Context-aware service discovery for dynamic grid environments

A dissertation submitted by

Mhlupheki George Sibiya

(20000658)

in fulfillment of the requirements for the degree of

Master of Science in Computer Science

to

The Department of Computer Science,

Faculty of Science and Agriculture,

University of Zululand

Supervisor: Professor S.S. Xulu

Co-supervisor: Professor M.O. Adigun

2010

Context-aware service discovery for dynamic grid environments

A dissertation submitted by

Mhlupheki George Sibiya

(20000658)

in fulfillment of the requirements for the degree of

Master of Science in Computer Science

to

The Department of Computer Science,

Faculty of Science and Agriculture,

University of Zululand

Supervisor: Professor S.S. Xulu

Co-supervisor: Professor M.O. Adigun

October 2010

DECLARATION

This dissertation represents the author's own research work and the work has not been submitted in any form to another tertiary institution for another degree or diploma. All the material used as source has been acknowledged in the text.

……

Signature

DEDICATION

I dedicate this work to my family for believing in me and for understanding when I had to spend most of my time away through the course of this work.

ACKNOWLEDGEMENTS

I would like to thank the Lord for maintaining my spiritual and physical strengths through the course of this work. Many thanks go to my supervisors Professor S.S. Xulu and Professor M.O. Adigun for guidance, support and for making this research work realizable. They also stood by from the beginning till the end. I would like to thank Mr. E. Jembere for the support in all aspects. Your down-to-earth spirit and passion for helping others to an extent that you can even sacrifice with your own time are very remarkable. I would like to thank Dr. J.S. Iyilade, Dr. O.O. Olugbara, Dr. S. Ekabua and and all fellow researchers at the centre for their contributions in laying a platform from which this work could be launched. Many thanks to my colleagues and friends, Mr. P.B. Dumakude, Mr. N.S. Ntetha and the rest of Dlangezwa High School staff for motivation and moral support. I would also like to thank Mr. S.K. Mthiyane for always believing in me and that has made me to always want to work hard so that I would not disappoint you. My sincere thanks are also extended to Ms. N.Z. Mkhize for her support throughout the course of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSARY

ACRONYM	DEFINATION	SHORT DESCRIPTION
CASDA	Context Aware Service Discovery Architecture	A service discovery approach proposed in our research work.
DAML-S	DARPA Agent Mark-up Language for Services	A semantic mark-up language for describing web services and their related ontologies.
DL	Description Logics	Formalisms for expressing structured knowledge and for accessing and reasoning with it in a principled way.
GUISET	Grid-based Utility Infrastructure for SMMEs Enabling Technologies	An ongoing project at the University of Zululand, Computer Science Department.
OWL	Web Ontology Language	It is a knowledge representation language used for ontology development.
OWL-S	Web Ontology Language for Web Services	A semantic web service mark-up language for describing web services and their ontologies.
OWLS-MX		An OWL-S based service matchmaker.
QoS	Quality of Service	It is the ability of a service to provide a certain level of performance.
SMMEs	Small, Medium and Micro Enterprises	Business entity categories.
SOA	Service Oriented Architecture	It is a set of principles used in the

		development and integration of interoperable service systems.
SOC	Service Oriented Computing	It a computing paradigm which embraces the principles of SOA.
UDDI	Universal Description, Discovery and Integration	It is an XML-based registry used by businesses to register themselves and their services on the internet.
WSDL	Web Service Description Language	An XML based web service description language which also provides a way to access them.
WSDL-S	Web Service Semantics	An extension of WSDL which support semantics.
XML	eXtensible Mark-up Language	It is a set of rules which specify a way to encode data in a machine readable format in a document.

ABSTRACT

The advent of the Service Oriented Computing (SOC) paradigm, web services, and grid services can contribute towards addressing the hardware and software acquisition problems for the resource constrained enterprises such as SMMEs by providing computing resources on-demand. In a Service Oriented Architecture (SOA) all participants including resource constrained mobile devices are not only expected to participate as service consumers, but also as service providers. However, service discovery in such environments is hindered by the fact that existing standards do not fully support dynamic service discovery at run time. There is, therefore, a need for a scalable and robust discovery mechanism that will also minimize false positive and false negatives in the discovered services. This is still an issue for the current widely adopted service discovery standards. Factors contributing negatively in service discovery range from the matchmaking approaches to the discovery mechanisms. There are many research efforts that attempted to address these issues in service discovery but they still leave a lot to be desired.

In this research work we address service discovery shortcomings by proposing an approach that utilizes both the pure syntactic and the semantic matchmaking techniques. This is realizable through our proposed Context Aware Service Discovery Architecture (CASDA) framework. CASDA proposes a description of services that allows the annotation of both the dynamic and static contexts of a service. It also provides a framework that utilizes requester context, service context and semantics without trading-off scalability. CASDA builds upon OWLS-MX's capability to utilize both the syntactic and logic techniques to service matching. It also takes into account the requester and service context in fulfilling the needs of the requester during the service discovery process. The results obtained showed an improvement

on scalability and recall when compared with OWLS-MX. However, there is a slight cost on precision.

CHAPTER 1

INTRODUCTION

1.1 Introductory Background

The University of Zululand Computer Science Department is currently working on the Grid-Based Utility Infrastructure for SMMEs Enabling Technologies (GUISET) (Adigun et al, 2006). GUISET takes advantage of the affordability and popularity of mobile devices technologies as well as advancements in SOA to address the hardware and software acquisition problems for resource constrained enterprises such as the Small, Medium and Micro Enterprises (SMMEs). In addressing these problems, GUISET implements SOA and hence the challenges towards realizing SOA are also GUISET issues.

Service-oriented computing (SOC) is an emerging computing paradigm which utilizes services as building blocks to support the development of fast, affordable and easy composition of applications in distributed systems (Vanrompay et. al., 2009). SOC provides an infrastructure utility on which SOA can be implemented. SOC has attracted significant investment in standardization and is being rapidly adopted by organizations of all types and sizes (Srivasan and Treadwell, 2005; Papazoglou et. al., 2007). What has contributed to the success of SOC is the usage of autonomous, platform-independent software entities as services. These services can be described, published, categorized, discovered, and dynamically assembled for developing distributed, interoperable and evolvable applications. Many companies have invested a lot of their resources and efforts in promoting the delivery of services in the form of web services and grid services.

Barrero, López and Población (2004) defined web services as software systems designed to support interoperable machine-to-machine interactions. Grid services, on the other hand, are web services that implement grid patterns in a grid environment (Sotomayor and Childers, 2006; Bocchi et. al. 2005). Srivasan and Treadwell (2005) remarked that there is considerable overlap between the goals of grid computing and the benefits of SOA based on web services. Therefore, both the web services and the grid services achieve a similar goal in SOA. In our discussion, the term *service* will thus be referring to both a web service and a grid service.

Despite the major success of SOA with regard to adoption, it is still facing some challenges. The SOA can be distinguished from other architectures by its loose coupling (Ort, 2005; Srivasan and Treadwell, 2005 and Papazoglou and van den Heuvel, 2005). In SOA, services and service clients are designed and implemented independently without any prior knowledge about the existence of each other. The potential client of a service, which can be an application or another service, needs to lookup in a services registry before binding and invoking a service. In most cases this lookup must be performed at runtime. This process, through which clients lookup services, is called service discovery. Service discovery is a prerequisite in SOA and yet it is still one of the challenges facing SOA (Suraci et. al., 2005; Papazoglou et. al., 2007; Klan, 2006; Zhang et. al., 2009). The traditional service discovery technologies match services based on syntax. Unfortunately, this results in poor results involving many false negatives and false positives. Many solution approaches (e.g. Klusch et. al., 2006) aimed at improving the quality of results by employing semantics tend to achieve this at the expense of scalability. In our work we advocate that there is a role that can be played by context in conjunction with semantics in enhancing the service discovery process without trading off scalability.

Advancements in mobile computing together with the developments in service-oriented computing have enabled mobile devices to participate in service-oriented environments both as service providers and service consumers (Doulkeridis et. al., 2006). Even though mobile devices are improving they still have shortcomings compared to their desktop-based counterparts. For example, mobile devices are battery powered and this has an impact on their design since all the hardware and software components must be designed to reduce energy consumption and to prolong the life time of batteries. Mobile connectivity can vary dynamically and thus adversely affect performance and reliability. Disconnections both intentional and unintentional are common in mobile devices. The link band can also vary by orders of magnitude over time and space. In cases where a mobile device is a service provider the service provided by that device may then enter and exit the system dynamically. In cases where the mobile device is a client, disconnection or battery failure may occur while the consumer is busy discovering or consuming a service. A service therefore may need to be rediscovered on recovery from a failure. This therefore incurs costs on the client side.

In a dynamic and service-rich environment like GUISET, the potential number of services with closely related functionalities can be extremely large. It, would, therefore be unrealistic to expect service consumers who use mobile devices to discover and select optimal services that meet their needs given the constraints on their devices. The issue of service availability at the time of the request is also one of the concerns as resources can be wasted on matching unavailable services. Service parameters can either be dynamic or static in nature (Sinclair et. al., 2005) and, therefore, may be used to expose important details of a service, such as its state. This is where context plays a role in enhancing discovery as services are filtered based on their contexts.

Papazoglou et. al., (2007) argue that enhanced service discovery requires explicating the semantics of both the service providers and the service requesters. Some research efforts (Jeckle et. al., 2003; ShaikhAli et. al., 2003; Miles et. al., 2003) address the issue of service discovery by augmenting the current standards like the Universal Description, Discovery, and Integration (UDDI). UDDI is a traditional widely adopted XML service directory which provides a mechanism through which services can be published and discovered (Clement, L., Hately, A., Von Riegen, C. and Rogers, T., 2008). A platform independent Web Service Description Language (WSDL) is a framework that explicitly defines the interface of a Web service in UDDI. These researchers augment UDDI by adding semantic annotations and the inclusion of descriptions of QoS characteristics which are not currently supported service definitions in WSDL. Other research efforts (e.g. Ludwig et. al., 2005; Doulkeridis et. al., 2006; Chen et. al., 2008; Toninelli et. al., 2008) proposed new techniques that are implemented independently of any traditional discovery standard. Unfortunately, all these proposed enhancements in service discovery mechanisms are unsatisfactory.

Enhancing service discovery is a broad concept as it could mean enhancing performance, scalability, security, etc. Our work aims to enhance the service discovery process in terms of the quality of the results, i.e. minimizing false positives and false negatives by improving recall and precision. We propose an approach which utilizes context and semantics in enhancing the service discovery process. We advocate that, with the services well described and the contexts of the requester taken into account, optimal services can be discovered with minimal user involvement or no involvement at all. There is a need to employ semantics in describing services to minimize false positives and false negatives in the discovered results. There is also a need for a robust and scalable matchmaking approach in any service discovery architecture.

1.2 Statement of the Problem

Solutions do exist that attempt to address outstanding issues with the traditional service discovery protocols that use pure syntactic matching such as the UDDI. Most of these solutions employ semantics but it is at the expense of scalability. We share the same sentiment expressed in (Schilit et. al., 1994; Lee and Helal, 2003; Doulkeridis et. al., 2006; Suraci et. al., 2008 and Yu and Reiff-Marganiec, 2009) that context can play a role in enhancing service discovery. In order to enhance service discovery in SOA such as GUISET, this study contributes in addressing the following issues:

1. Representation of service descriptions that would be expressive enough to include both static and dynamic contexts of a service, and
2. Provisioning of a corresponding framework that enables the utilisation of requester context, service contexts and semantics in the discovery process.

Our research study consisted of the following research activities:

1. Investigated and proposed a generic way of representing service context that is flexible enough to enable it to support a variety of contexts types and their values.
2. Enhanced the existing service registries like UDDI to provide support for service metadata and context.
3. Developed an algorithm that matches the requester context and service context during service discovery.

1.3 Purpose of the Project

This project aimed to enhance the process of service discovery by providing a mechanism that utilizes semantics and context data during the service discovery process and hence provide a mechanism for contextualized service selection.

1.4 Objectives

In order for the goal to be realized the following objectives needed to be achieved;

1. To provide a service description framework that would enable the matchmaking of service and requester context during the discovery process.
2. To develop a model that provides a solution to the issues raised in Section 1.2 (Paragraph 1).
3. To integrate a service discovery algorithm such that the context data is utilized during a service discovery process.
4. To develop a prototype of the model which will then be evaluated based on the relevance of the services discovered.

1.5 Motivation for the Study

Service-Oriented Computing (SOC) is a computing paradigm that utilizes services as basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments (Papazoglou et. al., 2007). The visionary promise of SOC is a world of cooperating services where application components are assembled with little effort into a network of services. These services are loosely coupled to create flexible dynamic business processes and agile applications that may span organizations and computing platforms. On its full implementation then, SOA can provide some solutions to SMMEs ICT problems by enabling easy access and utilization of computational resources

as services through mobile devices. One of the challenges in SOC is in service discovery where the main challenge is the usage of automated means for accurate discovery of services in a manner that demands minimal user involvement (Papazoglou et. al., 2007). In this regard our work provides a framework that utilizes requester context and service context in order to provide the requester with services that are most relevant to them with little effort from them. Current widely adopted registries like UDDI do not provide support for context awareness (Doulkeridis et. al., 2006). Although some studies on service discovery have dealt with the issue of context awareness (Chen G. et. al., (2008); Doulkeridis, C. et. al., 2006; Toninelli, A. et. al., 2008), more work still needs to be done to address this issue. As an illustration, consider the following internet based tutorial session that requires a context-aware service discovery mechanism:

> Menzi is a student to a distant learning college. He is attending an internet based tutorial session hosted by his remote instructor. Since real time voice or video materials need to be transmitted, the availability of the internet-based tutorial services must be 99% or higher. About an hour after the tutorial session starts, Menzi needs to drive to the college's local offices for an examination. On his way to the offices he wants to continue this ongoing internet-based tutorial session in a voice-based format.

This scenario requires context-aware services, which can automatically detect changes in Menzi's context states. A location tracking service needs to detect Menzi's position changes while he is moving. An adaptation service needs to adjust the internet based tutorial session formats to match Menzi's contextual changes. The changes among others might be location, activity, network, etc. Format changes include changes from fixed place session to driving

based session, from wired to wireless based session, from notebook to PDA-based session, and from video based to voice based session.

In the event that a currently running service cannot be adapted to new contextual parameters, it should be possible to discover and substitute it with another service without either interrupting the session or Menzi being explicitly involved in the process. This should be achieved by simply considering his context and the context of the required services. In such a scenario if service, access device, and requester contexts are not considered, the failure of a running service due to network failure might lead to it being substituted by non-functional or incompatible services which would eventually lead to the tutorial session being interrupted. There is, therefore, a need for an efficient, robust, scalable and context aware service discovery mechanism for such scenarios which is what our work addresses.

1.6 Methodology

This research takes a pragmatic stance (Easterbrook et. al., 2007) which adopts a mixed methods research approach. According to Easterbrook et. al. (2007), "Pragmatism adopts an engineering approach in research in that it values practical knowledge over abstract knowledge and hence uses all appropriate methods to obtain the knowledge." The research approach in this work is therefore, descriptive, formulative and evaluative, see Figure 1-1.

The descriptive part of this research involves literature survey, which comprise the work done in the SOA to address the issue of service discovery and context awareness. Theories gathered in the literature survey stage are used to analyze and evaluate the various solution approaches proposed by other researchers that address the issue of service discovery and context awareness. The information obtained in the study design stage is used to formulate

and develop a framework that addresses the service discovery issue through prototype implementation and experimentation.

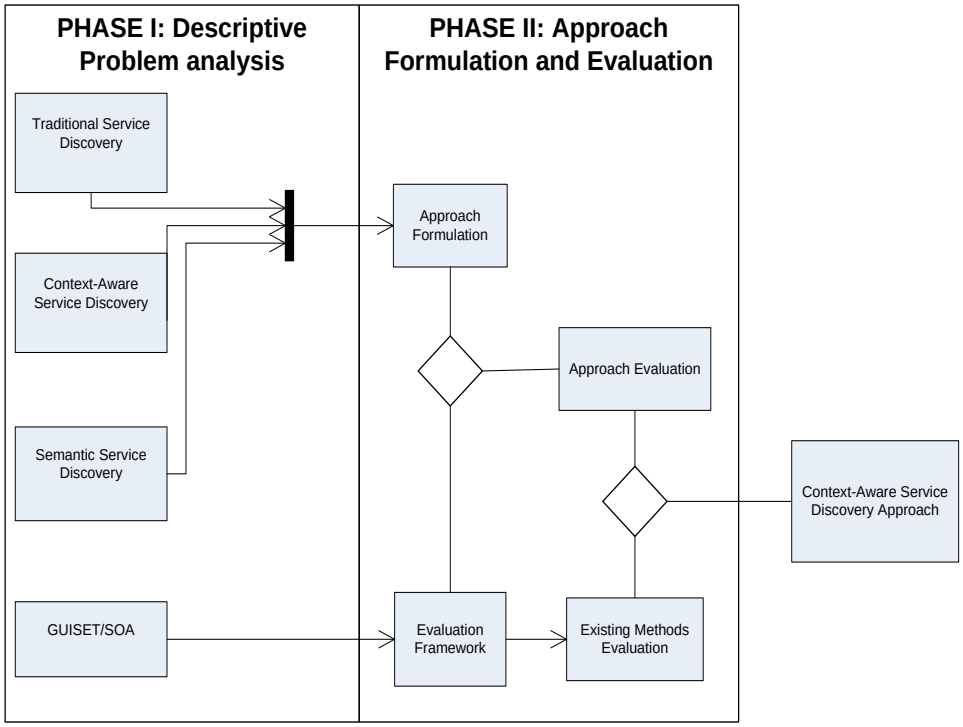


Figure 1-1 Research Approach

The model also consists of an integrated matchmaking algorithm. The performance measurement of the implemented prototype is based on the quality of the results retrieved. The metrics used to evaluate the model are: *recall, precision and response time*.

1.7 Chapter Summery and Organization of the Dissertation

In this chapter we highlighted service discovery issues associated with SOA. In SOA, services can be described, published, discovered, and dynamically assembled for developing massively distributed, interoperable and evolvable systems. Web services are currently the most promising technology in the implementation of Service-Oriented Computing (Papazoglou et. al., 2007). Web services provide the basis for the development and execution

of business processes that are distributed over the network and are available via standard interfaces and protocols. After services are published by providers, for them to be utilized by consumers, they need to be discovered and such a process is facilitated by registries such as Universal, Description, Discovery and Integration (UDDI).

In this chapter we indicated that this project aims to enhance the service discovery process by providing a mechanism that utilizes semantics and context during the discovery process. In order for this goal to be realized there are three things that need to be taken into account. First, it is the way services are described and whether those service descriptions cater for both semantics and context-awareness. Second, the approaches in which those service descriptions are matched against a request submitted by service consumers. The third one is the infrastructure through which these service descriptions and the matchmaking approaches would be supported. We refer to this infrastructure as *service discovery architecture*. Chapter 2, therefore, presents a background on the important concepts which include service description, context-awareness, matchmaking approaches and service discovery architectures. The chapter also presents a framework used to evaluate the matchmaking approaches and a framework to evaluate service discovery architectures in enhancing the service discovery process.

In this study we adopt a pragmatic stance hence, different research methods are adopted. One of the research methods adopted is a survey research method which is used in the problem description and analysis stage. The literature considered in the problem description and analysis stage is reviewed in Chapter 3. Chapter 3 examines the approaches proposed by other researchers in addressing the issue of service discovery. The literature is discussed according to the following aspects: *service description, matchmaking algorithms* and *service discovery architecture*. Chapter 3 also outlines lessons learnt from the evaluation of these approaches using an evaluation framework presented in Chapter 2.

The second research method adopted in this study is the design method. Based on the information gathered from problem description and analysis a solution approach is proposed in a form of a model. Chapter 4 presents the design of our model development. Chapter 4 first presents a service description approach adopted in our model. A context model for both services and service consumers that can be used with the adopted service description approach is presented. In Chapter 4 we then present the service matchmaking approach used in our model. Finally, the architecture which supports the service description framework, context model and the proposed matchmaking approach is presented.

The third research method adopted in this study is the experimentation. To evaluate the effectiveness of our solution approach experiments had to be conducted. To achieve this, our proposed model is instantiated as a prototype. The design, implementation and evaluation of the prototype are presented in Chapter 5.

Chapter 6 concludes our work and also presents gaps for future work.

CHAPTER 2

BACKGROUND

2.1 Introduction

Service descriptions, Service matchmaking algorithms and service discovery architecture contribute directly and indirectly into the efficiency of a service discovery process. Depending on whether the envisaged service discovery approach aims to support semantics and context or not, the framework used to describe the services contributes directly in enhancing the discovery process. If a service discovery approach aims to support context as an example, it is a requirement for a service description framework used to support it. One question that may be asked is whether the service description framework used can be adaptable to address new requirements or not. Another factor that contributes directly in a service discovery process is the matchmaking algorithm used to match the services. The efficiency of the matchmaking algorithm among other performance measures is important. Architecture on the other hand can contribute directly and indirectly into the efficiency of the service discovery. We refer to this infrastructure as service discovery architecture.

As indicated in the previous chapter, the aim of this research is to enhance the process of service discovery by utilizing semantics and context data. This chapter presents a background on important concepts in this study. Service Description and Context Awareness are introduced in Section 2.2 and 2.3 respectively. This is followed by discussions of Matchmaking algorithms taxonomy in Section 2.4 and Service Discovery Architectures Taxonomy in Section 2.5. Before concluding the chapter, in Section 2.6 we present an evaluation framework for services discovery architectures, and an evaluation framework for matchmaking algorithms.

2.2 Service Description

For services to be discovered they have to be well described and these descriptions need to be published in some registry or repository where they could be accessed. A description of services involves explicating their metadata. In a grid environment, metadata include user information such as name, address, profiles, and preferences. It also includes specifications of grid resource such as CPU, memory, and number of nodes (if parallel) for each computer. For software services, this includes location, compiler options, and perhaps, needed input data specifications (Fox, 2003). In web services, context attributes or metadata include functional attributes such as inputs, outputs and non-functional attributes such as QoS, security, cost, availability, etc. All these are attributes that are very important in determining the most suitable service in satisfying the goal of the service consumer. It is, therefore, important for this data to be supported in any service description framework that seeks to enhance a service discovery process. However, supporting such data is still a challenge in the current standard service discovery protocols such as WSDL used for UDDI.

In the current web services based approaches each service exhibits inputs and outputs channels or ports which are specified in practice by Uniform Resource Locators (URLs). Combining services as required in grid environments is a complex task since you have to match precise input and output service ports corresponding to a message-based remote procedure call implementation in the web or grid service model. A web service's procedure syntax is specified in its WSDL instance. Metadata adds method semantics, or meaning, in each service. Such enriched function calls (grid service ports) are characteristics of the semantic grid. We can think of the semantic or metadata information as a richer form of the documentation traditionally attached to programs. This semantic information is built in terms of keywords and relationships which domain experts in each field would need to develop.

These domain specific semantic frameworks are called *ontologies*. Semantics in a grid environment enhance the user interaction with the services in the environment. There are a number of frameworks which are used to model services ontologies such as Web Ontology Language (OWL) (Motik et. al., 2007), Web Ontology Language for Services (OWL-S) (Martin et. al., 2004), DAML-S (Ankolekar et. al., 2003) etc. These frameworks are discussed in Section 3.2.2.

2.3 Context-Awareness

Context is a broad concept which comes in many definitions. It is not easy to identify a widely accepted definition since researchers define it depending on the domains and environments in which they wish to apply it. In our work we adopt the definition by Dey et. al., (2001). Dey et. al., (2001) defined context as any information that characterizes a situation related to the interaction between users, applications, and the surrounding environment. Almost any information that occurs in the surroundings of a system usage can be described as context. For example, context may be based on the current time and date, the usage history, the user profile, the temperature at the user location, the position of the user, etc. Any system that takes such information into consideration while performing its processes, or functions, is said to be a context-aware system.

