

4.4.2.1 Type 1: Named Concepts having a Taxonomic Relation

When two concepts (C_R, C_A) are related through a taxonomy, the subsumption or taxonomic relation between these two concepts can fall into one of five categories. Assuming C_R is the requested concept and C_A is the advertised concept; these categories and the similarity between the concepts in each case are:

- C_R and C_A refer to the same concept or they are equivalent: In this case the similarity between C_A and C_B is obviously 1.
- C_R is a super concept of C_A: In this case since C_A is the subconcept, intuitively it possesses all the features desired of C_R and therefore Similarity(C_R,C_A) = 1.0[9].
- C_A is a super concept of C_R: In this case since C_A is the super concept, it may not have all the features desired of C_R and therefore C_A cannot fully satisfy the requirement. Hence Similarity(C_R,C_A) = t where $t \in [0, 1]$ [10]. The approach used for determining the value of t will be discussed later in this section.
- C_R and C_A intersect: In this case since the two concepts intersect, a given instance of C_A can also belong to C_R with some probability. Therefore there is some probability of the requirement C_R being satisfied by the concept C_A. Hence Similarity(C_R,C_A) = r where $r \in [0, 1]$.
- C_R and C_A are disjoint: In this case Similarity(C_R,C_A) = 0, since the concepts are disjoint; however, other associations between the two concepts may exist in certain situations (as discussed in Section 4.3.2) depending on the concepts involved. In that case another similarity measure must be applied here, which will be discussed in Section 4.4.2.2

The possible taxonomic relations and the similarity scores assigned in each case are summarised in Table 4.1.

[9] See the discussion in Section 4.3.1

[10] Again the discussion in Section 4.3.1 will clarify the position taken in this case

Taxonomic Relation Between C_R and C_A	**Similarity Score**
$C_A \equiv C_R$	1.0
$C_A \sqsubseteq C_R$	1.0
$C_R \sqsubseteq C_A$	t (where $t \in [0,1]$)
$\neg(C_R \sqcap C_A \sqsubseteq \bot)$	r (where $r \in [0,1]$)
$(C_R \sqcap C_A \sqsubseteq \bot)$	0.0

TABLE 4.1: Assignment of similarity scores when Subsumption Relation is considered.

Determining the Values of t and r:

For the two cases when C_A is a super concept of C_R and when C_R and C_A intersect; the similarity between the two concepts will be a value between 1 and 0. In this case we have to judge the similarity based on the probability of satisfying the given requirement. i.e. given that what is available is C_A, we have to judge the likelihood that it is also a C_R.

There have been a number of approaches for determining similarity between concepts in a taxonomy [110, 84], that are based on probability. Since the exact number of instances belonging to the classes in a taxonomy are not known; these approaches take into account the fact that, the number of instances of a class are inversely related to the depth of the class in the hierarchy; i.e. the number of its superclasses or ancestors. Based on this assumption, Skoutas et.al. [117] have provided an estimation for the similarity between two concepts C_R and C_A (the values for t and r in this case) as:

$$t \quad | \quad r \quad = \frac{|A(C_A) \cap A(C_R)|}{|A(C_R)|} \tag{4.1}$$

where $A(C)$ denotes the set of superclasses of a class C. Note that in the case when $C_R \sqsubseteq C_A$; $|A(C_A) \cap A(C_R)| = |A(C_A)|$. Therefore $t = \frac{|A(C_A)|}{|A(C_R)|}$.

Hence Similarity Score for two concepts C_R and C_A can be determined as:

$$SimilarityScore(C_R, C_A) = \begin{cases} 1 & if\ C_A \equiv C_R \\ \frac{|A(C_A)|}{|A(C_R)|} & if\ C_R \sqsubseteq C_A \\ 1 & if\ C_A \sqsubseteq C_R \\ \frac{|A(C_A) \cap A(C_R)|}{|A(C_R)|} & if\ \neg(C_R \sqcap C_A \sqsubseteq \bot) \\ 0 & if\ C_R \sqcap C_A \sqsubseteq \bot \end{cases} \tag{4.2}$$

4.4.2.2 Type 2: Named Concepts not having a Taxonomic Relation

As emphasised in Section 4.3.2; there may be certain classes of concepts where although no subsumption relation exists between them, some concepts can be thought of as being 'more closer or similar' to another concept than the rest. When properties involve such

concepts, some other method will have to be sought to find the similarity between such concepts.

Examples where such knowledge will be necessary are when reasoning with concepts such as Processor Type (e.g. Pentium 3, Pentium 4, and Athlon), Display Type (e.g. CRT, LCD and Plasma) or Paper Size (e.g. A0, A1, A2 and B1). Let us say that a service requester is looking for a computer with a Pentium 4 processor; how can we rank service advertisements having Pentium 3, Celeron and AMD Athlon processors as their processor type? In this case we have to use some similarity measure that indicates the closeness between the concepts (the different processor types in this example) in order to assign a sub-score with respect to the processor type requirement and thereby match the request and advertisement.

Several proposals for measuring concept similarity exist; Schwering in [116] and Borgida et. al. in [17] provides an overview of some of the existing approaches. For example Tversky et. al. in [131] has proposed a feature-based metric of similarity, in which common features tend to increase the perceived similarity of two concepts, and where feature differences tend to diminish perceived similarity. For instance, Tomato and Cherry are similar by virtue of their common features Round, Fruit, Red and Succulent. Likewise, there are dissimilar by virtue of their differences, namely Size (Large versus Small) and Seed (Stone versus NoStone). Hence in our work, if we wanted to find similarity between different Processor Types for example, the features/ properties of the Processors such as Clock Speed, Cache size and Manufactured-By will have to be used in measuring the similarity.

However measuring similarity between concepts is not within the scope of the current research and we assume that the knowledge of concept similarities between such concepts is available to the semantic matcher (either measured by using a third party approach for semantic similarity measurement or available as domain knowledge). This knowledge will then be used during the matching process by the semantic matcher, to obtain similarity values between Type 2 concepts.

Hence for the purpose of matching, Similarity Score for two Type 2 concepts C_R and C_A can be determined as:

$$SimilarityScore(C_R, C_A) = ConceptSimilarity(C_R, C_A) \tag{4.3}$$

4.4.2.3 Type 3: Constraints on Datatypes

As discussed previously in the motivating scenarios in Section 4.3.2, when available resources fail to meet requested characteristics with respect to numeric attributes, the domain users tend to evaluate the suitability of the available resources in proportion

to the violation of the requested numeric constraint. For instance if a resource seeker requires a computer with a *memory size of 1GB*, and there are two available advertisements of computers with *memory size of 512MB* and *256MB* these two advertisements both fail to meet the requirement set by the resource seeker. If only DL subsumption reasoning is used, both will be classified as failed matches. However, for effective approximate matching, they must be distinguished for the level of deviation from the original request and penalised accordingly during the matching process; i.e. the second advertisement (with the 256MB memory size) must be ranked lower when ranking.

Thus, when judging the similarity within individual requirements that involve numeric or datatype properties, the similarity measure has to be a judgement of the extent to which an available numeric value (in an advertisement) can satisfy the requested datatype criterion specified in a request. i.e. if a restriction '$>$*20*' applies, how well would values of '*21*', '*18*' and '*15*' satisfy this constraint? Assuming that is a flexible or imprecise criterion, intuitively we could say that '*21*' definitely satisfies the constraint and '*18*' and '*15*' satisfy the constraint only to a certain degree. Dealing with such cases of imprecision and vagueness is the principle behind fuzzy logic [143] introduced by Zadeh.

There have been many motivating scenarios in a variety of application domains, discussed in the literature [114, 121, 75] that stress the need for dealing with fuzziness and imprecision in the Semantic Web and description logics. Straccia in [122, 123, 124] has presented a fuzzy description logic that combines fuzzy logic with description logics. Typically, DLs are limited to dealing with crisp concepts; an individual is either an instance of a concept or it is not. In Fuzzy description logics, the concepts can be imprecise and thus an individual can belong to a concept only 'to a certain degree'; it allows for expressions of the form $\langle C(a)n\rangle, (n \in [0,1])$ which means 'the membership degree of individual a being an instance of the concept C is at least n'. For example, there can be a concept $Tall$ and an individual *tom* can belong to the concept $Tall$ to a degree of at least 0.7.

However, unlike in the domain described by [122, 124], the knowledge base dealt with in the proposed semantic matching framework is not fuzzy. i.e. it contains precise knowledge and crisp concepts. For example concepts such as Computer, Processor, Pentium4 are all crisp concepts and an individual is either an instance of such a concept or it is not. Also, the resource requests or advertisements do not contain any fuzzy predicates such as *Large Memory* and *High Capacity Disk*, but specify precise concepts or data values.

However in approximate matching, when judging similarity within individual requirements of a request that involves constraints on datatypes, it is desirable to consider these as soft constraints as already emphasised. Therefore, we consider the relevant data range restrictions to be fuzzy concepts or fuzzy boundaries and follow the approach discussed

in fuzzy description logic [124] when determining similarity between the required and the available property values. The approach followed in determining similarity within requirements that involve datatype property restrictions, is discussed in detail below.

Datatype constraints can be an exact, at least, at most or a range restriction. For example $\geq_{18}$ is a crisp predicate over the natural numbers denoting the set of integers greater than or equal to 18 and

$\geq_{18}(x) = \quad \{1 \quad if \quad x \geq 18, \quad 0 \quad otherwise$

In fuzzy description logics, this will be considered as a fuzzy domain predicate and one of the many membership functions [42, 79] in fuzzy set theory can be used to determine the degree of membership of a given x in the fuzzy set $\geq_{18}$. The membership functions that we use for the purpose of specifying the membership degrees are given in Equations 4.4, 4.5, 4.6 and 4.7.

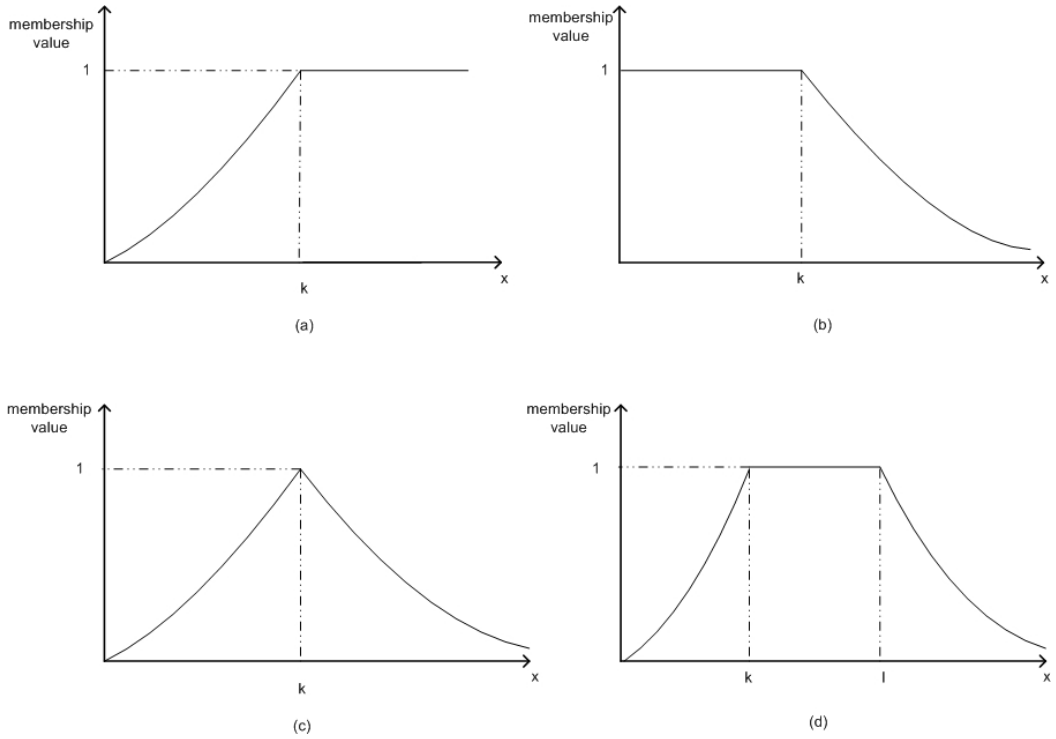

FIGURE 4.5: Fuzzy Membership Functions for Numeric Attribute Ranges

These functions can be defined as follows: let k and l be constants ($k < l$), then

$$\geq_k (x) \quad = \begin{cases} 1 & if \quad x > k, \\ 1 - \frac{|k-x|}{k} & otherwise \end{cases} \tag{4.4}$$

$$\leq_k (x) \quad = \quad \begin{cases} 1 & if \quad x < k, \\ 1 - \frac{|k-x|}{k} & if \quad k < x < 2k, \\ 0 & otherwise \end{cases} \tag{4.5}$$

$$=_k (x) \quad = \quad \begin{cases} 1 & if \quad x = k, \\ 1 - \frac{|k-x|}{k} & if \quad 0 < x < 2k, \\ 0 & otherwise \end{cases} \tag{4.6}$$

$$\geq_k, \leq_l (x) \quad = \quad \begin{cases} 1 & if \quad k < x < l, \\ 1 - \frac{|k-x|}{k} & if \quad x < k, \\ 1 - \frac{|l-x|}{l} & if \quad l < x < 2l, \\ 0 & otherwise \end{cases} \tag{4.7}$$

Thus datatype constraints specified will be considered as fuzzy boundaries and the deviation with respect to the specified constraint can be evaluated using the fuzzy membership functions defined above.

A constraint for a datatype property in a requirement ($c_{k,l}$) can take the form of ($= k$), ($\geq k$), ($\leq k$), or ($\geq k \sqcap \leq l$) for given constants k and l. If the value for the same datatype property in the advertisement is specified as v; then the similarity score between a constraint $c_{k,l}$ and v (indicating how well v satisfies the required constraint $c_{k,l}$) can be determined as:

$$SimilarityScore\ (c_{k,l}\ ,\ v) = \mu(v; k, l)$$

where μ denotes the membership function and
$\mu(x) \in \{\geq_k (x), \quad \leq_k (x), \quad =_k (x), \quad \geq_k, \leq_l (x)\}$

Representing Restrictions on Datatypes

The description of resource requests and advertisements for the purpose of semantic matching has been previously discussed in Section 4.4.1. As already mentioned, the request will specify the resource characteristics the resource seeker expects in a resource, for his needs to be satisfied. This will include the resource's properties the seeker is interested in and the ideal values or range of values desired of them.

The individual requirements of a request can concern either object properties or datatype properties. When the individual requirement of a request concerns datatype properties, the desired values or value ranges can have restrictions that involve mathematical inequality operators such as $<$, $\leq$, $>$ and $\geq$. For example a resource seeker may be looking for a Printer that supports a *Printing Resolution greater than 600*. Such restrictions on the properties of a resource must be essentially represented in the resource description for effective semantic matching to take place.

The OWL Web Ontology Language [140] which is the chosen resource description language in this framework, provides considerable expressive power to the Semantic Web. It

provides a number of OWL built-in datatypes [11] such as integer and float, and thereby facilitates the expression of datatype properties. However, OWL datatype formalism is insufficient for many applications as extensively discussed in [98, 99, 97]. In many applications, it is often necessary for users to be able to define their own datatypes or datatype predicates. For example in a service discovery scenario, a resource seeker may want to describe a request for a computer that has *disk space greater than 60 Gbytes* and *memory size greater than 1GB*.

The recent extension of the OWL language - OWL 1.1 [102], facilitates user-defined datatypes; it provides methods to define a data range by applying facets such as *maxExclusive* and *minExclusive* (as defined in the XML Schema Datatypes Specification [14]) to a particular data type. The approach for defining Data Ranges and Datatype Restrictions are discussed in detail in [93]. We follow this approach in describing the restrictions on datatype properties that occur in the resource descriptions.

4.4.3 Matching Process

The matching engine will compare the request with each available advertisement and depending on the suitability of the advertisement to satisfy the request, a score will be assigned. Depending on the score received by each advertisement, the advertisements can be ranked so that the resource seeker will know the order in which he should consider the available resources. As discussed in section 4.4.1, the resource request will described the characteristics or attributes that should be met by a potential resource in order for the request to be satisfied; it will consist of a number of individual requirements along with their priority values. The presence of any mandatory requirements that must be fully satisfied by any potential match will also be indicated by using the appropriate priority value as mentioned in 4.4.1.

The high-level algorithm for the matching process is illustrated in Section 4.4.3.1. During the matching process, the available resource will be checked to see if each mandatory individual requirement (RD) is satisfied in the advertisement description. If all the mandatory requirement(s) are met, then the advertisement will be evaluated through approximate matching.

In approximate matching, the available resources should be evaluated according to how well it satisfies each individual requirement specified in a request; i.e. the matching engine should quantify the extent to which each individual requirement description (RD) is satisfied by the resource advertisement. For this, the matching engine will check how similar the advertisement is with respect to each non-mandatory requirement (RD) specified in the request; the similarity will be determined depending on the semantic deviation of the expected value in request and the available value in advertisement for the same requirement, and a score will be assigned accordingly ($Score_i$).

Each characteristic specified in the request (r_{iR}) can be a named concept(C_R) or an existential restriction ($\exists p.C_R$). If it is a named concept, similarity will be compared between the corresponding concepts in request and advertisement ($Similarity(C_R, C_A)$). If it is an existential restriction, the corresponding existential restrictions (restrictions having equivalent or sub properties) will be found in the advertisement ($\exists p.C_A$) and the similarity will be compared between the corresponding concepts in request and advertisement. If it is a composite concept the similarity will be judged recursively. The score($Score_i$) for each individual characteristic in the request will be assigned depending on this similarity.

The degree of similarity between concepts will be determined depending on the type of concept or attribute involved; determination of similarity between concepts has been discussed in detail in Section 4.4.2.

A score ($Score_i$) is assigned for each individual requirement (RD) specified in the request. The score for the advertisement (match score) will be determined by using the weighted average of these individual scores (the weight will be the corresponding priority value of each individual requirement).

$MatchScore = \sum_{i=1}^{n} w_i.Score_i \div \sum_{i=1}^{n} w_i$

where w_i and $Score_i$ is the priority value and the score of the sub-requirement RD_i. The overall score for the advertisement provides an indication of how good the advertisement is in satisfying the given request. The score for an advertisement will in turn be used as the basis for ranking; the highest score will receive the highest rank and so on.

An example walk-through to illustrate the application of the matching solution in a pervasive scenario, is presented in Section 5.3.

4.4.3.1 Algorithm

function $Match(Request, Advert)$
  for each Mandatory Requirement (RD) in Request **do**
    {Check if Advert fully satisfies the requirement description (RD)}
    if $\neg(Advert \sqsubseteq RD)$ **then**
      Return 0
    end if
  end for
  {if all mandatory requirements are met proceed to approximate matching}
  $MatchScore = 0$
  for Each Non_mandatory Individual Requirement (RD_i) in Request **do**
    get the priorityValue (w_i) corresponding to RD_i
    $Score_i = ApproxMatch(RD_i, Advert)$

$MatchScore = MatchScore + (w_i * Score_i)$
end for

function $ApproxMatch(R, A)$
if $A \sqsubseteq R$ **then**
{then A completely satisfies R}
$finalScore = 1$
else if If R is an atomic concept or a Data Range **then**
{i.e. it is not defined by any Necessary and Sufficient conditions that indicates a class definition}
if A is an atomic concept or a Data Range **then**
{Judge similarity between R and A, depending on whether they are Type1, Type2 or Type3}
$finalScore = similarityScore(R, A)$
else
for each named concept A_i in A **do**
$subScore = similarityScore(R, A_i)$
{Get maximum subScore as the finalScore}
$finalScore =$ maximum $subScore$
end for
end if
else
for each necessary and sufficient (N&S) condition in R (r_i) **do**
{quantify the extent to which each (r_i) is satisfied by A}
if (r_i) is a named concept (C_R) **then**
for each named concept in advertisement (C_A) **do**
$subScore = ApproxMatch(C_R, C_A)$
Get maximum $subScore$ as the score for (r_i)
end for
else if it is an existential restriction **then**
{Find corresponding restriction(s) in the advertisement}
for each existential restriction in advertisement that has either an equivalent property or a sub property **do**
{Match the corresponding concepts or datatype values and restrictions}
$subScore = ApproxMatch(C_R, C_A)$
Get maximum $subScore$ as the score for (r_i)
end for
$finalScore = finalScore + subScore$
end if
end for
$finalScore = finalScore/number \quad of \quad N\&S \quad conditions$
end if

Return $finalScore$

4.5 Discussion

This chapter presented the proposed approach for semantic matching, which provides a pragmatic solution to match resource requests and advertisements. The approach provides a ranking mechanism, which will order the available resource advertisements according to their suitability to satisfy the given request.

The description of a request consists of a number of individual requirements. Each individual requirement specifies a characteristic or a constraint that should be satisfied by a potential advertisement. During the matching process, an advertisement is evaluated according to how well it satisfies each of the individual requirements in a request. Depending on this evaluation, each advertisement is assigned a match score which in turn will be used as the basis for ranking or ordering them according to their suitability. The individual requirements occurring in a request are categorised into three types (Type-1, Type-2 or Type-3), depending on the type of property or concept they involve. The extent to which an advertisement can satisfy a given individual requirement is then judged depending on the type or category of requirement into which it falls.