Context can also be considered as any information about circumstances, objects, or conditions by which a user is surrounded that is considered relevant to the interactions between the user and the environment (Dey et. al., 2001). Schilit et. al. (1994), divides context into three categories, namely: (1) *computing context* such as network connectivity, communication costs, communication bandwidth, and nearby resources such as printers, displays, and workstations; (2) *user context* such as user's profile, location, people nearby,

and even sometimes current social situation; and (3) *physical context* such as lighting, noise levels, traffic conditions, and temperature.

As can be seen in (Strang and Linnhof-Popein, 2004), there are various context modeling approaches. Examples of these are the key-value paired models, mark-up scheme models, graphical models, object oriented models, logic based models and ontology based models. Ontology based models are the most promising models that fully support heterogeneity and interoperability. da Rocha and Endler (2006) used an ontology based model in their quest to address the issues of generality, flexibility, performance and inter-operability in context models. Mahmud et. al. (2008) argues that context is a large data set that can be classified into numerous context types. Mahmud, et. al., who used an object oriented model, organizes the context types into categories which are spatial, temporal, environmental, activity, community and identification. A graph model was used by Doulkeridis et. al. (2006). In their approach, there were no context nodes but only atomic nodes existed. Context in their model is represented by edges while the atomic nodes represent services.

The introduction of context in the process of services discovery enables the discovery of personalized and customized services with minimal user involvement. This is required in dynamic Grid environments since the capability of the devices used by requesters includes constrained mobile devices. The requester cannot afford to be overwhelmed with a large number of services and be expected to filter the relevant ones manually. Context-aware service discovery would be of much assistance in such a situation as it would retrieve personalized and refined results to the requester.

In our research work, context plays a significant role and, we are, therefore, mainly interested in the service description frameworks that support context-awareness. OWL-S is an

extensible language which can be extended so that it is suitable for a given environment. To provide support for context awareness, Kirsch-Pinheiro et. al. (2008) took advantage of OWL-S extensibility and introduced a new element in the service profile of an OWL-S service description. This context element point to an XML document outside the semantic service description and the document contains the information about the context of the service. The reasoning behind storing the services in a separate file than the OWL-S profile is that some context atoms are dynamic and cannot be statically stored in the service profile. By extending OWL-S in this fashion, after matching the service semantically based on the inputs and outputs against the request, the matchmaker can then be instructed to further match the pair contextually.

2.4 Matchmaking algorithms taxonomy

Matchmaking algorithms can be classified into syntactic matchmaking and semantic matchmaking algorithms. Syntactic matching involves computing the syntactic similarity of service requests and service advertisements based on their service names and service descriptions (Le et. al, 2007). Le et. al., (2007) use "string-matching" to calculate how closely service names and service descriptions of two semantic services resemble each other. Traditionally, web service interfaces are described using syntactic WSDL documents. This restricts services matchmakers to key-word based matchmaking. Most solution approaches directed at addressing current issues with the traditional service discovery protocols adopt the semantic matchmaking route.

Section 2.4.1 through Section 2.4.6, present the semantic matchmaking approaches which are commonly used in research efforts that attempt to address the shortcomings of the traditional discovery protocols. Figure 2-1 shows a classification of these matchmaking algorithms. Bellur et. al., (2008) classify them into the following six categories: *Greedy approaches,*

Bipartite Graph Based Matching approaches, QoS enabled matching approaches, Across Heterogeneous Ontologies Based approaches, Description Logic (DL) Based approaches, and Ranked Instance Retrieval Based approaches. Our discussion below is based on semantic matchmaking algorithms as a means to address the issues with the traditional syntax based matchmakers and we follow Bellur et. al.'s classification.

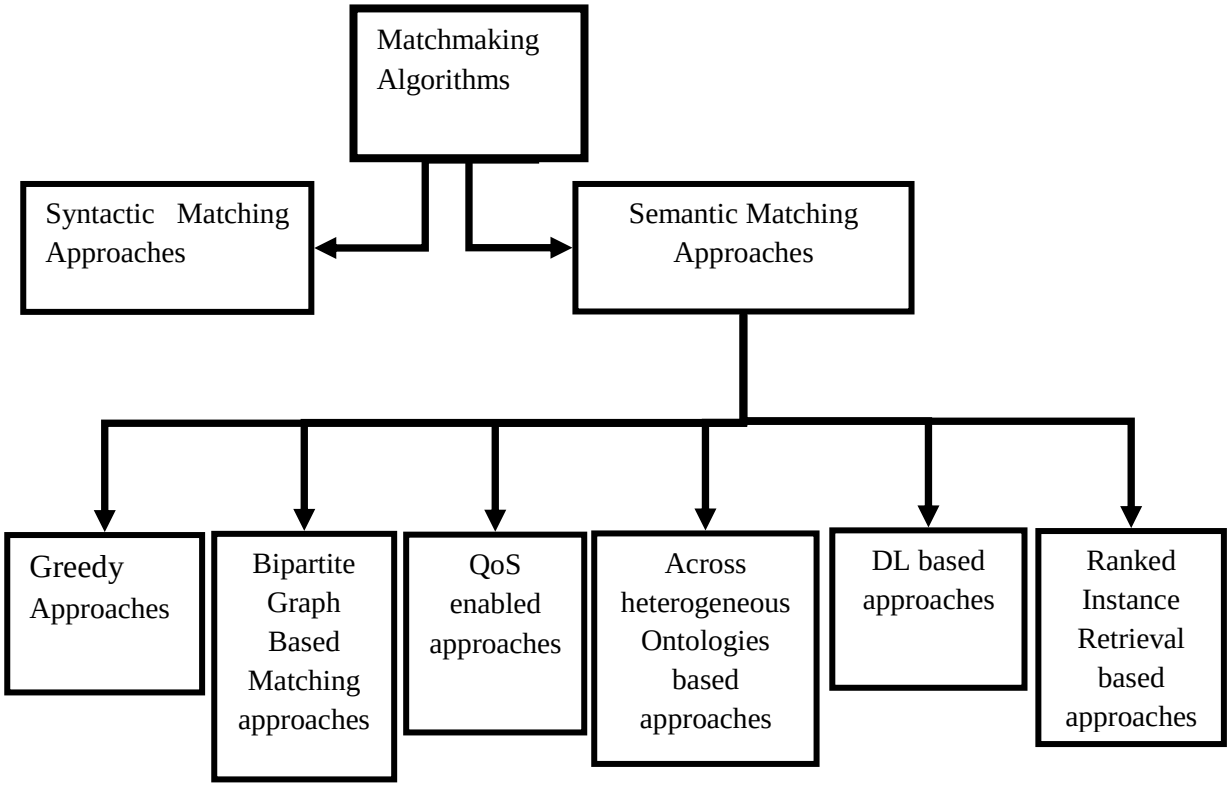


Figure 2-1: Matchmaking Algorithms Taxonomy

2.4.1 Greedy Approaches

Greedy algorithms are discussed in (Li and Guo, 2008; Sriharee and Senivongse, 2006; Hu et. al., 2008; and Payne et. al., 2002). Greedy algorithms are based on semantic matchmaking which matches services advertisements and requests based on inputs and outputs. The greedy matchmaking approaches consider the subsumption relations of concepts among service requests and service profiles. These algorithms try to match every output concept of a Query

with one of the concepts of advertisement. They start by matching all the output concepts of a request against concepts of an advertisement.

The greedy approaches rank the degree of matched results into four levels. The levels are as discussed below and summarized in Table 2-1 through Table 2-3. These tables are adapted from (Jaeger et. al., 2005).

Table 2-1: Inputs Matching (Jaeger et. al., 2005)

Result	Interpretation
0 FAIL	At least one input type of the request has not been successfully matched with one input type of the advertisement. The service cannot be executed properly.
1 PLUGIN	This matching result is provided, if the used categorization is not supported by the matching algorithm. However, this result is not the conclusion of an open world-assumption approach. The reasoning process still follows the closed world assumption, but due to the lack of understanding the conceptualization, no conclusion can be drawn.
2 SUBSUMES	For each input type of the advertisement exactly one input type of the request has been found, which is at least subsumed by the input type of the advertisement. This means that the advertised service might be invoked with a more specific input than expected, assuming the covariance of the input types of the advertised service.
3 EXACT	For every input type of the advertisement one equivalent input type of the request is found.

The greedy approaches are based on inputs, outputs and category for degrees of matching. The scheme is as follows;

• **Fail:** If no subsumption relation is found between a Request output and an advertisement output, then it is declared as failed match.

• **Plug-in:** If an advertisement output subsumes a Request output, then an advertisement output can be plugged in place of a Request output. In a plug-in match, it is assumed that service provider promises to provide output found in subclasses of an output of a service description.

• **Subsumes:** If an output of a service request subsumes an output of a service advertisement, then the service may fulfill the requirements of the service request. This is because the service advertisement provides output concept found in subclasses of the concept defined by a request output.

• **Exact:** If an advertisement output is an equivalent concept to a Request output, then they are labeled as Exact match. If a Request output is a subclass of an advertisement output, then the match is considered Exact under the assumption that service provider agrees to provide output in every possible subclasses of an advertisement output.

Table 2-1 through Table 2-3 present how these matching degree categories can be interpreted.

Table 2-2: Output Matching (Jaegers et. al., 2005)

Result	Interpretation
0 FAIL	At least one output of the request has not successfully been matched with an input of the advertisement.
1 PLUGIN	This matching result is provided, if the used categorization is not supported by the matching algorithm. As for the matching of inputs, this result is not the conclusion of an open-world-assumption approach, but due to the lack of understanding the conceptualization, no conclusion can be drawn.
2 SUBSUMES	The output types of the request subsume the output types of the advertisement or are equivalent to them. This means that the request might receive a more specific output type than expected. Assuming the contra-variance of the output types of the advertisement, it is still ensured, that the requirement of the service requester are met. Additionally for all output types of the request a successfully matching counterpart of the advertisement is identified.
EXACT	For each output type of the request one equivalent output type of the advertisement is found.

Table 2-3: Service Category Matching (Jaeger et. al., 2005)

Result	Interpretation
0 FAIL	The two concepts were not successfully matched
1 PLUGIN	Either the description of the advertised or the request is not classified or no reasoning functionality is available to determine matching for the types of categorization.
2 SUBSUME	The classification of the advertisement is subsumed by the classification of the request. This means that the advertisement offers more specific functionality than required.
3 EXACT	The classification of the advertisement and the classification of the request are equivalent.

The challenge associated with greedy algorithms is their assumption that the service provider can provide all subclasses of the output concepts. The ordering of concepts in request definition also determines the performance of the algorithms. To demonstrate the issue, consider the following example;

Lecturer and Student are subclasses of the Concept College. Through inferred relationship, Student is also a sub-concept of School. The output concept list for the service request and a service description would be:

Service output = {College, School}

Request output = {Student, Lecturer}

Firstly, Student would be matched against all concepts in the list of the service outputs. Secondly,

- College is superclass of Student and this implies an Exact match
- School subsumes Student and this implies a Plug-in match

Since the algorithm uses a greedy approach, it would match Student with College and remove them from their respective lists. There would only be one concept left in the service output list and therefore, matching Lecturer and School would result in Fail. The algorithm would therefore, return Failed match result between the service request and the service description. In essence,

- College is super class of Lecturer and this is an Exact match
- School subsumes Student which implies Plug-in match

The overall degree of matching for Request and service description is Plug-in. If the order of concepts in Request output list is changed, this matching would be achieved. The algorithm in Payne et. al., (2002) depends on the way in which concepts are ordered. This algorithm may, therefore, generally produce false negative results. Suppose for example that the output concept is not removed after matching and suppose that the advertisement is for {College, Cost} and the request is for {Lecturer, Student}. In this case, the algorithm would return an exact match since Lecturer and Student are immediate subclasses of College. The requester would receive College fees whereas they expected Lecturer information and Student information, hence a false positive result.

2.4.2 Bipartite graph based matching approaches

The bipartite matching approach exploits the expressive nature of the ontology language, OWL-S of the service and request profile. Functional elements such as input and output parameters are first extracted from each profile file. At the same time the parameters taken from the profile of an advertisement and the request profile are represented as vertices of the left and right side of the graph respectively. Each vertex is connected to all the vertices on the opposite side. At last, the bipartite graph matching algorithms then determines the set of

edges between the two parameter sets. The similarity of parameters class is computed, forming the weights of the edges in the bipartite graph.

In the bipartite-graph based matching the assumption that if an advertiser advertises a concept it would provide all the immediate subtypes of that concept as seen in greedy approaches is dropped. Hence, if an advertisement concept subsumes a request concept then it is a *subsume* match. If the request concept subsumes the advertisement concept the result it is a plug-in match. According to the rankings a plug-in match is higher than a subsume match. The matchmaking scheme takes an opposite approach to the one discussed in Section 2.3.1.
Bellur et al. (2008) defines a bipartite graph as a graph in which the set of vertices can be represented by two disjoint sets such that no edge of the graph lies between two elements of the same set. If there is a set of output concepts for service request and a set of output concepts for a service description, a graph having one vertex that corresponds to each concept in service request and service description is constructed. An edge is constructed if there is a match between a concept in a service request and a concept in a service description.

The constructed edge will have the degree of match as its weight. Matchmaking is needed in which all the output concepts of a service request will be matched against the service description. If such a matching occurs then the service description and the service request match. If a multiple of such matchmaking exist, the optimal one is chosen. If such a matching does not exist, the service request and the service description do not match. The best semantic matchmaking criterion then needs to be selected. Different weights with different degrees of match would then be assigned to those edges. Consider a case where a minimum weight would be assigned to an exact match. Given a maximum weighted edge in the matching, an

optimal matching would be a complete matching with the maximum weighted edge at its minimum.

In order to determine an overall matching for a weighted bipartite graph, Bellur et al. (2008) suggest the usage of the Hungarian algorithm (Kuhn, 1955).

In graph based matchmaking, the algorithm takes service request as input and tries to match its input/output concepts against the input/output concepts of each service description. If there is a match among input concepts as well as among output concepts, the service is included in the results set. To perform the input/output matchmaking, the Hungarian algorithm (Kuhn, 1955) is applied on a created graph. The weights, as explained above, are used to determine an optimal matching of the graph. The results obtained from the matching consist of a list of services. The service list is sorted based on the degree of match of the input and output concepts. The bipartite-graph matchmaking algorithm eliminates correctness issues found in greedy algorithms described in Section 2.3.1. This algorithm does regulate false positives and false negatives. The algorithm does not allow the service consumers to rank their preferred concepts. The algorithm works with shared ontologies among service descriptions and requests. The algorithms therefore, may not work well in heterogeneous ontology and distributed environments.

Many semantic matchmaking algorithms, including the one proposed by Bellur and Kulkarni, (2007) and the other examples given so far, are based on matching of input and output terms only. The bipartite graph based approaches can be applied in preconditions and effects based matchmaking. The algorithm for condition matching works in two phases which are *parameter comparability* and *condition equivalence*, briefly discussed below.

• **Parameter Compatibility:** it checks the equivalence of the parameters used between the request and the advertisement. During the input and output matching the mapping between the concepts used in a service request and the concepts used in an advertisement is determined.

• **Condition Equivalence:** This occurs if the preconditions and effects have a structural similarity.

2.4.3 QoS-enabled matchmaking approaches

The matchmaking techniques discussed in Section 2.3.2.1 and Section 2.3.2.2 are based on the function properties of a service. These matchmaking techniques overwhelm the requester as there are hundreds of services that are functionally equivalent. The matchmaking process can be improved by considering non-functional properties such as the QoS properties of a service (Kritikos and Plexousakis, 2008). The QoS properties of a service differ from one service to another. For a service in a network environment, QoS plays a role in determining whether the service can satisfy the goals of the requester or not. It is, therefore, important for a semantic matchmaking approach to consider QoS properties to perform a match that satisfies the goals of the user. To enable support for QoS while performing a matchmaking process, there is a need to determine how well a service satisfies the goals of the requester. One of the strategies that can be used to gather such information about a service is through an interface that can be used by consumers to rate the service. This information would then be stored and used as a service profile.

While discovering services, data from the following sources can be taken into account (Vu et. al., 2006):-

• QoS properties as provided by service providers while publishing the service.

• QoS ratings by the service consumers as mentioned above, and

• Reports from QoS monitoring agents.

Most service matchmaking approaches can be adapted to perform QoS based matchmaking. In such cases services would be matched based on their functional and non-functional properties. The results obtained from QoS based matchmaking and functional properties matchmaking would then be combined to obtain a service ranking.

2.4.4 Across heterogeneous ontologies based approaches

These algorithms provide a mechanism through which requests and services can be matched even when their descriptions use different ontologies. The algorithms determine the similarity of concepts through a function. The similarity is computed based on three methods. The first method is based on synonym sets, the second one on semantic neighborhood, and the third one on distinguishing features of concepts. The algorithm though is not flexible enough to allow service consumers to specify their preferred concepts.

This matchmaking approach relaxes the requirement for service descriptions to use a single ontology. There are different methods that are used to determine the similarity among concepts found in cross ontologies, (Guo et. al., 2005). Some of these methods are *Synonym sets; Distinguishing features of concepts*, and *semantic neighborhoods and relations*

Synonym sets: they are similar words or words with similar meanings and hence, they can be considered as a same entity. Wordnet is the most popular tool that can be used to derive the synonym set of a concept. In a case of cross ontologies the words car and vehicle may refer to one entity. The words are different but may have the same semantic meaning.

Distinguishing features of concepts: these are used to incorporate semantics into concepts with common features. The object properties and data type properties are used to perform the similarity evaluation. Different concepts with common features stand a chance to have similar meanings.

Semantic neighborhood: this is based on the idea that concepts that are being compared belong to classes that are in the same set of classes.

The similarity results obtained from each of these methods are finally added up to obtain the overall similarity. These three similarity measures are used to determine the edge weights of edges in the bipartite graph discussed in Section 2.3.3. The function concepts are taken from both the service description and the service request. These concepts are then used as nodes in constructing a graph. The bipartite graph matchmaking algorithms are then applied on the generated graph. The output will then be a list of relevant services ranked according to their matching score.

2.4.5 DL based Approaches

The semantic matchmaking based on DL algorithms provides a ranking of the matched services which was not the case with the semantic matching across heterogeneous ontologies. DLs are a family of formal languages used for knowledge representation (Kutz and Mossakowski, 2008). The DL is used to model the domain knowledge of a service in a format that is machine interpretable.

In a DL algorithm the basic syntax is based on predicate logic and consists of three kinds of descriptions which are *concept names, number restrictions* and *universal quantifications*.

- Concept names: they stand for names of objects such as car, house etc.

• Number restrictions: it gives numerical values to concepts

• Universal quantifications**:** it specifies restrictions on the objects belonging to a concept.

Advertisements and requests are described as a combination of the described concepts. The matchmaking algorithm then matches the request against a service description and each time a service is matched, the degree of matching is given.

The advantage with this algorithm is that it can be adapted to allow service consumers to specify their preferred concepts. Service consumers attach weight to different concepts and hence the final results are ranked according to their preferred concepts. Service weights can be learnt by the system simply by considering the rate at which they are used by service consumers. Services can then be ranked according to their usage.

2.4.6 Ranked instance retrieval based approaches

This Section presents a method of service matchmaking that allows service consumers to input their preference on concepts (Beck and Freitag, 2006). The algorithm applies a ranking tree to match a request against a service advertisement. A typical service request would return a large number of services. The ability of service requesters to input preference would help narrow the search. The ability to use concept preference also affords them expressive power. This enables them to express their requirements to obtain the best possible results from the matchmaking process. Given a service request annotated with users preferred concepts Bellur et. al. (2008) describe below how the ranking tree of the advertisement must be evaluated.

"The ranking tree is evaluated by calling the routine recursively for each subquery. The evaluated results are then used to form a top-level ranking tree. The rank of top-level rank tree is an average of the user preferences of all the subqueries which were

satisfied. For a negated query, the rank of top-level tree is replaced by the average of user preferences of all the concepts which are not satisfied by the query. An atomic concept belonging to the query is satisfied by advertisement if it is contained in it."

The advantage with this algorithm is that service requesters can specify their preferred concepts.

2.5 Service Discovery Architectures Taxonomy

This section presents the service discovery architectures taxonomy as shown in Figure 2-2. The service discovery architectures are basically divided into Registry based architectures, peer-to-peer architectures and Hybrid architectures (Ververidis and Polyzos, 2007; Seno et. al., 2007; Meshkova et. al., 2008; Mian et. al., 2009; Le et. al., 2007).

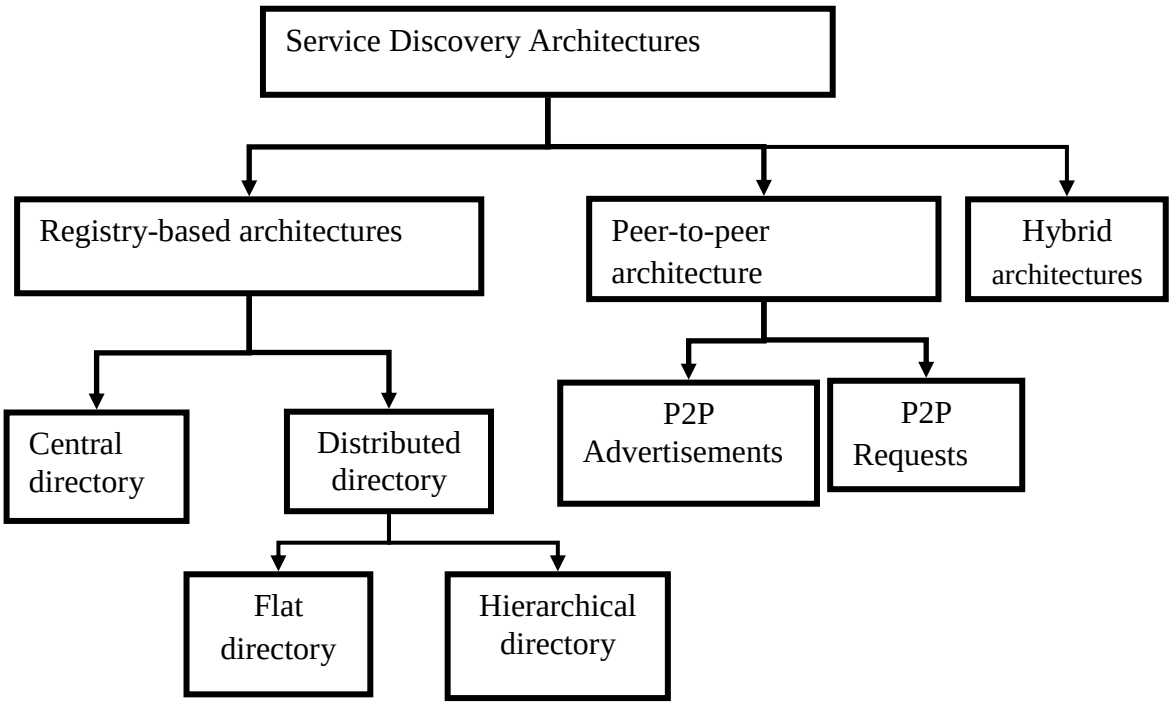


Figure 2-2: Taxonomy of service discovery architectures, adopted from Moreno-Vozmediano (2008).

The registry based architectures are either centralized or distributed, with the distributed directory further divided into a hierarchical directory and flat directory. The peer-to-peer architecture is either a peer-to-peer advertisement or peer-to-peer request. These discovery architectures are described below.

Registry-based Architectures

In these architectures participating devices in the network play the basic roles of a service host, a client or a service Registry. Service providers register their services to service directories and service requesters look up for services through this registry.

In practice, registry architectures can be implemented as centralised or distributed as mentioned above. In *Centralised* registry based architectures fixed node which takes up the role of a directory. The centralised registry architectures have challenges when used in mobile networks. These challenges include the following:

- mobile devices are not always available in the network
- the centralised registries may become bottle necks of failure
- Scalability is also an issue as mobile devices are resource constrained and hence cannot handle a large number of requests

Distributed registry based architectures are more suitable for networks where mobility is supported. A basic question is whether it would be possible for every service requester to discover and invoke any service available anywhere within the network when the registry is distributed. One simple solution is to replicate every service advertisement so that all the directories in the network would contain all the services published in the network.