This matching approach has a number of desirable properties, which are:

- Ranking of matches: The ranked list of matches provides the resource seeker with a useful aid in understanding the appropriateness of the advertisements and the order in which they should consider the potential matches.
- Approximate matching: The individual requirements of a request are categorised into three types depending on the concept or property it involves. This allows the matcher to employ the appropriate function to judge the similarity and suitability within the requirements, during the matching process.
- Consideration of priorities on the individual requirements and mandatory requirements: This allows different weights or priorities to be associated with the requirements of a request and to specify whether any of the requirements are mandatory. Considering these priorities and mandatory requirements in the matching process, allows the matcher to return results that agrees better with the resource seeker's context.

The proposed matching approach provides an effective approximating criterion while not being limited or specific to any particular domain as opposed to rule based matching approaches such as that proposed in [128]. That is, it could be generally applied to any domain where the resources are described as restrictions over the properties applicable.

Some concerns that could arise regarding the proposed matching approach, are discussed in the following sections.

4.5.1 Commitment to a Common Ontology

In this semantic matching approach, we have assumed that all the resource providers and requesters use the same ontology for resource descriptions. However, commitment to a common ontology may not always be feasible in many practical situations. Although ontology mapping techniques (that provide mechanisms to map concepts from one ontology to another) are a subject of much research, there is no widely accepted solution as yet. However, if an ontology mapping technique (such as the approaches proposed in [73] and [92]) can be successfully applied to identify appropriate mappings between resource descriptions described in different ontologies, the matching mechanism can be applied on the resource descriptions thereafter.

4.5.2 The Approach for Judging Similarity Within Requirements

For the Type-3 requirements (i.e. datatype properties), we have stressed the need for judging the suitability of the available resources in proportion to the violation of the requested numeric constraint. Therefore, we employ the functions illustrated in Figure 4.5 to judge the similarity score, for all Type-3 attributes. It can be argued, that for different datatype properties, a variation in the value of the property will have varied sensitivity on the suitability. For example, referring to a *computer* ontology, a 50% variation from the required range for the *memory size* property, may be felt more severely than a 50% variation from the required range for the *disk space* property. I.e. a resource seeker may feel that a computer having 50% less *memory* than the requirement is much worse than a computer that has 50% less *disk space*. However, this varied sensitivity on the suitability, is not limited to datatype properties; it can be present for symbolic properties as well.

The objective of this research, is to provide a general approach that will "estimate" the suitability of an advertisement during the matching process and to provide a recommendation of their appropriateness to satisfy a request. To this effect, the approximate matching criterion fulfils the purpose[11].

An alternative approach for matching that can provide specific similarity functions, for each type of concept or property, is a rule based approach such as that presented in [128]; in this case, matching rules will be specified for each property or concept. Such approaches will then become specific to the domain for which these rules were developed, and therefore can no longer serve as a general approach.

[11] This can be seen from the results obtained from the effectiveness evaluation experiments, which are discussed in Section 6.1

However, the possibility of employing different functions for different types of concepts and datatype properties, can be explored as a part of future work for this research. For example, the possibility of specifying rules to compute similarity, that will override the general similarity function for a particular type of attribute, whilst keeping the approach general, can be investigated further.

The proposed semantic matching approach is implemented in a pervasive context to match device based services, which will be discussed in Chapter 5. Also, it must be evaluated in terms of its retrieval effectiveness and performance; the empirical evaluations carried out in this respect, are discussed in detail in Chapter 6

Chapter 5

The Application of Semantic Matching in a Pervasive Environment

In this chapter we will discuss the application of the proposed semantic matching framework presented in Chapter 4 in the context of a pervasive environment; specifically the meeting room scenario discussed in Section 5.1. The resources concerned in a pervasive environment are devices and their services and hence the discovery process will involve description and matching of device requests and advertisements. The rest of this chapter is organised as follows: In Section 5.1 we will first introduce the Meeting Room scenario for which we apply the semantic matching approach. Then in Section 5.2 we will present the details of this implementation. In Section 5.3, we present an example illustration of this application. Finally in Section 5.4 we discuss the practical concerns of applying semantic matching in a pervasive context.

5.1 The Meeting Room Scenario

The Semantic Matcher presented in Chapter 4 is applied in a meeting room scenario, which has been initially constructed for the FEEL project [43]. The scenario involves a number of co-workers who are gathered in a meeting room in a discussion. The room is a device rich environment where there are a number of wired and wireless devices. The co-workers will be having their own personal devices such as mobile phones, PDAs and laptops. There are a variety of devices, both inside the meeting room and outside (within the organisation), available for use during the discussion. These may range from computers, printers and display units to projectors, cameras and camcorders. During the discussion the participants will get involved in various tasks which may give rise to the need for specific devices. For example one of the participants may be interested

in showing a multimedia file stored in a certain flash media card, and hence would want to find a device that could read and render the file. The participants may then decide that they want to view the file on a more suitable display before continuing the discussion; depending on the file to be viewed they may require a display with specific characteristics and will raise a request specifying the requirements in order to find the best device available. Later on, the file may need to be printed on suitable media, hence a printer with specific properties may be required. In another instance the participants may want to execute a particular software application and therefore a computer that meets certain requirements may be needed.

Thus to cater for these resource needs, an appropriate discovery mechanism should be in place where the information about the available resources are accessible and an effective matching approach will find the best possible resource depending on the context.

5.2 Implementation of the Semantic Matcher

As discussed in Chapter 2 there are three distinct components involved in service discovery, which are: the language or model used for service description, the matching mechanism used to determine which available service(s) match the given request and the discovery architecture that determines the communications and protocols used when advertising and querying. As has been already emphasised, the matching mechanism used to compare service advertisements and requests are independent of the discovery architecture and communication protocols used between the service advertisers and requesters.

The purpose of this implementation is to investigate the applicability of the proposed semantic matching mechanism in a pervasive scenario and to evaluate its effectiveness and efficiency when applied in a pervasive context. Evaluating any particular discovery architecture or communication protocol is not within the scope of this thesis and hence we do not emulate/involve any particular communication platform in this implementation exercise.

The high level architecture of the components involved in this implementation of the Semantic Matcher is illustrated in Figure 5.1.

The semantic matching functionality is carried out by the Matching Engine. This is implemented in Java. The Pellet description logic reasoner [104, 60] is used to facilitate the necessary reasoning tasks during the matching process. The matching engine communicates with the Pellet reasoner through the Pellet-API provided.

The resource requests and advertisements are described in OWL as discussed in Section 4.4.1. Since the resources involved in this domain are devices and their services, the Device Ontology discussed in Chapter 3 was used for the description of the individual

requirements and properties in the requests and advertisements. The ontologies were developed with the Protégé ontology editor and converted to OWL using the Protégé OWL Plug-In. The resource requests and resource advertisements encoded in OWL are available to the Matching Engine as OWL files. The OWL files are read and parsed by the Matching Engine by using the OWL-API [1].

5.2.1 The Semantic Matching Process

The resource request concerned and all the available resource advertisements are compiled into OWL format in a .owl file[2]. Then this OWL file is presented to the Matching Engine. Once the matching system receives the OWL descriptions, it checks for the consistency of the descriptions. If they are consistent the matching process begins.

The Matching Engine will parse the resource request and will compare and evaluate each of the advertisements provided in the OWL file, according to the matching mechanism presented in Section 4.4.3. If there are any mandatory requirements specified in the request, the advertisement will be first checked to see if these are satisfied. If they are not satisfied, the advertisement will receive a score of 0. If the mandatory requirements are satisfied, approximate matching will be carried out with respect to the non-mandatory requirements. As a result of this matching process, a score will be assigned to each advertisement, which will indicate the suitability of the advertisement to satisfy the request. Once all the advertisements are evaluated and scored, the advertisements are ranked on the basis of the score they have received. Then the Matching Engine will return the advertisements together with their scores and rankings.

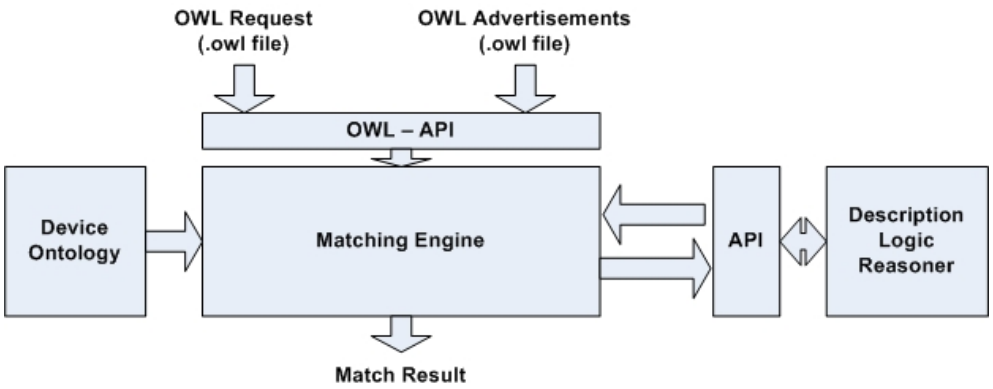

FIGURE 5.1: The Matching System

[1] http://owlapi.sourceforge.net/index.html

[2] For the sake of this implementation we assume that the resource seeking agents/users and the resource providers will compile the requests and advertisements in OWL. However in practice, the infrastructure involved may support the compilation of the requests and advertisements, for example through a GUI.

5.3 Experiments

We have tested the Semantic Matcher implementation using several use cases which were subsequently used for the evaluation experiments. For each of the use cases, we generated the descriptions for the request and the advertisements using Protégé and saved them into .owl files using the Protégé OWL plug-in. These were then used by the Matching Engine as the input request and advertisement descriptions. The Semantic Matcher is then invoked to obtain the match results, which provides the match score and match rank for each of the advertisements.

We illustrate the application of the Semantic Matcher implementation using the following two examples.

Example Illustration 1: ***Seeking a Colour Laser Printer that can print on Paper Size A2***

We assume a scenario where a user in a pervasive environment seeks a printer with certain characteristics. The request concerned is a *Colour*, *Laser* Printer, that can print on the paper size *A2*. We also assume that the Paper Size *A2* is the most important (highest priority) attribute under the context involved and that the other two attributes (the Print Technology being *Laser* and the *Colour* printing capability) are of lesser priority. For the sake of this example illustration, the priority values or weights for the three attributes are assumed to be set by the resource seeker as 0.6 for the paper size attribute and 0.2 for both the other attributes (the printing colour and printer technology)[3]. Since the request is for a *Printer*, this is stated as a mandatory requirement with priority value 2.0 (2.0 should be the associated priority value for any mandatory requirement present in a request, as described in detail in Section 4.4.1). The request will be described in description logic notation as:

$$
\begin{aligned}
&Request \sqsubseteq \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority. = 2.0\ \sqcap \\
&\exists hasRequirementDescription.RD1) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority. = 0.6\ \sqcap \\
&\exists hasRequirementDescription.RD2) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority. = 0.2\ \sqcap \\
&\exists hasRequirementDescription.RD3) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority. = 0.2\ \sqcap
\end{aligned}
$$

[3]In practice, the priority values for each individual requirement should be assigned by the resource seeker. These priority values will reflect the relative importance of each requirement, in the context in which the request is raised. In cases where the resource seekers are human, they can be aided by Graphical User Interface components such as sliders, in determining the priority values.

$\exists hasRequirementDescription.RD4)$

$RD1 \equiv ptr : Printer$

$RD2 \equiv \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A2)$

$RD3 \equiv \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPrintTechnology\ .\ ptr : Laser)$

$RD4 \equiv \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPrintingColour\ .\ ptr : Colour)$

For the purpose of this evaluation experiment we assume the availability of six advertisements (with varying characteristics). The descriptions of these advertisements are as follows:

$Advert1 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A2\ \sqcap$
$\exists ptr : hasPrintTechnology\ .\ ptr : Laser \sqcap \exists ptr : hasPrintingColour\ .\ ptr : BW)$

$Advert2 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A2 \sqcap$
$\exists ptr : hasPrintTechnology\ .\ ptr : Laser \sqcap \exists ptr : hasPrintingColour\ .\ ptr : Colour)$

$Advert3 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A2 \sqcap$
$\exists ptr : hasPrintTechnology\ .\ ptr : Inkjet \sqcap \exists ptr : hasPrintingColour\ .\ ptr : BW)$

$Advert4 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A4 \sqcap$
$\exists ptr : hasPrintTechnology\ .\ ptr : Inkjet \sqcap \exists ptr : hasPrintingColour\ .\ ptr : Colour)$

$Advert5 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A3 \sqcap$

$\exists ptr : hasPrintTechnology\ .\ ptr : Inkjet \sqcap \exists ptr : hasPrintingColour\ .\ ptr : BW)$

$Advert6 \sqsubseteq\ ptr : Printer \sqcap \exists dev : hasHardwareDescription$
$(dev : HardwareDescription \sqcap \exists ptr : hasPaperSize\ .\ ptr : A4 \sqcap$
$\exists ptr : hasPrintTechnology\ .\ ptr : Inkjet \sqcap \exists ptr : hasPrintingColour\ .\ ptr : BW)$

Considering the attributes involved in this example: The Paper Size attribute is a Type-2 attribute and we assume that the similarity values between A2, A3 and A2, A4 are given as 0.6 and 0.25. Printer Technology attribute is also a Type-2 attribute and we assume the similarity value between Laser and Inkjet is given as 0.7. Printing Colour is a Type-1 attribute, where the concept Colour is defined as a subclass of the concept BW (since all colour printers can print black & white as well). Given the knowledge that $Colour \sqsubseteq BW \sqsubseteq PrintingColour$, $SimilarityScore(Colour, BW)$ can be computed using Equation 4.2, which gives 0.66. Considering Advert6, this satisfies the mandatory requirement of being a *Printer* and therefore will proceed through to the approximate matching process. This will get subscores of 0.25, 0.7 and 0.66 for the attributes of Paper Size, Printer Technology and Printing Colour. By evaluating the final match score according to the algorithm stated in Section 4.4.3.1 (using the associated priority values of the requirements), this Advert will get a match score of 0.711. Similarly, the other advertisements can be evaluated in the same way and by considering the match score, the advertisements could be ranked accordingly.

The match results produced are illustrated in Table 5.1. This indicates the match score and rank received by each advertisement.

Advertisement	***Match Score***	***Match Rank***
Advert 1	0.966	2
Advert 2	1.0	1
Advert 3	0.936	3
Advert 4	0.745	5
Advert 5	0.816	4
Advert 6	0.711	6

TABLE 5.1: Match Results for the Example Illustration 1: Printer Request

Example Illustration 2: ***Seeking a Wide Screen, Flat Panel display unit that has a Diagonal Size of at least 19 inches***

Let us assume a scenario where a user in a pervasive environment seeks a display device with certain properties. The request concerned is a *WidescreenDisplay*, that has *LCD* display technology and that has a diagonal size of at least *19* inches; let's also assume that the priority values for each of these individual requirements as assigned by the resource seeker is, 2.0, 0.3 and 0.7 respectively (the *WidescreenDisplay* requirement is

a mandatory requirement). This request can be described in description logic notation as:

$$
\begin{aligned}
&Request \sqsubseteq \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{2.0}\ \ \sqcap \\
&\exists hasRequirementDescription.RD1) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{0.3}\ \ \sqcap \\
&\exists hasRequirementDescription.RD2) \sqcap \\
&\exists\ hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{0.7}\ \ \sqcap \\
&\exists hasRequirementDescription.RD3)
\end{aligned}
$$

$$RD1 \equiv WidescreenDisplay$$

$$
\begin{aligned}
&RD2 \equiv \exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap \\
&\exists dev : hasDisplay(dev : Display \sqcap \\
&\exists dev : hasDisplayTechnology\ .\ Plasma))
\end{aligned}
$$

$$
\begin{aligned}
&RD3 \equiv \exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap \\
&\exists dev : hasDisplay(dev : Display \sqcap \\
&\exists dev : hasDisplaySize(DisplaySize \sqcap \\
&\exists dev : hasDiagonalSize\ .\ =_{19})))
\end{aligned}
$$

Lets assume that the following advertisements for display devices are available:

$$
\begin{aligned}
&Advert1 \sqsubseteq\ dev : DisplayDevice \sqcap \\
&\exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap \\
&\exists dev : hasDisplay(dev : Display \sqcap \\
&\exists dev : hasAspectRatio\ .\ as16\_10 \sqcap \\
&\exists dev : hasDisplayTechnology\ .\ LCD \sqcap \\
&\exists dev : hasDisplaySize(DisplaySize \sqcap \\
&\exists dev : hasDiagonalSize\ .\ =_{17})))
\end{aligned}
$$

$$
\begin{aligned}
&Advert2 \sqsubseteq\ dev : DisplayDevice \sqcap \\
&\exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap \\
&\exists dev : hasDisplay(dev : Display \sqcap \\
&\exists dev : hasAspectRatio\ .\ as16\_10 \sqcap \\
&\exists dev : hasDisplayTechnology\ .\ Plasma \sqcap \\
&\exists dev : hasDisplaySize(DisplaySize \sqcap \\
&\exists dev : hasDiagonalSize\ .\ =_{42})))
\end{aligned}
$$

$$
\begin{aligned}
&Advert3 \sqsubseteq\ dev : DisplayDevice \sqcap \\
&\exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap
\end{aligned}
$$

$\exists dev : hasDisplay(dev : Display \sqcap$
$\exists dev : hasAspectRatio\ .\ as4\_3 \sqcap$
$\exists dev : hasDisplayTechnology\ .\ LCD \sqcap$
$\exists dev : hasDisplaySize(DisplaySize \sqcap$
$\exists dev : hasDiagonalSize\ .\ =_{17})))$

$Advert4 \sqsubseteq\ dev : DisplayDevice \sqcap$
$\exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap$
$\exists dev : hasDisplay(dev : Display \sqcap$
$\exists dev : hasAspectRatio\ .\ as16\_9 \sqcap$
$\exists dev : hasDisplayTechnology\ .\ CRT \sqcap$
$\exists dev : hasDisplaySize(DisplaySize \sqcap$
$\exists dev : hasDiagonalSize\ .\ =_{32})))$

The Device Ontology also contains the knowledge that, WideScreen Displays are display devices with certain aspect ratios. The axiom that states this fact can be expressed in description logic notation as:

$dev : WidescreenDisplay \equiv dev : DisplayDevice \sqcap$
$\exists dev : hasHardwareDescription(dev : HardwareDescription \sqcap$
$\exists dev : hasDisplay(dev : Display \sqcap \exists dev : hasAspectRatio\ .\ (as15\_9 \sqcup as16\_10 \sqcup as16\_9)))$

Considering the attributes involved in this example: The *DisplayTechnology* attribute is a Type-2 attribute and we assume that the similarity values between *LCD* and *Plasma* and *CRT* and *Plasma* are given as 0.8 and 0.3 respectively. *DiagonalSize* is a Type-3 attribute and the similarity between the requested value and the available value are determined using a membership function as described in Section 4.4.2.3. Considering the Advert1, this satisfies the mandatory requirement of being a *WidescreenDisplay* (through the use of reasoning, the matcher will identify that the Advert1 satisfies this requirement since it is specified as a *Display* with an aspect_ratio of *as16_10*) and therefore will proceed through to the approximate matching process. This will get subscores of 0.8 and 0.89 for the attributes of *DisplayTechnology* and *DiagonalSize*. By evaluating the final match score according to the algorithm stated in Section 4.4.3.1 (using the associated priority values of the requirements), this advertisement will get a score of 0.97. Similarly, the other advertisements can be evaluated in the same way and by considering the match score, the advertisements can be ranked.

The match results produced are illustrated in Table 5.2. This indicates the match score and rank received by each advertisement.

Advertisement	***Match Score***	***Match Rank***
Advert 1	0.97	2
Advert 2	1.0	1
Advert 3	0.0	4
Advert 4	0.94	3

TABLE 5.2: Match Results for the Example Illustration 2: Display Device Request

5.4 Discussion

Although semantic approaches to service discovery can provide many benefits over syntactic approaches, we have to bear in mind the fact that certain resources in pervasive environments (small mobile devices such as mobile phones and PDAs), are constrained in terms of computing power and memory. On the other hand the standard semantic web tools and technologies such as description logic reasoners, can be too heavy-weight for such resources. Hence a feasible architecture has to be chosen for the discovery process while facilitating the use of semantic descriptions and reasoning mechanisms to provide effective description and matching of services.

In service discovery, a certain entity or entities will act as a repository or directory for storing information about available services. Solutions for service information storage vary between two end-points: completely centralised and completely distributed (peer-to-peer) architectures (See [89, 35] for a detailed discussion on this topic). A centralised approach for a directory service may be a straightforward way to implement efficient access to data and low overhead traffic, although it creates a single point of failure. SLP with a Directory Agent [118] is an example that has a central server for information storage. In distributed approaches, communication is typically based on broadcast or multi-cast mechanisms. This technique is common for protocols designed to work in local area networks and ad hoc networks. The network size and type is important in determining the design for a directory system. There is a wide range of hybrid solutions that have been proposed for achieving better results under specific circumstances or assumptions. Marin-Perianu et. al. in [89] presents a brief survey of the commonly used solutions for directory systems and discusses the advantages and disadvantages concerned with each type of solution.