In traditional approach like Jini, there are Lookup Servers which act as service registries. Jini differs from other distributed approaches since there is no communication among the lookup servers. Service providers need to advertise and maintain their services in more than one server. In current approaches the distributed registries communicate to each other and hence service advertisements are automatically replicated. The service providers only publish services in only one server and they are replicated automatically. One of the approaches used by the distributed directory architectures is managing a backbone of nodes which serve as directories. In such approaches service providers only publish services in one or more directory servers which belong to the backbone. In (Sailhan and Issarny, 2005), the members of the backbone frequently exchange the service advertisements. When a request is submitted it is forwarded to all the nodes that might contain the requested service.

Clustering approach: other research works like (Klein et. al., 2003b, Ververidis and Polyzos, 2007; Meshkova et. al., 2008) use a clustering approach which employ service rings. A service ring is made up of clusters. These clusters are formed based on physical distance among nodes and the similarity of services they provide.

The Distributed Hash Table (DHT) based technique is another solution in attempts to enable service requester to access services from anywhere in the network when the replication cannot be performed by the directory servers. DHT based techniques are described in (Yu et. al., 2003 and Chen et. al., 2008). In these techniques the network is divided into geographical regions. Each region has a responsibility of managing a set of keys that represent services in the same classification. These keys are each mapped to a region based on a hash-table mapping scheme. In each region there is a single node which stores these keys thus acting as registry. The ability to discover a remote service is possible since requesters and service providers use the same hashing function. The service requesters then find the location of the registry where the service description is stored. Using this location information, the request is

routed to that directory. In the case of clusters, the routing node forwards discovery requests and also store services advertisements published by the nodes within the region. To enable a discovery that covers a wide scope of the network, the gateways forward service requests among themselves.

The *election of nodes* that would take a role of a directory and the *election of nodes* that would participate in a directory structure is an important issue that needs to be addressed for registry-based service discovery architectures. In (Klein and König-Ries, 2002) an election mechanism is not provided, while in other approaches a directory node election is performed randomly. In more elaborate approaches, like Sailhan and Issarny, 2005, packet loss rate, effective connectivity to neighbours and capacity criteria are used to select nodes.

Registry based approaches bring additional communication costs of maintaining the structure in the network. These costs are also for replicating service information among directory member nodes for consistency.

Peer-to-peer Architectures (also referred to as Registry-less architectures)

A peer-to-peer architecture differs from the registry-based architecture in that there are no service directories that exist between service providers and service consumers. Service providers are responsible for the development of the services and broadcasting them. Service requesters broadcast service requests to the network instead of a central registry.

In architectures where there are no registries, there is a need to determine how frequent should service advertisements be broadcast to avoid network congestion and redundant transmissions. In normal cases service providers periodically broadcast service advertisements to their nearest neighbours. These service advertisements are then stored in the local cache of the receiving node. Service providers, whose services advertisements are about to expire or have expired, are prioritised in assigning them to make the next broadcast.

A similar approach is employed in (Campo et. al., 2005) where providers periodically advertise their services along with services that they became aware of provided by neighbouring nodes. A service provider reschedules advertising its services for a certain amount of time if it receives its own service description from its nearest neighbours. This means that its nearest neighbours are aware of the service it hosts.

Multicasting is another technique used to lower network loads. In multicasting service providers send their service descriptions on a fixed multicast group. The service requesters send their requests in the same way. Service requests can be sent to the nearest neighbours only or messages can be sent to the whole network as in (Helal et. al., 2003).

Broadcasting and multicasting are costly when messages are intended to cover the whole network. Alternative techniques to broadcasting and multicasting are: *Advertisement range*; *Selective forwarding, probabilistic forwarding, intelligent forwarding*; *Peer-to-peer information caching* and *Intermediate node responding to service requests* (Ververidis and Polyzos, 2007).

In Allia (Ratsimor et. al., 2002) an advertisement range which is determined from the number of hops to the service advertisement message destination is used. In this approach, a P2P information caching technique is used. The technique allows the service advertisement messages to reach a maximum number of nodes in the network. After receiving a service advertisement message, a node caches it. When advertising its services, the cached advertisement is also included.

In *selective forwarding*, after receiving a service query that it cannot satisfy the node forwards the request to the nodes that host the service.

The *peer-to-peer information caching* technique is used along with *probabilistic forwarding* instead of selective forwarding. In probabilistic forwarding, a node that receives a service query that it cannot satisfy forwards it. The query is forwarded with a probability that is inversely proportional to the number of hops already travelled by the query.

In intelligent forwarding, every node continuously monitors its second neighbourhood which is the nearest neighbours to its immediate neighbours. The advertising process starts with the service provider sending advertisements to its nearest neighbours. In the next service advertisement cycle, it will be the service provider's nearest neighbours that forward the advertisement they received to their other nearest neighbour. This node is service a second nearest neighbour to the service provider. Broadcasting is replaced by unicasts.

Intermediate nodes may also be able to respond to service requests directed to the services that they previously used or forwarded (Motegi et. al., 2002). In this way the network communication costs are reduced as the request need not to travel all the way to the service provider as the intermediate nodes can respond to the requests. *Location information* can be used as proposed in (Tchakarov and Vaidya, 2004). In this approach, service providers periodically send advertisements in cross-shaped directions. A node, along the path where the service descriptions are sent sets up a pointer that leads to the path from which the advertisement came. Whenever a node needs the service whose advertisement it received, it simply sends a request in that direction.

Hybrid Architectures

In hybrid approaches, these architectures service providers publish their services with service directories accessible to them. If they do not locate any registry, they broadcast their

messages to the entire network. After sending their queries, the service requesters may receive their replies from either the service providers or the registries.

2.6 Evaluation Frameworks

This section presents the evaluation frameworks that can be used to evaluate the different approaches in matchmaking algorithms. This is followed by a presentation of the evaluation frameworks that can be used to evaluate services discovery approaches.

2.6.1 Evaluation Framework for Matchmaking Algorithms

In this section we discuss both the *qualitative* and *quantitative* aspects on which matchmaking algorithms can be evaluated. This discussion of the evaluation framework for matchmaking algorithms is based on the work by Bellur et. al., (2008). This evaluation framework will be used in Chapter 3, Section 3.3 to compare and contrast different matchmaking algorithms presented.

QUALITATIVE ASPECTS

The qualitative aspect of evaluation framework focuses on the metrics that evaluate the satisfiability of the goals of the requester. Under qualitative aspect we consider the following characteristics *Semantic matching, false positives and false negatives, Flexible matching, Soft constraints, Preference of concepts, User defined matching, Heterogeneous ontologies, QoS enabled discovery.* These are briefly discussed below;

***Semantic matchmaking*:** any *matchmaking* approach needs to be able to be considering the meanings of concepts used in service descriptions and service requests while performing the matchmaking. The matchmaker also needs to be able to consider the relations which exist between those concepts in the ontology. In syntactic matching concepts are considered without their meanings.

False positives and false negatives: False positives are the services included in the results matchmaker even though they cannot fulfill the requester's needs. On the contrary, a false negative is when the matchmaker fails to return services that are most relevant to the requester. A trade-off exists between the two. The more flexible the matchmaker is, the more the number of false positives will be obtained in every matchmaking results. It is therefore important to control the flexibility of the matchmaker in order to balance the false positives and false negatives proportions.

Flexible matching: The semantic matchmaking algorithm should encourage service providers to be more precise in describing their services. This can be done by providing a degree of match for the matched service descriptions. The degree of match should be higher for more precise service description. If service providers are not forced to be precise in describing their services, they would make their service descriptions to be general to increase chances of being selected instead of specifying what they can offer.

Soft constraints: Soft constraints are constraints which might be preferred by service requesters but not necessarily to be satisfied.

Preference of concepts: The service requesters should be able to specify which concepts they prefer should be considered in the matchmaking process.

User defined matching: service requesters should have some control over the matching process. The algorithm should give the service requesters the ability to regulate the flexibility of the algorithm, the quality of matching etc. The ability of the service requesters to control the matchmaking helps them make the matchmaking to be more suited to their needs.

Heterogeneous ontologies: in most discovery approaches, the service description and the service request share the same ontology. In truly distributed environments services may not share the same ontology as the request. Even in such cases of heterogeneous ontologies the matchmaker needs to be able to perform the matchmaking.

QoS enabled discovery: the other aspects discussed so far consider the matchmaking based on the functional properties of a service. Service consumers may consider non-functional properties such as QoS properties (which are availability, throughput etc.) to be important. It is therefore important for the matchmaker to take these into account.

QUANTITATIVE MEASURES

This aspect focuses on the number of service returned than the satisfiability of the goal of the requester. In this aspect we consider the following characteristics; *Efficiency, Precision, Recall, and F1 and break-even.*

Efficiency*:* usually a service discovery algorithm needs to match services from any domain. In the current size of the web the number of service advertisements would be very large. This requires that the computational complexity of the algorithm not to be high.

Precision*:* Precision is defined as the number of relevant services to the number of the total retrieved services ratio. As in False positives and false negatives, the flexibility of the matchmaker determines the precision value. The more flexible it is, the higher is the rate of false positives and hence precision is low. As the algorithm becomes more rigid, the number of false positives decreases and hence precision increases. Therefore, to obtain higher precision the semantic matchmaking algorithm has to be more rigid.

Recall: Recall is defined as the number of relevant and retrieved services to the total number of relevant services. As the algorithm becomes more flexible, the number of false negatives decreases and hence recall becomes high.

F1 and break-even*:* for recall and precision to be balanced, the flexibility of the matchmaking algorithm needs to be regulated. F1 is a harmonic mean of recall and precision and when minimized, it would set precision and recall at acceptable values. Break-even is where recall and precision curves intersect. Precision and recall measures are coarse grained as they categorize service descriptions as discrete relevant or irrelevant. In semantic matchmaking where degrees of matching are important, these are not best performance metrics of matchmaking. A fine-grained evaluative approach is proposed by (Anagnostopoulos et. al., 2006). Anagnostopoulos et. al. (2006) proposes a method based on fuzzy logic which provides fuzzy equivalents of precision and recall as measures to quantify performance of matchmaking. Details on these approaches are beyond the scope of our work.

2.6.2 Evaluation Framework for service Discovery Architectures

This section presents the evaluation criteria that are used to evaluate the architectures presented in Section 3.4 of the next chapter. The service discovery architectures can be evaluated based on the criteria suggested by KÄuster et. al. (2008), Toma et. al. (2005) and Yu and Reiff-Marganiec, (2008). These evaluation criteria are: *service availability, latency, Model for non-functional properties, automated reasoning support, Scalability, Efficiency of Matchmaking, Decoupling, Query and Advertising languages expressiveness, Matchmaking vs. Brokering and User Preferences.* Each of these is discussed below.

***Service availability*:** it can be regarded as a QoS attribute of a service. A service that is frequently unavailable is viewed by requesters or users as having poor quality. A service discovery mechanism needs to be able to take such attributes into consideration.

***Latency*:** this is the time taken between when a service consumer sends a request and when it receives a response. For a Service discovery mechanisms it is important to consider latency so that requesters may not be kept waiting unnecessarily.

***The Model for non-functional properties*:** service requesters need to be able to objectively distinguish services based on their non-functional properties to make the most appropriate choice among a number of services with similar functionality. In the light of the above it is therefore required for any solution approach to present a model that supports non-functional properties that can be used in matchmaking during the discovery process.

***Automated reasoning support*:** in automation a service designer would still specify data for the service when making it available and a user would still be able to specify requirements. The discovery and the selection though would be performed without human intervention. Such automatic discovery and selection methods are essential when considering context aware service provisioning which is very important for resource constrained mobile devices. The provision of an automatic discovery mechanism can be significantly improved if machine interpretable service and request descriptions are used. During the discovery process the service descriptions and service requests are matched against each other. A new knowledge when needed is inferred based on the existing facts found in the domain ontologies.

***Scalability*:** scalability means that the approach can consider numbers of properties and also that many ranking processes are taking place simultaneously. It is important to analyze how a system reacts when there is a growth in one or more of its parameters. The scalability of a

service discovery system is influenced by the scalability of its subcomponents which are query processing, storage, etc.

Efficiency of matchmaking: efficiency is used to describe properties of an approach relating to how much of various types of resources it consumes. In a discovery mechanism the question that needs to be answered is, is it possible to match the value of arbitrary services and requests and if so how is it achieved. It is crucial that a solution approach's complexity should not trade off its efficiency.

Decoupling**:** a discovery approach needs to take independence of service descriptions and service requests into account. If a service requester writes request without knowledge of the existing service, the question would be whether the language being used offer enough guidance to ensure that relevant services will be found by the discovery mechanism.

Advertising and Query language expressiveness**:** is it possible to describe real services and real service requests in sufficient details to meet the automation criteria and this can be done without reasonable effort? It is important to determine how expressive is the query language used and how easy it is to formulate queries and advertisements. Special attention is normally given to whether semantics are supported by the language.

Matchmaking versus Brokering**:** during the discovery process two approaches can be distinguished, namely (i) *Matchmaking* - it matches requests against service descriptions but does not play a role in the next process which is the interaction between the service and the service consumer, and, (ii) *Brokering*- it matches requests against service descriptions but also plays a role in the service and service consumer interactions.

User Preferences**:** service requesters have varying preferences for the non-functional criteria depending on the situation that they find themselves in. An example is where users may consider the security property as more important than privacy when requesting a financial

service. Hence the approach needs to provide mechanisms for users to specify their preferences.

2.7 Conclusion

This chapter has presented a background on Service Description, Context Awareness, Matchmaking algorithms taxonomy and the Service Discovery Architectures Taxonomy. It has been highlighted that the challenge is not only about using context in service discovery processes but also about defining and managing context. This chapter, therefore, has clarified the angle from which we view context and how it can be used to enhance the service discovery process. A number of service matchmaking algorithms has also been presented. Lastly we presented the evaluation frameworks for both the matchmaking algorithms and the discovery architectures. In the next chapter, Chapter 3, we present the state of the art where we analyze other research efforts that enhance service discovery.

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

The widely adopted service discovery specification, UDDI, has challenges which among others, include the following: it does not address the issues of service data such as non-functional properties of a service; it is unable to aggregate information about the state of a service (Sinclair et. al., 2005); it does not support context-awareness in a straightforward and unambiguous manner (Doulkeridis et al., 2006). These are some of the short comings of the UDDI and other traditional standard discovery protocols that prevent them from meeting the requirements of SOA. Application components in SOA need to be assembled with little effort into a network of services that can be loosely coupled to create flexible dynamic processes and agile applications. Given the challenges mentioned above, traditional discovery protocols cannot support the dynamic service discovery as required in SOA.

Although a lot of research effort has been directed at addressing the limitations of the traditional service discovery mechanisms and context awareness in service discovery processes, the results of these attempts still leave a lot to be desired. This work, as already indicated, is aimed at enhancing the service discovery process by proposing a framework that addresses the quality of matched results in traditional and other related service discovery approaches. This chapter brings to light the outcomes of descriptive problem analysis phase of our research. We first discuss related work on service discovery architectures, matchmaking algorithms, and their corresponding evaluation frameworks. From this discussion, we identify strengths and weaknesses in current algorithms. The insight gained

from this analysis will assist us to formulate our solution approach. Our solution approach advocates that context can play a role in improving the results of the matchmaking process in the service discovery process. Subsequent sections of this chapter are arranged as follows: Section 3.2 discusses service descriptions and how these are used in addressing issues in service discovery. In Section 3.3 we discuss related matchmaking algorithms and their evaluation based on the framework discussed in Section 2.6.1. The related Service discovery architectures and their evaluation that is based on the framework discussed in Section 2.6.2 are discussed in Section 3.4. Section 3.5 concludes this chapter.

3.2 Service Discovery and Service Description Standards

This section presents a review on service discovery. This is followed by a discussion of service description frameworks.

3.2.1 Service Discovery

Services can be defined as autonomous, platform independent computational entities that can be used in a platform independent way (Capilla, 2006; Papazoglou et. al., 2007; Vanrompay et. al., 2009). In a typical service-based scenario, a service provider hosts a network-accessible software module which is an implementation of a service. The service provider then defines a service description through which the service is published. A client discovers a service and retrieves the service description directly from the service, possibly through metadata exchange or from a registry or repository such as UDDI (Papazoglou, 2007).

Service discovery is important in SOC as it enables a runtime association of services with service consumers in the network (Mokhtar et. al., 2007). There are three roles which can be

played by participants in SOA, namely: *Service Provider*, which is the role played by a software entity that offers service in a network; *Service Requester*, which is a role of an entity seeking to utilize a particular service; and *Service Directory*, which is the role of an entity maintaining the advertisements of the available services and a way to bind with them. Service description formalisms are used to describe the capabilities and other non-functional properties of a service. Hence service descriptions, given a service discovery protocol, enable service providers, requesters and repositories to interact with one another.

Unfortunately, service discovery is still a challenge in SOA and in dynamic grid environments in particular due to the fact that both the software service and potential software service consumers are designed, developed, and deployed independently. Doulkeridis (2007) pointed out that the main challenge in service discovery is using automated means to accurately discover services with minimal user involvement. By explicating the semantics of both the service provider and the service consumer, automated service discovery can be achieved.

In order to address challenges in service discovery, Mokhtar et. al. (2007) listed three basic requirements as follows;

(i) *Semantic Matching*: during service discovery, the underlying semantic service description and requests should be considered. The matching in traditional widely adopted service discovery standards is based on assessing the conformance of their syntactic interfaces. Syntactic matchmaking approaches achievable poor results in open dynamic environments. There is, therefore, a need to use higher level of abstractions which is independent of the low-level

syntactic realizations to dentate service semantics specific to the SOA technologies.

(ii) *Degree of Conformance*: matching should be able to identify the degree of conformance between services and clients, and rate services with respect to their suitability for a specific client or user.

(iii) *Context awareness*: the user centered service discovery calls for the discovery systems to be awareness of context and QoS.

However, there is still a need for a service description framework that will lay a ground for the above requirements to be met. Section 3.2.2 therefore, presents the different service description frameworks proposed from other research efforts.

3.2.2 Service Description Frameworks

A lot of research efforts have focused on adding semantic information to web services (Beco et. al., 2005). There are a number of standard frameworks which have been proposed that are used to specify the service semantics. Among these frameworks are the following: Web Ontology Language (OWL), Web Ontology Language for web services (OWL-S) (Martin et. al., 2004), Web Service Modeling Ontology (WSMO) (Roman et. al., 2006) and WSDL-S (Akkiraju et. al., 2005). OWL-S is ontology of services based on OWL. It can be viewed as a language for describing services, reflecting the fact that it provides a standard vocabulary to create service descriptions. WSMO provides ontological specifications for describing the core elements of semantic web services and consists of four main elements: ontologies, goals, web services descriptions and mediators. The WSDL-S defines a mechanism to semantically annotate the services described using WSDL with different ontology languages (e.g. OWL, UML).

Some researchers use extensions of these standard frameworks to describe services so that they can suite their domain requirements. Examples of those researchers are Beco et. al. (2005), and Bocchi et. al. (2005), who both extended OWL-S. Beco et. al. (2005) identified the process enactment as one of the key requirements for the semantic grid. They argue that the system needs descriptions, such as workflows to facilitate composition of multiple resources, and mechanisms for creating and enacting these in a distributed manner. Workflow is the orchestration of a set of activities to accomplish a larger and sophisticated goal. In its raw form OWL-S cannot describe Workflows and hence, Beco et. al. (2005) proposed a semantic workflow representation language, OWL-WS, which is an extension of OWL-S. Bocchi et. al. (2005) extended OWL-S to make it compatible for Grid environments where it would support the Grid related concepts of a grid service such as job and Job Submission Description.

The most widely used semantic service description framework among the afore-mentioned frameworks is OWL-S. OWL-S is also recommended in service-oriented computing research (Kirsch-Pinheiro et. al., 2008). Despite its being widely adopted, it still has some challenges. The challenges include its applicability when addressing context awareness of services. Researcher efforts, in addressing these issues, took advantage of its generality and flexibility. Researchers (such as Balzer et. al., 2004; Aggarwal, 2004; KÄuster et. al., 2008) investigated the practical applicability of the OWL-S and concluded that though it gained too much attention at its release there was still more work to be done. The pitfalls identified were in the areas of validity of the OWL-S documents, Matchmaking of Profiles and execution of process models.

Firstly, under the issue of OWL-S documents, it had emerged that OWL-S ontologies are not compatible with DL reasoners. This problem can be solved by tailoring down the ontologies to OWL DL. Secondly, when Matchmaking profiles, OWL-S does not provide concrete concepts for the publication and discovery of services. This is a problem since the matchmaking algorithms depend on the capabilities of the requester and the way functional and non-functional parameters of a service are modeled. Specifying the whole set of Input, Output, Preconditions and Effects (IOPE) in a request in OWL-S is a problem. This is so because even if a Service fulfils the intended goal, i.e., Outputs and Effects (OEs) such a service may not be discovered if the Inputs and Outputs (IOs) of the request do not match those of the service description. Thirdly, under the Execution of the Process Model, the two identified shortcomings of OWL-S are: OWL-S lacks the specification of a model for representing conditions, and the concept of Extensible Style sheet Language (XSL) transformation for mapping data types is not sufficient.

3.3 Matchmaking Algorithms

The main purpose of service discovery is to enable the client, which could be a human or a software entity, to identify and select services which will help fulfill its goal at any given time. Given that there is large number of dynamic services in a Grid environment, we require an efficient algorithm which is scalable and is capable of minimizing false positives and false negatives during the discovery process. Various algorithm techniques have been developed by researchers which are aimed at solving specific problems. Some are meant for grid services and some are meant for web services while others are meant for both grid services and web services. We introduced matchmaking algorithms in Section 2.4 of the previous chapter. In subsections below we review Matchmaking approaches in Section 3.3.1, and present their evaluation in Section 3.3.2.

3.3.1 Matching Approaches

In Section 2.4.2 a background on the matchmaking approaches was presented. The commonly used algorithms in matchmaking which employ semantics are Greedy algorithms, QoS enabled matching algorithms and the Bipartite graph matching algorithms. Examples of the applications of greedy algorithms are in the following research works (Klusch et. al., 2006; Fu et. al., 2007; Ludwig and Reyhani, 2005; Yonglei et. al., 2006; and Jaeger et. al., 2005). Solution approaches such as OWLS-MX in (Klusch, M. et. al., 2006), match services based on their functional properties which are inputs and outputs. Other research efforts like (Fu et. al, 2007; Hu et. al., 2008; Kritikos and Plexousakis, 2008; Vu et. al., 2006) embrace the idea of accommodating non-functional properties of services to enhance precision of the matchmaking algorithms. This is referred to as a QoS enabled matchmaking discussed in Section 2.3.2.3.

A bipartite graph matching is used in (Guo et. al., 2005 and Zilin et. al., 2006). Bipartite matching approach has proved useful in addressing the issues with the widely adopted traditional discovery approaches which have a high rate of false positives and false negatives in their matchmaking results. The bipartite graph matching approach as presented in Section 2.4.2.2 brings in the issue of latency. This is so with all semantic based matchmaking approaches as they use the ontology resources which are process intensive. We envisage that the issue of latency can be addressed by using context matching that is restricted to syntactic matching to filter the services before they are matched semantically.

Semantic matchmaking approaches discussed in this section are based on subsumption relations of concepts among advertisements and service profiles. Friesen and Altenhofen

(2005) address the issue of latency in semantic matchmaking approaches by proposing that matchmaking be performed at publishing time. They attempted to ease the process of service matching, especially for composed web services, so that during runtime the problem is reduced to selection of services rather than first matching services that will be composed to fulfill the requester's goal. The semantic matching between the goal of the composed web service and the appropriate component services is performed at publishing time. The Goal-Capability Association storage introduced in this approach is stored as an association between goals in the Web Service Description of the composed service and Web Service Descriptions of the potential services. The algorithm in Friesen and Altenhofen (2005) performs the matchmaking at publishing time. It keeps the goal-capability association storage consistent for any changes of goals or service capabilities published to the registry. The solution approach by Friesen and Altenhofen (2005) has weaknesses such as high memory consumption.