In view of the limited computing power available on many mobile devices, an appropriate architecture has to be chosen for the discovery process where the directory service (which is responsible for storing service descriptions and executing the matching process), could run centrally on the network and the devices could communicate through the network as necessary.

The main objective of this implementation exercise was to provide a prototype of the Semantic Matcher which can be subsequently used to evaluate the effectiveness of the proposed semantic matching mechanism. Therefore, the architectural aspects of a practi-

cal deployment of the Semantic Matcher has not been the focus of this implementation. However, in a real world deployment of the semantic matching solution, there are a number of design issues that needs to be taken into consideration. These are:

- *Storage of the Available Resource Advertisements:*

  The Semantic Matcher implementation has assumed that the descriptions of the resource advertisements are are available to the Matching Engine in an .owl file. However, in a practical deployment of the semantic matching solution, persistent storage will have to be used for the storage of OWL descriptions of the advertisements. A number of solutions for the storage of Semantic Web data (RDF Triple Stores) exist, such as Mulgara[4], 3store[5] and Kowari[6]. A survey and an evaluation of the available RDF triple store systems is provided in [80].

- *Indexing/Search Mechanism*

  In its current state of the Semantic Matcher implementation, the Matching Engine will compare and match all the available advertisements sequentially, during the matching process. However, this may become intractable when very large numbers of advertisements are present. Therefore it will be useful to investigate possible means of developing directories that can be searched and managed efficiently, for example through the creation of search indexes. Constantinescu et.al. [32] have proposed such a method for the efficient search and maintenance of directories to support semantic matchmaking.

- *Compiling the Resource Request*

  The resource request should be available to the Matching Engine in OWL format. However, in practice, it will not be reasonable to assume that a human user will compile the request in OWL. Therefore the system must support the compilation of the requests, using an appropriate method, for example through a graphical user interface.

Another practical concern is the efficiency and performance of the semantic matching process; the computational cost of involving semantic web technologies in the matching process needs to be investigated. We empirically evaluate the efficiency and performance of the service matching process and present the results in Section 6.2.

[4]http://mulgara.org/
[5]http://sourceforge.net/projects/threestore/
[6]http://kowari.org/

Chapter 6

Evaluation

In this chapter we present an empirical evaluation of the semantic matching framework (described in Chapter 4) and discuss the results obtained and the conclusions drawn from them. We evaluate the matching framework with respect to two aspects: effectiveness and performance. Firstly and most importantly, we wish to evaluate the proposed matching framework with respect to how effective it is, i.e. how good the system is in discovering the relevant or suitable resources. Secondly it is important to gain an understanding of the performance and scalability of the matching framework, to justify that any compromise in performance resulting from the involvement of reasoning mechanisms, is outweighed by the benefits gained from semantic matching.

In section 6.1 we discuss how the effectiveness of a semantic matching approach can be evaluated, the experiments carried out to evaluate this aspect and the conclusions drawn from them. Next in 6.2 we discuss the evaluation of the performance of the proposed semantic matching approach.

6.1 Evaluating Effectiveness of Semantic Matching

The effectiveness of a Semantic Matching approach refers to its ability on retrieving relevant matches in relation to the given resource request. Tsetsos et. al. in [130] provides a generalised view of evaluating retrieval effectiveness in a semantic matching context. As depicted in Figure 6.1, a matching engine will compute a degree of match(e(R, A_i)) for each advertised resource(A_i) in relation to a resource request (R) raised by a user. In order to evaluate the matching engine effectiveness, some expert or user mapping r(R, A_i) (a relevance assessment between R and A_i) must be available. Thus the vectors e and r are defined as:

$$r : Q \times A \rightarrow W$$

$$e : Q \times A \rightarrow W$$

where Q is the set of all possible resource requests, A is the set of resource advertisements and W is the set of values denoting the degree of relevance (for r) or degree of match (for e) between a request from Q and advertisement from A. Both r and e may assume various types of values: Boolean (W={0,1}), real numbers (W=[0,1]) and classification (W={ exact, plugin, subsume, ... }). Given these informal definitions, *the evaluation of a matching engine is the determination of how closely the vector* $\boldsymbol{e}$ *(delivered by the engine) approximates the vector* $\boldsymbol{r}$ *(specified by domain experts/users).*

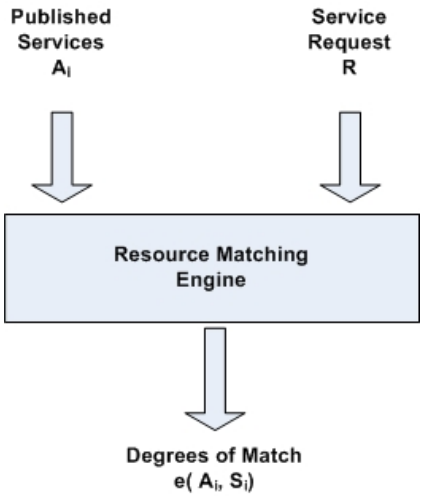

FIGURE 6.1: Resource Retrieval Evaluation

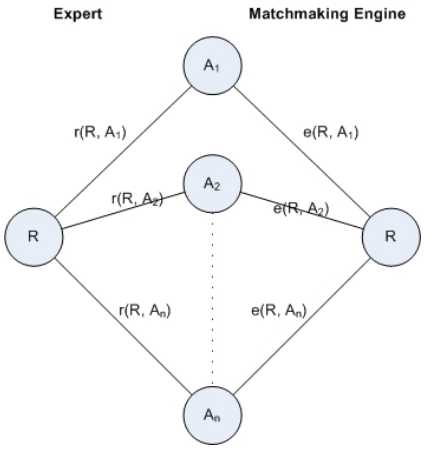

FIGURE 6.2: Expert and Matching Engine Relevance Assessments

In the following sections we discuss the current methods used for evaluating retrieval effectiveness and explain the criteria used for the evaluation of the proposed semantic matching work. Then the specific objectives of the evaluation exercise will be discussed followed by the details of the experiments and their results.

6.1.1 Current Practices used for Evaluating Retrieval Effectiveness

Currently there is no agreed "best practice" methods or heuristics which are used to evaluate effectiveness of a matchmaking solution. As pointed out by Tsetsos et. al in [130], although many semantic matching research efforts have conducted experiments to evaluate the performance with respect to scalability and response times, they lack a quantitative analysis of the retrieval effectiveness of their solutions: the main reason for this being the lack of established evaluation metrics and methodologies for semantic matching schemes.

However in the Information Retrieval (IR) domain (where relevant documents must be extracted from a collection of documents), which addresses a somewhat similar problem to matchmaking (identifying suitable resources out of several resources); Precision and Recall metrics and the associated F-measure [9] (see Equation 6.2, Equation 6.3 and Equation 6.1) are used to judge the effectiveness[1].

$$precision = \frac{Number \quad of \quad Retrieved \quad Resources \quad that \quad are \quad Relevant}{Number \quad of \quad Retrieved \quad Resources} \tag{6.2}$$

$$recall = \frac{Number \quad of \quad Retrieved \quad Resources \quad that \quad are \quad Relevant}{Number \quad of \quad Relevant \quad Resources} \tag{6.3}$$

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{6.4}$$

However most information retrieval tasks assume a boolean relevance, i.e. the document is either relevant or completely irrelevant. The measures of precision and recall are thus based on this assumption; the positive and negative matches returned by an IR system are compared against those deemed to be relevant or irrelevant (as determined by a human subject) in order to arrive at the precision and recall metrics.

Semantic matching solutions typically aim to provide a more flexible approach for matching, rather than just classifying the available services into crisp sets of relevant and irrelevant cases. Therefore, the matches can have different levels of relevancy or suitability

[1]Recall is the proportion of relevant resources actually retrieved in answer to a search request. Precision is the proportion of retrieved resources that is actually relevant.

A single measure combining recall and precision is the F-measure or weighted harmonic mean (Equation 6.4). The general formula for non-negative real α is given by:

$$F_\alpha = \frac{(1+\alpha)(precision * recall)}{\alpha * precision + recall} \tag{6.1}$$

Choosing $\alpha > 1$, weights recall more than precision. When $\alpha = 1$, recall and precision are both equally weighted which gives F or F_1 as in Equation 6.4.

as compared to the given request. Depending on this degree of relevancy or suitability, the matches are either: classified to a number of sets as in [101, 82]; or ranked usually based on a real number score assigned to each resource. To evaluate such a matching approach based on precision and recall, the range of service rankings must be divided into two complementary sets through the use of a threshold value (a minimum acceptable degree of match). All the advertisements having a ranking greater than the threshold will be taken as relevant, and the others will be taken as irrelevant. Similarly the experts/ users of the domain will have to classify the advertisements as relevant or irrelevant as opposed to any other classification or ranking. However, there are several issues associated in using such an evaluation scheme to evaluate a semantic matching approach that employs multiple degrees of match (i.e. the potential matches can have different levels of relevancy or suitability):

- One of the main objectives of a semantic matching scheme is to facilitate flexible matching rather than just classifying the advertisements into two sets corresponding to either exact or failed matches. Re-classifying the resulting degrees of match into relevant and non-relevant matches for the purpose of employing an evaluation scheme using precision/ recall metrics, will mean that the extra information and semantics provided by the matcher are disregarded.
- In order to re-classify the match results into relevant and non-relevant matches, a suitable threshold has to be agreed. Agreeing on such a threshold value for this purpose is a problematic task since such a threshold value may depend on the context and on how stringent one wants the retrieval process to be.
- To apply precision/ recall metrics, the domain expert or user will have to assign a boolean relevance to each advertisement; determining if an advertisement is relevant or not, may not be straightforward in certain cases and is in contrast to the objective of semantic matching, which aims for more flexible and accurate resource retrieval.

Hence an evaluation approach has to be adopted which takes in to consideration the objectives and semantics of the matching approach and the type of classification or ranking provided. In the following section we discuss some of the approaches employed by related efforts to evaluate their semantic matching frameworks and then explain the evaluation criteria adopted for the evaluation of the proposed semantic matching framework.

6.1.1.1 Evaluation Methods used in other Service Matching Research

The web service matching approaches presented by Dong et. al in [41] and Wang et.al in [141] have adopted precision and recall metrics to evaluate the effectiveness of the

proposed matching solutions. The matcher results are compared against the boolean relevances assigned to the available services (services are deemed to be either relevant or irrelevant by the authors). In [141], these assigned relevances are compared against the resources that receive a score greater than 50% by their matching system[2]. In [41] again, precision, recall and variants of precision are used to evaluate the matching solution.

The matching system proposed in [139] also adopt precision and recall metrics to evaluate their system. The domain involved in this research is the recruitment of human resources. The resource request involved is a job specification and the advertisements will be the skills of the available job seekers. Their evaluation study involves five subjects who are human resource experts. The matchmaker will assign a score for each advertisement in relation to the job request, and the advertisements that get a score greater than 60% are deemed to be the "positive" matches returned by the system. The human experts also assign a score to each advertisement, and those that get a score greater than 60% are taken as the "relavant" matches for calculation of precision and recall metrics. In this situation however, there is a prior known "marking scheme" for scoring the skills specifications in relation to a given job request. Therefore it has been possible to adopt precision and recall metrics in this research. However, the reason for choosing a threshold value of 60%, is not justified in the literature.

Tsetsos et. al in [130] presented a generalised evaluation scheme to judge retrieval effectiveness of semantic web service matchmaking. However, the evaluation scheme presented is specific to matchmaking systems where, the matches are classified into an agreed set of classes as in [101, 82]. They have used generalised recall and precision metrics (which will be discussed later in Section 6.1.1.2) to measure the correspondence between the relevance assessments delivered by the matching engine and by the experts/ users.

In the semantic matching approach presented in [40] by di Noia et. al, the available advertisements are ranked or ordered by their suitability to satisfy the given request. They have evaluated their matching approach by comparing the matchmaker results with human perception; the closeness between the ranking produced by the matching engine and the rankings obtained by domain users in the same scenario, is estimated through the use of standard deviation. This gives an understanding of how close the matching engine can approximate human judgement. Also, in the matchmaking work presented in [139], the ranking provided by the matchmaker is compared with the averaged human ranking provided by domain experts to find the closeness between the matcher ranking and human ranking, and thereby estimate the ability of the matching engine to approximate human judgement.

As explained earlier, effectiveness evaluation involves determining how well the matcher

[2]The authors have assumed that a service receiving a score of less than 50% is likely to be irrelevant to the request

can approximate expert or user judgement. Although there is no accepted "best practice" measure for evaluating semantic matchers, it can be observed that various efforts have adopted precision and recall (or extended and modified versions of precision and recall) from the information retrieval domain for this purpose.

6.1.1.2 Precision and Recall for Generalised Systems

As pointed out earlier in Section 6.1, although *precision* and *recall* metrics have been widely accepted to measure effectiveness of information retrieval systems, they are only applicable to systems that return a boolean relevance. These metrics cannot be directly adopted to measure effectiveness of systems that return a fuzzy value for the relevance (i.e. systems that return a value $\in [0, 1]$ as the degree of relevance for an advertisement A w.r.t. a request R). Therefore to measure effectiveness in such systems, Buell et. al. in [22] have presented generalised measures of *precision* and *recall*.

Let $r(A_i)$ denote the degree of relevance assigned by the expert or user for the advertisement A_i, let $e(A_i)$ denote the degree of relevance assigned by the system for the advertisement A_i[3] and n denote the number of advertisements. Then the generalised recall and precision measures are defines as:

$$Recall = \frac{\sum_{i=1}^{n} \min(r(A_i), e(A_i))}{\sum_{i=1}^{n} r(A_i)} \tag{6.5}$$

$$Precision = \frac{\sum_{i=1}^{n} \min(r(A_i), e(A_i))}{\sum_{i=1}^{n} e(A_i)} \tag{6.6}$$

The purpose of the evaluation is to measure how closely e can approximate r. It can be observed that, when the system estimates for relevance values are more stringent than the expert estimates, precision is maximised. When the expert estimates are more stringent, then the recall is maximised.

[3]That is, $e(A_i)$ can be seen as the fuzzy membership function (defined by the system) that defines the membership value of A_i in the set of advertisements that satisfy the given request. Similarly, $r(A_i)$ can be seen as the fuzzy membership function (defined by the expert) that defines the membership value of A_i in the set of advertisements that satisfy the request

6.1.2 Evaluating the Semantic Matcher

As discussed in the previous sections, there are no agreed, best practice evaluation methods that can be used to evaluate semantic matching solutions. The precision and recall metrics in IR domain cannot be directly applied in this case due to the aforementioned problems.

The ultimate result of the proposed semantic matching framework in this research will be the ranking of the available advertisements, indicating which is the best match, which is the next best and so on. It is much easier to obtain such a ranking from a domain user, as opposed to obtaining a percentage score for an advertisement. As stated in the previous discussion, evaluation of semantic matching is the determination of how closely the rankings/classifications delivered by the engine, approximates the rankings/-classifications specified by domain experts/users. Hence to judge the effectiveness or correctness of how the matches are ranked or classified, the resultant ranking or classification will have to be compared against what a human subject or expert will view as the correct ranking or classification.

For the purpose of the evaluation of retrieval effectiveness of this semantic matching solution, we will compare the rankings delivered by the matching engine with that provided by the domain users in the same context. The domain users' rankings will be obtained through studies conducted and the average user ranking will be obtained[4].

The closeness between the average user ranking and the ranking from the matching engine for the same situation will be judged quantitatively and graphically which in turn helps to evaluate the effectiveness of the matcher. We outline the methods and metrics used for this purpose in the following sections.

6.1.2.1 Adapting the Generalised Precision and Recall to Evaluate the Proposed Semantic Matching Approach

As discussed in Section 6.1.1.2, the well known measures of precision and recall have been extended to measure effectiveness of systems that return a fuzzy value for the relevance. The equations for generalised precision and recall are given in Equation 6.5 and Equation 6.6. To use this equation to evaluate a matcher, the matching system should return a value $\in [0, 1]$ as the degree of relevance for an advertisement. However, the proposed semantic matcher returns a ranking ($\in [1, n]$, where n is the number of advertisements considered during the matching process), where the best resource advertisement gets rank 1, the second best gets 2 and so on[5]. To exploit the generalised

[4]Human subjects can have subjective differences; for example what one views as the third best match could be viewed as the fourth best by another. By averaging the rankings obtained by a number of subjects, the effects of subjective judgements can be minimised.

[5]Also, it is much easier for a domain user to rank the advertisements rather than assigning a relevance score

precision and recall as a metric for evaluation, the rank should be adjusted to a fuzzy relevance value $\in [0, 1]$.

The rank can be adjusted to obtain a value $\in [0, 1]$ which will indicate the fuzzy relevance for an advertisement. We use the following equation to obtain a fuzzy relevance (f) from the rank. The fuzzy relevance f_i for the i th advertisement that has rank $rank_i$, can be obtained by:

$$f_i = \frac{n - rank_i}{n} \tag{6.7}$$

where n denotes the number of advertisements considered during the matching process (and therefore the maximum value that can be taken by the $rank$). The measure f can then be used in Equation 6.6 and Equation 6.5 for calculating generalised precision and recall.

6.1.2.2 Chosen Evaluation Criteria

As pointed out by Tsetsos et. al. in [130], the service matching domain lacks established metrics and methods for evaluating the retrieval effectiveness and only a few semantic matching efforts have carried out a quantitative analysis of effectiveness of their proposed approaches. However, precision and recall metrics (or their generalised versions) have been adopted for evaluating certain matching solutions [41, 141, 130]. However, precision and recall metrics (as given in Equation 6.2 and Equation 6.3) will mean that the output of a semantic matcher (that returns a fuzzy relevance as the output) has to be converted into a boolean relevance; this approach has limitations as identified in Section 6.1.1. These limitations can be overcome by using the generalised precision and recall metrics as discussed in Tsetsos et. al.[130]; they have also adopted these metrics for evaluating semantic matchers that classify available services into an agreed set of classes as in [101, 82]. These generalised precision and recall metrics can also be extended for the evaluation of matchers that rank the available services as identified in Section 6.1.2.1.

Thus in view of the above discussion, we use the following metrics and methods to judge the effectiveness of the Semantic Matching Approach.

- **Generalised Recall and Precision and associated F-measure:** We use the fuzzy relevance scores obtained from the rankings (through Equation 6.7) to compute the generalised precision and recall (Equation 6.6 and Equation 6.5). These values of precision and recall are then used to compute the F-measure (Equation 6.4) which gives a combined measure of effectiveness.
- **Standard Deviation:** The Standard Deviation between the matcher ranking and the average human ranking is computed. This quantitatively indicates the

deviation between the two rankings[6].

- **Graphical Illustration of the Rankings:** Although this does not give a quantitative value, it is useful to gain an understanding of the variance between the semantic matcher ranking and the human ranking through visual inspection.

6.1.3 Evaluation Study

As discussed earlier in this chapter, effectiveness evaluation of a semantic matching solution is the determination of how closely the ranking/ classification delivered by the system, approximates the ranking/ classification specified by domain users. Hence to judge the effectiveness or correctness of how the matches are ranked by the proposed approach, the resultant ranking will have to be compared against what a human subject or domain user will view as the correct ranking. In this section, we will discuss the details of the human participant study conducted to obtain the human ranking and also the factors affecting the design of the experiments.

6.1.3.1 Human Participant Study

To obtain the human ranking for this evaluation exercise we conducted a study involving human subjects. For each experiment, a scenario or use case is devised that will involve a resource seeking situation where the seeker raises a query for a resource with certain property requirements. The use cases are devised in a pervasive environment context, specifically the meeting room scenario described in Chapter 5.

For each use case we construct a questionnaire which specifies: the device and the property/ functionality requirements that the resource seeker is interested in, the context that has given rise to the need of the device and the available devices and their properties. The questionnaires that have been used in this study are included in Appendix A. We handed out the questionnaire to the subjects involved and asked them, to assume that they are the resource seeker in the given context and rank the available devices specified, in the order they would consider them for utilising for the specified need (i.e. 1 for the device that they consider as the best to suit the requirement, 2 for the next best and so on). For each use case we involved at least 10 subjects and averaged the ranking provided for the purpose of comparison with the matchmaker results.

According the policy of the Dept. of Electronics and Computer Science in the University of Southampton, any study involving human participants, whether questionnaires, lab

[6]A high value for the standard deviation will mean that the difference between the ranking concerned and the human ranking is high; a low value will mean that the ranking agrees better with the human ranking.

studies, field observations and so on, must be reviewed to see if it is ethical[7]. Therefore, the ethics committee approval was obtained prior to the commencement of this study.

Participants of the study:

The participants of the study will be general domain users of a pervasive environment. Since the questionnaires will be requesting the participants to rank common computing resources (such as computers, printers etc.) with certain characteristics, in relation to some given request, this will require some basic sense of the features and properties present in such a resource. Therefore the participants must be computer literate.

The participants were selected from the University research labs in the Department of Electronics and Computer Science, since they can be considered as general domain users of pervasive computing resources. Since the study must be able to capture any variations in the subjective preferences of different users, a sample size between 10 and 15 will be selected for each questionnaire.

Recruitment of participants:

The participants will be approached verbally and through email communication and if they are willing they can return the completed questionnaire. Participants will be our colleagues in the university. The purpose of the study will be briefly explained to the participants (verbally or through e-mail) prior to handing out the questionnaire. The participants are free to return the questionnaires after completion or they can choose not to respond. No written consent is obtained in this study.