Ludwig and Reyhani (2005) identified seven requirements for a matchmaking framework in grid environments which are as follows:

High degree of flexibility and expressiveness: different advertisers may want to describe their services with different degrees of complexity and completeness and hence the description tool or language should support these needs.

Support for subsumption: services should not be matched based on simple service names but a type system with subsumption relationships is required.

Support for data types: attributes such as quantities should be part of the service descriptions and the way to compare this by means of data types.

The matching process should be efficient: the process should minimize delays, which would compromise its effectiveness.

Appropriate syntax for the grid: the matchmaker should be compatible with grid/web technologies and the information must be in a format appropriate for a grid environment.

Flexible and modular structure: the framework should be flexible enough to allow grid applications to describe their context semantics and grid services to describe their semantics in a modular manner.

Lookup for matched services: the framework should provide a mechanism to allow the lookup an invocation of matched services.

Ludwig and Reyhani (2005) proposed a multi-phased service discovery framework for Grid environments which is comprised of three stages: *context selection* where the request is matched within the appropriate context, *semantic selection* where request is matched semantically, and *registry selection* where the lookup is performed. They use of context is aimed at improving the quality of results. Unfortunately, Ludwig and Reyhani (2005) do not address scalability. They use semantics in all phases and hence the scalability is hindered in the discovery process. Though multi-phased approach where context is matched separately is commended, the excessive use of ontology resources in all the phases results into an unnecessary overhead and hence efficiency is compromised.

The matchmaking algorithms discussed in this section have strengths and weaknesses in addressing service discovery at the matchmaking level. These are summarized under evaluative analysis in Section 3.3.2.

3.3.2 Evaluative analysis of the Matchmaking algorithms

This section presents an evaluation framework for the Matchmaking algorithms. It also draws conclusions on the presented matchmaking algorithms based on the evaluation framework. For evaluation purposes of matchmaking algorithms in our research work, we use *precision* and *recall* as they are the standard and are the most often used performance measures. A number of matchmaking approaches together with the requirements suggested by Ludwig and

Reyhani were presented in Section 3.3.1. In Table 3-1 we present the service description languages they use and their precision and recall results. In the table the algorithms are classified according to whether they address issues in Web service environments or Grid based environments. Among the considered literature, only the approaches by Zilin et. al., (2006) and Ludwig et. al., (2005) are grid based. The services descriptions in the approaches are in OWL-S with the exception of Friesen and Altenhofen (2005) who uses WSMO. Fu et. al., (2007) takes advantage of the extensibility of OWL-S to include non-functional properties in the service description which are QoS properties. From the considered research works in Table 3-1, only Zilin et. al., (2005), Fu et. al., (2007), Ludwig and Reyhani (2005) and Klusch et. al. (2006) considered precision-recall as the evaluation metrics for their approaches.

The evaluation of the matchmaking algorithms helped shed some light on the progress made by researchers in enhancing service discovery. The evaluated research works were selected based on a requirement they address in service discovery. Guo et. al., (2005) addresses the issue of heterogeneous ontologies requirement by presenting a similarity function which relaxes the requirements of a single ontology and thereby accounts for different ontologies.

This is a requirement for distributed systems. Zilin et. al., (2006) and Ludwig and Reyhani (2005) take advantage of the richness of the profiles and the extensibility of OWL-S to describe grid services. This strengthens our opinion that OWL-S framework can be used to semantically describe grid services.

Table 3-1: Matchmaking Algorithms Comparison

Algorithm	**Grid/Web Service Based**	**Advertisement Language**	**Precision** (Average/Range)	**Recall** (Average/Range)
Bipartite graph Matching by Guo R. et. al. (2005)	Web Services	OWL-S	Not Available	Not Available
Extended Kuhn-Munkres algorithm by Zilin et. al. (2006)	Grid Based	OWL-S	75	98
Optmization algorithm by Friesen et. al. (2005)	Web Services	WSMO 1.0 and WSMO 1.1	Not Available	Not Available
Multistrategy algorithm by Fu et. al. (2007)	Web Services	OWL-S extended with SWRL and QoS	80	58
Ludwig S.A. et. al. (2005)	Grid Based	DAML-S/OWL-S	45-100	15-100
OWL/OWL-S based algorithm by Yonglei et. al. (2006)	Web Services	OWL-S	Not Available	Not Available
OWLS-MX by Klusch, M. et. al. (2006),	Web Services	OWL-S	65-93	10-94
Owl-s Matcher by by Jaeger et. al. (2005)	Web Services	OWL-S	Not Available	Not Available

Most of the matchmaking algorithms are based on subsumption relations among service request and advertisements (Friesen et. al.:2005, Fu et. al., 2007; Yonglei, 2006; Klusch et. al., 2005 and Jaeger, 2005). The differences among them are the similarity measures, and the service parameters that they also consider on top of the inputs and outputs.

3.4 Service Discovery Architectures

More than addressing or enhancing the discovery process at matchmaking level, it is also important to consider infrastructure on which it is performed which are service discovery architectures. Service discovery architectures provide an infrastructure through which all the participants interact to provide the service requester, with the most relevant services in as shortest time as possible. Regarding service information dissemination, there are three basic architectures that a service discovery approach may adopt as discussed in Section 2.5 (Ververidis and Polyzos, 2007; Meshkova et. al., 2008; Sena et. al. 2007, Moreno-Vozmediano, 2008; Mian et. al., 2009). These architectures are Registry based architectures, Peer-to-peer architecture and hybrid architecture.

Despite the large number of publications on each of the service discovery architectures described in Section 2.5, it is not clear from research which one is best compared to the other two. The reason that makes it difficult to conclude on an architecture that is most suitable for dynamic environments is that the decision depends on many factors. Some of these factors relate to the characteristics of a network and others to tuneable parameters to be supported. In a mobile network with high mobility and a low service request rate, a distributed architecture without caching is preferable. It is only preferable when compared to registry-based architectures. This is because registry-based architecture would keep outdated service information in directories, or would require costly constant network communication for maintaining service information integrity and coping with mobility. In a case where there was a high service request rate, a registry-based architecture could be more preferable. In this case, a peer-to-peer architecture would require that service consumers frequently broadcast their queries to the whole network which is also costly. This network congestion would outweigh the network congestion created in a registry-based architecture.

Generally, none of the three architectures can outperform the other two. The underlying routing protocol may also hinder the performance of service discovery architecture as shown in (Sailhan and Issarny, 2005). In general proactively routed networks where mobility is supported, the hybrid architecture proves to be the in terms of service availability (Ververidis et. al., 2007). When it comes to messaging overhead the peer-to-peer architecture is the best compared to the other two. For reactively routed networks such as in (Sailhan and Issarny, 2005), where mobility is supported, it was shown that a peer-to-peer architecture may outperform a hybrid one, both in terms of higher service availability and lower message overhead. Delays are almost the same in both architectures.

In essence, it would be good to have a flexible architecture that can self tune its parameters and change from registry-based to peer-to-peer or hybrid, based on a mobile network's dynamic characteristics.

The next sub-sections, Section 3.4.1 presents research works that adopt or adapt these architectures and the evaluation of these research efforts in Section 3.4.2.

3.4.1 Service Discovery Architecture Approaches

The registry architecture classifications were discussed in Section 2.5. Examples of Registry-based Architectures are (Kozat and Tassiulas, 2003 and Tyan and Mahmoud, 2004). The Peer-to-peer architectures (Registry-less architectures) are used in (Varshavsky et. al., 2005; Lenders et. al., 2005). A registry-less architecture is much simpler from registry-based architectures. This is because there is no need for directory selection and maintenance mechanisms.

Some solution architectures have been proposed which include Miles et. al. (2003), Doulkeridis et. al. (2006), Chen et. al., (2008) and Toninelli et. al., (2008) among others.

Criteria that can be used to evaluate the effectiveness of service discovery architectures are *service availability, messaging overhead* and *latency* (Ververidis et. al., 2007). In addition to these are the requirements that need to be satisfied, namely: *Model for non-functional properties, Automated reasoning support, Scalability, Decoupling of service descriptions and requests, Query and Advertising languages expressiveness, Match making vs. Brokering and lookup for matched services***.** These are outlined in KÄuster et. al., (2008), Toma et. al., (2005) and Yu and Reiff-Marganiec (2008).

In SOA there can be a number of services with similar functionalities and hence, for the requester to be able to select the most appropriate service there is a need for the consideration of non-functional properties. There are a considerable number of efforts on techniques that can be used in expressing non-functional properties such as implementing them as operator graphs (e.g. Chen et. al., 2008). They can also be implemented in RDF as in Miles et. al., (2003). Scalability is one of the most important requirements that need to be addressed by any solution approach architecture as the number of services can grow considerably in SOA and especially for the resource constrained mobile devices. Examples of techniques to address it include performing matching on the server side (Miles et. al., 2003), through adopting a distributed filter-and-pipe software architecture (Chen et. al., 2008), to adaptation (Toninelli et. al., 2008).

Based on service discovery requirements raised in section 3.3.1, the authors developed architecture as shown in Figure 3-1. In the architecture the Grid application sends out a request to the service discovery matchmaker (1).

The request goes through the context matching module first where it is matched within the appropriate context of the application ontology. This means that depending on the service request, which came from one of the applications, the appropriate context ontology is chosen

and the first match is performed. Additional parameters are attached to the request and forwarded to the semantic matching module (2). In this module the semantic matching is performed where the service request is matched using the semantics (metadata) of services. Having all necessary semantic data, a service lookup is done using a service registry (3). This lookup information is sent back to the Grid application (4) to be used for the Grid service call (5).

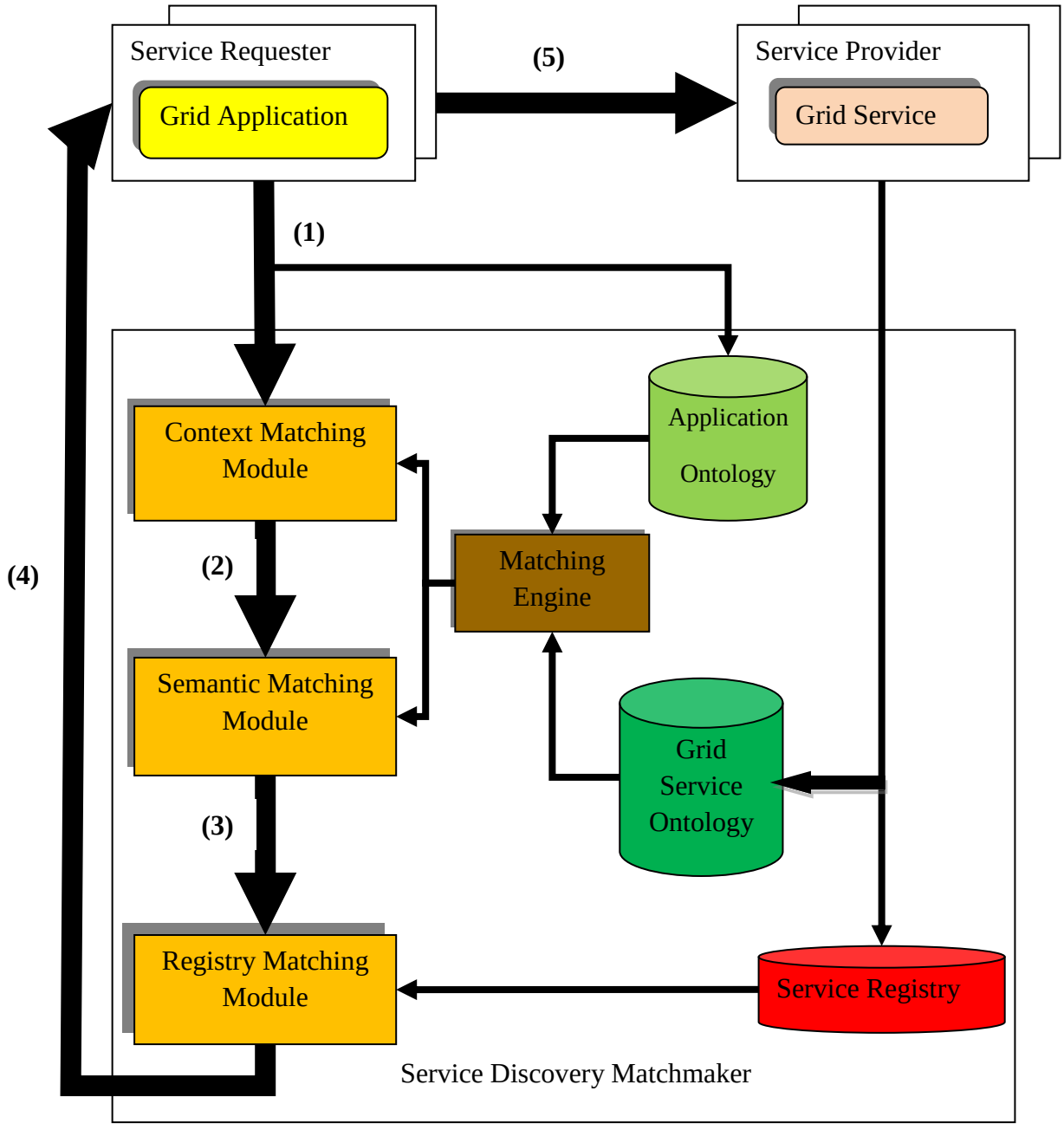


Figure 3-1: Semantic Service Matchmaker (Ludwig, 2005)

3.4.2 Evaluative analysis of Service Discovery Architectures

The evaluation for service discovery architectures is as summarized in Table 3-2. Context-Awareness is regarded as a prerequisite in most of the related approaches considered. Service context, which we use interchangeably with non-functional properties of a service, can be implemented as graphs. The graph can then be searched using graph based search algorithm (Miles et. al., 2003; Doulkeridis et. al., 2006 and Chen, 2008).

Table 3-2: Discovery Architecture Evaluation

Approach	**Non-functional properties support (Context)**	**Automated reasoning**	**Scalability**	**Decoupling**	**Efficiency**
UDDI-M^T by Miles et. al., 2003.	✓	Through semantics	Server Side Matching	✓	Server Side Matching
Solar by Chen G. et. al., 2008.	✓	Not supported	The adopt a filter-and-pipe software architecture	×	Persistent Context sensitive queries
Doulkeridis, C. et. al., 2006)	✓	Through Semantics	Scalable context updates are traded for improved searching	✓	Use best first search algorithm
AIDAS by Toninelli, A. et. al., 2008	✓	Through Semantics	Through Adaptation to access devices	✓	Matching on server side

Most of the service discovery approaches discussed in this section adopt the semantic matchmaking which we strongly support. Unfortunately, if the whole discovery process is purely semantic it comes with a cost of inefficiency. This is due to the computational intensity of the ontology repositories and the inference tools. It is, therefore, necessary to accommodate syntactic matching in the discovery process which we call hybrid matchmaking. For scalability reasons the discovery architecture should allow the computational intensive sub processes of the discovery process to be performed at the server side.

3.5 Conclusion

In this chapter we have presented state of the art approaches aimed at enhancing service discovery in SOA. We first presented a state of the art on service description framework. The discussion on the service description framework is mainly on semantic description frameworks. From the discussion it came out that despite its shortcomings, OWL-S is the most widely adopted semantic service description framework. One of the reasons for its wide acceptance is its extensibility. These service descriptions are made available so that they can be matched against service requests. There are different types of matchmaking algorithms which perform this task. In this chapter we have also presented a state of the art on these matchmaking algorithms. Of the matchmaking algorithms presented chapter in Section 2.4, the most commonly used approaches are greedy algorithms, QoS enabled matchmaking and the bipartite graph matching. The infrastructure enabling usage of the service descriptions has to be considered in the matchmaking process hence, the service discovery architectures. The service discovery architectures are registry-based, registry-less and the hybrid architecture. Using the evaluation frameworks adopted strengths and weaknesses of matchmaking algorithms and the architectures were identified. These strengths and weakness are as

summarized in Table 3-1 and Table 3-2. One of the lessons learnt from the considered research works are the different ways of incorporating context into the service discovery process. The main weakness prevailing in most of them is that they employ semantics but scalability is traded off in the process. These weaknesses and strengths informed the design of our own solution approach presented in the next chapter, Chapter 4.

CHAPTER 4

MODEL DESIGN

4.1 Introduction

To address the service discovery issues raised in Section 1.2 and to meet the objectives defined in Section 1.4 a model is developed in our research work. Our model provides a way of describing service that supports both static and dynamic service and user contexts. Our model also provides a framework that enables the utilization of semantics and context data in the discovery process. Our investigation in this study revealed that there was a need for describing services in a way that will enable us to address issues of semantics and context awareness. This cannot be addressed by the current widely adopted restrictive technologies such as WSDL used for UDDI (Doulkeridis and Vazirgiannis, 2004; ShaikhAli et. al., 2003; Kirsch-Pinheiro et. al., 2008). Our investigation also revealed that current service discovery technologies that support automated service composition do not support a personalized user centric discovery (Toninelli et. al., 2008). Many solutions exist in literature which attempt to enhance the service discovery process. These solutions were discussed in Chapter 3. Most of them employ semantics but do so at the expense of scalability. Building upon existing solutions such as Ludwig and Reyhani, (2005) and Klusch, et. al., (2006), gave rise to a Context Aware Service Discovery (CASDA) model, which improves the quality of discovered results without trading off scalability.

To achieve an enhanced service discovery there was a need to investigate other aspects that contribute to it. These other aspects are service description, matchmaking algorithms and the service discovery architecture. In this chapter, therefore, we present service description framework adopted for CASDA as one aspect that contributes towards the realization of an

enhanced service discovery. From literature (e.g. Doulkeridis et. al., 2006; Klan, 2006 and Krisch-Pinheiro et. al., 2008) it was seen that context can play a role in addressing the raised issues in service discovery. The context model presented in this chapter enables the context to be utilized in the service discovery process. With service context and requester context defined, there is a need for them to be matched. In this study we limit matchmaking of context to a pure syntactic matching. An algorithm which matches these contexts is presented. The functional properties of the services are matched semantically and a semantic matchmaking approach used in CASDA is presented. As a core contribution of this chapter, we present our overall solution approach to the raised issues by presenting the integrated model Architecture. We also give a brief description of each component.

4.2 Service Description

The first issue that this work aimed to address as a means of enhancing the service discovery process was finding a way of representing service descriptions that is expressive enough to cater for both static and dynamic contexts of a service. In addressing the issue of service description we adopt the OWL-S. We chose OWL-S over others because it is generic and extensible and hence it can allow the annotation of non-functional properties of a service. Secondly, as stated in Section 3.2, OWL-S is recommend for describing the semantics of services in service-oriented computing research (Kirsch-Pinheiro et. al., 2008). Another reason is that although OWL-S is meant for web services it is rich and general enough to describe any service. This makes it more useful for GUISET.

To enable automated reasoning during service discovery, the services need to be semantically described. Our investigation as seen in Section 3.2 revealed that for a semantic and yet expressive language for our solution approach OWL-S is the best candidate. Many solution approaches which use OWL-S exist (such as Beco et. al., 2005, Bocchi et. al., 2005 and

Jaeger et. al., 2005), with some of them capitalizing on its extensibility to address various issues in service discovery. In our work for each published service, there is a standard OWL-S service description. A separate repository is used to manage service context descriptions. This is to allow the modularity and efficiency in the matchmaking engine. The context matchmaker makes use of the service context descriptions and it matches them syntactically. The second phase in the matchmaking engine is the semantic profile matching which makes use of the OWL-S service descriptions. The separation of the matchmaking phases into a purely syntactic and semantic helps to minimize the usage of the computational intensive ontology tools. The minimal usage of the ontology tools improves the efficiency of the matchmaking engine.

4.3 Context Modeling

In our approach it was important to adopt a proper context model that is flexible yet expressive enough to describe both the static and dynamic contexts of the service and the requester (Mahmud et. al., 2007). Context can be modeled in different ways depending on the domain in which it is intended to be used. For example, as discussed in Section 2.2, there are key-value paired models, mark-up scheme models, graphical models, object oriented models, Logic based models and ontology based models. Ontology based models are the most promising models which fully support heterogeneity and interoperability (Strang and Linnhof-Popein, 2004). Our solution approach utilizes ontology during semantic service match making phase. We therefore adopt the Object oriented models combined with the graphical model to minimize the overhead that would be caused by the utilization of the ontology resources. The object model was chosen because of its generality and flexibility, performance and inter-operability. Context exists in different types as was classified by Dey et. al., (2001). Context is also a large data set and can be classified into numerous context

types. Therefore the object model combined with the graphical model fits perfectly in a dynamic grid environment.

In our approach context entities are treated as objects and hence as atomic nodes. The relationship among the nodes is represented by edges and hence a graphical model. For illustration purposes, we use a non-exhaustive context model and hence we only consider *Time* and *Location* which we consider as also playing a role in the service and requester interaction. *Time* and *Location* are described in the Table 4-1. Section 4.3.1 discusses the requester context model. Section 4.3.2 discusses the service context model.

Table 4-1: CASDA context description

Context Data Type	Description
Time	This refers to the time of the day as some services may be available at specific times of the day.
Spatial	This is the context that is related to the location parameters. For the mobile requester or service it is the expected location

4.3.1 Requester context

In our approach we represent the requester, the context entities such as a location names and time as nodes. If a requester is in a certain location at a particular time of the day, say Durban at 08H00, edges are created between the requester node and its current location and between the requester node and its current time to represent the relationship as in Figure 4-1.

In this model more context nodes can easily be integrated in the context model without tampering with the overall model structure. With a model such as this the requester's context is determined by determining the nearest neighbor nodes. In Figure 4.1 the requester is associated with two nodes within the graph model which are *Location-1* and *Time-1*.

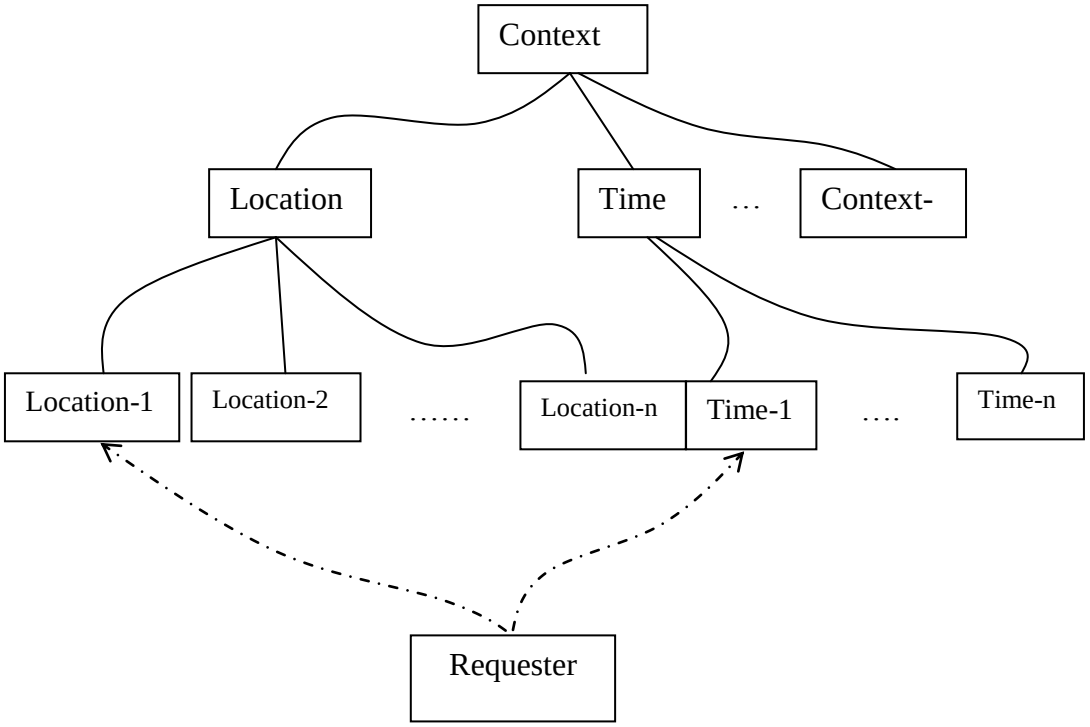


Figure 4-1: CASDA Requester Context Model

The matchmaker will only allow services matching the location and time to be passed on to the process intensive semantic matchmaker. This context can be updated dynamically even during the session. For example, if the requester moves from one location to another his association with the previous location node is disabled and a new one is established. Depending on the time of the day and location, a single requester may access services using different devices in different situations. i.e. during office hours a Web meeting participant might be using his or her office desktop or laptop, whereas during the rush hours in the morning or in the afternoon he might be driving or in a taxi and hence might be using a portable device such as a PDA or a smart phone.