Risks and benefits to the participants:

The study involves completing a short questionnaire which will take around 10 minutes approximately. The responses are completely anonymous and non-personal in nature. Therefore there is no risk (or benefit) to the participant anticipated in connection with this study.

Privacy and confidentiality:

The responses collected from the questionnaires are non-personal in nature and the responses will not be associated with any individual participant. As stated earlier, the questionnaire simply requests the participant to rank available resources in relation to a given request in a certain context, and does not involve revealing of any personal information, personal opinions, beliefs, etc.

[7] `http://www.soton.ac.uk/inf/ethics_policy.html`

6.1.3.2 Determining the Use Cases for the Experiments

The use cases involved for each of the experiments must be as close as possible to realistic device requests that will occur in a pervasive environment; i.e. the use cases must correspond to usual device requirements that can occur in a day-to-day pervasive context. This way, the participants involved in the human participant study can give correct responses and also we can obtain concrete results from the evaluation experiments. If a use case involves a scenario where the judgement (i.e. the choice of the best device out of a set of available devices) requires specific knowledge, the participants may not be able to give correct answers and this will bias the results of the experiments. For example if a use case involved a request for a *Surveillance Camera* with particular features, an average computer user may not understand the terminology used and may not have the knowledge required to respond correctly to the questionnaire(s) in the human participant study. Also, the participants must be able to complete each questionnaire in a reasonable amount of time. Therefore the device requests involved in the use cases, must contain only a small number of requirements, so that each questionnaire will require only a limited amount of mental calculation and reasoning. Therefore in this evaluation study, we adhere to use cases that involve relatively common device requests that can occur in the pervasive environment and each request will contain a maximum of five individual requirements.

6.1.3.3 Subjects involved in the Human Participant Study

As stated in the previous section, the subjects of the study will be general domain users of a pervasive environment. In the evaluation experiments conducted by Dong et. al. in [41] and Wang et.al in [141] (these have been discussed in more detail in Section 6.1.1.1), the matcher results are compared against the boolean relevances assigned to the available services by the authors. I.e. there have been no human participant studies used to obtain any relevance values and only the author(s) opinion is taken into account for the classifying of matches. Human participant studies have been used by Noia et. al [40] and Veit et. al. [139], to obtain average human rankings to evaluate the proposed resource matching solutions (these evaluation approaches have also been discussed in Section 6.1.1.1). Veit et. al. in their evaluation study have used 5 participants to obtain the average human ranking. Noia et. al in [40] have used 20 participants to obtain the average human ranking for the purpose of comparing against their proposed semantic matching solution.

In view of the use cases involved in our application scenario, there can be some variation present in the rankings assigned to the available advertisements, due to subjective preferences. Therefore it will be useful to obtain the rankings from several participants and to use their average for the sake of comparing the Semantic Matcher results. However, since the scenarios used are relatively straight forward, there can be only a limited

level of variation that can be present in the assigned rankings. Therefore we choose the number of participants in our study to be between 10 and 15[8].

6.1.4 Objectives of the Effectiveness Evaluation Experiments

As pointed out earlier in this chapter, the evaluation of a matching solution is the determination of how closely the matcher output approximates the domain expert's or user's output. Therefore it is important that the results of the proposed semantic matching approach (hereafter referred to as the Semantic Matcher in the rest of this chapter) are as close as possible to human perception. That is, the resultant rankings or classifications computed by the matchmaker system must agree reasonably well with that computed by a domain user for the same situation.

The objective of these evaluation experiments is to show that the results of our proposed matchmaker correlate reasonably well with human perception. We have described several desirable properties for a matching mechanism in Section 4.2. In this section we show why those properties are beneficial and that the presence of these properties in a matchmaker will cause the results to be more effective.

Specifically we intend to find answers to the following questions:

- **Is a ranked list of matches more beneficial to the domain users than a classified set of matches?** Most existing semantic matching approaches produce a classified set of matches as its output (as in [101, 82]) and some others produce a boolean result where a resource can be either a "match" or "no match" ([57, 128]). We will show that a ranked list of matches are more beneficial and serves the expectations of domain users involved, by comparing the human ranking with the ranking obtained with the proposed matching approach and a classified result set obtained by a match classification approach.

- **Does the involvement of approximate reasoning in the matching process, produce results that better agree with human judgement as opposed to using subsumption reasoning alone?** In the discussion in Section 4.2 we have stated that the involvement of approximate reasoning in a matchmaker system will be more beneficial than using subsumption reasoning alone[9]. Given the fact that the available advertisements are ranked, we show that using the

[8]Using a larger number of subjects in the study may provide greater confidence in the results. However, we believe that providing an evaluation of the match results using a reasonable number of subjects is important to provide an estimate of the effectiveness of a semantic matching solution. As discussed earlier in Section 6.1.1, most semantic matching research efforts have not provided any evaluation of its retrieval effectiveness and therefore we cannot gain an understanding of their practical utility.

[9]We have identified three types of concepts or properties (Type-1, Type-2 and Type-3) occurring in the individual requirements of a request and have argued that subsumption reasoning alone is not sufficient for finding approximate matches when Type-2 and Type-3 properties are involved.

proposed approximate reasoning to produce the ranking is more effective than using subsumption reasoning alone.

- **Does the involvement of priorities in the service request and the matching process, produce matchmaker results that better capture context dependencies and subjective preferences of the domain users?** In Section 4.2 we have discussed the issue, when a resource seeker can have varying priorities or weights on individual property requirements of a request due to subjective preferences and the context involved. Given the fact that the available advertisements are ranked using approximate reasoning, we will show that considering priorities (placed on individual requirements in a request) in the matching process will further improve the effectiveness of the matcher results. Handling mandatory requirements[10] in the matching process, can be viewed as a specific case of priority consideration.

In view of the objectives specified above, the test hypotheses for the effectiveness evaluation experiments are as follows:

H1: *Ranking of potential matches is more effective than the classification of potential matches.*

To test this hypothesis we obtain results from the proposed matcher and the results of the classification scheme proposed in [101] for the same scenario. These results are then compared with the human ranking to test which scheme can better agree with human judgement. The experiment conducted to test this hypothesis is discussed in Section 6.1.5.1

H2: *Given the fact that the available advertisements are ranked, the use of the proposed approximate reasoning in the matching process will produce ranking results that are more effective, as opposed to the case where subsumption reasoning alone is used for ranking.*

To test the effectiveness of approximate reasoning, we obtain matcher results for a scenario where both Type-2 and Type-3 properties occur. The results are also obtained for a matcher that uses subsumption reasoning alone. The human rankings are then obtained and compared with both result sets to find which has a better correlation. Section 6.1.5.2 discusses the experiment conducted to test this hypothesis.

H3: *Given the fact that the available advertisements are ranked using approximate reasoning, priority consideration in the matching algorithm will further improve the effec-*

[10]Mandatory requirements, as discussed in Section 4.2 are requirements which must be strictly met by any potential resource advertisement.

tiveness of the matcher ranking, when varying priorities are associated with the individual requirements in a request.

This hypothesis is tested through a case study involving a context that places varying priorities or weights on the individual requirements of the request. The matcher results are obtained when priority values on the requirements are considered in the matching process. For the sake of comparison and to illustrate the utility of priority consideration, the match results are also obtained in the case where priority values are not considered in the matching process (i.e. when all the requirements are assumed to have equal priority). Human rankings are obtained for the context involved, which is then is compared with the matcher ranking obtained with priority consideration for the context and the matcher results without priority consideration. By analysing the results we will show that when priorities are taken into account in the matching process, the matcher rankings agree better with the human rankings obtained for the same context. Section 6.1.5.3 presents the details and results of the experiment carried out to test this hypothesis.

H4: *Given the fact that the available advertisements are ranked using approximate reasoning, the consideration of mandatory individual requirements in a request during the matching process, will further improve the effectiveness of the matcher ranking.*

As discussed in Section 4.2, mandatory requirements occur when the resource seekers require certain individual property requirement(s) in a request, to be strictly met by any potential resource advertisement; i.e. they will not want to consider any advertisement that will have even a minor deviation, with respect to that property. In the experiments, we will show that strict matching with respect to such mandatory requirements in a request, will further improve the effectiveness of the resultant matcher ranking, when the context involves such mandatory requirements. Handling mandatory requirements in the matching process, can in fact be considered as a specific case of priority matching. The experiment conducted to test this hypothesis is discussed in Section 6.1.5.4.

6.1.5 Experiments & Results

In this section we discuss in detail, the experiments carried out to test the hypotheses stated in the previous section and present the results obtained. For each experiment, a scenario or use case is devised that will involve a resource seeking situation (in a pervasive environment context) where the seeker raises a query for a resource with certain property requirements. A human participant study is carried out (as explained in Section 6.1.3.1) in order to obtain the human ranking for the scenarios involved in each of the experiments. The matcher results are also obtained for the same scenario and in each case the averaged human ranking is used for analysis and comparison of the ranking provided by the proposed matching framework.

6.1.5.1 Ranking of Potential Matches vs Classification of Matches

In Section 4.2, we have argued that a ranked list of potential matches are more beneficial to the users of the matching system than a classified set of matches (as in [101]). In this section we discuss the experiment conducted to investigate this fact and thereby test the hypothesis *H1* stated in the previous section.

In ranking, a number is assigned to each potential match which indicates the order in which it could be considered by the resource seeker (rank 1 for the best match, 2 for the next best and so on). When classifying, each potential match is assigned into a class out of a set of discrete classes. This class assignment can then be used to interpret the degree of match for the potential match involved. For the sake of comparison in this experiment we will take the classification scheme proposed in [101], where the potential matches are classified into Exact, Subsumes, Plug-in and Fail[11]. The Exact matches and Subsumes matches (advertisements that are more specific than the request) are the most preferable, Plug-in matches (advertisements that are more general than the request) can be taken as the next best and the Failed matches are the lowest level[12].

In this experiment we construct a scenario where a resource seeker raises a request for a resource with certain characteristics. The available advertisements in this experiment are chosen so that all four classes of Exact, Plug-In, Subsumes and Fail occur in the match set when classifying the potential matches. We obtain the human rankings for this scenario through a study (as discussed in Section 6.1.3.1) and also obtain the rankings provided by the proposed matchmaker and compare with the results provided with the classification scheme described above.

The specific scenario developed for this experiment is as follows: a resource seeker requires a computer for a certain purpose and specifies the required properties in the resource request which are:

Computer which
- Has Processor Pentium4
- Has Operating System Windows XP
- Has USB 2.0 port
- Has Minimum Physical Memory of 512MB
- Has Minimum Disk Space of 120 Gbytes

The available computers (advertisements) considered for the experiment and their properties are tabulated in Table 6.1. For the purpose of this experiment, an advertisement

[11]The details of these efforts were discussed in more detail in Section 2.4

[12]It could be argued if Subsumes matches are equally good as Exact matches or whether it comes at the second level or third level (after Plug-in matches). However this will depend on the domain and the context involved and the discussion provided in Section 4.3.1 will clarify the point of view applicable in our case.

of a computer is considered to be a tuple of the form[13]:

$Advertisement = (OS, Processor, USBport, Memory, DiskSpace)$, where
$OS = \{Windows, WinXP, WinXPprof, WIN2000, Unix, Linux, SunOS\}$,
$Processor = \{P4, P3, P2, P1, AthlonXP, Athlon\}$,
$USBport = \{USB, USB1.0, USB2.0\}$ and
$Memory$ and $DiskSpace$ are numeric attributes.

Further knowledge about $USBport$ and OS are given as:

$USB1.0 \sqsubseteq USB$, $USB2.0 \sqsubseteq USB$ (i.e. both $USB1.0$ and $USB2.0$ are types of USB ports)

$WinXPprof \sqsubseteq WinXP \sqsubseteq Windows$, $WIN2000 \sqsubseteq Windows$,
$Linux \sqsubseteq Unix$, $SunOS \sqsubseteq Unix$

In order to judge how a domain user would rank the available advertisements in this scenario, a human participant study was conducted using the Questionnaire 1 given in Appendix A. Rankings were obtained from 12 subjects; the summarised observations of the completed questionnaires are also included in Appendix A. The matcher rankings were also obtained for these advertisements in response to the request. Table 6.2 tabulates the average human ranking and the rankings delivered by the Semantic Matcher. This also indicates the classification given for each advertisement under the classification scheme discussed in [101] (which will be one of Exact, Subsumes, Plug-in or Fail, as described earlier in this section). For the sake of comparison, these class assignments are then interpreted into a ranking depending on the degree of match: i.e. by taking into account the fact that Exact and Subsumes matches are the best, Plug-In matches are the next best and the Failed matches are the worst.

Figure 6.3 graphically illustrates: (1) the difference between the average human ranking and the Semantic Matcher ranking (2) the difference between the average human ranking and the ranking obtained by interpreting the match class. From this graph it can be observed that the Semantic Matcher ranking is much closer to the human ranking as opposed to the ranking obtained though the classified results.

By analysing the ranking given in Table 6.2, we can observe that although certain advertisements have been assigned to the same match class (and thus can be interpreted as being equally good since they have the same "degree of match"), the domain users tend to distinguish between the suitability of such advertisements. For example, in this experiment the advertisements that have been classified as Plug-In matches (Advert-3, Advert-4, Advert-5 which are all ranked in the 3rd place) have been assigned distinct

[13]The Device Ontology supports the description of further characteristics and functionality of computers and devices in general, as discussed in Chapter 3. However, in this section, we only describe what is relevant for the purpose of this experiment.

Advert	***OS***	***Processor Type***	***USB port***	***Memory (Mbytes)***	***Disk Space (Gbytes)***
Ad 1	Win XP	P4	USB 2.0	512	120
Ad 2	Win XP Prof	P4	USB 2.0	512	120
Ad 3	Windows	P4	USB 2.0	1024	160
Ad 4	Windows	P4	USB	1024	120
Ad 5	Windows	P4	USB	768	Unknown
Ad 7	Win XP Prof	P4	USB 2.0	512	80
Ad 8	Win XP Prof	P4	USB 2.0	128	80
Ad 9	Linux	P4	USB 2.0	128	60
Ad 10	Linux	P2	USB 1.0	128	60

TABLE 6.1: Available Advertisements of Computers

Advert	***Semantic Matcher Rank***	***Match Class***	***Scaled Rank (for comparison)***	***Average Human Rank***
Advert 1	1	Exact	1	1.54
Advert 2	1	Subsumes	1	1.23
Advert 3	3	Plug-In	3	3
Advert 4	4	Plug-In	3	4.66
Advert 5	6	Plug-In	3	6.31
Advert 6	4	Fail	6	4.46
Advert 7	6	Fail	6	5.69
Advert 8	8	Fail	6	6.78
Advert 9	9	Fail	6	8.31

TABLE 6.2: Semantic Matcher Ranking, Match Classification Results and Average Human Ranking

ranks as per average human ranking. The same observation can be made with respect to Failed matches: Advert-6, Advert-7, Advert-8, Advert-9 which are ranked last. Thus a classification approach cannot distinguish between the suitability of advertisements within each match class. However, the matcher rank has been more in agreement with the human ranking as can be observed.

	precision	***recall***	***F-measure***	***standard deviation (%)***
Classifier	0.80	0.95	0.87	17.48
Semantic Matcher	0.94	0.94	0.94	6.54

TABLE 6.3: Precision, recall, F-measure and standard deviation for the Semantic Matcher and the classifier

The precision, recall, F-measure and the standard deviation for both the Semantic

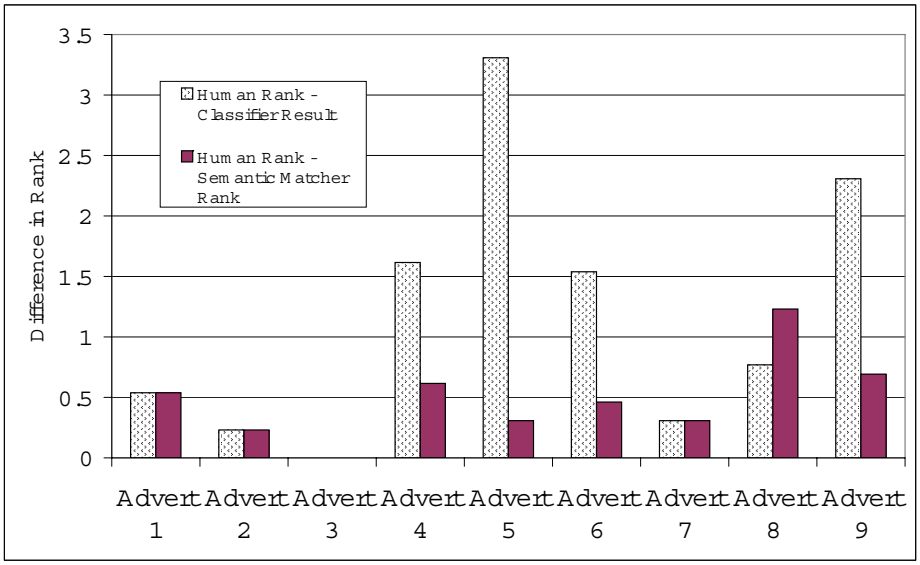

FIGURE 6.3: The Difference Between the Rankings: Human Rank - Semantic Matcher Rank and Human Rank - Rank interpreted from Match Classification

Matcher ranking and the ranking obtained through the classifier are given in Table 6.3. The standard deviation for the classifier (which indicates the deviation between the resultant rankings and the human rankings) is 17.48% which is much higher than the standard deviation for the Semantic Matcher which is 6.54%. This indicates that the classifier has a much higher deviation from the human ranking as opposed to the Semantic Matcher. The F-measure (which gives a combined measure of effectiveness using recall and precision) for the Semantic Matcher is 0.94 which is higher than the F-measure for the classifier which is 0.87. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that the Semantic Matcher ranking agrees better with human perception as opposed to the classifier output.

Hence the results support the hypothesis *H1* that, ranking of potential matches is more effective than the classification of potential matches.

6.1.5.2 Effect of Approximate Reasoning in the Matching Process

In Section 4.2, we have discussed the importance of approximate or non-exact matching; we have mentioned that the advertisements should be ranked depending on how much it deviates from the request concerned[14]. To do this, the matching mechanism will have

[14]The theory behind the use of approximate reasoning in the matching process has been discussed in detail in Chapter 4

to consider to what extent the advertisements deviate with respect to each required attribute in the request and it will have to consider the type of attribute involved in the resource description when approximating and match ranking. There are three types of attributes considered in the matching process (that has been discussed in section Section 4.4.2); which are Type-1, Type-2 and Type-3. We have argued that reasoning based on the subsumption relation alone is not sufficient when Type-2 and Type-3 attributes are involved in the request and advertisement descriptions. Approximate matching will have to be carried out with respect to these two attribute types in order to provide effective ranking of available resources. We have conducted an experiment to test the hypothesis *H2* and thereby show that approximate reasoning with respect to both these attribute types are more effective and agrees better with human ranking as opposed to when using subsumption reasoning alone.

To show the added utility of using approximate reasoning in the matching process when compared to the use of subsumption reasoning alone, we devise a scenario where the resource request contains both Type-2 and Type-3 properties. The advertisements will have varying values for these properties; certain advertisements will have values within the requested range specified in the request, others will deviate from the specified range in different extents. Human rankings for these advertisements are obtained to identify the human perception related to these deviations of the properties. The average human rankings will then be compared with the resultant ranking of the Semantic Matcher and the rankings obtained through the *subsumption matcher*[15].

The specific scenario constructed for this experiment is as follows: a resource seeker needs to make use of a Computer to run a certain application. Depending on the requirement, he/ she raises the request which specifies the characteristics that needs to be present in a potential matching resource. The request for the Computer is:

Computer with

Processor Type Pentium4
Has Minimum Physical Memory of 512MB
Has Minimum Disk Space of 120 Gbytes

The request specified in this experiment has one Type-2 property (Processor Type) and two Type-3 properties (Memory Size and Disk Space) involved. The available advertisements for computers are chosen so that they will have different values for these

[15]For the sake of this experiment, we compare the resultant rankings of the Semantic Matcher with the rankings produced by a *subsumption matcher*. To allow for a fair comparison, the subsumption matcher used here, will match the request and advertisements by each property or attribute specified in the request when ranking, but will only use subsumption reasoning to measure the deviation within each attribute: i.e. it will follow the same matching process as the Semantic Matcher, but will consider all the requirements of a request as Type-1 properties.

attributes, some meets the specified requirement and some do not and they can deviate from the specified criteria in different levels. The available computers considered for the experiment are tabulated in Table 6.4. For this experiment, an advertisement is considered to be a tuple of the form: $Advertisement = (Processor, Memory, DiskSpace)$, where $Processor = \{P4, P3, P2, P1, AthlonXP, Athlon\}$, $Memory$ and $DiskSpace$ are numeric attributes.

Advert	***Memory (Mbytes)***	***Disk Space(Gbytes)***	***Processor Type***
Advert 1	512	120	P4
Advert 2	512	110	P4
Advert 3	1024	160	P4
Advert 4	1024	80	P4
Advert 5	768	60	P4
Advert 6	256	120	P4
Advert 7	128	120	P4
Advert 8	256	60	P4
Advert 9	256	110	P4
Advert 10	512	120	AthalonXP
Advert 11	512	140	P1
Advert 12	512	140	P3

TABLE 6.4: Available Advertisements of Computers

We have obtained human ranking for this use case from 12 individuals and the average human ranking has been computed. The Questionnaire 2 given in Appendix A has been used to obtain responses for this experiment. The summarised observations of the responses to this questionnaire are also included in Appendix A. Table 6.5 illustrates the averaged human ranking along with the ranking provided by the Semantic Matcher and the ranking that can be obtained from the *subsumption matcher*.

Advert	***Averaged Human Ranking***	***Subsumption Matching***	***Semantic Matcher***
Advert 1	1.38	1	1
Advert 2	4.00	3	3
Advert 3	1.15	1	1
Advert 4	5.54	3	6
Advert 5	7.08	3	7
Advert 6	6.38	3	8
Advert 7	9.69	3	11
Advert 8	9.54	11	12
Advert 9	8.46	11	9
Advert 10	2.54	3	4
Advert 11	9.92	3	10
Advert 12	7.54	3	5

TABLE 6.5: Averaged Human Ranking, Subsumption Matcher Ranking and Semantic Matcher Ranking

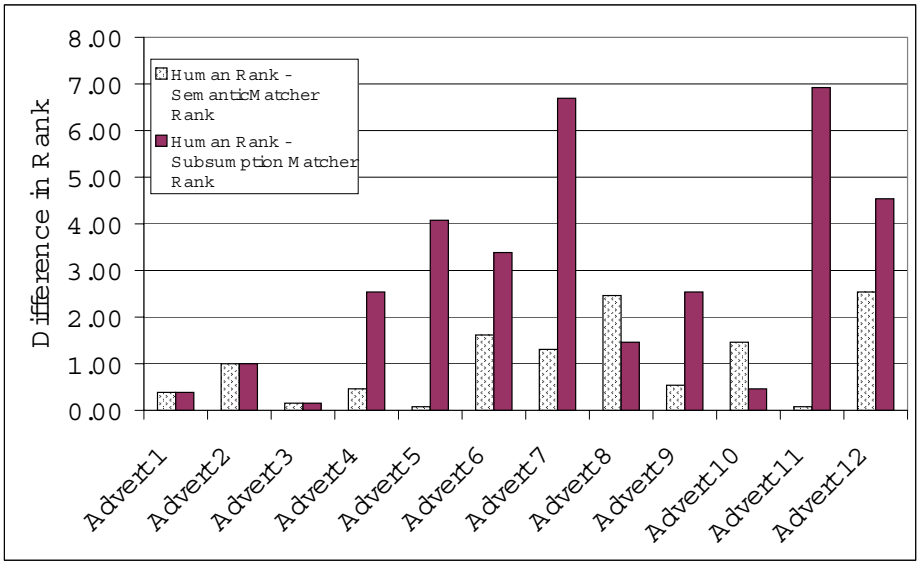

FIGURE 6.4: The Difference Between the Rankings: Human Rank - Subsumption Matcher Rank and Human Rank - Semantic Matcher Rank

On closer scrutiny of the ranks received by the advertisements; it can be observed that certain advertisements have been ranked equally by the subsumption matcher although they have been ranked quite differently according to average human ranking. The Semantic Matcher ranking however is more in agreement to the human ranking. For example, the subsumption matcher has ranked both Advert-2 and Advert-5 equally in 3rd place since both these advertisements do not meet the specified criteria. However these have received different ranks in the average human ranking (Advert-2 as 4.0 and Advert-5 as 7.08), since these deviate from the specified Disk Space requirement in different amounts. Since the Semantic Matcher employs approximate reasoning with respect to Type-3 properties (Disk Space is a Type-3 property), it is able to distinguish between these two advertisements and has given Advert-2 and Advert-5 a rank of 3 and 7 respectively. A similar observation can be made w.r.t. Advert-10 and Advert-11: these have been ranked equally by the subsumption matcher because both advertisements do not meet the Processor requirement. However, they have been ranked as 2.54 and 9.92 respectively according to human ranking (since Advert-10 has "AthalonXP" processor which is considered as having a similar performance level to the requested "P4" processor and Advert-11 has "P1" processor which has a much lesser performance compared to "P4"). Since the proposed matcher employs approximate reasoning with respect to Type-2 properties (Processor is a Type-2 property), it is able to distinguish between these two advertisements appropriately.

Figure 6.4 graphically illustrates: the difference between the average human ranking and

	precision	***recall***	***F-measure***	***standard deviation (%)***
Subsumption Matcher	0.69	0.93	0.79	30.22
Semantic Matcher	0.94	0.89	0.91	10.93

TABLE 6.6: Precision, recall, F-measure and standard deviation for the subsumption matcher and the Semantic Matcher

the Semantic Matcher ranking and the difference between the average human ranking and the subsumption matcher ranking. From this graph it is evident that the proposed matcher ranking is much closer to the human ranking as opposed to the subsumption matcher.

The precision, recall, F-measure and the standard deviation for both the Semantic Matcher and the subsumption matcher are given in Table 6.6. The standard deviation for the subsumption matcher (which indicates the deviation between the subsumption matcher rankings and the human rankings) is 30.22% which is much higher than the standard deviation for the Semantic Matcher which is 10.93%. This indicates that the subsumption matcher has a much higher deviation from the human ranking as opposed to the Semantic Matcher. The F-measure (which gives a combined measure of effectiveness using recall and precision) for the Semantic Matcher is 0.91 as opposed to the subsumption matcher which has 0.79 for the same metric. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that the Semantic Matcher through the involvement of approximate matching, has delivered results that better agrees with human perception as opposed to the subsumption matcher and thus is more effective.

Hence the results of this experiment supports the hypothesis *H2* that: given the fact that the available advertisements are ranked, the use of the proposed approximate reasoning in the matching process will produce ranking results that are more effective, as opposed to the case where subsumption reasoning alone is used for ranking.

6.1.5.3 Matching when the Individual Property Requirements of a Request have Varying Priorities

The motivation for allowing priorities for individual property requirements was discussed in Section 4.2.4. We have pointed out that in many practical scenarios, the resource seekers will consider certain service requirements as being more important than others. Thus the service description should allow for the description of such priorities[16] and these priorities must be considered during the matching process. In this section we discuss the experiment conducted to investigate the effect of considering such priority

[16]description of priorities in the service description is discussed in Section 4.4.1

requirements in the matching process, and thereby test the hypothesis *H3* (stated in Section 6.1.4).

To evaluate the benefits of priority consideration, we devise a scenario where a resource seeker needs a device with a number of properties. The resource request is raised under a context which places varying priorities on the different property requirements: i.e. some requirements of the request are more important than the others.

The specific scenario developed for the purpose of this experiment is as follows: a resource request for a printer is raised under a certain context; the request specifies the need for a *Laser* printer that prints in *colour* and supports *A2* paper size (three property requirements). Under the context considered, the topmost priority will be placed on the paper size property. The context of the request is: the resource seeker considers that the printed content will appear best on A2 paper size, hence Paper Size is considered as the topmost priority (other two properties will have equal priority).

The printer request considered for the experiment is:

Printer that
Can Print in Colour
Supports Paper Size A2
Has Printer Technology Laser

The available advertisements of printers are as tabulated in Table 6.7. For this experiment, an advertisement of a printer is considered to be a tuple of the form:

$Advertisement = (PrintingColour, PrintingTechnology, PaperSize)$, where
$PrintingColour = \{Colour, BW\}$,
$PrintingTechnology = \{Laser, Inkjet\}$ and
$PaperSize = \{A2, A3, A4\}$.

The human rankings for the scenario were obtained from 12 individuals through the human participant study: Questionnaire 3 given in Appendix A is used to obtain the human ranking and the responses obtained are also given in the appendix. The participating subjects were asked to assume that the printer is sought under the context involved (that the paper size property is more important than the other properties) and to rank the available printers accordingly. The Semantic Matcher results have also been obtained, considering the fact that the paper size property has higher priority than the others[17].

[17]For the sake of this experiment, we have assumed that the paper size requirement has a priority of 0.6 and both the other requirements have a priority of 0.2. In practical situations, the priorities for the individual requirements will be specified in the request by the resource seeker. Section 4.4.1 explains how the priorities can be specified in the request description.

Advert	***Max Supported Paper Size***	***Printing Technology***	***Printing Colour***
Advert 1	A2	Laser	BW
Advert 2	A2	Laser	Colour
Advert 3	A2	Inkjet	Colour
Advert 4	A2	Inkjet	BW
Advert 5	A3	Laser	BW
Advert 6	A3	Laser	Colour
Advert 7	A3	Inkjet	Colour
Advert 8	A4	Laser	BW
Advert 9	A4	Laser	Colour
Advert 10	A4	Inkjet	Colour
Advert 11	A3	Inkjet	BW
Advert 12	A4	Inkjet	BW

TABLE 6.7: Available advertisements of printers

To illustrate the added effectiveness of priority consideration, the matcher results have to be compared with the case when there is no priority consideration in the matching process. Thus we also obtain the results from the Semantic Matcher, assuming that there are equal priorities on all three properties. This is equivalent to the case when there is no priority consideration in the matching process. Table 6.8 illustrates the averaged human ranking obtained and the ranking provided by the Semantic Matcher (with priority consideration and without priority consideration).

Advert	***Average Human Ranking***	***Semantic Matcher Rank (Without Priority Consideration)***	***Semantic Matcher Rank (With Paper Size as 1st Priority)***
Ad 1	3.23	3	3
Ad 2	1	1	1
Ad 3	2.08	2	2
Ad 4	4.42	5	4
Ad 5	6.83	8	7
Ad 6	4.67	3	5
Ad 7	5.67	5	6
Ad 8	10.08	11	11
Ad 9	7.92	7	9
Ad 10	8.92	9	10
Ad 11	7.67	10	8
Ad 12	11.08	12	12

TABLE 6.8: Average Human Ranking and the Semantic Matcher Rankings with and without Priority Consideration

On closer scrutiny of the rankings received by the advertisements in Table 6.8, it can be observed that the Semantic Matcher ranking with priority consideration is closer to the average human ranking as opposed to the Semantic Matcher ranking without priority consideration. For example, Advert-9 (ranked 7th) has been ranked as being better

than Advert-5 (ranked 8th) by the matcher when ignoring the priority on the paper size attribute. However, according to the average human ranking Advert-5 (ranked 6.83) is better than Advert-9 (ranked 7.92); intuitively this is because Advert-5 meets the paper size criterion although it is worse off in the colour attribute. When priority is considered in the matching process Advert-5 is ranked better than Advert-9, and is thus in agreement with the human ranking.

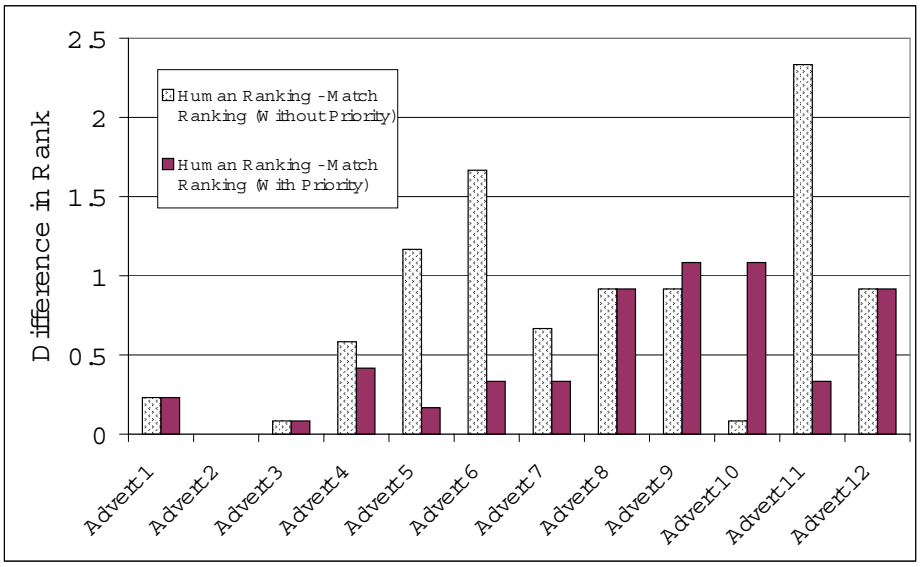

FIGURE 6.5: *The Difference Between the Averaged Human Ranking and the Semantic Matcher Rankings: With and Without Priority Consideration*

	precision	***recall***	***F-measure***	***standard deviation (%)***
Semantic Matcher (without priority consideration)	0.91	0.94	0.93	8.67
Semantic Matcher (with priority consideration)	0.93	0.99	0.96	5.17

TABLE 6.9: Precision, recall, F-measure and standard deviation for the Semantic Matcher: with and without priority consideration

Figure 6.4 graphically illustrates: the difference between the average human ranking and the Semantic Matcher ranking when priorities are not considered (i.e. assuming all requirements have equal priorities) and the difference between the average human ranking and the Semantic Matcher ranking with priority consideration. From this graph it can be observed that the Semantic Matcher ranking with priority consideration agrees better with the human ranking (since the difference between the two rankings is smaller)

as opposed to the case when there is no priority consideration.

The precision, recall, F-measure and the standard deviation for both cases (the Semantic Matcher ranking with priority consideration and Semantic Matcher ranking without priority consideration) are given in Table 6.9. The standard deviation for the Semantic Matcher without priority consideration is 8.67% which is higher than the standard deviation for the Semantic Matcher with priority consideration, which is 5.17%. This indicates that the Semantic Matcher ranking obtained when priorities are disregarded, has a higher deviation from the human ranking as opposed to the case when priorities are considered by the Semantic Matcher. The F-measure for the Semantic Matcher with priority consideration is 0.96 as opposed to the Semantic Matcher without priority consideration which has 0.93 for the same metric. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that when priorities are considered by the matcher, the delivered results agrees better with human perception as opposed to the case when priorities are disregarded. That is, the Semantic Matcher results have become more effective when the priorities on the individual requirements are taken into account during the matching process.

Hence the results of this experiment supports the hypothesis *H3* that: given the fact that the available advertisements are ranked using approximate reasoning, priority consideration in the matching algorithm will further improve the effectiveness of the matcher ranking, when varying priorities are associated with the individual requirements in a request.

6.1.5.4 Matching when Mandatory Requirements are present in the Request

The motivation for considering such mandatory requirements[18] in the matching process has been discussed in Section 4.2.4. We have argued that when such mandatory requirements are present, strict matching will have to be carried out with respect to those requirements, in order to produce match results that are effective for the context involved.

In this section we discuss the experiment conducted to investigate the effect of considering mandatory requirements in the matching process and thereby test the hypothesis *H4* (stated in Section 6.1.4).

For this experiment we devise a scenario where a resource seeker needs to utilise a certain device with a number of properties. The request will specify the properties and their required values. To demonstrate the utility of incorporating mandatory attributes,

[18]Mandatory requirements are individual property requirements in a request, that the resource seeker requires to be strictly met by any potential resource advertisements; i.e. he will not want to consider any advertisements that will have even a minor deviation, with respect to that property. Such mandatory requirements can occur due to the context that has given rise to the resource need.

the scenario is constructed such that both mandatory and non-mandatory individual requirements are present in the request. The advertisements are chosen such that some of the advertisements meet the mandatory requirement and the others do not. The Semantic Matcher results will be obtained, when mandatory requirement(s) are considered in the matching process. For the sake of comparison we also obtain the results assuming that the mandatory requirement(s) are not considered, i.e. when all the attributes are considered for flexible/ approximate matching. By comparing the two resultant rankings with that provided by domain users, we analyse the effects of considering mandatory requirements in the matching process, on the match results produced. The averaged human rankings were obtained using a human participant study.

The specific scenario developed for this experiment is as follows: a resource seeker needs to utilise a computer to run a certain application. Depending on the job at hand he specifies the individual characteristics of the computer he is looking for. The request for the Computer is:

Computer with

Processor Type Pentium4
Has Operating System Windows
Has Minimum Physical Memory of 512Mbytes
Has Minimum Disk Space of 120 Gbytes

The context of the scenario is that, the computer is needed to run a particular application that will only run on a Windows based OS and hence cannot run on a SunOS or Linux. Thus the *Operating System* requirement is a mandatory individual requirement in this case. Therefore this requirement must be strictly met in any potential advertisement for it to be considered as a match. The available computers considered for the experiment are tabulated in Table 6.10. For the purpose of this experiment, an advertisement is considered to be a tuple of the form:

$Advertisement = (OS, Processor, Memory, DiskSpace)$, where
$OS = \{Windows, WinXP, WinXPprof, WIN2000, Unix, Linux, SunOS\}$,
$Processor = \{P4, P3, P2, P1, AthlonXP, Athlon\}$,
$Memory$ and $DiskSpace$ are numeric attributes.

Further knowledge about OS is given as:

$WinXP \sqsubseteq Windows$, $WIN2000 \sqsubseteq Windows$,
$Linux \sqsubseteq Unix$, $SunOS \sqsubseteq Unix$

A human participant study was conducted using the Questionnaire 4 given in Appendix A. The subjects were expected to rank the available computers, bearing the

Advert	***Processor***	***Operating System***	***Memory (Mbytes)***	***Disk Space (Gbytes)***
Advert 1	P4	WIN XP	512	160
Advert 2	AthlonXP	WIN XP	512	180
Advert 3	P4	LINUX	1024	140
Advert 4	P4	WIN XP	1024	110
Advert 5	P4	SUN OS	768	110
Advert 6	P4	WIN XP	512	120
Advert 7	P4	SUN OS	512	20
Advert 8	P2	WIN XP	256	100
Advert 9	P3	WIN 2000	256	70
Advert 10	P2	WIN XP	128	60

TABLE 6.10: Advertisements of available computers

context in mind. The rankings were obtained from 12 individuals and the responses obtained are also given in Appendix A.

The averaged human ranking is depicted in Table 6.11. We have also obtained the Semantic Matcher results; first by allowing approximate or flexible matching for all property requirements (i.e. any deviations in the requirement will be scored appropriately); secondly by considering the Operating System requirement as a mandatory requirement (thereby all the computers not having a Windows OS are ranked last). Table 6.11 also illustrates these resultant rankings in both cases, when the mandatory requirement is considered and is not considered.

Advert	***Avg. Human Ranking***	***Semantic Matcher (when disregarding the mandatory OS requirement)***	***Semantic Matcher (when considering mandatory requirement)***
Advert 1	1.15	1	1
Advert 2	2.31	4	4
Advert 3	7.23	5	8
Advert 4	3.38	3	3
Advert 5	7.84	6	8
Advert 6	1.84	1	1
Advert 7	8.61	9	8
Advert 8	5.77	8	6
Advert 9	5.54	7	5
Advert 10	6.84	10	7

TABLE 6.11: Average Human Ranking and Semantic Matcher Rankings obtained with and without consideration of the Mandatory Requirement

On closer observation of the advertisements' properties and the rankings provided by humans and the Semantic Matcher, it can be observed that the matcher rankings agrees better with the human rankings when the mandatory requirement is considered. For ex-

ample, according to both the human ranking and the Semantic Matcher ranking (when mandatory requirements are considered), Advert-3, Advert-5 and Advert-7 have received the worst ranks out of the 10 advertisements. Intuitively, this is because all these three advertisements do not meet the mandatory OS requirement. However, when the mandatory requirement is disregarded, the Semantic Matcher has ranked these differently and we can see Advert-3 has a relative higher rank (5th out of 10 advertisements). Thus we can observe that, when the Semantic Matcher provides strict matching with respect to mandatory requirements, it is able to appropriately penalise any advertisements that do not meet the mandatory requirements, thereby giving such advertisements a lower rank. Thus the match results become more suitable for the context involved.

Figure 6.6 graphically illustrates: (1) the difference between the average human ranking and the Semantic Matcher ranking when the mandatory requirement is not considered (i.e. allowing approximate matching for all requirements) and (2) the difference between the average human ranking and the Semantic Matcher ranking when the mandatory requirement is considered. From this graph it can be observed, that the Semantic Matcher ranking obtained when the mandatory requirement is considered agrees better with the human ranking (since the difference between the two rankings is lesser) as opposed to the case when it is not considered.