In such a scenario it is important that the discovery mechanism retrieves services that are compatible with the device capabilities. The requester location context helps the reasoners to determine the services to be loaded to the semantic matchmaker. By loading only the

services that match contextually overhead is reduced on the ontology resources by not loading all the services from different domains and locations available in the registry.

The location information is given in symbolic names and in coordinates as in (da Rocha and Endler, 2006). In our approach the location nodes or objects are named in symbolic name while the coordinates are attributes. The activity context also helps in narrowing down the domain of services to be loaded. For an example, during office hours it could be inferred that the requester is engaged with professional activities and hence entertainment services would be irrelevant.

4.3.2 Service Context

The context of a service can be the location of the service, version, provider's identity, and the type of returned results and its cost of use (Doulkeridis et. al., 2006). The service context model (Figure 4-2) is similar to the requester context model. Here the services which are associated with a similar set of context entities are grouped together and this is the list that is retrieved by the syntactic matchmaker phase of the matchmaker module. Within a service context there is a volatile *service list* which lists services that share the same contexts. This is the list of services which are returned when the requester queries services and specifies the contexts corresponding to the one within which the list falls under.

The identification of the service is by name or its alias in the model. The identification information is a link between the service description files which are in different repositories. There are two types of repositories, one with WSDL and one with OWL-S files. In a grid environment, where state is of essence, the activity of a service at a given time is crucial.

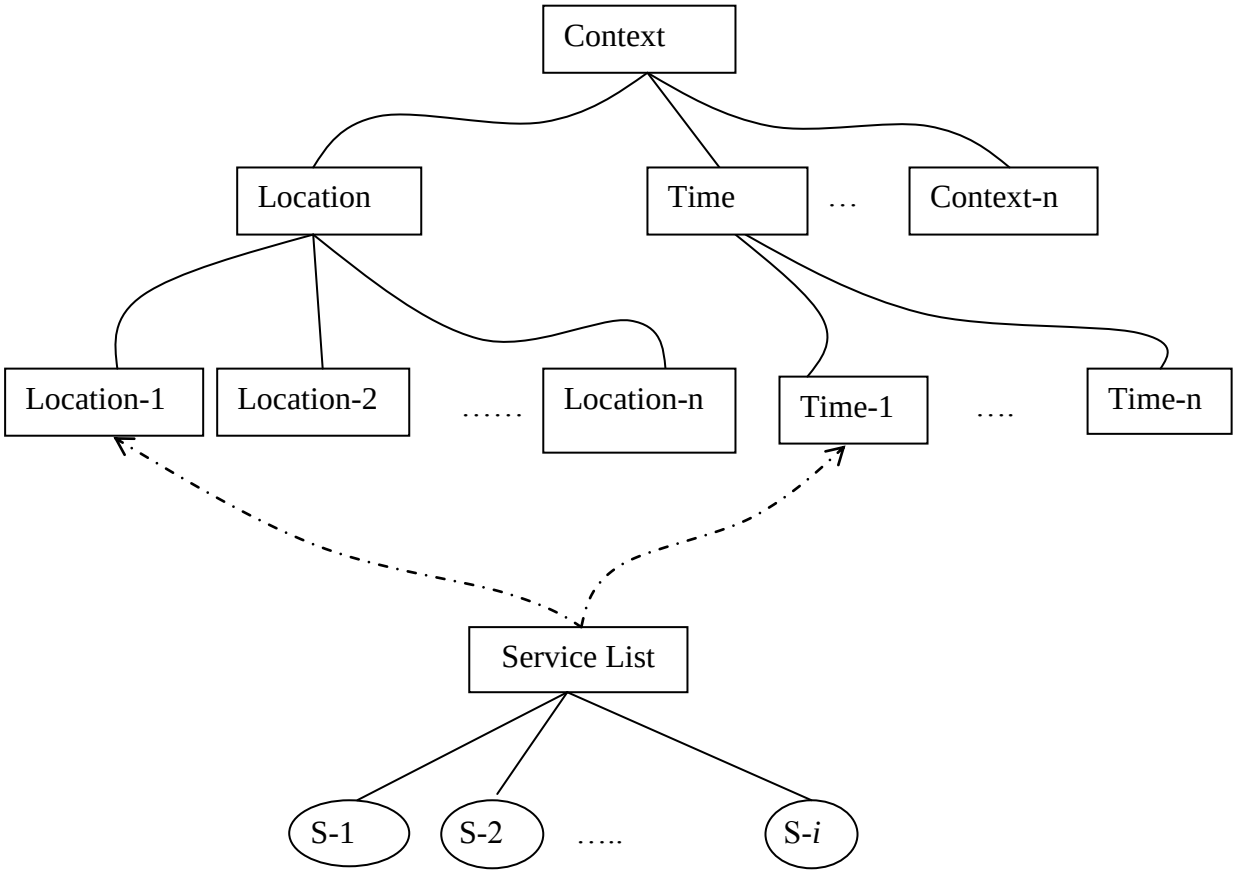


Figure 4-2: CASDA Service Context Model

The activity information include the process state, status of the service, whether it is busy, idle or out of service and the current load at a given time. During discovery the capabilities of a requesting device may be matched against the requirements of the service, and if they do not match then it means the service cannot execute or perform its function on the requesting device. Requesters may also have preference over the service providers and hence this is also catered for. The failure rate as a QoS is also of much importance in the selection of the appropriate services during service discovery. All this information can be added with ease in the context model due to its flexibility.

4.4 Context Matchmaking algorithms

This section presents two algorithms that are used to query both the service context and the requester context. These algorithms query and organize the context data in such a way that their matching can be achieved at the context matchmaking stage. These algorithms adopt a best-first-search strategy with context being represented in a graphical model (Sbihi, 2006; Kacholia, 2005).

4.4.1 Requester Context Querying Algorithm

The algorithm presented in this section is for searching for a requester stored as a node in a graph model. This algorithm is adapted from (Pearl, 1984 and Kacholia et. al., 2005). Its output is a list of context attributes which will be matched against those of the services. A brief outline (and notation) of the algorithm follows while its pseudo-code is shown in Fig 4.3

The algorithm receives a query from a requester which consists of a set of keyword concepts $Q = (t_1, t_2, t_3 \ldots t_n)$. We denote the set of nodes matching keyword t_i by S_i. Other notations used in the algorithm are briefly described in Table 4-2.

The search algorithm attempts to find best rooted directed trees connecting to at least one node from each set, S_i. The algorithm manages two queues: one queue for the set of nodes to be expanded (L_{open}), and another queue for the set of already expanded nodes (L_{closed}). For each node u being explored at a given iterative search, a track of the best known path from u to any node in S_i succeeding u is kept. At each step of the algorithm, among the incoming iterator node with highest priority, i.e., u' , a child node that u should follow to reach a node in S_i in the best known path is scheduled for exploration.

A node u is explored as follows: Outgoing edges are traversed to propagate keyword-distance information from u to adjacent nodes, and the node moved from L_{open} to L_{closed}. Additionally, if the node is found to have been reached from all keywords specified in the query Q, it is added to an output queue. Answers are output to the output queue when the algorithm decides that no answers with higher scores will be generated in future.

Table 4-2: Notations used in the requester querying algorithm (adapted from Kacholia et. al., 2005)

Q	Set of keyword concepts $t_1, t_2, \ldots t_n$
C	Set of concepts defining the user context
L_{open}	A set of nodes that are yet to be explored
L_{closed}	A set of nodes already explored and expanded
S_i	A set of nodes matching the terms specified in the query
S	A union of sets of nodes matching each term specified in the query
u	The current node being explored
u'	The best child node of u to be followed to reach a goal node t_i in S_i

The distance of a node u from the keyword nodes i.e., the number of edges from the nearest keyword node, as determined when it is inserted into L_{closed} after the goal node was returned is stored. A depth cut-off value can be specified to prevent generation of answers that would be unintuitive due to excessive path lengths. Best-first search algorithm adapted in our algorithm in Figure 4-3, in its most basic form is as follows:

The L_{open} list starts with a single node which is the starting node. The L_{closed} list starts empty and it is where the examined nodes will be added. The while loop starts by checking whether L_{open} is empty or not. If it is empty, then the algorithm returns failure or no registered

requester and exits. Otherwise the first node which will be the starting node is removed from L_{open} and placed in L_{closed} which is a list of Nodes that have been visited. The node is then expanded, where expansion is the identification of successor nodes of the starting node. The successor nodes are checked to see whether or not one of them is the goal node. If any successor is a goal node, the algorithm returns success and the solution, which consists of a path traced backwards from the goal to the start node.

INPUT: Q
Output: C

Initialization:
$L_{open} \leftarrow S$;
$L_{closed} = \emptyset$;
$C = \emptyset$

Main Steps:

WHILE $(L_{open} \neq \emptyset)$ **do**
/*if the open list is empty it means the graph has been exhausted*/

MOVE(u) /* *the node u from the open list to closed list*
Remove u from L_{open} *;* $L_{closed} \leftarrow u;$ */
IF (*u is the goal node)*
/* return *node u neighbors*/*
FOR ($\forall$ *successor nodes of node* $u \notin L_{open} \vee L_{closed}$)
Add the successor nodes to C;
ENDFOR
IF (any successor nodes, u', of node *u* is the requested node)
EXPAND(u)*;*
FOR ($\forall$ *successor nodes of node* $u' \notin L_{open} \vee L_{closed}$)
Add the successor nodes to C;
ENDFOR
ENDIF
ENDWHILE
EXIT

Figure 4-3: Requester Context Querying Algorithm

This loop will only be broken if the algorithm returns the shortest path leading to the goal node or, failure if L_{open} is empty and the requested node was not found.

4.4.2 Service Context Querying and matchmaking Algorithm

The Requester Context Querying Algorithm discussed in the previous section obtained attributes describing context of the requester from the graph. In this section, we present an algorithm, shown in Figure 4-4, which uses the results obtained above. It retrieves service context attributes from the graph and matches them against those of the requester. Table 4-3 gives a brief description of the notations used in the context matching algorithm.

Table 4-3: Notations used in the context matching algorithm

Q	Set of keyword concepts t_1, t_2, ... t_n
L_{open}	A set of service nodes that are yet to be explored
L_{closed}	A set of nodes already explored and expanded
S	Set of nodes representing services available in the graph
$service_j$	The current service node being explored
$service_{j,status}$	A Boolean attribute of a service with values, functional or non-functional
$service_{j,weight}$	Weight of a service determined from the matching context attributes
c_{ij}	the *i*-th context attribute of the *j*-th service
C	Set of concepts defining a service context
$ServiceList$	A set of services matching with requester context

The algorithm takes Q as input where, Q consist of the keyword concepts defining the requester context obtained from the algorithm in Section 4.4.1 i.e., $Q \leftarrow Q \cup C$. The requester profile consists of the context attributes obtained from the algorithm in Section 4.4.1. Like the algorithm in Section 4.4.1 it also adopts the best-first search algorithm. Let $ServiceList$

represent a set of services ready to be passed on to the semantic matchmaker phase. The algorithm first initializes $ServiceList$, L_{open} and L_{closed} lists. A service is represented by a Node in the search tree. In our work we aim to minimize wastage of resources such as matching non-functional services. Hence before the service context can be matched against that of the requester the status is checked to verify whether the service is functional or non functional. Next the list L_{open} is checked if it contains any service Nodes and if so the algorithm proceeds otherwise it exits with a message, no services are available. Let $context_{ij}$ represent the i-th context attribute of the j-th service. Let $service_{j,weig\,ht}$ be the service weight average value which is the value with which the service context matches with that of the requester. Let $C = \{c_{1,}c_2, c_3, \dots, c_n\}$ be a set of n context attributes obtained from the requester context querying algorithm to be matched against the services contexts. And let $P_i = \{p_1, p_2, p_3, \dots, p_n\}$ be the set of the corresponding weights that the requester attaches to each context attribute when submitting the request. c_i is compares with c_{ij} and if it matches it is used to determine the service weight. Then the normalized service weight average for each service can be given by:

$$service_{j,weig\,ht} = \sum_{i=0}^{n} p_i c_i \qquad \text{4.1}$$

Where, n stands for the number of context attributes submitted by the requester. The service weight is then compared with the threshold. It then checks all the services and verifies the functionality of $service_j$. If a service weight after matching does not meet the requester specified threshold it is discarded otherwise the service is listed among the services to be matched at semantic level, i.e. added to $ServiceList$.

The requester and the service contexts both form the backbone of our work. As much as we fully support the semantic matchmaking in our work we would first like the services to match the requester context versus the service contexts purely syntactic. Hence the services have to

first meet context matching value threshold before they can be passed on for semantic matchmaking.

INPUT: Q
OUTPUT: $ServiceList$

Initialization:

$ServiceList = \emptyset$;
$L_{open} \leftarrow S$;
$L_{closed} = \emptyset$;

Main steps:

WHILE($L_{open} \neq \emptyset$) **do**
$service_j \leftarrow first\ node\ of\ L_{open}$;
*/*For all c_i matching $c_{i,j}$*/*
$\forall c_i \in Q \wedge c_i \equiv c_{i,j}$
/ Determine the service weight by means of eq. 4.1*/*
$service_{j,weig\ ht} = \sum_{i=0}^{n} p_i c_i$

IF ($service_{j,weig\ ht} \geq threshhold \wedge service_{j,status} = functional$)
*/*Add service to service results*/;*
$ServiceList \leftarrow ServiceList \cup \{service_j\}$
$L_{closed} \leftarrow L_{clo\,sed} \cup \{service_j\}$

ELSE
$L_{closed} \leftarrow L_{closed} \cup \{service_j\}$

ENDIF
ENDWHILE
RETURN $ServiceList$
EXIT

Figure 4-4: Service Context Querying Algorithm

This is aimed at saving the resources utilized by the ontology reasoners during the semantic matchmaking by not allowing services which do not meet the contextual requirements of the requester to be processed by the reasoners. The requester in his request provides the contextual requirements which must be met by the services. All the services meeting the requirements are listed for semantic matchmaking. If there is none of the available services that meet the requirements all the available services are passed for semantic matchmaking

phase so that the discovery process does not terminate as a result of failure to meet specified contexts.

4.5 Semantic Service Matchmaker

After the context matchmaking has been performed it is then the semantic matchmaking process that is performed. For this purpose we utilize the services of OWLS-MX (Klusch et. al., 2006). This is aimed at addressing the problem with the current widely adopted standard services discovery techniques such as UDDI which are solely based on syntax. The semantic matchmaker matches the services based on their Input and Output (I/O).

In Section 4.5.1, we discuss OWLS-MX in details as it is adopted in the semantic matchmaking phase of our solution approach. Klusch et. al., (2006) proposed a matchmaker (OWLS-MX) which is a hybrid semantic web services match maker that complements logic based reasoning with approximation based on Information Retrieval similarity computations. Klusch et. al., (2006) is of the idea that under certain constraints the logic based only OWL-S Input/output matchmaking approaches can be outperformed by hybrid ones. The hybrid semantic service matching performed by the matchmaker OWLS-MX exploits both logic-based reasoning and content-based information retrieval techniques for OWL-S service profile I/O matching. Section 4.5.2 discusses the subsumption relations used by OWLS-MX.

4.5.1 Generic OWLS-MX matching algorithm

The OWLS-MX matchmaker takes any OWL-S service as a query, and returns an ordered set of relevant services that match the query each of which is annotated with its individual degree of matching, and syntactic similarity value. OWLS-MX then first classifies the service query I/O concepts into its local service I/O concept ontology. For this purpose, it is assumed that

the type of computed terminological subsumption relation determines the degree of semantic relation between pairs of input and concepts.

Auxiliary information on whether an individual concept is used as an input or output concept by any registered service is attached to this concept in the ontology. The respective lists of service identifiers are used by the matchmaker to compute the set of relevant services that I/O match the given query according to its five filters. In particular, OWLS-MX does not only pair wisely determine the degree of logical match but also determines the syntactic similarity between the conjunctive I/O concept expressions in OWL Lite. These expressions are built by recursively unfolding each query and service input (output) concept in the local matchmaker ontology. As a result, the unfolded concept expressions are including primitive components of a basic shared vocabulary only.

Table 4-4: Notations used in the semantic matchmaking algorithm

S	Set of matching services
R	Request
dom	Degree of match of a service against a request
α	Service matching threshold
INPUTS_R	Set of input concepts from the service request profile
OUTPUTS_R	Set of output concepts from service request profile
INPUTS_S	Set of input concepts from the service profile
OUTPUTS_S	Set of output concepts from the service profile
SIM_{IR}	Similarity computing function

Any failure of logical concept subsumption produced by the integrated DL reasoner of OWLS-MX will be tolerated, if and only if the degree of syntactic similarity between the respective unfolded service and request concept expressions exceeds a given similarity threshold. The generic matchmaking algorithm for the semantic matchmaker is represented formally as shown in Figure 4-5 (Klusch et. al., 2006):

Let $INPUTS_S = \{IN_S,i|0 \leq i \leq s\}$, $INPUTS_R = \{ IN_R,j|0 \leq j \leq n\}$, $OUTPUTS_S$ $= \{ out_S,k|0 \leq k \leq r\}$, $OUTPUTS_R = \{ out_R,t|0 \leq t \leq m\}$, set of input and output concepts used in the profile I/O parameters HASINPUT and HASOUTPUT of registered service *S* in the set *Advertisements*, and the service request *R*, respectively. Attached to each concept in the matchmaker ontology are auxiliary data that informs about which registered service is using this concept as an input and/or output concept.

Algorithm *Match*: Find advertised services S that best hybrid match with a given request R; returns set of (*S, degreeOfMatch,SIMIR*(*R, S*)) with maximum degree of successful match (*dom*), and syntactic similarity value exceeding a given threshold α.

FUNCTION MATCH(*Request R*, α)
LOCAL *result, degreeOfMatch, hybridFilters* = {SUBSUMED-BY, NEAREST NEIGHBOUR*}*
FORALL (*S, dom*) $\in$ CANDIDATES*inputset*($INPUTS_R$) $\wedge$ (*S, dom'*) $\in$ CANDIDATES*outputset*($OUTPUTS_R$)
do
degreeOfMatch $\leftarrow$ MIN(*dom, dom'*)
IF *degreeOfMatch* $\geq$ *minDegree* $\wedge$ (*degreeOfMatch* $\notin$ *hybridFilters* $\vee SIM_{IR}(R,S) \geq \alpha$)
then
result := *result* $\cup$ { (*S, degreeOfMatch*, $SIM_{IR}(R,S)$) }
ENDIF
ENDFOR
RETURN *result*
ENDFUNCTION

Figure 4-5: Semantic Matchmaking Algorithm (Klusch et. al., 2006)

The first variant of OWLS-MX, *OWLS-M0* utilizes the logic-based semantic filters Exact, Plug-in, and Subsumes which were described in Section 2.3.1, whereas the hybrid filter Subsumed-By is utilized without checking the syntactic similarity constraint.

The other variants of OWLS-MX *OWLS-M1*, *OWLS-M2*, *OWLS-M3*, and *OWLSM4* compute the syntactic similarity value $Sim_{IR}\ (out_S, \mathrm{out_R})$ by use of the loss-of-information measure, extended Jacquard similarity coefficient, the cosine similarity value, and the Jensen-Shannon information divergence based similarity value, respectively. One of the symmetric token-based string similarity measures, the cosine similarity metric is defined as follows. The cosine similarity was chosen as it is one of the best performers in the OWLS-MX variants.

→ The cosine similarity metric

$$Sim_{cos}(S,R) = \frac{\vec{R}.\vec{S}}{\|\vec{R}\|_2^2 \cdot \|\vec{S}\|_2^2} \quad \text{.....4.2}$$

With standard TFIDF term weighting scheme and the unfolded concept expressions of request R and service S are represented as n-dimensional weight index term vector $\vec{R}$ and $\vec{S}$ respectively. $\vec{R} \cdot \vec{S} = \sum_{i=1}^{n} \omega_{i,R} \times \omega_{i,S}$, $\|X\|_2 = \sqrt{\sum_i^n \omega_{i,X}^2}$, and $\omega_{i,X}$ denotes the weight of the *i*-th index term in vector X.

4.5.2 Subsumption Relations in OWLS-MX

The semantic service matchmaking algorithm in OWLS-MX, in Section 4.5.1 uses Subsumption relations. This section presents the subsumption relations for OWLS-MX as discussed in (Klusch et. al., 2006). Let T be the terminology of the OWLS-MX matchmaker

ontology specified in OWL-Lite (SHIF (D)) or OWL-DL (SHOIN (D)); CT_T the concept subsumption hierarchy of T;

LSC(C) the set of least specific concepts (direct children) C' of C, i.e. C' is immediate sub-concept of C in CT_T ; LGC(C) the set of least generic concepts (direct parents) C' of C, i.e., C' is immediate super-concept of C in CT_T ; $Sim_{IR}(A,B)\epsilon[0,1]$ the numeric degree of syntactic similarity between strings A and B according to chosen Information Retrieval (IR) metric with used term weighting scheme and document collection, and $\alpha \in [0;1]$ given syntactic similarity threshold; $\doteq$ and $\dot{\geq}$ denote terminological concept equivalence and subsumption, respectively.

Exact match. Service S exactly matches request R,

$$\Leftrightarrow (\forall IN_S \exists IN_R : IN_S \triangleq IN_R) \wedge (\forall OUT_R \exists OUT_S : OUT_R \triangleq OUT_S)$$

The service I/O signature perfectly matches with the request with respect to logic-based equivalence of their formal semantics.

Plug-in match. Service S plugs into request R,

$$\Leftrightarrow (\forall IN_S \exists IN_R : IN_S \dot{\geq} IN_R) \wedge (\forall OUT_R \exists OUT_S : OUT_S \in LSC(OUT_R))$$

Relaxing the exact matching constraint, service S may require lesser inputs than it has been specified in the request R. This guarantees at a minimum that S will be executable with the provided input if the involved OWL input concepts can be equivalently mapped to

WSDL input messages and corresponding service signature data types. This is assumed to be a necessary constraint of each of the subsequent filters.

Subsumes match. Request R subsumes service S,

$\Leftrightarrow (\forall\, IN_S \exists IN_R : IN_S \dot{\geq} IN_R) \wedge (\forall\, OUT_R \exists OUT_S : OUT_R \dot{\geq} OUT_S)$.

This filter is weaker than the plug-in filter with respect to the extent the returned output is more specific than requested by the user, since it relaxes the constraint of immediate output concept subsumption. As a consequence, the returned set of relevant services is extended in principle.

Subsumed-by match. Request R is subsumed by service S,

$\Leftrightarrow (\forall\, IN_S \exists IN_R : IN_S \dot{\geq} IN_R) \wedge \forall OUT_R \exists OUT_S : (OUT_S \triangleq OUT_R \vee OUT_S \in$

$LGC(OUT_R)) \wedge Sim_{IR}(S, R) \geq \propto$.

This filter selects services whose output data is more general than requested, hence, in this sense, subsumes the request. The focus is on direct parent output concepts to avoid selecting services returning data which is thought to be too general. It then depends on the individual perspective taken by the user, the application domain, and the granularity of the underlying ontology at hand, whether a relaxation of this constraint is appropriate, or not.