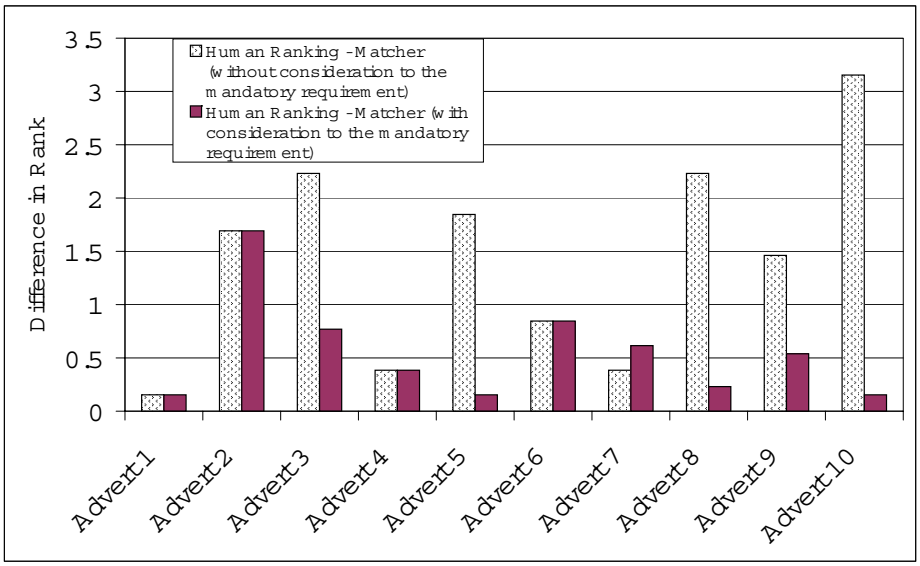

FIGURE 6.6: The Difference Between the Averaged Human Ranking and the Semantic Matcher Rankings: With and Without Consideration to the Mandatory Requirement

The respective values for precision, recall, F-measure and the standard deviation are given in Table 6.12. The standard deviation for the matcher with mandatory requirement consideration (which indicates the deviation between the Semantic Matcher rankings

	precision	***recall***	***F-measure***	***standard deviation (%)***
Semantic Matcher (without mandatory requirement consideration)	0.88	0.82	0.85	14.37
Semantic Matcher (with mandatory requirement consideration)	0.95	0.94	0.94	6.52

TABLE 6.12: Recall, precision, F-measure and standard deviation for the Semantic Matcher: with and without consideration to mandatory requirements

and the human rankings) is 6.52% which is much less than the standard deviation for the Semantic Matcher rankings that disregards the mandatory requirement, which is 14.37%. This indicates that the Semantic Matcher ranking obtained when mandatory requirements are disregarded, has a higher deviation from the human ranking as opposed to the case when they are considered as mandatory by the matcher. The F-measure for the Semantic Matcher with mandatory requirement consideration is 0.94 as opposed to the Semantic Matcher without mandatory requirement consideration, which has 0.85 for the same metric. Therefore, through the anlysis of the overall results obtained from the above experiment it can be observed that when mandatory requirements are considered by the Semantic Matcher (i.e. strict matching is carried out w.r.t. the mandatory requirements), the delivered results agrees better with human perception, as opposed to the case when mandatory requirements are disregarded. That is, the matcher results become more effective when strict matching is employed with respect to mandatory requirements in the request.

Hence the results of this experiment supports the hypothesis *H4* that: Given the fact that the available advertisements are ranked using approximate reasoning, the consideration of mandatory individual requirements in a request during the matching process, will further improve the effectiveness of the matcher ranking.

6.1.6 Discussion

In this section we have presented an evaluation of the retrieval effectiveness of the Semantic Matcher. As pointed out by Tsetsos et. al. in [130], most available semantic matching approaches have not undergone any systematic evaluation, due to the fact that there are no widely accepted methods for evaluating effectiveness of semantic matching solutions.

In information retrieval however, the precision and recall metrics have been widely used for measuring the effectiveness of the solutions. Certain research efforts (presented

in [141], [41]) have adopted precision and recall metrics for effectiveness evaluation. However, Tsetsos et. al. in [130] have identified limitations associated with using the traditional recall and precision metrics for the evaluation of semantic matching solutions that return a fuzzy relevance value. Buell et. al. in [22] have presented the generalised recall and precision measures which can be used to evaluate systems that return a fuzzy relevance. We have used these generalised measures of recall and precision together with standard deviation and graphical plots to measure the effectiveness of the Semantic Matcher results.

A human participant study has been carried out to obtain the human rankings applicable to the scenarios used in the experiments. These rankings were then used to compare against the Semantic Matcher ranking when measuring effectiveness.

We have described several desirable properties for a matching mechanism in Section 4.2. In this section we have conducted several experiments to investigate how the presence of these properties can improve the effectiveness of the Semantic Matcher. Through the analysis of the experimental results obtained, we have shown that:

- Ranking of potential matches agrees better with human perception and is more effective than the classification of potential matches.
- The Involvement of approximate matching in the Semantic Matcher has delivered rankings that better agree with human perception and thus is more effective, as opposed to the case where subsumption reasoning alone is used for ranking.
- When varying priorities are associated with the individual requirements in a request, priority consideration in the matching process will improve the effectiveness of the matcher ranking, as opposed to the case when all requirements are assumed to have equal priority.
- When mandatory requirements are considered by the Semantic Matcher, the delivered results agree better with human perception, as opposed to the case when mandatory requirements are disregarded. That is, the Semantic Matcher results become more effective when strict matching is employed with respect to mandatory requirements in the request.

The human participant study conducted to obtain the human rankings for the experiments, involved handing out questionnaires to the participants and obtaining responses/ rankings for the scenarios involved. The questionnaires used to obtain human rankings and the aggregated responses to the questionnaires are included in the Appendix A .

When the overall responses to the questionnaires are considered, it can be observed that in most instances, the subjects seem to agree on the rankings of the advertisements;

particularly there is agreement about the "best" and the "worst" advertisements applicable in the scenario. However, there is a noticeable variance in the responses/ rankings given by the different subjects involved in the study. This variation can be due to subjective preferences. The subjects have assumed that certain properties or requirements are more important than others. For example, certain subjects have assumed that the *Colour* property is more important than the *Printing Technology* property, when completing Questionnaire 3. Another reason for this variance can be human error. However since the rankings are averaged, the effects due to such variations are minimised.

There were instances where the subjects involved in the study, have assumed there own context when ranking the resources given on the questionnaire. For example, when responding to Questionnaire-1, a certain subject have assumed that the resource is requested for connecting an external device and hence have ignored variations in the advertisements w.r.t. the memory and hard disk properties. In another instance, one subject has assumed that *disk space* can be easily upgraded and therefore has disregarded variations *disk space capacity* in the advertisements when ranking.

The questionnaire requests the subjects to *assume* a particular context that the request is raised in; thus the study involves "artificially" created contexts and requests. However, since the scenarios involved are common situations that a typical computer user encounters during everyday life, it is relatively straight forward to complete the questionnaire.

6.2 Evaluating the Performance of Semantic Matching

The proposed semantic matching approach must have a reasonable level of performance (w.r.t. matching time) for its practical use in facilitating the discovery of resources. Therefore, we evaluate the performance of the solution using the prototype implementation of this system, through the use of two experiments. Specifically, we investigate the scalability of the solution in terms of the number of advertisements matched and the size of the resource request. The objective of this evaluation exercise is to investigate the variation in execution time of the matching process, when the number of advertisements matched and the size of the resource request increases. If the Semantic Matcher is scalable, the execution times must be acceptable, for reasonable numbers of advertisements and request sizes.

We have devised two tests in this regard:

1. Matching time against increasing numbers of advertisements: In these experiments, the execution time taken for the matching process was obtained for increasing numbers of advertisements. The hypothesis tested is:

H5: *The execution time for the matching process is 'satisfactory', for 'reasonable' numbers of advertisements (for most practical purposes)*[19].

The test conducted in this respect is discussed in detail in Section 6.2.1.

2. Matching time against increasing request sizes: The execution time was obtained for increasing request sizes. The size of a request refers to the number of individual requirements it contains. The test hypothesis is:

   H6: *The execution time for the matching process is 'satisfactory' for 'reasonable' request sizes.*[20]

   Section 6.2.2 discusses the test conducted in this respect.

The experiments were carried out using a 3.2GHz, Intel PentiumD PC with 2GB of memory and the execution times were averaged over 30 runs.

6.2.1 Scalability w.r.t. Number of Advertisements

We test the scalability of the system in terms of the number of advertisements involved in the matching process. The number of advertisements available for matching is varied between 10 and 10000 and the execution time taken for the matching process is measured in milliseconds. Since we are evaluating the variation of execution time in response to the variation of the number of advertisements, the size of the resource request must be kept constant. For this experiment, we keep the size of the request constant at 4 individual requirements.

We obtain two sets of results:

1. When the advertisements are described using the Printer Ontology[21] which contains 126 concepts, 67 properties and 65 restrictions.
2. When the advertisements are described using the Computer Ontology[22] which contains 156 concepts, 103 properties and 75 restrictions.

The execution times obtained in each case are tabulated in Table 6.13. Figure 6.7 graphically illustrates the execution times for both ontologies.

[19] For example, we can assume that, the maximum number of devices available in a large enterprise will be typically around 500 - 1000. Later in this chapter, we discuss the possible solutions that can be applied, when the number of advertisements go beyond these limits. We can also assume that, an execution time of approximately 10s is an acceptable time for matching.

[20] Assuming that, the number of requirements that can be expected in a device request in most typical pervasive environments, could range from 3-6.

[21] The Printer Ontology is a specialisation of the generic Device Ontology and defines additional concepts and properties necessary to describe printers (such as printer resolution, supported media types and printing speed). Creating specialised ontologies to describe specific devices was discussed in Section 3.4.

[22] Again, this is a specialisation of the generic Device Ontology and defines additional concepts and properties necessary to describe computers.

Number of Advertisements	***Printer Ontology (ms)***	***Computer Ontology (ms)***
10	535	594
50	1562	1766
100	2578	2937
200	4594	5141
500	9797	11266
1000	18719	20328
2000	37156	40328
5000	98110	104938
10000	212500	221156

TABLE 6.13: Average execution time for increasing numbers of advertisements

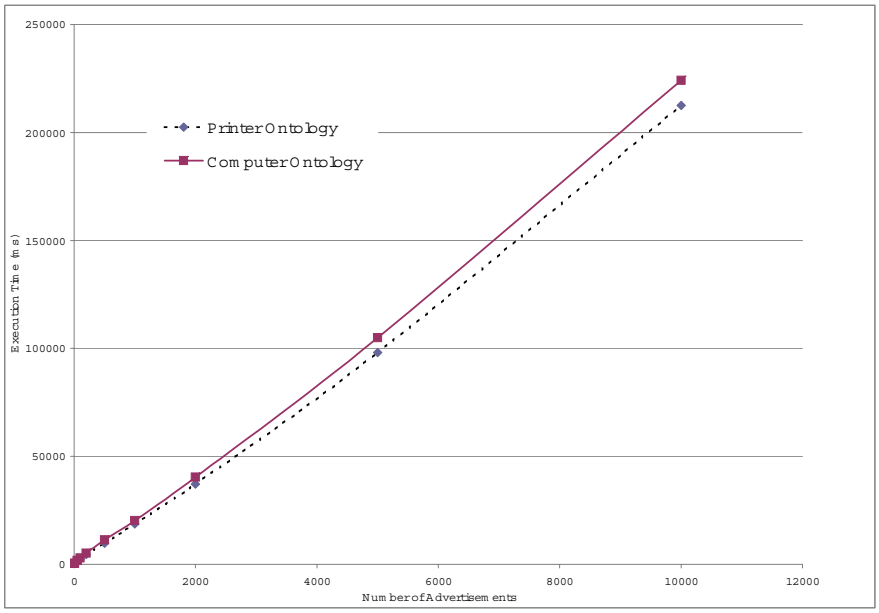

FIGURE 6.7: Number of Advertisements Vs Execution Time

It can be observed from the two plots, that for both ontologies the execution times for the matching process keeps increasing. The execution time becomes noticeably high, when the number of advertisements involved is high. For example, it has taken approximately 37 seconds to match 2000 advertisements with a request; this will mean a response time of 37 seconds when 2000 advertisements are present. Although the matching times are relatively low for small numbers of advertisements, these response times may become undesirable in the presence of a large number of advertisements. To overcome this issue, load balancing solutions that will distribute the matching load between a number of nodes [77, 19], can be used. Such solutions will help to lower the overall time taken for matching and thereby improve the resultant response times. However, any detailed

investigation into such solutions, is not within the scope of this thesis and hence will not be discussed in this document.

It can also be observed that, the execution times taken when the advertisements and requests are described using the Computer ontology (which is the larger ontology), are generally higher when compared to the execution times related to the Printer ontology. This may be due to the fact that, when the size of the ontology is larger, the knowledge base that the reasoning mechanism has to deal with becomes larger and thus this can affect the execution time.

Although the plots seems almost linear, on closer observation of the execution times, it can be seen that the gradient of the plot keeps gradually increasing (from 17.34 to 22.88 for the plot related to the Computer ontology) when the number of advertisements increases. However, the rate of the increase observed is low.

The execution time taken to match reasonable numbers of advertisements, can be observed to be within acceptable limits. For example, when the number of advertisements is 200 and 500, the matching time taken is approximately 4.5s and 9.8s respectively (for Printer Ontology). Thus, the results support the hypothesis *H5* that, the execution time for the matching process is satisfactory, for reasonable numbers of advertisements.

6.2.2 Scalability w.r.t. the Size of the Resource Request

In this experiment, we test the scalability of the system in terms of the size of the resource request; i.e. the number of individual requirements involved in the request. We vary the number of individual requirements in the resource request between 1 and 7 and measure the execution time taken by the matching process (while keeping the number of advertisements constant at 50).

For this case again, we obtain two sets of results for the two ontologies:

1. When the advertisements are described using the Printer Ontology.
2. When the advertisements are described using the Computer Ontology.

The execution times obtained are as tabulated in Table 6.14. Figure 6.8 graphically illustrates the execution times for both ontologies.

From the graph it can be observed, that for both ontologies the execution times for the matching process keeps increasing, when the request size is increased. The matching time for a request that has 5 individual requirements specified (when described with the Computer ontology, in the presence of 50 advertisements to be matched), is approximately 1.8 seconds, which can be acceptable, given the benefits provided by semantic

Request Size: Number of Requirements	***Printer Ontology (ms)***	***Computer Ontology (ms)***
1	538	675
2	875	1181
3	1056	1291
4	1388	1559
5	1481	1841
6	1678	2116
7	1947	2412

TABLE 6.14: Average execution time for increasing request sizes

matching. As with the previous experiment, the same observation can be made regarding the execution times related to the two ontologies; the execution time related to the larger ontology (the Computer ontology) is higher than for the smaller ontology. The plots related to both ontologies are approximately linear.

The execution times for the matching process for increasing request sizes (up to a size of 7), can be observed to be acceptable. For example, when the request size is 4, the matching time is 1.4s approximately (for Printer Ontology); when the request size is 6, the matching time is 1.7s. Thus, these results support the hypothesis *H6* that, the execution time for the matching process is satisfactory for reasonable request sizes.

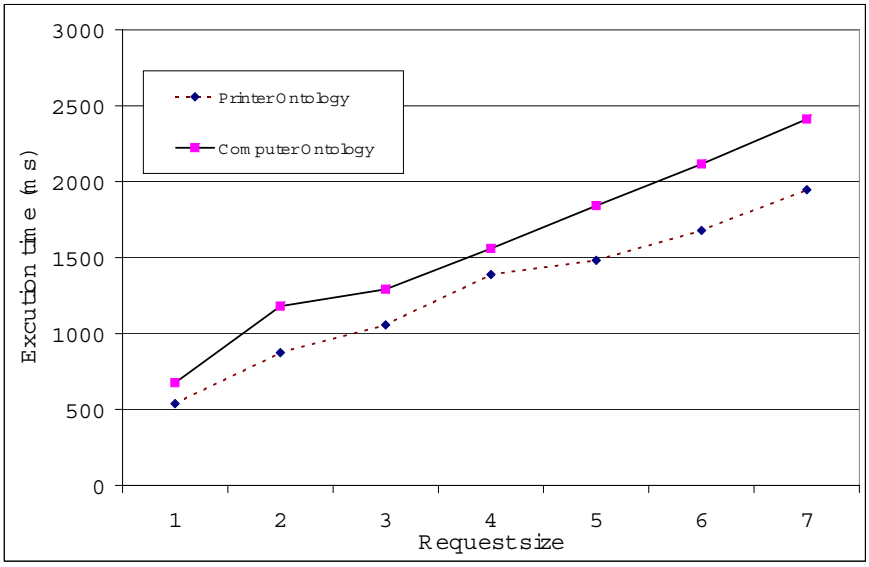

FIGURE 6.8: Request Size Vs Execution Time

6.2.3 Discussion

In this section, we have presented an investigation of the scalability of the Semantic Matcher. Specifically, we have carried out two tests to investigate the scalability of the Semantic Matcher with respect to: (1) the number of advertisements matched and (2) the size of the request in terms of the number of individual requirements.

From the experimental results obtained, we can observe that:

- For reasonable numbers of advertisements, the matching time is acceptable.
- For reasonable request sizes, the matching time is acceptable.
- The size of the ontology used for service description, affects the matching time; larger ontologies tend to increase the time taken for matching.

From the overall results, we can observe that the Semantic Matcher is scalable against increasing numbers of advertisements and increasing request sizes. For large numbers of advertisements, noticeable high matching time has been observed (for example, when the number of advertisements is 1000, the matching time taken is approximately 19s for the Printer Ontology as seen from Table 6.13). However, effective solutions to distribute the matching load [77, 19], can help to improve the response times obtained.

The first test was devised to monitor the variation in matching time in response to increasing numbers of advertisements. At the same time, we intended to monitor the effects of the size of the ontology (that was used to describe the requests and advertisements) on the matching time as well. Therefore, the advertisements were selected to be of the same device type; i.e. when the number of advertisements were increased from 10 to 10000, all of these advertisements were printers (for the Printer Ontology) or computers (for the Computer Ontology). This way, the advertisements can be described using the same ontology (either the Printer ontology or the Computer ontology in this case). However, in a realistic environment it is often the case that the available resources are of various device types (for example there could be computers, printers, cameras, display devices and so on) and these devices can have various security levels and access rights. For example in a large corporate building, there can be around 500 - 1000 advertised resources/devices. However, if we consider a single resource seeker raising a request for a printer which satisfies certain criteria, there may be at most fifty printers which he/ she can access (depending on the security constraints). The majority of resource advertisements will be filtered out by device type and security constraints, before reaching the detailed matching stage. Therefore, the number of advertisements that will go into the detailed matching stage will be far less than the total number of available advertisements.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Recent technological trends in electronics have resulted in a change in lifestyle, whereby pervasive mobile devices such as mobile phones, PDAs and GPS devices have become an integral part of everyday life. This trend, together with the advancement in wireless communications which has resulted in an increasingly wireless world, have raised users' expectations about the accessibility of services in pervasive environments. This has raised challenges for service discovery in a dynamic environment, where the services accessible to a user keeps changing continuously. This scenario has provided the inspiration for this research of exploring the means to support service discovery in pervasive environments.

Service discovery is a widely investigated area in a variety of domains, not limited to pervasive computing. There are several traditional approaches to service discovery that have been investigated in Chapter 2; in general these provide syntactic approaches to service description and discovery, whereby locating appropriate services rely on matching service descriptions based on keywords or interfaces. These will not be able to detect a match in cases where the service descriptions involve different representations of conceptually equivalent content and thus poses a serious limitation. With the advent of the Semantic Web, many research efforts have focused on using ontologies and reasoning mechanisms to facilitate the service discovery process.

Chapter 2 has provided a discussion of the existing research efforts in the area of semantic service matching. The use of reasoning mechanisms in the matching process have enabled these approaches to match service descriptions based on their conceptual content and therefore it offers service providers, the flexibility in how they describe the services. One of the objectives of semantic matching approaches are to provide flexible or approximate matches; i.e. the matching mechanism should be able to identify service advertisements that can approximately match a request, although they do not exactly meet all the

requirements specified. This will allow the resource seekers to choose the best service out of the available ones, in the absence of an exact match.

In general, these semantic matching solutions have provided important research directions in overcoming the limitations present in the traditional approaches for service discovery. However, these solutions have a number of overlooked issues and lacks certain desirable properties that must be present in a pragmatic solution to support effective service discovery. Particularly, these approaches lack an appropriate criterion to approximate the available service advertisements with respect to a given request and to rank them accordingly.

The objective of this research has been to provide a pragmatic approach to service matching with a particular focus on pervasive resources. A semantic service matching approach has been developed and presented in Chapter 4 to this effect. This provides a ranking mechanism, which will order the available resource advertisements according to their suitability to satisfy the resource request under concern. A resource request will typically specify a number of individual requirements or constraints that must be satisfied by a potential advertisement. The proposed matching approach will evaluate an available advertisement, according to how well it satisfies each of these requirements. Depending on this evaluation, a match score will be assigned for each available advertisement and then they will be ranked depending on the score. The individual requirements occurring in a request are categorised into three types (namely Type-1, Type-2 and Type-3), depending on the type of property or concept they involve. The extent to which an advertisement can satisfy a given individual requirement is then judged depending on the type or category of requirement it falls into. The approach followed in judging the suitability within each requirement, is described in detail in Chapter 4. In summary, the proposed Semantic Matcher has the following desirable properties.

- *Approximate Matching:* It provides a novel technique to find approximate matches during the semantic matching process. This is different to what is offered by description logic approaches to semantic matching, where the subsumption relation is used to find approximate or flexible matches. There are certain practical situations where the subsumption relation alone is not sufficient in finding the approximate matches. In such cases the proposed approximate matching technique can find inexact matches appropriately and thus provides a more pragmatic approach. This approximate matching in turn provides a mechanism to score the available advertisements and therefore the non-exact matches can be ranked. The ranking of available advertisements is useful for the resource seeker to gain an understanding of the suitability and the order in which he/ she should consider the advertisements.

- *Ranking of Potential Matches:* Ranking or classifying is the ordering of the possible matching advertisements in the order of their suitability to satisfy the given

request. Ranking helps the service seekers to gain an understanding of the suitability of a service and the order in which they should consider the potential matches. The approximate matching mechanism used in the proposed Semantic Matcher can evaluate how well an advertisement can meet the requirements of a request and assign a score accordingly. This evaluation in turn is used to rank the non-exact matches or advertisements. Therefore the Semantic Matcher will return a ranked list of potential matches.