Logic-based fail. Service S fails to match with request R according to the above logic-based semantic filter criteria.

Nearest-neighbor match. Service S is nearest neighbor of a request R,

$\Leftrightarrow (\forall\, IN_S \exists IN_R : IN_S \dot{\geq} IN_R) \wedge (\forall OUT_R \exists OUT_S : (OUT_R \dot{\geq} OUT_S) \vee (Sim_{IR}(S, R) \geq \propto)$.

Fail. Service S does not match with request R according to any of the above filters.

The OWLS-MX matching filters are sorted according to the size of results they would return, in other words according to how relaxed the semantic matching. In this respect, it is assumed

that service output data that are more general than requested relaxes a semantic match with a given query. As a consequence, the following total order of matching filters is obtained

$$Exact < Plug-In < Subsumes < Subsumed-By < Logic-based\ Fail < Nearest-neighbor < Fail.$$

4.6 The Model Architecture

The SOA makes it possible to have a large number of services without their clients in mind. Applications are also designed without the services offered in mind. In an event of failure of a service, an application needs to be able to seamlessly unplug from one service and plug into a new service without disrupting the activity at hand. This, therefore, requires that services be dynamically discovered at runtime. However, with the current service discovery technologies which do not support service context and semantics, such a requirement cannot be achieved.

To overcome such problems we then proposed CASDA, a service discovery model which combines a pure syntactic matching and a semantic matchmaking in a service discovery process. In Section 4.6.1 we discuss the model design considerations and the architecture with its components are discussed in Section 4.6.2. CASDA employs both pure syntactic and semantic matchmaking techniques to address the issues in service discovery raised in this work. Context is also taken into account in the matchmaking process.

4.6.1 Model Design Considerations

In order to take the work further from other research efforts we make the following assumptions:

- Service providers have subscribed and hence the trust and security issues have been addressed.
- The services have already been published to the infrastructure.

Based on these assumptions the following design considerations are then taken into account:

Latency

In our model the issue of latency is addressed by using context matching that is restricted to syntactic matching to filter the services before they are matched semantically.

Efficiency of the Matchmaking process

Though the discovery process must be more requester centric (hence the use of context), efficiency should not be traded. The matchmaking process should be efficient which means that it should not burden the requester with excessive delays that would reduce its effectiveness (Ludwig, 2005). And this is achieved in our model by allowing a pure syntactic matchmaking at the requester versus service context level.

Robustness against internet Failures

During the semantic service matchmaking process the matchmaking engine makes use of ontologies of which in some cases might need to be imported (Gagnes, 2006). Without internet connection these ontologies cannot be imported and hence the matchmaker fails to evaluate the services for use. Our model hence has a component which manages both local and imported ontologies which is used in cases of internet failure.

Reusability

In AIDAS (Toninelli et. al., 2008) the context manager, query processor/manager and profile matchmaking are treated as separate components for reusability purpose. For this reason in our solution approach we adopt such an approach. Since the model is not to be based on a

single or particular type of registry which may become available or unavailable dynamically, there is a service manager component which handles the configurations of the registries. Context is also very dynamic and also has context on its own hence, it is important that the context model be managed separately so that the model can be extended, modified or updated without affecting the other modules.

Registry independence

Our work addresses the issues with the current service discovery standards which are solely based on key word search and are very restrictive in expressing service data. Our work provides a solution for XML based service registries. We would, therefore, like the model to be independent of any type of XML registry (Doulkeridis, 2006) and hence the registry is managed separately and the communication is through interfaces like Java API for XML Registries (JAXR) for Java based implementations.

4.6.2 CASDA Architectural Components

This section presents the components of CASDA which provides a solution to the issue of context awareness in service discovery. The components are represented in Figure 4-6 with their sub-components and a brief discussion is given below which outlines the problem aspects which those components address.

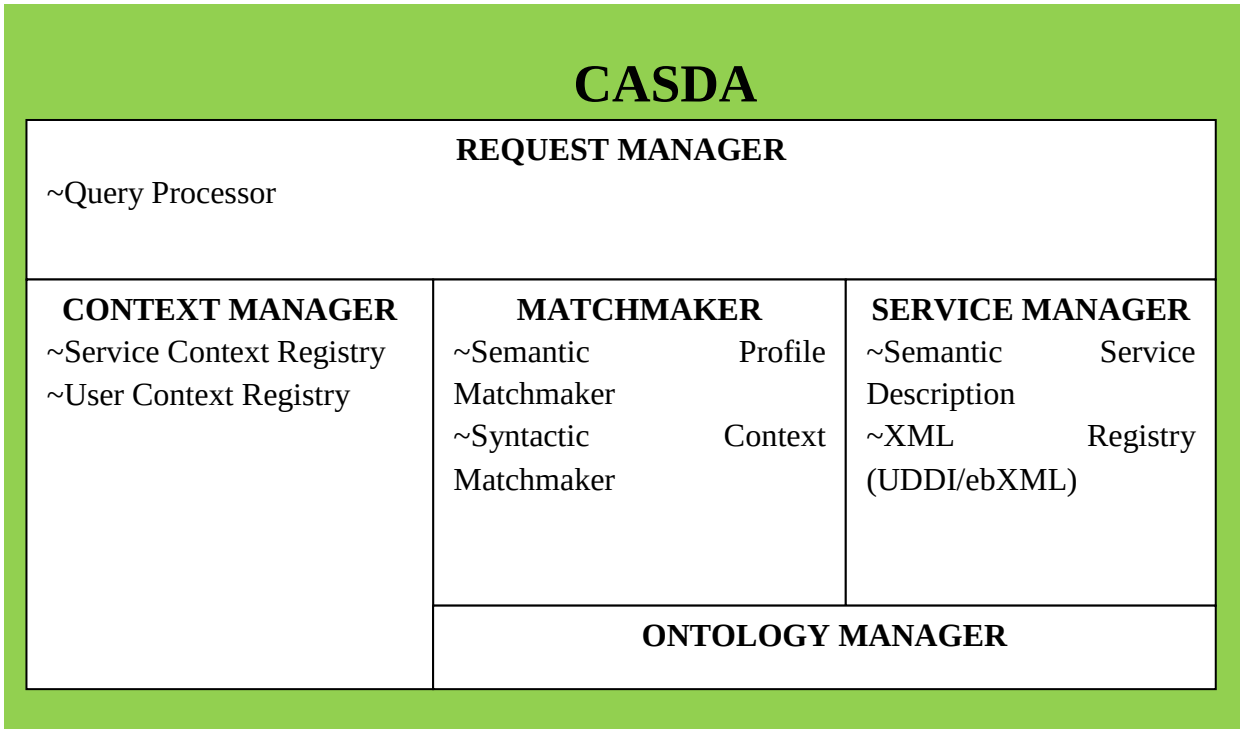


Figure 4-6 CASDA Conceptual Architecture

4.6.2.1 Context Manager

The context manager consists of registries for services and requester context data which store the service and requester context implementations. Its core responsibility is keeping the context data up to date. Every time a requester initiates a session the information such as a change in location or any other contextual information such as the device in use the context information is dynamically updated accordingly. The service information is also kept up to date whereby this component pings the service to check its availability. The services also use this component to notify the registry of any status changes.

4.6.2.2 Service Manager

The service manager consists of an XML service registry which is enhanced to support context aware semantic service discovery. It also consists of two repositories, one consisting

of semantic services descriptions which are in OWL-S and the other consisting of the standard WSDL service grounding descriptions.

4.6.2.3 Matchmaker

The matchmaker consists of two components i.e. the semantic matchmaker and the pure syntactic context matchmaker and is aimed at addressing the question of context utilization during service discovery. The semantic matchmaker first matches the services based on their Inputs and Outputs (IOs). The matching here is solely based on the Service Profile in their OWL-S description. The context matchmaker refines the list of services available in the registry before they are matched by the process intensive semantic matchmaker.

4.6.2.4 Request Manager

The request manager component provides an interface for the requesters to request services. Its core responsibility is the formulation of the requests for the discovery of services while the prototype performs the Syntactic context matchmaking this component formulates an OWL-S request which will be matched against the services from the Syntactic matchmaking phase in the Semantic matchmaking phase.

4.6.2.5 Ontology Manager

This ontology manager manages both the local as well as the imported ontologies which are acquired during active internet connections. This component addresses the case of internet failure where the ontologies can be accessed locally for reasoning purposes during the semantic matchmaking phase

4.7 Conclusion

In this chapter we have presented a design of our model which resulted in the Context-aware Services Discovery Architecture (CASDA) as our solution approach to enhancing a service discovery process. We then presented our service description approach which supports service context. Our service description approach makes use of OWL-S. The OWL-S was chosen among other service description frameworks, because it is expressive and can be extended to also include non-functional properties of a service as done in (Beco et. al., 2005) and (Bocchi et. al., 2005). We have also presented service and requester context models and their matchmaking algorithms. Matchmaking among service context and requester context is limited to pure syntactic matchmaking. This is done so as to limit the utilization of the ontology resources which are computationally expensive. Semantic matchmaking is used to match functional properties of a service. We therefore presented semantic matchmaking algorithm employed by CASDA. Our semantic matchmaker builds upon the capabilities of OWLS-MX (Klusch et. al., 2006). The CASDA model was presented in a form of architecture. As stated above, CASDA exploits both pure syntactic and semantic techniques in performing the service matchmaking. Chapter 5 deals with prototyping and evaluation of our proposed model, CASDA.

CHAPTER 5

MODEL PROTOTYPING AND EVALUATION

5.1 Introduction

In chapter 4 we presented a Context-Aware Service Discovery Architecture (CASDA) model which provides our solution approach towards enhancing the service discovery process. As indicated earlier, the aim of this research was to enhance a service discovery process in a service oriented environment such as GUISET (Adigun et. al., 2006). CASDA exploits both the pure syntactic similarity and the semantic similarity techniques in addressing the issue of service discovery.

CASDA model prototype is developed as a proof of concept and a tool for evaluation of our solution approach. In this chapter we present the design of CASDA prototype in the form of UML diagrams which includes, use cases, sequence diagram and activity diagrams. The CASDA prototype makes use of JWSDP 1.5 which comes bundled with a UDDI registry. In CASDA, UDDI is used for the publication of services. The ontologies are obtained from an online repository and published locally on an XAMP HTTP server. These ontologies are utilized in the semantic service profile matchmaking stage. The service profiles are described using OWL-S. The implementation of a pure syntactic matchmaking algorithm that matches a requester context against service context is also presented. To prove the effectiveness of CASDA the prototype is evaluated based on scalability, precision and recall.

5.2 CASDA Prototype Design

CASDA prototype implements the basic functions of our model. In this section we present the prototype design of our model detailing CASDA use case scenario. Given the use case scenario presented under the motivation for the study in Section 1.5, we present the design of the prototype supporting Menzi's needs. Menzi's tutorial session needs not to be interrupted as a result of failure of any participating service at runtime. In this case there is a need for services to be discovered at runtime as Menzi changes context or after a failure of any participating service. We present the UML models of the discovery architecture proposed in our work as a solution approach to the issue of service discovery in SOA. We present the Use case Model, the sequence diagram and the activity diagram.

5.2.1 Use case Modeling

Based on assumptions and design considerations presented in Section 4.6.1 in the previous chapter, a use case scenario is painted through which the CASDA prototype can be evaluated. Figure 5-1 represents the use case diagram for the prototype. There are two actors which are Requester and the Request Manager. Here the requester is a software agent acting on behalf of the user. CASDA has two use cases:

1. The requester agent requesting or discovering services.

2. The Request Manager supplying a request to the matchmaker

The first use case which is requesting for services is extended by two subtasks. The first one matches service context and requester context. The second one matches request and service profiles semantically.

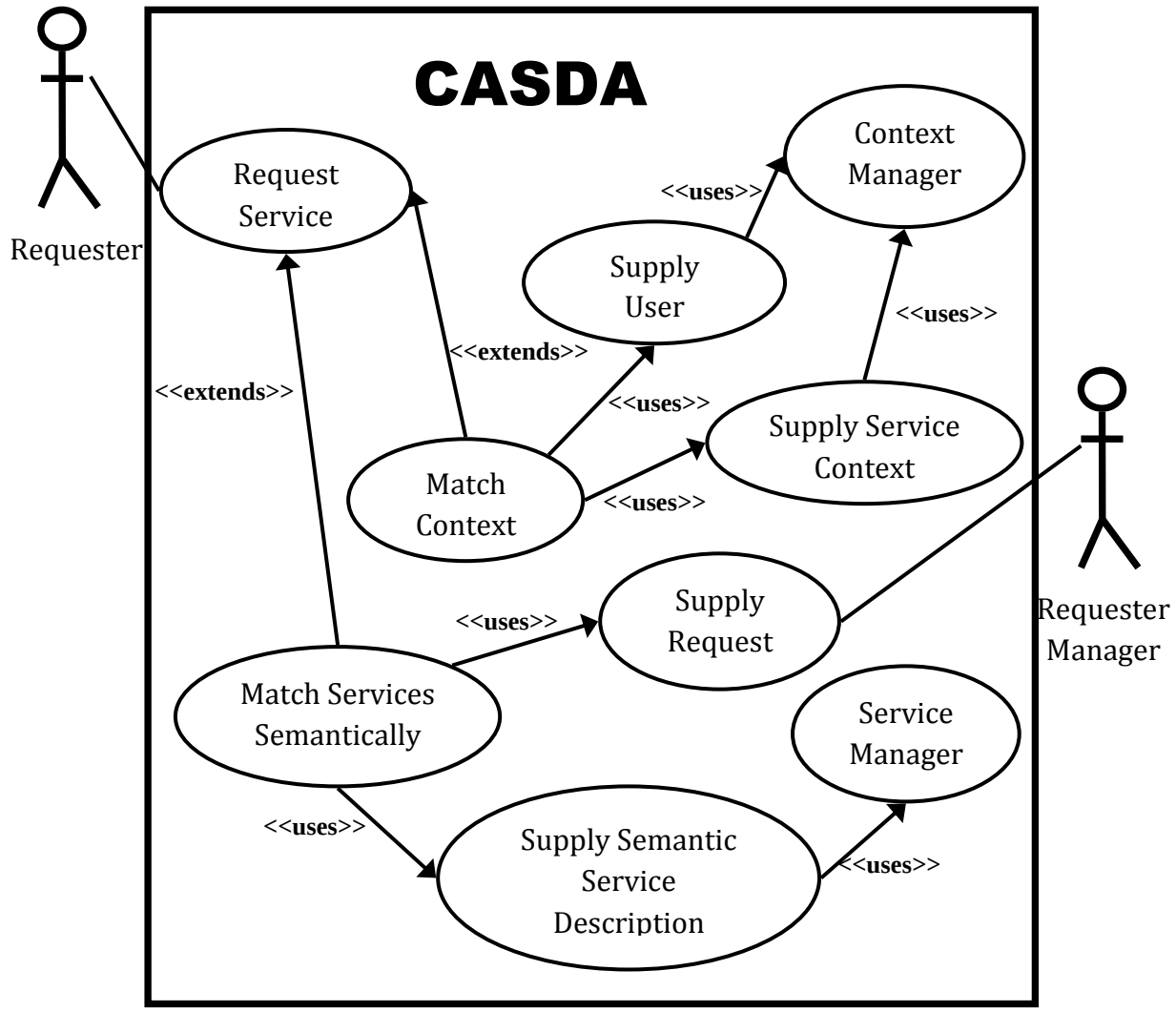


Figure 5-1: CASDA Use Case Diagram

When a service is requested the match context subtask is called. The match context subtask obtains the service context and the requester contexts from the context manager. The context manager is responsible for the management of contexts for both the published services and the requesters. After the Context matching phase has been performed the results are passed on to the pure Semantic Matchmaker. The results from the context matching phase are a list of service names or IDs. The Semantic matching phase makes use of the semantic request from the Request Manager and the semantic service description from the service manager.

The results list obtained from the context matching phase is used to filter the service descriptions to be loaded from the service manager.

Requester

The requester is a software agent or an application which act on behalf of the requester. It is responsible for updating the requester profile and context in the CASDA. It acts as a listener to the requester and as the requester's context attributes changes it updates them accordingly in the CASDA. The requester also request for the available services in the CASDA.

Context Manager

The context manager is modeled as a graphical structure which stores the users, services and context objects as nodes. Every time the requester updates the context the manager updates the graph accordingly. The Context Manager also registers as a listener to registered services where in any changes in their context is updated accordingly in the graph. In Figure 5-2 "*an*"s represents Nodes and "*ae*"s edges e.g. there are node labels with names corresponding to the services they represent. We also have Cape Town and Durban as node labels and they represent location context entities corresponding to the names. We also have edges which give association between node/entities e.g. the *activity_nationalpark_service* which is available at 16H00 and the *price_fishservice* available in Pretoria.

Request Manager

The Request Manager is a software entity which generates semantic service request. After the pure Syntactic Context matching phase has been performed, the Request Manager supplies the semantic request to the CASDA's Matchmaker component. This semantic request is matched against the semantic service descriptions from the Service Manager in the Semantic Matching phase.

```
an "degreenationalgovernment_scholarship_service"
an "monographpersoncreditcardaccount_recommendedprice_service"
an "car_report_service"
an "academic-degreegovernment_lending_service"
an "maxprice_colabreadorbiscuit_Bothservice"
an "sportslegal-agent_destination_service"
an "company_profession_service"
an "Cape Town"
an "title_actionfilmrecommendedpricequality_service"
an "academic-degree_funding_GermanGovservice"
an "avocado_price_service"
an "Durban"
an "_Toyotaprice_service"
an "HDP2_service"
an "countrycapital-city_hotel_service"
an "tea_taxedprice_service"
an "4wheeledcar_technology_service"
an "_book_Oracleservice"
.
.
.
.
ae "activity_nationalpark_service16H00" "activity_nationalpark_service"  "16H00"
ae  "AcceptCostAndHealingPlan_serviceCape  Town"  "AcceptCostAndHealingPlan_service"
"Cape Town"
ae "employee_postal-address_XYZServiceEmpangeni" "employee_postal-address_XYZService"
"Empangeni"
ae                                          "award_lendingduration_GermanGovservice16H00"
"award_lendingduration_GermanGovservice"  "16H00"
ae              "geopolitical-entity_occupationaltrade_service16H00"              "geopolitical-
entity_occupationaltrade_service"  "16H00"
ae "_price_FishservicePretoria" "_price_Fishservice"  "Pretoria"
ae "sports_beach_servicePretoria" "sports_beach_service"  "Pretoria"
ae               "academic-degreegovernment_funding_serviceDurban"               "academic-
degreegovernment_funding_service"  "Durban"
ae       "maxprice_whiskeycolabeer_servicePretoria"       "maxprice_whiskeycolabeer_service"
"Pretoria"
ae "_polling_GermanserviceCape Town" "_polling_Germanservice"  "Cape Town"
ae  "dvdplayermp3player_price_MDservice08H00"  "dvdplayermp3player_price_MDservice"
"08H00"
ae "AuthorizePhysician_servicePolokwane" "AuthorizePhysician_service"  "Polokwane"
ae "coffee_recommendedprice_service12H00" "coffee_recommendedprice_service"  "12H00"
ae "maxprice_liquid_serviceJohannesburg" "maxprice_liquid_service"  "Johannesburg"
ae"mp3playerdvdplayer_Recpriceshipping_service16H00"
"mp3playerdvdplayer_Recpriceshipping_service"  "16H00"
```

Figure 5-2: Representation of nodes and edges in CASDA

Service Manager

The Service Manager consists of a Repository which stores Semantic service descriptions of the published services. These descriptions can be generated dynamically during the service publication and they are used in the Semantic Matchmaking phase of the CASDA.

5.2.2 Sequence Diagram Modeling

The Sequence diagram in Figure 5-3 shows the sequence of the flow of messages among the components and actors in CASDA. The flow of messages is initiated by the requester or the user while registering its details or requesting for services. When the requester initiates a discovery process the Context Manager handles all the relevant requester context updates. The main flow of messages that involves all the components is when the requester queries for services. The Requester submits user context into the Matchmaker which then queries the service contexts from the Context Manager. After receiving the service contexts from the Context Manager the Matchmaker matches received contexts syntactically against the context of the requester. The Matchmaker then calls the Request Manager to formulate a semantic request based on the user's context. The Matchmaker then forwards a list of service from the Syntactic Matchmaking phase to the Service Manager which responds by sending back the Semantic Service description of the listed services.

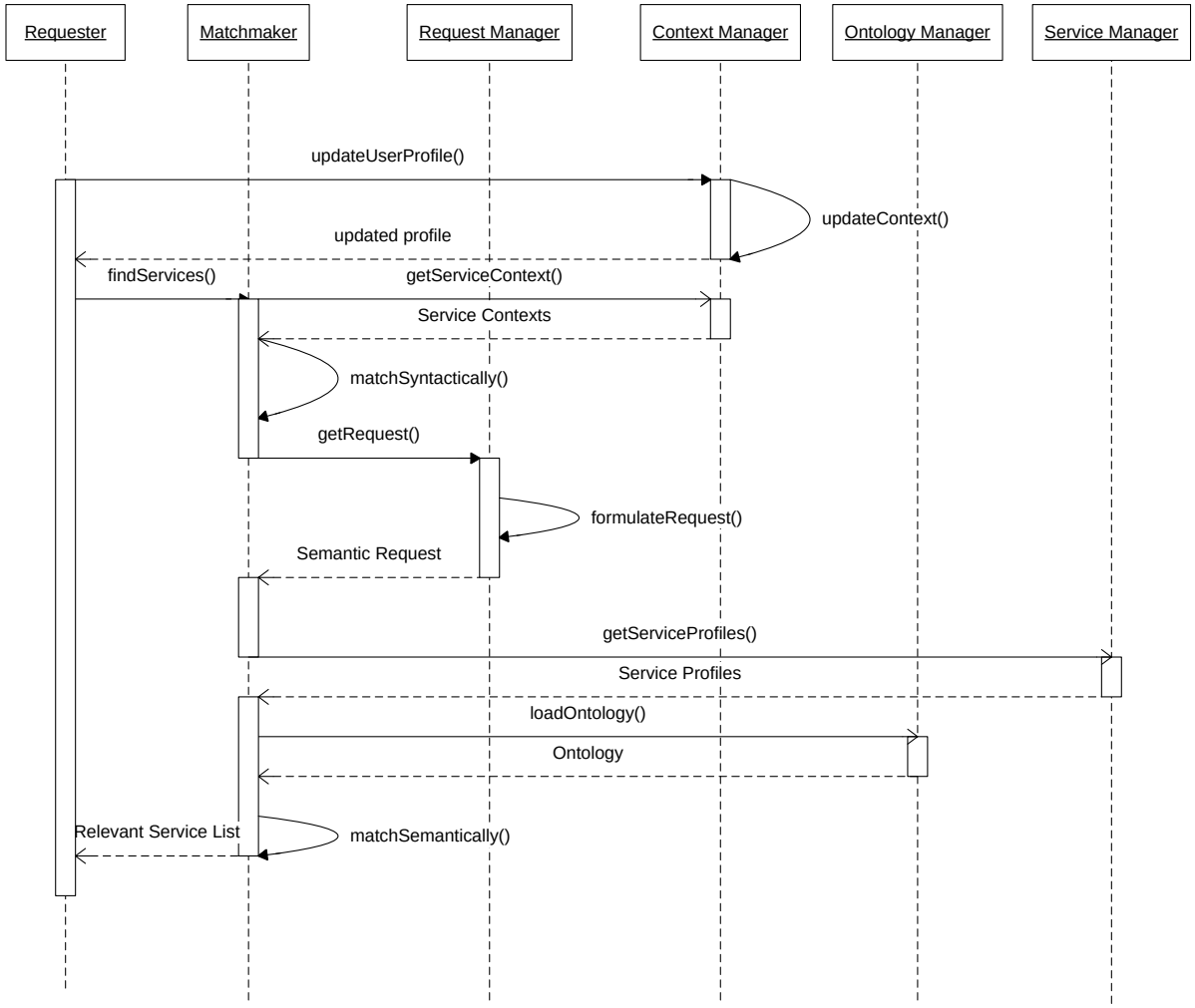


Figure 5-3: CASDA Sequence Diagram

The matchmaker finally loads the ontologies from the Ontology Manager which it uses to reason when performing the semantic matchmaking. After successfully performing the Semantic matchmaking phase the list of all the relevant services are returned to the requester.

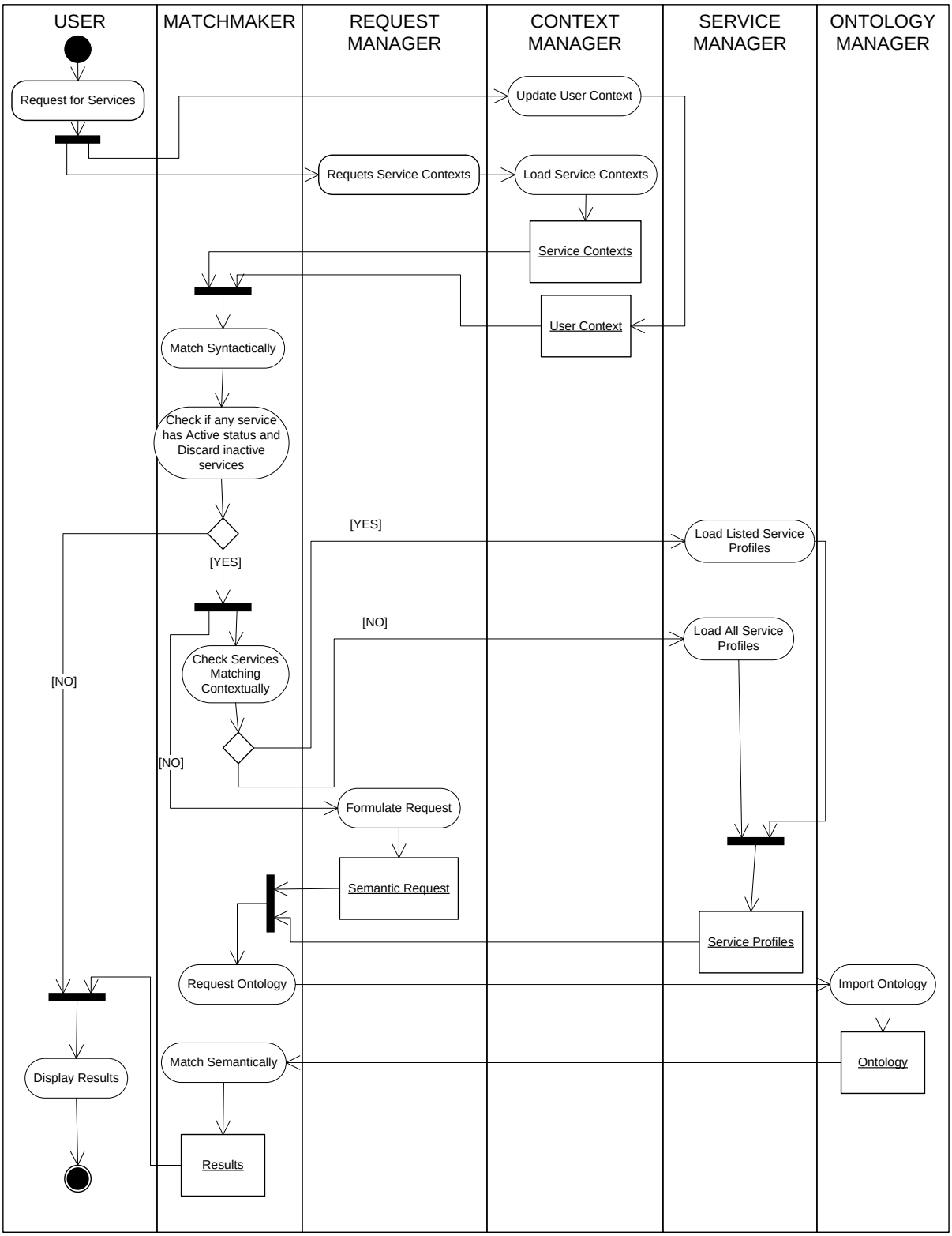


Figure 5-4: CASDA Activity Diagram

5.2.3 Activity Diagram

Figure 5-4 shows the Activity diagram of our service discovery architecture. The process starts with the user submitting a request to the request manager while updating the user profile and context. The context information is stored by the context manager in a graph model. The request manager then instructs the context manager to also load the service contexts to the matchmaker. The service context is also stored as a graphical model within the context manager. The combined graph which consists of the user node, service nodes and the context atoms together with the edges associating the user and services with those context atoms are loaded to the matchmaker. The matchmaker then matches the user versus the services contexts using the context querying algorithm described in Section 4.4.

In the context manager every change in state of a service is updated dynamically so that non functional services at the time of request can be easily identified and discarded. Therefore, after matching the services against the user or requester contextually, the matching services are then checked whether they are functional or not. If none of the services is functional at the time of request the system exits. This happens because there will be no published service that can serve the goal of the requester at the time of request.

All the service profiles of the services that match the requester contextually are loaded to the semantic matchmaker from the service manager. If there are no services matching the requester contextually, then the matchmaker loads the semantic profiles of all the services that listed. The semantic profiles are stored in a repository in the Service Manager. After loading the semantic profiles the ontology, which will be used in the semantic matchmaking process, is verified and imported if necessary. The matchmaker also simultaneously gets the semantic request profile from the request manager. The semantic request profile is generated

by the request manager from the service inputs/outputs specified by the requester. The matchmaker then matches the services against the semantic request using the algorithm by Klusch et. al. (2006) as described in details in Section 4.5. The results are then returned to the requester ranked according to their matching degrees.

5.3 CASDA Implementation

Chapter 4 presented the development of CASDA. Section 5.2 presented the prototype design. This section therefore presents the implementation and the evaluation of CASDA model. CASDA was implemented on a Microsoft Vista environment with 2GB of Ram and 1.67 GHZ processor. The tools used in the development process was Netbeans IDE 7, JDK 1.6, JWSDP 1.5 which comes bundled with the UDDI registry, XAMP HTTP server to publish the ontologies locally. The following subsections present the implementation details of CASDA structured into Service description in Section 5.3.1; Context modeling in Section 5.3.2; Context matchmaking in Section 5.3.3 and the Semantic Matchmaker in Section 5.3.4.

5.3.1 Service description

Services in CASDA are described using OWL-S as shown in Figure 5-5. Service descriptions are published in a repository. All nodes representing service in the context manager have corresponding OWL-S files in the Service Manager's repository. These OWL-S files are loaded into the matchmaker as soon as the semantic matchmaker is called. From Figure 5-5 it can be seen that the ontologies that require being imported are stored locally. Within the service description only the profile elements are visible as they are the only part used during the semantic matchmaking process. The ontologies were obtained from http://www.daml.org/ontologies. Figure 5-5 represents one sample service from the repository and is identified as *APPLE_PRICE_SERVICE*. It takes *_APPLE* as an input and it

gives *_PRICE* as an output. They inputs and outputs of the services are only used by the Hybrid OWLS-MX matchmaker as the pure syntactic matchmaking phase of CASDA matches the non-functional contextual properties of the services.

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF  xmlns:owl       = "http://www.w3.org/2002/07/owl#"
xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:service   = "http://www.daml.org/services/owl-s/1.1/Service.owl#"
xmlns:process   = "http://www.daml.org/services/owl-s/1.1/Process.owl#"
xmlns:profile    = "http://www.daml.org/services/owl-s/1.1/Profile.owl#"
xmlns:grounding = "http://www.daml.org/services/owl-s/1.1/Grounding.owl#"

xml:base        = "http://127.0.0.1/services/1.1/apple_price_service.owls">

<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://127.0.0.1/ontology/Service.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Process.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Profile.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Grounding.owl" />
<owl:imports    rdf:resource="http://127.0.0.1/ontology/Mid-level-ontology.owl"
/>
<owl:imports rdf:resource="http://127.0.0.1/ontology/concept.owl" />
</owl:Ontology>

<service:Service rdf:ID="APPLE_PRICE_SERVICE">
<service:presents rdf:resource="#APPLE_PRICE_PROFILE"/>
<service:describedBy rdf:resource="#APPLE_PRICE_PROCESS_MODEL"/>
<service:supports rdf:resource="#APPLE_PRICE_GROUNDING"/>
</service:Service>

<profile:Profile rdf:ID="APPLE_PRICE_PROFILE">
<service:isPresentedBy rdf:resource="#APPLE_PRICE_SERVICE"/>
<profile:serviceName xml:lang="en">
ApplePriceService
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service informs the price of certain apple.
</profile:textDescription>
<profile:hasInput  rdf:resource="#_APPLE"/>
<profile:hasOutput rdf:resource="#_PRICE"/>

<profile:has_process rdf:resource="APPLE_PRICE_PROCESS" /></profile:Profile>

.
.
.

</rdf:RDF>
```

Figure 5-5: OWL-S Service Description (from Owl-s test collections by Klusch et al., 2006)

5.3.2 Context Modeling

For illustrations purposes in our work the context type considered are Time and Location only The location and time are used by the matchmaker to infer the context domain under

which the requester exists. A context domain in this case is a combination of location and time e.g. in Cape Town at 14H00. The context model which covers context for both the services and requesters is implemented using graphical object model using GraphStream (Dutot et. al., 2007). At publishing time the services are entered into the graph as nodes with their service id's or names as labels of those nodes. The service nodes are then associated with context atoms which are also represented as nodes within the graph.

5.3.3 Context Matching Algorithm

```
public Set findServices(String location, String time) throws IOException, GraphParseException{

        Set set1 = new HashSet();
        Set set2 = new HashSet();

        String nodeName;
        g = new DefaultGraph();
        GraphReader reader =
GraphReaderFactory.readerFor("C:/Users/Boffin/Desktop/file.dgs");
        GraphReaderListenerHelper helper = new GraphReaderListenerHelper(g);
        reader.addGraphReaderListener(helper);
        reader.read("C:/Users/Boffin/Desktop/file.dgs");

        Node n1 = g.getNode(location);
        Node n2 = g.getNode(time);

        Iterator<? extends Node> iter = n1.getNeighborNodeIterator();

        while (iter.hasNext()) {
           Node m = iter.next();
           set1.add(m.getId());
        }

        Iterator<? extends Node> n2Neighbours = n2.getNeighborNodeIterator();

        while (n2Neighbours.hasNext()) {
           Node m = n2Neighbours.next();
           set2.add(m.getId());
        }

        Set result = SetOperations.intersection(set1, set2);
        ArrayList syntacticRes = new ArrayList(result);
        System.out.println(result);
        sm.matchServices(syntacticRes);
        return result;
    }
```

Figure 5-6: Context Matching Algorithm

The code fragment in Figure 5-6 is part of the pure syntactic matchmaking algorithm that combines the algorithms presented in Section 4.4.1 and Section 4.4.2. The context matching algorithm in Figure 5-6 takes as an input, time and location. The algorithm makes use of the Breadth-first search algorithm introduced in Section 4.3.1 is embedded in GraphStream. The algorithm then takes the location ID or Label specified in the request determines its nearest neighbors. The nearest neighbors to the location node are service nodes. The location nodes and service nodes neighborhood are updated dynamically as mobile services change locations. The nearest neighbors of the specified time are also determined. The nearest neighbors of the specified location are stored in *set1* and the nearest neighbors of the specified time are stored in *set2*. An intersection of the two sets gives a list of service available at that specified location and time.

The results obtained from this step are a list of service ID's. This list is used to determine the service profiles from the repository that will be loaded into the semantic matchmaker component.

The implemented graph based context model is flexible and more context types can be added in the graph in addition to current *Location* and *Time* context types. The matching could still be performed on the nearest neighborhood of services to the specified contexts.

5.3.4 Semantic Matchmaker

The matchmaker is implemented with two phases or levels of matching. The first phase is restricted to syntactic matching while the second phase is performing the semantic matching. The OWLS-MX 2.0 comes in as a readymade solution for CASDA's second matchmaking and semantic phase. The code fragment in Figure 5-7 presents part of the algorithm as presented in Section 4.5.1.

```
/**
* Performs purely semantic matching
* Basically it performs a intersection of the candidates for input- and output concepts
* @param inConcepts Vector of URIs of input concepts of the request
* @param outConcepts Vector of URIs of output concepts of the request
* @return          SortedSet of relevant services
*/
protected SortedSet semanticMatch(Vector inConcepts, Vector outConcepts) throws
URISyntaxException, MatchingException {
      Map input = getInputCandidates(inConcepts);
          if ( (outConcepts==null) || (outConcepts.size()<=0) )
                  return inputCandidatesToResult(input);
          Map output       =          getOutputCandidates(outConcepts);