- *Facilitating Priorities and Mandatory Requirements:* Another novelty in this semantic matching approach is its facility for the resource seeker to describe priorities or weights associated with individual requirements in a request and to specify whether the requirements are mandatory or optional. There are many cases where the individual requirements of a request can have unequal importance, i.e. certain requirements can be more important than the others. This can be either due to the context involved or due to subjective preferences. In such cases, this approach allows the resource seekers to describe priorities or weights associated with these individual requirements in a request. These priorities are taken into account during the semantic matching process, so that the match results produced are more suitable for the context in which they were raised.
- *General Applicability of the Semantic Matching Approach:* In Section 2.4.4.4 we have discussed the general applicability of the semantic matching approaches; i.e. the possibility of applying a particular semantic matching approach, for matching of resources in a domain other than the domain it was originally intended for. The rule-based approaches (such as those presented in [128, 66]) are domain specific approaches for resource matching, since the matching rules must describe the specific conditions under which a resource can be taken as a match for a given request. The matching mechanism of the proposed Semantic Matcher is based on the categorisation of the requirements/properties of a resource into 3 types (explained in detail in Section 4.4.2) and does not depend on specific matching rules. Therefore, the proposed semantic matching mechanism provides a more general approach for resource discovery and is not limited to a particular domain.

For semantic service matching to be effective, it must be supported by an appropriate approach for service description. Since service matching for pervasive environments will have to deal with devices and their services, we have explored appropriate methods of describing device based services. Current approaches and ontologies for describing devices have been investigated; although there are several device description proposals that aim to provide solutions for specific applications, it was observed that there is no general framework that supports the description of device characteristics and capabilities. Hence the Device Ontology presented in Chapter 3, has been developed to provide such a general framework for describing the devices and their services, which in turn will facilitate semantic service discovery in pervasive scenarios. The ontology was developed by

following a systematic design process, going through the activities and tasks applicable to each design stage, as suggested in the Methontology [46, 85] ontology design methodology. The ontology has been evaluated by using the current best practice approaches for ontology evaluation [55], specifically the OntoClean method [63] and ODEval[54].

The proposed semantic matching approach has been implemented in a pervasive scenario, using the Device Ontology for describing the resource advertisements and requests. This implementation was then used to evaluate the Semantic Matcher in terms of its retrieval effectiveness and performance.

Current methods of evaluating retrieval effectiveness has been investigated and it was observed that there is no established "best practice" methods or metrics to guide the evaluation process. Due to this reason, most semantic matching research efforts have not undergone any systematic evaluation process [130] to identify their retrieval effectiveness. However, by exploring the existing literature on effectiveness evaluation and the available metrics in information retrieval, we have adopted a suitable metric (specifically, the generalised recall and precision) to quantify the effectiveness of the match rankings, and have followed a systematic evaluation process (aided by a human participant study) to judge the retrieval effectiveness of the Semantic Matcher. The results from the effectiveness evaluation experiments have shown that the Semantic Matcher results are compatible with human judgement and thus is effective in retrieving the relevant matches. Specifically, through the experimental results we have shown, that each of the desirable properties present in the Semantic Matcher, namely: ranking of matches, approximate matching, consideration of priorities on individual requirements and consideration of mandatory requirements in the matching process, has caused the match results to be more effective.

The proposed semantic matching approach must have a reasonable level of performance (w.r.t. matching time) for its practical use in facilitating the discovery of resources. Therefore, we have evaluated the performance of the solution using the prototype implementation of the Semantic Matcher, through the use of two experiments. Specifically, we investigated the scalability of the solution in terms of the number of advertisements matched and the size of the resource request (in terms of the number of individual requirements). The objective of this evaluation exercise was to investigate the variation in execution time of the matching process, when the number of advertisements matched and the size of the resource request increases. If the Semantic Matcher is scalable, the execution times must be acceptable, for reasonable numbers of advertisements and request sizes. It has been shown through these performance evaluation experiments, that the Semantic Matcher has an acceptable and manageable performance level.

In conclusion, the contribution of this thesis is a framework for the semantic description and matching of resources in a pervasive environment. It provides a pragmatic approach for semantic matching that effectively matches and ranks available resources, depending

on their suitability to satisfy the resource request under concern. It possesses desirable properties that must be essentially present in a practical solution for resource matching and thus provides an important step in the direction towards providing pragmatic semantic matching solutions, that will facilitate enhanced service discovery in real world applications.

7.1.1 Benefits and Feasibility of Semantic Matching Approaches

Resource discovery has become an essential element in a number of domains (such as Web Services, grid environments and pervasive environments), in assisting humans and agents to find the appropriate resource(s). With the changes in technology, these environments are becoming increasingly heterogeneous and dynamic and therefore effective means to facilitate automatic resource discovery is becoming even more crucial.

There are a number of existing solutions for service discovery, such as Jini [6], Salutation [111] and UDDI[129]), which were discussed in Chapter 2. In these approaches, the services are characterised by using predefined service categories, fixed attribute value pairs and interfaces. Such descriptions are inflexible and difficult to extend to new concepts and characteristics, and since these descriptions do not describe devices or services at a conceptual level, no form of inferencing can be carried out on them. Hence the matching techniques in these service discovery approaches are limited to syntactic comparisons based on attributes or interfaces. Thus, the above mentioned discovery approaches cannot provide effective discovery of devices and their services in dynamic environments since they will fail to identify equivalent concepts (service requests and advertisements) which are syntactically different, or approximate matches that deviate from the service request in certain aspects.

On the other hand, the semantic matching approaches through the use of logical reasoning mechanisms, consider the conceptual content of the resource descriptions during the matching stage and therefore are able to identify equivalent concepts even if they have different syntactic representations; this is the main benefit of semantic matching approaches. As a consequence, semantic matching solutions have a number of advantages over syntactic solutions: they provide the resource providers and seekers, the flexibility in how they describe resources, they can offer automatic classification of resources and consistency checking of resource descriptions during the service discovery process. Furthermore, semantic matching supports flexible or approximate matching based on the semantic descriptions of resources, so that in the absence of an exact match that completely satisfies the requirement, the matching approach can present flexible matches depending on their suitability. Motivating scenarios for the semantic description and matching of resources has been presented and discussed in detail in Section 3.2 and Section 4.2.

Section 6.2 provided an empirical evaluation of scalability and performance of the proposed Semantic Matcher. From the overall results it was observed that the Semantic Matcher is scalable against increasing numbers of advertisements and increasing request sizes. However, for large numbers of advertisements, significantly high matching time was observed. To overcome the problem of high response times, solutions such as load balancing [77, 19] can be potentially used, and this possibility needs to be investigated further. It was also observed that, when the resources were described using a larger ontology, the matching times were generally higher (when compared to the matching times related to the smaller ontology); i.e. when the size of the ontology used to describe resources increases, the matching times also increase. The performance effects of using larger and more complex ontologies to facilitate resource description during semantic matching need further investigation.

From the empirical results obtained through the performance evaluation experiments of this thesis and from the investigations carried out in other semantic matching research efforts [82, 7, 87, 88], it can be observed that the use of logical reasoning mechanisms for resource matching has a significant cost on the performance. Therefore, the benefits of the semantic matching solutions have to be weighed against their performance cost on the service discovery process. A detailed analysis of the benefits and an evaluation of the performance has to be carried out for the particular domain or context in which the semantic matching will be potentially utilised, to justify practical deployment.

7.2 Future Work

The presented service matching approach can be improved and enhanced in several ways. Also, the approach can be further evaluated to investigate the effectiveness and feasibility in several aspects. The potential work that can be carried out in this direction is outlined below.

7.2.1 Matching combinations of resources to satisfy a request

In its current state, the Semantic Matcher only considers matching a single advertisement that can satisfy a request; this does not consider the service composition aspect during the matching process. However, in practice, there are situations where a resource request can be satisfied by a combination of two or more resources. For example, a resource seeker may want to print an image stored on a *SD card*, and hence may raise a resource request for a printer that has a *SD card* slot. This request can be also satisfied by the combination of a computer that has a *SD card* slot and a printer. Therefore, it will be useful if such combinations can be considered during the matching process and be returned in the match results as appropriate. This will provide a valuable and practical

enhancement to the current Semantic Matcher and hence this aspect can be further investigated.

7.2.2 Security and privacy considerations

When searching for suitable service advertisements to satisfy a request, the semantic matching approach has so far not taken into account any security and privacy considerations. However, if the Semantic Matcher is to be used in practice to facilitate service discovery, it is vital to consider issues concerning security and privacy. For example, a resource request for a printer with certain properties, must only be matched with the printers that the resource seeker is allowed to access. If the matcher considers and returns any printers which the seeker cannot access, it will only create unnecessary overhead. To address this issue, approaches for encoding such security and privacy information in the service descriptions, must be first investigated. Thereafter, the service matching process can be enhanced to take into account any constraints posed by security and privacy concerns.

7.2.3 Allowing fuzzy or vague terms in request description

In the proposed semantic matching framework, the requests will be described in terms of precisely specified requirements. It does not contain vague terms such as *large* and *high*. In most resource seeking situations in pervasive scenarios, the resource seeker will have a clear idea of what is required and thus is less likely feel the to need to use fuzzy or vague terms in service requests. For example, a typical domain user in a pervasive environment raising a request for a computer, will want to specify approximate figures for the requirements of *memory size* and *disk capacity* rather than specifying the requirements as *"large" memory size* or *"high" disk capacity*. However, there can be situations where such fuzzy terms can be used; for instance, one may want to make use of a printer with a *"short" print queue*. There can be a stronger need for the use of such vague terms in service descriptions, in certain other domains. For example, Stoilos et.al. in [120] and Agarwal et. al. in [1], have provided motivating scenarios in the domains of information retrieval and e-commerce, where users may need to raise queries with vague specifications. Therefore, if the semantic matching approach is to be applied in such domains, allowing such fuzzy terms in the service descriptions and interpreting them for subsequent use in the matching process can become useful. Thus, investigating the possibility of employing fuzzy descriptions, may provide a potential enhancement to the current work.

7.2.4 Further evaluation experiments

Several evaluation experiments have been conducted and presented in Section 6.1. These experiments were carried out to test the retrieval effectiveness of the Semantic Matcher and to demonstrate the improved effectiveness provided by the properties it possesses, which are: match ranking, approximate matching, priority consideration and consideration of mandatory requirements. The scalability of the Semantic Matcher has also been investigated, in terms of the number of advertisements involved in the matching process and the size of the resource request (i.e. the number of requirements in the request). The experimental results have been discussed in Section 6.2.

Other aspects of performance can be also investigated, which will help towards judging the usability of the Semantic Matcher in practical environments. For example, when the Semantic Matcher is deployed on a network to support service discovery, the transmission times between the resource seekers/ providers and the directory service can be measured to test the communication overhead involved. Further, solutions for improving the performance of the Semantic Matcher can be investigated. For example, as observed from the experimental results presented in Section 6.2.1, matching times were relatively high when a large number of advertisements were involved. Methods for improving the performance in this respect can be investigated. A possible approach that can be explored further, is the application of a load balancing solution that can distribute the matching load, which can help to improve the resultant response times.

Appendix A

Human Participant Study: Questionnaires and Responses

In this Appendix we include the questionnaires that were used for the human participant study and the summarised results obtained from the completed questionnaires.

Questionnaire 1: Human Ranking of Resources in a Pervasive Environment

Assume you are looking for a computer on the local network for a certain requirement.

The characteristics you are looking for are as follows:

Request**:** Computer
Has Processor Type Pentium4
Has Operating System Windows XP
Has USB 2.0 port
Has Minimum Physical Memory of 512MB
Has Minimum Disk Space of 120 GB

Assume we have the following information about the computers available:

Advertisement	OS	Processor Type	USB port	Memory (MB)	Disk Space (GB)	Rank
Ad 1	Win XP	P4	USB 2.0	512	120	
Ad 2	Win XP Prof	P4	USB 2.0	512	120	
Ad 3	Windows	P4	USB 2.0	1024	160	
Ad 4	Windows	P4	USB[1]	1024	120	
Ad 5	Windows	P4	USB	768	Unknown	
Ad 6	Win XP Prof	P4	USB 2.0	512	80	
Ad 8	Win XP Prof	P4	USB 2.0	128	80	
Ad 9	Linux	P4	USB 2.0	128	60	
Ad 10	Linux	P2	USB 1.0	128	60	

Can you please indicate the order of preference you would have for each advertised computer in the right most column in the table above. E.g. Rank 1 for the best choice, 2 for the second best etc.

Please also make the following assumptions:

- More than one advertisement could have the same rank. E.g. there could be two advertisements both assigned rank 3.
- Assume that as long as the "Minimum" requirement is met with respect to a certain attribute, then the quantity in that attribute does not affect the decision. (The "more the better" does not apply here)

Approximately, how much time did you spend to complete the questionnaire?

Please write down any assumptions you had to make or any comments you may have.

[1] This could refer to either USB 1.0 or USB 2.0. The specific USB version is not given here.

Questionnaire 2: Human Ranking of Resources in a Pervasive Environment

Assume you are looking for a computer on the local network that suits particular requirements (say to run a particular application).

The requirements you are looking for are as follows:

Request**:** Computer
Has Processor Type Pentium4
Has Minimum Physical Memory of 512MB
Has Minimum Disk Space of 120 GB

Assume we have the following computers available with the properties as shown in the table below:

Advertisement	Memory (MB)	Disk Space (GB)	Processor Type	Rank
Advert 1	512	120	P4	
Advert 2	512	110	P4	
Advert 3	1024	160	P4	
Advert 4	1024	80	P4	
Advert 5	768	60	P4	
Advert 6	256	120	P4	
Advert 7	128	120	P4	
Advert 8	256	60	P4	
Advert 9	256	110	P4	
Advert 10	512	120	AthlonXP	
Advert 11	512	140	P1	
Advert 12	512	140	P3	

Can you please indicate the order of preference you would have for each advertised computer in the right most column in the table above. E.g. Rank 1 for the best choice, 2 for the second best etc.

Please also make the following assumptions:

- More than one advertisement could have the same rank. E.g. there could be two advertisements both assigned rank 3.
- AthlonXP is the AMD processor that gives "more or less" an equivalent performance to Intel P4.
- Assume that as long as the "Minimum" requirement is met with respect to a certain attribute, then the quantity in that attribute does not affect the decision. (The "more the better" does not apply here)

How much time did you have to spend to complete the questionnaire?

Please write down any assumptions you had to make or any comments you may have.

Questionnaire 3: Human Ranking of Resources in a Pervasive Environment

Assume you are looking for a printer to print a certain file.

Depending on the job at hand, the requirements you are looking for are as follows:

Request**:** **Colour** Printer that
Supports Paper Size **A2**
Has Printer Technology **Laser**

Assume that the property - ***Paper Size is the topmost priority***, since you think the printed content will appear best on A2 paper size.

The following printers are available with the properties as shown in the table below:

Advertisement	Maximum Supported Paper Size	Printing Technology	Colour/ BW	Rank
Advert 1	A2	Laser	BW	
Advert 2	A2	Laser	Colour	
Advert 3	A2	Inkjet	Colour	
Advert 4	A2	Inkjet	BW	
Advert 5	A3	Laser	BW	
Advert 6	A3	Laser	Colour	
Advert 7	A3	Inkjet	Colour	
Advert 8	A4	Laser	BW	
Advert 9	A4	Laser	Colour	
Advert 10	A4	Inkjet	Colour	
Advert 11	A3	Inkjet	BW	
Advert 12	A4	Inkjet	BW	

Can you please indicate the order of preference you would have for each advertised printer in the right most column in the table above. E.g. Rank 1 for the best choice, 2 for the second best etc.

Knowledge Available & Assumptions to be made:

- More than one advertisement could have the same rank. E.g. there could be two advertisements both assigned rank 3.
- A2 is the largest paper size, A3 is smaller and A4 is smallest.

How much time did you have to spend to complete the questionnaire?

Please write down any assumptions you had to make or any comments you may have.

Questionnaire 4: Human Ranking of Resources in a Pervasive Environment

Assume you are looking for a computer on the local network that suits particular requirements, to run a certain application. The ***application requires a Windows based operating system***.

The requirements you are looking for are as follows:

Request**:** Computer
Has Processor Type Pentium4
Has Operating System Windows[1]
Has Minimum Physical Memory of 512MB
Has Minimum Disk Space of 120 GB

Assume we have the following computers available with the properties as shown in the table below:

Advertisement	Processor Type	Operating System	Memory (MB)	Disk Space (GB)	Rank
Advert1	P4	WIN XP	512	160	
Advert2	AthlonXP	WIN XP	512	180	
Advert3	P4	LINUX	1024	140	
Advert4	P4	WIN XP	1024	110	
Advert5	P4	SUN OS	768	110	
Advert6	P4	WIN XP	512	120	
Advert7	P4	SUN OS	512	20	
Advert8	P2	WIN XP	256	100	
Advert9	P3	WIN 2000	256	70	
Advert10	P2	WIN XP	128	60	

Can you please indicate the order of preference you would have for each advertised computer in the right most column in the table above. E.g. Rank 1 for the best choice, 2 for the second best etc.

Please also make the following assumptions:

- More than one advertisement could have the same rank. E.g. there could be two advertisements both assigned rank 3.
- AthlonXP is the AMD processor that gives "more or less" an equivalent performance to Intel P4.
- Assume that as long as the "Minimum" requirement is met with respect to a certain attribute, then the quantity in that attribute does not affect the decision. (The "more the better" does not apply here)

How much time did you have to spend to complete the questionnaire?

Please write down any assumptions you had to make or any comments you may have.

[1] "Windows" can refer to any Windows OS (WIN XP, WIN 2000, etc.)

Human Participant Study: Responses for Questionnaire 1

Advertisement	Subjects													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
Advert 1	1	4	1	1	1	1	1	2	2	3	1	1	1	1.54
Advert 2	1	3	1	2	1	1	1	1	1	1	1	1	1	1.23
Advert 3	3	1	1	3	3	5	3	3	3	5	3	3	3	3.00
Advert 4	7	2	4	4	4	6	4	4	7	6	4	4	4	4.62
Advert 5	7	5	8	4	5	9	5	5	8	9	7	5	5	6.31
Advert 6	4	6	5	4	5	3	5	6	4	2	4	5	5	4.46
Advert 7	4	7	6	4	7	4	7	7	5	3	6	7	7	5.69
Advert 8	4	7	6	4	8	7	8	8	6	7	7	8	8	6.77
Advert 9	7	9	9	4	9	8	9	9	9	8	9	9	9	8.31

Human Participant Study: Responses for Questionnaire 2

Advertisement	Subjects													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
Advert 1	2	2	1	1	2	1	1	1	2	2	1	1	1	1.38
Advert 2	5	3	4	4	4	4	4	4	4	5	3	4	4	4.00
Advert 3	1	1	1	3	1	1	1	1	1	1	1	1	1	1.15
Advert 4	7	4	8	7	8	5	6	5	5	4	3	5	5	5.54
Advert 5	7	6	9	7	9	6	10	6	6	6	8	6	6	7.08
Advert 6	9	7	5	7	5	7	7	7	7	7	3	6	6	6.38
Advert 7	9	10	6	7	12	8	11	11	10	10	8	12	12	9.69
Advert 8	9	9	10	7	11	10	12	7	9	9	11	10	10	9.54
Advert 9	9	8	7	7	10	9	9	9	8	8	8	9	9	8.46
Advert 10	3	5	1	2	2	1	3	1	3	3	3	3	3	2.54
Advert 11	6	12	12	5	7	12	7	12	12	12	12	10	10	9.92
Advert 12	3	11	11	5	6	11	4	10	11	11	3	6	6	7.54

Human Participant Study: Responses for Questionnaire 3

Advertisement	Subjects													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
Advert 1	3	3	3	7	3	3	3	3	3	3	2	3	3	3.23
Advert 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1.00
Advert 3	2	2	2	4	1	2	2	2	2	2	2	2	2	2.08
Advert 4	4	4	4	10	3	4	4	4	4	4	4	4	4	4.42
Advert 5	7	7	7	7	8	7	7	7	5	7	6	7	7	6.83
Advert 6	5	5	5	2	5	5	5	5	5	5	4	5	5	4.67
Advert 7	6	6	6	4	5	6	6	6	5	6	6	6	6	5.67
Advert 8	11	11	11	7	11	11	11	11	5	11	10	11	11	10.08
Advert 9	9	9	9	2	8	9	9	9	5	9	8	9	9	7.92
Advert 10	10	10	10	4	8	10	10	10	5	10	10	10	10	8.92
Advert 11	8	8	8	10	5	8	8	8	5	8	8	8	8	7.67
Advert 12	12	12	12	10	11	12	12	12	5	12	11	12	12	11.08

Human Participant Study: Responses for Questionnaire 4

Advertisement	Subjects													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
Advert 1	1	2	1	2	1	1	1	1	1	1	1	1	1	1.15
Advert 2	3	4	1	3	1	1	3	1	2	2	3	3	3	2.31
Advert 3	5	6	8	5	8	8	8	9	8	8	5	8	8	7.23
Advert 4	4	1	4	4	4	4	4	4	3	3	3	3	3	3.38
Advert 5	9	6	8	5	8	9	9	8	9	9	6	8	8	7.84
Advert 6	2	3	1	1	1	1	1	3	4	4	1	1	1	1.84
Advert 7	10	6	8	5	8	10	10	10	10	10	9	8	8	8.61
Advert 8	6	5	6	5	6	6	5	5	6	5	8	6	6	5.77
Advert 9	6	6	5	5	5	5	5	7	5	7	6	5	5	5.54
Advert 10	8	6	7	5	7	7	7	6	7	6	9	7	7	6.84

Bibliography

[1] Sudhir Agarwal and Steffen Lamparter. Smart - a semantic matchmaking portal for electronic markets. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pages 405–408, Washington, DC, USA, 2005. IEEE Computer Society.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Bpel4ws specification: Business process execution language for web services version 1.1, 2003. URL: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[3] Jürgen Angele and Georg Lausen. Ontologies in F-Logic. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 29–50. Springer, 2004.

[4] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara. Daml-s: Web service description for the semantic web. In *International Semantic Web Conference (ISWC 2002)*, pages 348–363, 2002.

[5] Faiz Arni, KayLiang Ong, Shalom Tsur, Haixun Wang, and Carlo Zaniolo. The deductive database system LDL++. *Theory and Practice of Logic Programming Journal (TPLP)*, 3(1):61–94, 2003.

[6] K. Arnold, B. OSullivan, R. W. Scheifler, and A. Wollrath J. Waldo. *The Jini Specification*. Addison-Wesley, 1999.

[7] S. Avancha, A. Joshi, and T. Finin. Enhancing the Bluetooth service discovery protocol. Technical report, University of Maryland Baltimore County, 2001.

[8] F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *Description Logic Handbook - Theory, Implementation and Applications*. Cambridge university press, 2003.

[9] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[10] Ayomi Bandara, Terry R. Payne, David de Roure, and Gary Clemo. An ontological framework for semantic description of devices (poster). In *3rd International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, 2004.

[11] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl, web ontology language - w3c recommendation 10 february 2004, 2004. URL: http://www.w3.org/TR/owl-ref/.

[12] A. Bernaras, I. Laresgoiti, and J. Corera. Building and reusing ontologies for electrical network applications. In *European Conference on Artificial Intelligence (ECAI 96)*, pages 298–302, 1996.

[13] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[14] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes second edition - w3c recommendation 28 october 2004, 2004. URL: http://www.w3.org/TR/xmlschema-2/.

[15] Specification of the Bluetooth system, core version 1.1, 2001. URL: `https://www.bluetooth.org/foundry/specification/document/Bluetooth_Core_1.1_vol_1`.

[16] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture, w3c working group note, 2004. URL: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[17] A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *The 2005 International Workshop on Description Logics (DL2005)*, Edinburgh, Scotland, UK, 2005.

[18] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[19] Tony Bourke. *Server load balancing.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[20] J. Brank, M. Grobelnik, and D. Mladenic. A survey of ontology evaluation techniques. In *Conference On Data Mining & Data Warehousing. (SiKDD 2005)*, 2005.

[21] Christopher Brewster, Harith Alani, Srinandan Dasmahapatra, and Yorick Wilks. Data driven ontology evaluation. In *Language Resources and Evaluation Conference (LREC 2004)*, Lisbon, Portugal, 2004.

[22] Duncan A. Buell and Donald H. Kraft. Performance measurement in a fuzzy retrieval environment. In *Fourth International Conference on Information Storage and Retrieval, Oakland, California, USA, May 31 - June 2, 1981*, pages 56–62, 1981.

[23] US Census Bureau. North american industry classification system (NAICS), 2002. URL: http://www.census.gov/epcd/www/naics.html.

[24] Composite capability/preference profiles (CC/PP): Structure and vocabularies 1.0, w3c recommendation 15 jan 2004, 2004. URL: http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.

[25] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and A. Joshi. Dreggie: Semantic service discovery for m-commerce applications. In *Workshop on Reliable and Secure Applications in Mobile Environment, Symposium on Reliable Distributed Systems, 2001*, 2001.

[26] Abhijit Chaudhury and Jean-Pierre Kuilboer. *E-Business and E-Commerce Infrastructure: Technologies Supporting the E-Business Initiative*. McGraw-Hill Higher Education, 2001.

[27] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2004)*, pages 22–26, Boston, MA, August 2004.

[28] Nancy Chinchor. Muc-4 evaluation metrics. In *MUC4 '92: Proceedings of the 4th conference on Message understanding*, pages 22–29, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[29] R. Chinnici, J-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) 2.0 part 1: Core language, world wide web consortium, 3 august 2005, 2005. URL: http://www.w3.org/TR/2005/WD-wsdl20-adjuncts-20050803.

[30] Shy Cohen. Ontology and taxonomy of services in a service-oriented architecture. *The Achitecture Journal. MSDN. Journal 11*, April 2007.

[31] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello, and Marco Mottola. A formal approach to ontology-based semantic match of skills descriptions. *J. UCS*, 9(12):1437–1454, 2003.

[32] Ion Constantinescu and Boi Faltings. Efficient matchmaking and directory services. In *WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, page 75, Washington, DC, USA, 2003. IEEE Computer Society.

[33] C Runtime Library (RTL). URL: `http://msdn.microsoft.com/en-us/library/abx4dbyh(VS.80).aspx`.

[34] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing, 2001.

[35] C. Dabrowski, K. Mills, and J. Elder. Understanding consistency maintenance in service discovery architectures during communications failure. In *Third Annual Workshop on Software Performance*, Rome, Italy, 2002.

[36] Daml+oil language, 2001. URL: http://www.daml.org/2001/03/daml+oil-index.html.

[37] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. Knig-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg. Web services modelling language (wsml), wsml final draft 5th october 2005, 2005. URL: http://www.wsmo.org/TR/d16/d16.1/v0.21/#cha:wsml-basic-syntax.

[38] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. Knig-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg. Web services modelling ontology (wsmo), w3c member submission, 3rd june 2005, 2005. URL: http://www.w3.org/Submission/WSMO/.

[39] K. Decker, K. Sycara, and M. Williamson. Middle agents for the internet. In *15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[40] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of Twelfth International World WideWeb Conference WWW 2003 ACM press*, pages 321–330, Budapest, Hungary, May 2003. ACM. in refereed paper track.

[41] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *The 30th VLDB Conference*, Toronto, Canada, 2004.

[42] D. Dubois and H. Prade. *Fuzzy sets and systems - Theory and applications*. Academic press, New York, 1980.

[43] The FEEL project: Focussed, efficient and enjoyable local activities. URL: `http://dsv.su.se/FEEL/index2.htm`.

[44] D. Fensel and C. Bussler. The web service modeling framework (wsmf). *Electronic Commerce: Research and Applications*, 1(2):113–137, 2002.

[45] Dieter Fensel, F. van Harmelen, Ying Ding, M. Klein, H. Akkermans, J. Broekstra, A. Kampman, J. van der Meer, Y. Sure, R. Studer, U. Krohn, J. Davies, R. Engels, V. Iosif, A. Kiryakov, T. Lau, , and U. Reimer. On-to-knowledge: Semantic web enabled knowledge management. *Special Issue of IEEE Computer on Web Intelligence (WI)*, 2003.

[46] Mariano Fernandez, A. Gomez-Perez, and N. Juristo. Methontology: From ontological art towards ontological engineering. In *Symposium on Ontological Engineering of AAAI*, Stanford, California, 1997.

[47] M. Fernandez-Lopez. Overview of methodologies for building ontologies. In *IJCAI 99 Workshop on Ontologies and Problem Solving Methods*, 1999.

[48] Mariano Fernandez-Lopez. A survey on methodlogies for developing, maintaining, evaluating and reengineering ontologies, knowledge web project deliverable, 2002.

[49] FIPA device ontology specification, 2002. URL: http://www.fipa.org/specs/fipa00091/.

[50] Frdric Frst and Francky Trichet. Heavyweight ontology engineering. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 38–39. Springer, 2006.

[51] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. Translator-C. Franzke.

[52] R.L. Goldstone and J. Son. *The Cambridge Handbook of Thinking and Reasoning*. Cambridge University Press, April 2005.

[53] A. Gomez-Perez and O. Corcho. Ontology languages for the semantic web. *IEEE Intelligent Systems*, Jan/Feb 2002.

[54] A. Gomez-Perez and M.C. Suarez-Figueroa. Results of taxonomic evaluation of rdf(s) and daml+oil ontologies using rdf(s) and daml+oil validation tools and ontology platforms import services. In *the 2nd International Workshop on Evaluation of Ontology-based Tools held at the 2nd International Semantic Web Conference (ISWC 2003)*, 2003.

[55] Asunción Gómez-Pérez. Ontology evaluation. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 251–274. Springer, 2004.

[56] Grid organizational memory: Grid ontologies. URL: `http://gom.kwfgrid.net/web/space/Grid+Ontologies`.

[57] J Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44. CEUR Workshop Proceedings, 2001.

[58] K. Gottschalk. Web services architecture overview: The next stage of evolution for e-business, 2000. URL: http://www-106. ibm.com/developerworks/ library/w-ovr.

[59] D. A. Gratton. *Bluetooth Profiles: The Definitive Guide*. Prentice Hall: New Jersey, 2002.

[60] B.C. Grau and B. Parsia. From shoq(d) toward e-connections (poster). In *International Workshop on Description Logics, (DL2004)*, 2004.

[61] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[62] M. Gruninger and M.S. Fox. Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada, 1995.

[63] N. Guarino and A. Welty. An overview of ontoclean. *Handbook on Ontologies*, page 151172, 2004.

[64] V. Haarslev and R. Moller. Racer system description. In *International Joint Conference on Automated Reasoning, IJCAR'2001*, 2001.

[65] Uwe Hansmann. *Pervasive Computing: The Mobile World*. Springer, 2003.

[66] A. Harth, Y. He, H. Tangmunarunkit, S. Decker, and C. Kesselman. A semantic matchmaker service on the grid. poster. In *The 13th International World Wide Web Conference*, New York, USA, 2004.

[67] J. Hartmann, Y. Sure, A. Giboin, D. Maynard, M. del Carmen Surez-Figueroa, and R. Cuel. Methods for ontology evaluation. knowledge web project deliverable, 2005.

[68] J. Hendler. J. hendler, on beyond ontology, keynote talk, international semantic web conference, 2003. URL: http://iswc2003.semanticweb.org/hendler_files/ v3_document.htm.

[69] Ian Horrocks. Fact reference manual version 1.6. august 1998., 1998.

[70] Michael C. Jaeger and Stefan Tang. Ranked matching for service descriptions using daml-s. In *CAiSE'04 Workshops, 2004*, Riga, Latvia, June 2004.

[71] Java Platform. URL: `http://java.sun.com/javase/`.

[72] D.M. Jones, T.J.M. Bench-Capon, and P.R.S Visser. Methodologies for ontology development. In *IT&KNOWS Information Technology and Knowledge Systems, 15th IFIP World Computer Congress*, Vienna, Austria and Budapest, Bulgaria, 1998.

[73] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1):1–31, January 2003.

[74] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[75] Michael Kifer. Requirements for an expressive rule language on the semantic web. In *W3C Workshop on Rule Languages for Interoperability*, April 2005.

[76] Yoshinobu Kitamura and Riichiro Mizoguchi. Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Syst. Appl.*, 24(2):153–166, 2003.

[77] Chandra Kopparapu. *Load Balancing Servers, Firewalls, and Caches*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[78] Bartosz Kryza, Jan Pieczykolan, Marta Majewska, Renata Slota, Marian Babik, Adrian Toth, Jacek Kitowski, and Ladislav Hluchy. Grid organizational memory - semantic framework for metadata management in the grid. In M. Bubak, M. Turala, and K. Wiatr, editors, *Cracow Grid Workshop*, volume 2. ACC CYFRONET AGH, October 2006.

[79] Sukhamay Kundu and Jianhua Chen. Fuzzy logic or lukasiewicz logic: A clarification. In *ISMIS '94: Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, pages 56–64, London, UK, 1994. Springer-Verlag.

[80] Ryan Lee. Scalability report on triple store applications. Technical report, Massachusetts Institute of Technology, 2004.

[81] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[82] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331–339. ACM, 2003.

[83] Lei Li and Ian Horrocks. Matchmaking using an instance store: Some preliminary results. In *Proceedings of the 2003 International Workshop on Description Logics (DL'2003), poster paper*, 2003.

[84] Dekang Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[85] Mariano Fernández López, Asunción Gómez-Pérez, Juan Pazos Sierra, and Alejandro Pazos Sierra. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1):37–46, 1999.

[86] A. Lozano-Tello and A. Gomez-Perez. Ontometric: A method to choose the appropriate ontology. *Journal of Database Management. Special Issue on Ontological analysis, Evaluation, and Engineering of Business Systems Analysis Methods*, 15(2), April/June 2004.

[87] S.A. Ludwig and S.M.S. Reyhani. Semantic approach to service discovery in a grid environment. *Journal of Web Semantics*, 4(1):1–13, 2006. http://sorma.fzi.de/protected/Lud05b.pdf.

[88] Simone A. Ludwig and S. M. S. Reyhani. Introduction of semantic matchmaking to grid computing. *Journal of Parallel Distributed Computing*, 65(12):1533–1541, 2005.

[89] R.S. Marin-Perianu, P.H. Hartel, and J. Scholten. A classification of service discovery protocols. Technical report, Centre for Telematics and Information Technology, University of Twente, Enschede, 2005.

[90] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services, w3c technical note member submission, 2nd november 2004, 2004. URL: http://www.w3.org/Submission/2004/07/.

[91] S. A. McIlraith and D. L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, page 9093, January/February 2003.

[92] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. *Lecture Notes in Computer Science*, 1777:86–100, 2000.

[93] Boris Motik, Peter F. Patel-Schneider, and Ian Horrocks. Owl 1.1, web ontology language, structural specification and functional style syntax, 2007. URL: http://www.webont.org/owl/1.1/owl_specification.html.

[94] R. W. Fergerson N. F. Noy and M. A. Musen. The knowledge model of protege-2000: Combining interoperability and flexibility. In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, pages 17–32. Springer-Verlag, 2000.

[95] Marian H. Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm C. Taylor, and Amy Unruh. Active information gathering in infosleuth. *Int. J. Cooperative Inf. Syst.*, 9(1-2):3–28, 2000.

[96] Web ontology language candidate recommendations, 2003. URL: http://www.w3.org/2003/08/owl-pressrelease.

[97] Jeff Z. Pan. Reasoning Support for OWL-E (Extended Abstract). In *Proc. of Doctoral Programme in the 2004 International Joint Conference of Automated Reasoning (IJCAR2004)*, July 2004.

[98] Jeff Z. Pan and Ian Horrocks. OWL-E: Extending OWL with Expressive Datatype Expressions. Technical report, School of Computer Science, the University of Manchester, April 2004.

[99] Jeff Z. Pan and Ian Horrocks. OWL-Eu: Adding Customised Datatypes into OWL. In *Proc. of Second European Semantic Web Conference (ESWC 2005)*, 2005.

[100] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Importing the semantic web in uddi. In *E-Services and the Semantic Web Workshop, 2002*, 2002.

[101] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.

[102] Peter Patel-Schneider and Ian Horrocks. Owl 1.1, web ontology language, 2007. URL: http://www.w3.org/Submission/owl11-overview/.

[103] Terry R. Payne, Massimo Paolucci, Rahul Singh, and Katia Sycara. Communicating agents in open multi agent systems. In *First GSFC/JPL Workshop on Radical Agent Concepts (WRAC)*, pages 365–371, 2002.

[104] Pellet OWL reasoner, 2003. URL: `http://www.mindswap.org/2003/pellet/index.shtml`.

[105] C. Priest. A conceptual architecture for semantic web services. In *International Semantic Web Conference (ISWC)*, 2002.

[106] Uta Priss. Formal concept analysis in information science. http://www.upriss.org.uk/papers/arist.pdf.

[107] RDF primer, 2004. URL: http://www.w3.org/TR/rdf-primer/.

[108] RDF vocabulary description language 1.0: RDF schema, 2004. URL: http://www.w3.org/TR/rdf-schema/.

[109] Marie-Laure Reinberger and Peter Spyns. Discovering knowledge in texts for the learning of dogma-inspired ontologies. In P Buitelaar, S. Handschuh, and

B. Magnini, editors, *ECAI 2004 Workshop on Ontology Learning and Population*, pages 19–24, 2004.

[110] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[111] Golden G. Richard and M. Spencer. *Service Discovery Protocols and Programming*. McGraw-Hill Professional, 2001.

[112] Ray Richardson, Alan F. Smeaton, and J. Murphy. Using WordNet as a knowledge base for measuring semantic similarity between words. Technical Report CA-1294, Dublin City University, Dublin, Ireland, 1994.

[113] Konstantinos Sagonas, Terrance Swift, and David S. Warren. XSB as an efficient deductive database engine. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 442–453, New York, NY, USA, 1994. ACM.

[114] Elie Sanchez, editor. *Fuzzy Logic and the Semantic Web*. Capturing Intelligence. Elsevier, Amsterdam, 2006.

[115] Semantic annotations for wsdl and xml schema - w3c recommendation 28 august 2007, 2007. URL: http://www.w3.org/TR/sawsdl/.

[116] Angela Schwering. Hybrid model for semantic similarity measurement. *Lecture notes in computer science*, pages 1449–1465, 2005.

[117] Dimitrios Skoutas, Alkis Simitsis, and Timos K. Sellis. A ranking mechanism for semanticweb service discovery. In *IEEE SCW*, pages 41–48, 2007.

[118] An introduction to slp. draft white paper, 1999. URL: http://www.openslp.org/doc/html/IntroductionToSLP/index.html.

[119] IEEE Computer Society. Ieee standard glossary of software engineering terminology, 1990.

[120] G. Stoilos, G. Stamou, V. Tzouvaras, and J.Z. Pan. Uncertainty and ruleml rulebases: A preliminary report. In *International Conference on Rules and Rule Markup Languages for the Semantic Web*. International Conference on Rules and Rule Markup Languages for the Semantic Web, Galway, 2005, (Short Paper), 2005.

[121] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrock. A Fuzzy Description Logic for Multimedia Knowledge Representation. In *Proc. of the International Workshop on Multimedia and the Semantic Web*, 2005.

[122] Umberto Straccia. A fuzzy description logic. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, Madison, US, 1998.

[123] Umberto Straccia. Reasoning within fuzzy description logics. *J. Artif. Intell. Res.*, 14:137–166, 2001.

[124] Umberto Straccia. A fuzzy description logic for the semantic web. *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, 2005.

[125] B. Swartout, P. Ramesh, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Symposium on Ontological Engineering of AAAI*, pages 138–148, March 1997.

[126] Katia Sycara, Jianguo Lu, Matthias Klusch, and Seth Widoff. Matchmaking among heterogeneous agents on the internet. In *Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace*, March 1999.

[127] Katia P. Sycara, Matthias Klusch, Seth Widoff, and Jianguo Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1):47–53, 1999.

[128] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid - the grid meets the semantic web. In *First Workshop on Semantics in Peer-to-Peer and Grid Computing. In conjunction with the Twelfth International World Wide Web Conference 2003*, Budapest, Hungary, 2003.

[129] OASIS UDDI Specification TC. The UDDI technical white paper. Technical report, OASIS, 2004. Available from http://uddi.org/pubs/uddi-tech-wp.pdf.

[130] Vassileios Tsetsos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades. On the evaluation of semantic web service matchmaking systems. In *ECOWS '06: Proceedings of the European Conference on Web Services*, pages 255–264, Washington, DC, USA, 2006. IEEE Computer Society.

[131] Amos Tversky. Features of similarity. *Psychological Review*, 84:327– 352, 1977.

[132] Uaprof specification, 1999. URL: www.wapforum.org/what/ technical/SPEC-UAProf-19991110.pdf.

[133] United Nations Standard Products and Services Code, 2008. URL: http://www.unspsc.org/.

[134] UPnP, 2000. URL: `http://www.upnp.org/download/UPnPDA10_20000613.htm`.

[135] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[136] G. van Heijst, A.Th. Schreiber, and B.J. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human-Computer Studies*, 46(2-3):184–292, Feb/March 1997.

[137] C. J. van Rijsbergen. Getting into information retrieval. *Lectures on information retrieval*, pages 1–20, 2001.

[138] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides G.M. Petrakis, and Evangelos E. Milios. Semantic similarity methods in wordnet and their application to information retrieval on the web. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16, New York, NY, USA, 2005. ACM Press.

[139] D. Veit, J. P. Muller, and Ch. Weinhardt. An empirical evaluation of multidimensional matchmaking. In *Workshop on Agent Mediated Electronic Commerce IV (AMEC-IV), In conjunction with the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2002.

[140] W3C. Owl, web ontology language - w3c candidate recommendation 18 august 2003, 2003. URL: http://www.w3.org/TR/2003/CR-owl-features-20030818/.

[141] Yiqiao Wang and Eleni Stroulia. Semantic structure matching for assessing web-service similarity. In *International Conference on Service Oriented Computing*, pages 194–207, 2003.

[142] XSB Logic Programming and Deductive Database system. URL: `http://xsb.sourceforge.net/`.

[143] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.