      SortedSet result = new TreeSet();
      Map.Entry me;
      Integer ID;
      ExtendedServiceInformation inInfo,outInfo;
      DOM degree;
      Iterator iter = input.entrySet().iterator();
      while (iter.hasNext()) {
         me = (Map.Entry) iter.next();
         ID = (Integer)me.getKey();
         if (output.containsKey(ID)) {
            inInfo=(ExtendedServiceInformation) me.getValue();
            degree = ((DOM) output.get(ID));
            outInfo=degree.getBestDegree();
            if ( (inInfo.degreeOfMatch>outInfo.degreeOfMatch) ||
              ( (inInfo.degreeOfMatch==outInfo.degreeOfMatch) &&
               (inInfo.similarity<outInfo.similarity) ) )
               result.add(new MatchingResult(inInfo , inInfo.unfoldedconcept,
outInfo.unfoldedconcept) );
            else
              result.add(new MatchingResult(outInfo , inInfo.unfoldedconcept,
outInfo.unfoldedconcept) );
         }
      }
      return result;
   }
```

Figure 5-7: Semantic Profile Matchmaker (DFKI GmbH, Germany, 2005)

It performs the matchmaking on the semantic request created by the Request Manager against the OWL-S service descriptions loaded after the syntactic matchmaking phase has been concluded.

5.4 Model Evaluation

To evaluate our model we used the OWL-S Service Retrieval Test Collection by Khalid and Kapahnke, (2008). We chose the OWL-S 1.1 services with a total of 1007 services and 29 requests with each request having a predefined relevant set from the 1007 services. The experiments that were carried out aimed to investigate the performance metrics which are scalability, recall and precision. These metrics were introduced in Section 2.6.1 and Section 2.6.2. The interface used to test our solution approach is shown in Figure 5-8.

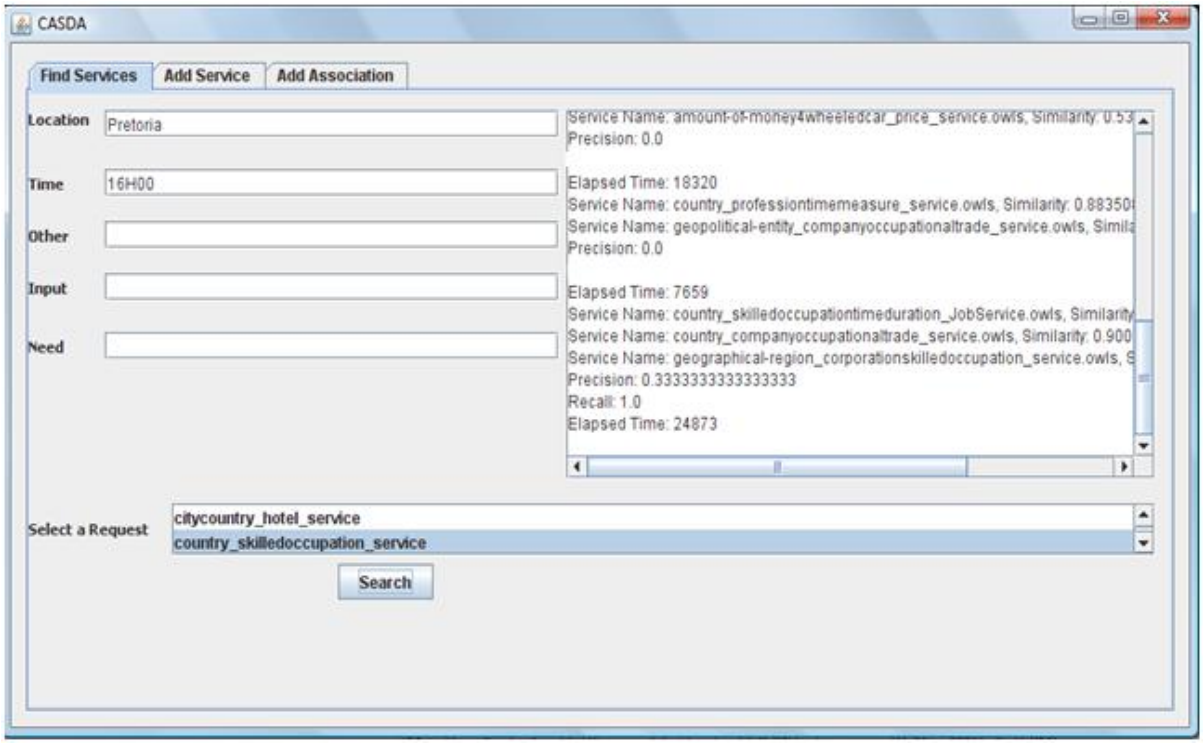


Figure 5-8: CASDA Evaluation Window

5.4.1 Scalability of our Solution Approach

The section presents the experimental design and the evaluation results of CASDA based on scalability.

5.4.1.1 Experimental Design

For testing purposes in the context model we use Time and Location. There are 6 locations and 5 different times of a day which result into 30 different contexts. The 1007 service are then randomly allocated in each context. OWLS-MX as discussed in Section 4.5.2 is hybrid matchmaker which combines both syntactic and logic based matchmaking. OWLS-MX consists of six variants and one of them is the cosine similarity which we use in the semantic matchmaking phase of our CASDA. OWLS-MX variant is set to its lowest matching degree. The matching degree is set to its lowest to allow it to retrieve a maximum number of services during the execution of the requests. Each request is then executed 30 times with each of the 30 location-request combinations. The purpose of this test is to observe the processing time against the numbers of services to be matched. The published services are published in sets of 200, 400, 600, 800 and 1000 for each test. A similar procedure is carried out with OWLS-MX using the same cosine similarity variant used in CASDA. In OWLS-MX case each request is executed five times for each set of services and the execution time recorded.

5.4.1.2 Test Results

Even though both OWLS-MX and CASDA show good results while investigating scalability as in Figure 5-9 CASDA has a better performance in terms of the time it takes to process requests. OWLS-MX starts at about 100 000 milliseconds to about 700000 milliseconds for 1000 services matched against one request and this averages to 750 milliseconds per service published. On the other hand, CASDA starts at about 5000 milliseconds for 200 services and ends at about 11000 milliseconds for 1000 service and these averages to 7.5 milliseconds per service published which is 100 times that of OWLS-MX.

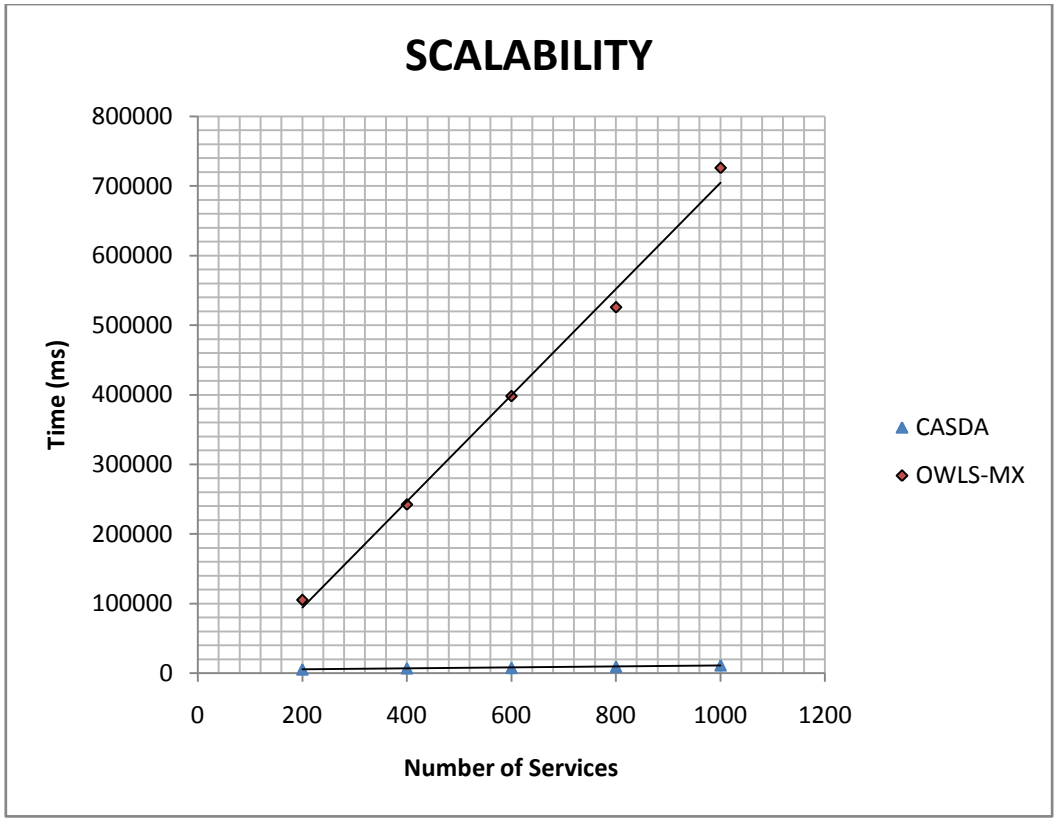


Figure 5-9: CASDA and OWLS-MX Scalability

The main contributing factor to this is that CASDA relieves the semantic matchmaker which utilizes process intensive ontology by filtering the published services using context. CASDA allows a pure syntactic similarity matching at the context matching level which is not as compute intensive as the semantic matching. At the context matching phase the services are filtered and only those that match the requester contextually are passed on to the semantic matchmaker.

5.4.2 Precision-Recall evaluation

In this section we present the experimental procedures performed to evaluate CASDA prototype implementation based on scalability and recall.

5.4.2.1 Experimental Design

The experimental setup is the same as done when evaluating the scalability in Section 5.4.1.1. For recall and precision experimentation, the services are classified as relevant and irrelevant for each of the requests. The number of relevant services is kept constant for every time test set size is varied in the order of 200 services. The change in the number of services in Figure 5-10 and Figure 5-11 thus changes the ratio of the number of relevant services to the total number of published services. In our experiment precision and recall are dependent on this ration (Cormack et. al., 1998; Gunawardana and Shani, 2009). Precision and recall are measured using the equations:

$$precision = \frac{number\ \ of\ relevant\ \ and\ returned\ \ services}{number\ \ of\ returned\ \ services}$$

And

$$recall = \frac{number\ \ of\ relevant\ \ and\ returned\ \ services}{number\ \ of\ publis\ hed\ relevant\ \ services}$$

These metrics are measured against the number of services published.

5.4.2.2 Experimental Results

Figure 5-10 shows there is no improvement on precision in CASDA compared to OWLS-MX. From Figure 5-11 though it can be seen that there is an improvement in recall which shows precision is traded in CASDA. CASDA is at a lower range of Precision compared to OWLS-MX but, the cut off Recall is at 90% compared to that of OWLS-MX which is at 70%.

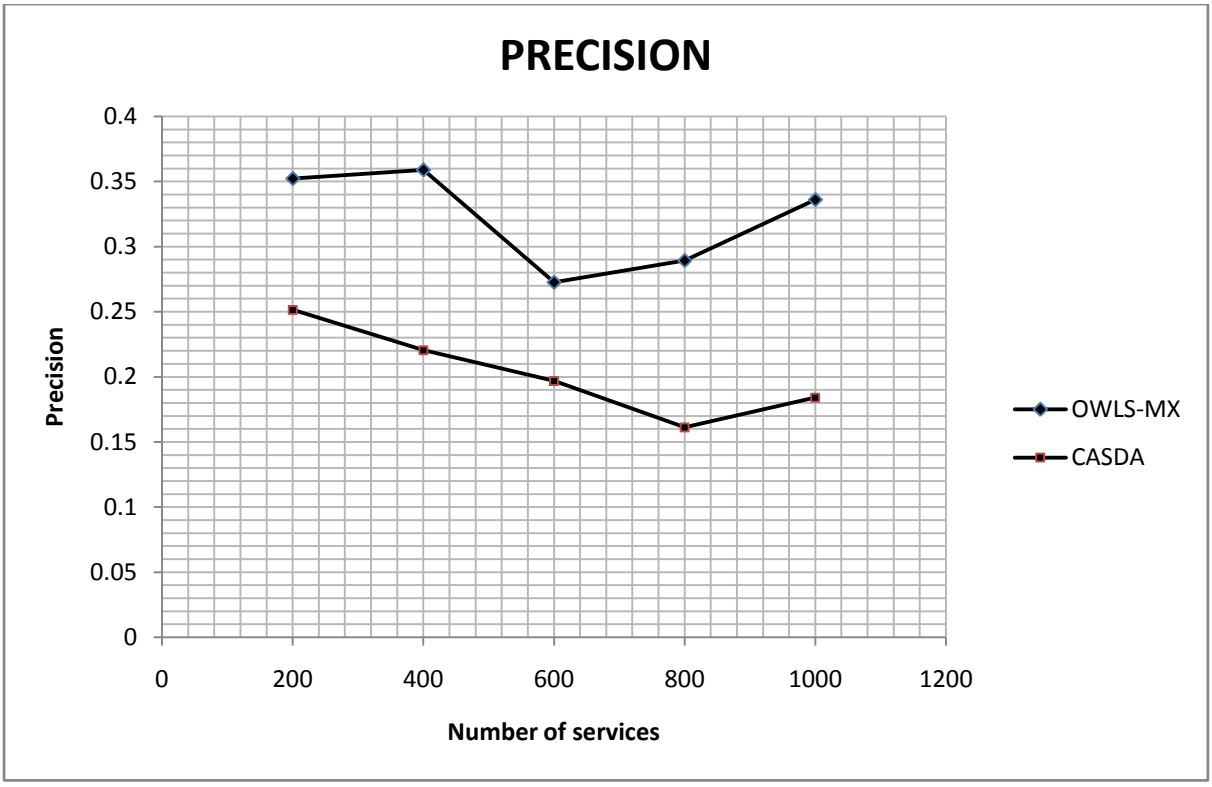


Figure 5-10: CASDA and OWLS-MX Precision

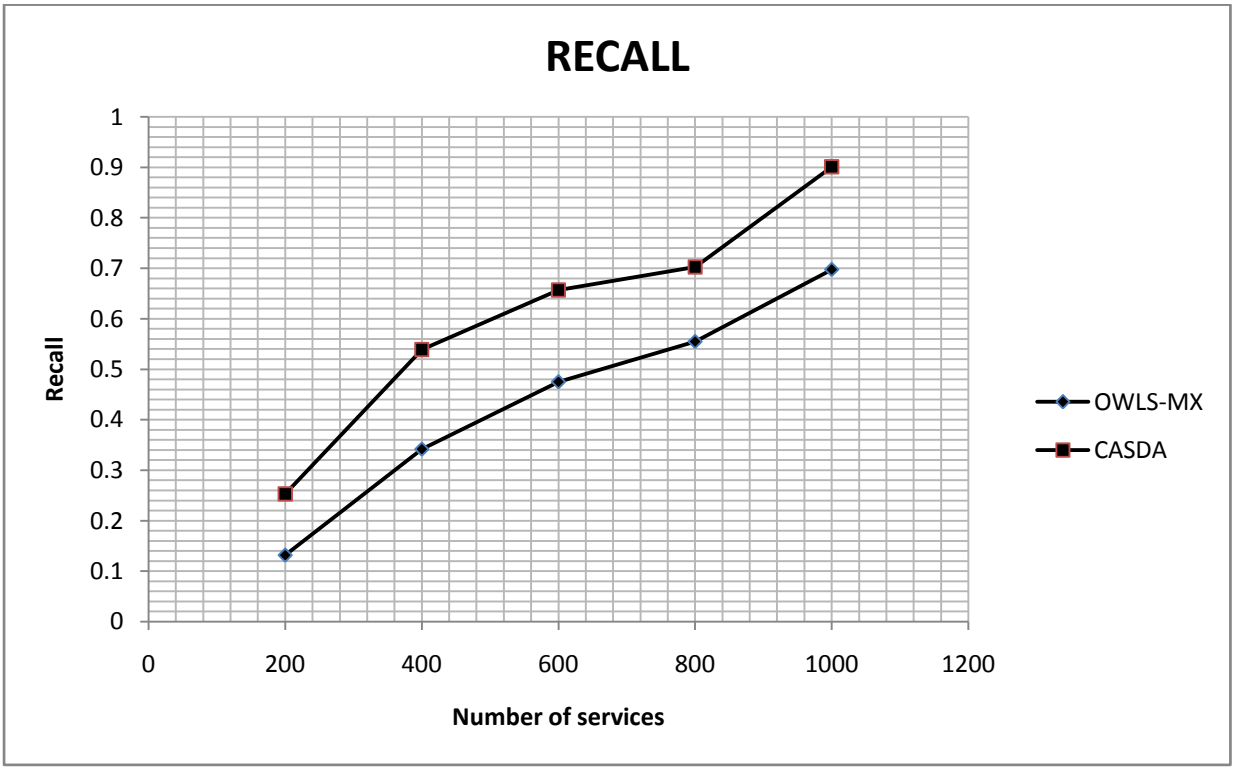


Figure 5-11: CASDA and OWLS-MX Recall

CASDA is outperformed by OWLS-MX when the evaluation is based on precision-recall. CASDA is at precision range lower than that of OWLS-MX with a maximum at 25%. On recall though CASDA is higher than OWLS-MX as CASDA starts at 25% and the cut off is at 90% where as OWLS-MX starts at 13% and cuts off at 70%.

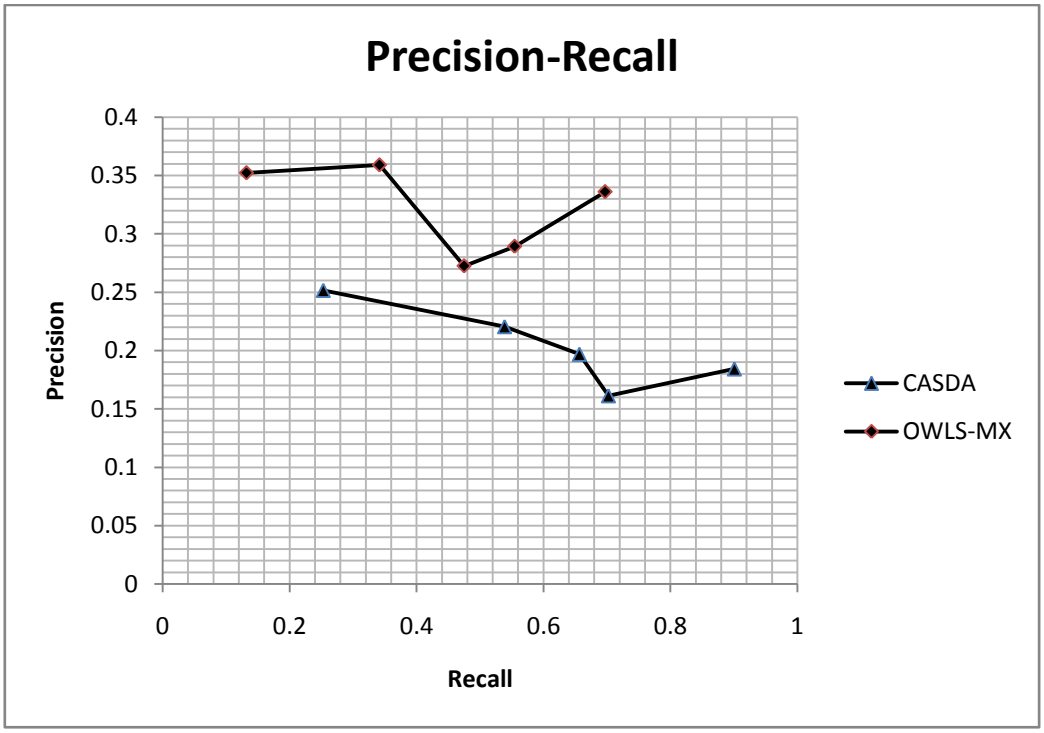


Figure 5-12: Precision-Recall for OWLS-MX and CASDA

5.5 Discussion

Both CASDA and OWLS-MX show an improvement in terms of scalability. From Figure 5.9, it can be seen that CASDA outperforms OWLS-MX in terms of the time it takes to process the requests (Latency). The contributing factor is that CASDA eliminates services that do not match the requester context before they matched semantically. In Figure 5-10

there is an improvement in CASDA compared to OWLS-MX in terms of recall. In Figure 5-11, however, it can be seen that CASDA improved at the expense of recall.

It is worth noting that the quantitative analysis results of CASDA and OWLS-MX presented are obtained in a limited environment. The ontologies used are the ones managed locally. Also for demonstration purposes there are only two context attributes considered. In practice heterogeneous ontologies, more context attributes are considered. Also the number of advertised services will be very high. Hence the difference between the performances of CASDA against OWLS-MX will not be as high as obtained above. It is pleasing though to note that as the number of services grows in Figure 5-9, CASDA remains at a low level in terms of request processing time.

5.6 Conclusion

It this chapter we have presented the design, implementation and evaluation of our solution approach which addresses the issue of service discovery in Service Oriented environments. The prototype implemented and presented in this chapter was aimed to be used at investigating the efficiency of the matchmaking approach proposed in CASDA. CASDA builds upon the capabilities of semantic service matchmakers such as OWLS-MX in enhancing service discovery. The semantic matchmakers are aimed at addressing the issues of service discovery found in traditional syntax based discovery protocols such as UDDI. OWLS-MX is one of the semantic matchmakers and it semantically matches services against a request based on their functional properties. CASDA takes context into account while matching services and context matching is limited to pure syntactic matchmaking. The use of context is aimed at delivering the most appropriate services to service requesters. In this chapter therefore CASDA has been evaluated based on scalability, precision and recall. The results have then been compared to those of OWLS-MX, which does not take context into

account. Based on the results which have been presented in this chapter, in Chapter 6 we present the overall conclusion to this study.

CHAPTER 6

SUMMARY, CONCLUSION AND FUTURE WORK

6.1 Introduction

In the first chapter of this dissertation we presented the service discovery problem which this work is addressing. The purpose of this research work was to enhance the service discovery process in a services oriented environment. The enhancement of service discovery process was to be achieved by providing a discovery mechanism that utilizes context. A solution approach is then proposed in a form of architecture, which was then prototyped and evaluated. Our solution approach provides architecture and a matchmaking approach which together enable an enhanced service discovery process. The proposed architecture provides an environment through which essential components can interact for an enhanced service discovery process. The matchmaking algorithm maintains the quality of discovered results without trading-off scalability. The matchmaker algorithm consists of two phases which are, context matching phase and the semantic profile matching phase. The context matching phase is purely syntactic while the second service profile matching phase is semantic. This is done to allow the computation intensive ontology resources to be utilized in only one phase and in that way scalability is not traded-off.

This chapter summarizes and concludes the work done in this research. The remainder of this chapter is arranged as follows: Section 6.2 summarizes this research work and presents the conclusions drawn from this research. Section 6.3 then presents the future work.

6.2 Summary and Conclusions

Services are created and published to be consumed. Before a client can consume a service, the service needs to be discovered first. If it cannot be discovered the service is as good as not being there. Hence, service providers make their services known by publishing them in registries such as the Universal Description, Discovery and Integration (UDDI). Service requesters discover these services from registries by explicitly submitting service requests. Services can also be implicitly discovered depending on the discovery mechanism used.

Our investigation through the course of this research has shown that service discovery is still an issue in SOA. Traditional service discovery mechanisms such as the UDDI use syntactic matchmaking techniques which result in poor results. This means that there are services that are discarded in the service discovery process which are capable of fulfilling the requester's needs. On the other hand there are services that are returned but which cannot fulfill the purpose of the requester. These are the so-called false negatives and false positives whose occurrences are high in traditional discovery mechanisms.

Most solution approaches that attempt to address the challenges in traditional service discovery mechanisms are purely semantic. They consider the semantic similarities of the requests and the service descriptions rather than the syntactic similarities. The disadvantage with semantic approaches is that they cause overhead in the discovery process. This is so because they make use of ontology reasoners which are computationally expensive and require a lot of memory. This eventually results into delays which might frustrate service requesters and lead them to abandon their discovery processes without their goals being fulfilled.

There are other solution approaches which do consider both syntax and semantics in service discovery but, unfortunately they leave out context. In this research we advocate that context

plays a major role in fulfilling the needs of the requester while performing the discovery process. A service might match a request either syntactically or semantically and hence, be capable of fulfilling the needs of the requester. But if the context is wrong, for example if the service requires that the requester have a video streaming capable device for it to run properly, then without such a device the service is useless to the requester. In such a scenario there is a waste of resources and time on the side of the requester. If context had been taken into account in such a discovery, then costs could have been eliminated. Ensuring that services that do not match contextually are not returned to the requester results in saving of resources and time.

This research addressed the issue of service discovery in SOA by proposing a CASDA whose contribution is in the following aspects:

a) The discovery architecture and
b) matchmaking approach

CASDA stands out from the solution approaches considered in the literature because:

a) it offers a flexible context model which is suitable for the volatile and dynamic grid environments
b) it offers a mechanism in which services are filtered contextually before they are matched semantically and hence scalability is improved considerably

The architecture proposed in our work comprises of the following components: a *Request Manager, Service Manager, Context Manager, Matchmaker* and the *Ontology Manager*. The proposition and integration of these components was informed by the following design considerations which were addressed:

Efficiency of the Matchmaking Process: the services and requester context are managed separately by the *context manger* which updates those contexts accordingly as they change dynamically. This eases the matchmaker from constructing new contexts every time a request is submitted. Hence unnecessary delays in the discovery process are avoided.

Robustness against internet failure: in cases of weak connections, disconnections may happen frequently while the discovery process is in progress. As the matchmaker might need to import ontologies through the internet, the discovery process might be disturbed by the disconnections. Recovery of the discovery process is facilitated by the ontology manager which locally stores the ontologies that were already imported, while there was internet connection.

Registry Independence: Our architecture is designed to coexist with the traditional XML registries such as the UDDI and the ebXml. The services manager handles the configurations of a new registry.

Our solution approach addresses the lack of scalability found in pure semantic matchmakers, the high rate of false positives and false negatives found in pure syntactic matchmakers and lack of context utilization. These issues were addressed in our work as follows:

The scalability issue which results from employing semantics is addressed by limiting the context matchmaking to pure syntactic matching. The issue of the high rate of false positives and false negatives was addressed by taking context into consideration and employing semantics in the discovery process. The issue of the lack of context awareness was addressed by introducing a context model which represented the entities interacting with the matchmaker in a bipartite graph. In a dynamic environment the contexts of the entities involved change continuously and hence there was a need for a flexible model that can capture new contexts as well as context types as they change. The graphical model was

chosen to serve this purpose due to its flexibility. The entities represented in the graph, such as the requester and the services, are easily matched syntactically by considering the context atoms that they are associated with.

The tests conducted on CASDA showed an improvement compared to OWLS-MX (Klusch, M. et. al., 2006) in terms of *scalability*. Scalability was but one aspect of enhancing the discovery process in a Service Oriented Environment. In spite of the results showing an improvement in scalability, the other aspect which is *precision* dropped considerably on CASDA when compared to OWLS-MX. This drop is attributed to the fact that services relevant to each request are distributed randomly across the thirty contexts used for testing. The proportion of the relevant services in each context domain is, therefore, smaller and, for other requests there are no relevant services in some of the domains and hence a request executing in that domain obtains results which have a higher recall but smaller precision. It is, therefore, important to redefine the relevance of the test services to the requests with context taken into account in the test data and that is beyond the scope of our work.

6.3 Limitations and Future Work

For the context model in our experiments for CASDA we only considered Time and Location for demonstration purposes. Also, locations were only specified in symbolic names. There is a need to specify them as positional coordinates. A location can also be known by more than one symbolic name e.g. Pretoria is also known as Tshwane. In practice there are more context attributes which may be considered and these depend on the application level. In the light of the above, it is clear that there is still a need to capture semantics to some extent without trading off scalability. There is also a need to consider models with additional features. This research work made the following assumptions:

i) There are no trust and security issues as the service providers are already subscribed to the infrastructure.

ii) The service requests are already submitted and ready to be executed

For future work, therefore, it is important to also consider the trust and security issues among the participants which are service providers and service consumers in the infrastructure. There is also a need to investigate an efficient way of dynamically generating service requests that will capture the service requester preferences.

REFERENCES

[1]. Adigun, M.O, Emuoyibofarhe O. J., and Migiro, S.O., (2006). "Challenges to Access and Opportunity to use SMME enabling Technologies in Africa", a presentation at 1st All Africa Technology Diffusion Conference, June 12-14, 2006, Johannesburg South Africa.

[2]. Aggarwal, R. (2004). Semantic Web Services Languages and Technologies: Comparison and Discussion.

[3]. Akkiraju, R., Farrell J., Miller, J., Nagarajan M., Schmidt M., Sheth A., and Verma K., (2005)."Web Service Semantics - WSDL-S," A joint UGA-IBM Technical Note, version 1.0, April 18, 2005. http://lsdis.cs.uga.edu/projects/METEOR-S/WSDL-S.

[4]. Anagnostopoulos, C., Tsetsos, V. and Hadjiefthymiades, S. (2006). On the evaluation of semantic web service matchmaking systems. ECOWS '06: Proceedings of the European Conference on Web Services, IEEE Computer Society, pages 255–264.

[5]. Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., McDermott, Martin D., McIlraith S. A., Narayanan S., Paolucci M., Payne T. and Sycara K. (2003) DAML-S: Semantic Markup for We Services, 2003. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.6236&rep=rep1&type=pdf

[6]. Balzer, S., Liebig T. and Wagner, M. (2004). Pitfalls of OWLS–A Practical Semantic Web Use Case, ICSOC'04, November 15–19, 2004, New York, New York, USA.

[7]. Barrero, D. F., López, D. R. and Población, Ó. G. (2004). Distributed metainformation searching: an approach to information retrieval in the age of the semantic web. Selected Papers from the TERENA Networking Conference (2004).

[8]. Beck, M. and Freitag, B. (2006). Semantic matchmaking using ranked instance retrieval. SMR '06: Proceedings of the 1st International Workshop on Semantic Matchmaking and Resource Retrieval, Co-located with VLDB, 178 of CEUR Workshop Proceedings, 2006.

[9]. Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N. and Surridge, M. (2005). OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition, Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05), IEEE,.

[10]. Bellur, U. and Kulkarni, R. (2007). Improved matchmaking algorithm for semantic web services based on bipartite graph matching. ICWS 2007. IEEE International Conference on web services, 2007.

[11]. Bellur, U. Vadodaria, H. and Gupta A. (2008). Semantic Matchmaking Algorithms. Advances in Greedy Algorithms, Book edited by: Witold Bednorz, ISBN 978-953-7619-27-5, pp. 586, November 2008, I-Tech, Vienna, Austria.

[12]. Bocchi, L. Ciancarini, P. Moretti, R. Presutti, V. and Rossi, D. (2005). *An OWLS Based Approach to Express Grid Services Coordination.* ACM Symposium on Applied Computing. SAC '05 March 13-17, 2005, Santa Fe, New Mexico, USA

[13]. Campo, C. Munoz, M. Perea, J. C. Marın, A. and Garcıa-Rubio, C. (2005). PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems," Proc. 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005 Workshops), Kauai Island, Hawaii.

[14]. Capilla, R. (2006). Context-aware Architectures for Building Service-Oriented Systems. Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06).

[15]. Chakraborty, D., Joshi, A., Yesha, Y., and Finin, T., (2002). "GSD: A Novel Group-Based Service Discovery Protocol for Manets," Proc. 4th IEEE Conf. Mobile and Wireless Communications Networks (MWCN 02), IEEE Press, 2002, pp. 140–144.

[16]. Chen, G., Li, M., and Kotz, D., (2008)., "Data-centric middleware for context-aware pervasive computing", Pervasive and Mobile Computing 4(2), ELSEVIER.

[17]. Cheng, L. and Marsic, I. (2000). "Service Discovery and Invocation for Mobile Ad Hoc Networked Appliances," Proc. 2nd Int'l Workshop Networked Appliances (IWNA 00), Dec. 2000; www.cse.lehigh.edu/~cheng/papers/ iwna-manet.pdf.

[18]. Clement, L., Hately, A., Von Riegen, C. and Rogers, T. (2008). UDDI Version 3.0.2 UDDI Spec Technical Committee Draft, Dated 20041019, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm. Last date accessed: 1 June 2008

[19]. Cormack, G.V., Palmer, C.R., Van Biesbrouck, M., Clarke, C.L.A., (1998). Deriving Very Short Queries for High Precision and Recall, MultiText Experiments for TREC-7.

[20]. da Rocha R.C.A. and Endler, M. (2006). Supporting Context-Aware Applications: Scenarios, Models and Architecture. Assessoria de Biblioteca, Documenta¸c˜ao e Informa¸c˜ao PUC-Rio Departamento de Inform´atica Rua Marquˆes de S˜ao Vicente, 225 - G´avea 22453-900 Rio de Janeiro RJ Brasil.

[21]. Dawes, A. M. (2009). http://www.math.uwo.ca/~mdawes/courses/344/kuhn-munkres.pdf, last accessed Septe mber 11, 2009.

[22]. Dey, A.K., Abowd, G.D. and Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. HUMAN-COMPUTER INTERACTION, 2001, Volume 16, pp. 97–166, Lawrence Erlbaum Associates, Inc.

[23]. Donini, F. M., Di Noia, T., Di Sciascio, E., and Mongiello, M. (2003). Semantic matchmaking in a p2p electronic marketplace. SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, pages 582–586, 2003.

[24]. Doulkeridis, C., and Vazirgiannis, M. (2004). Querying and Updating a Context-Aware Service Directory in Mobile Environments. Web Intelligence 2004: 562-565

[25]. Doulkeridis, C., Loutas N. and Vazirgiannis M. (2006). "A System Architecture for Context-Aware Service Discovery" in: Electronic Notes in Theoretical Computer Science 146 pp. 101-106.

[26]. Dutot A., Guinand, F., Olivier, D.and Pigné, Y. (2007). Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In EPNACS: Emergent Properties in Natural and Artificial Complex Systems, 2007.

[27]. Easterbrook, S., Singer, J., Storey, M. and Damian, D. (2007). Selecting Empirical Methods for Software Engineering Research,2007.

[28]. Engelstad, P.E. and Zheng, Y., (2005). "Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks," Proc. 2nd annual conference on Wireless On-demand Networks and Services (WONS 2005), St.Moritz, Switzerland.

[29]. Fox, G. (2003). "Data and metadata on the semantic Grid" Computing in Science & EngineeringVolume 5, Issue 5, Sept.-Oct. 2003 Page(s): 76 – 78 DOI 10.1109/MCISE.2003.1225865.

[30]. Friesen, A., and Altenhofen, M. (2005). Matching Composed Semantic Web Services at Publishing Time, E-Commerce Technology Workshops, 2005. Seventh IEEE International Conference on Volume , Issue , 19 July 2005 Page(s): 64 – 70 DOI 10.1109/CECW.2005.14

[31]. Fu, Y., Jin, T., Ling, X., Liu Q., and Cui, Z. (2007). "A Multistrategy Semantic Web Service Matching Approach," International Conference on Convergence Information Technology (ICCIT), pp.253-256.

[32]. Gunawardana, A. & Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Journal of Machine Learning Research Vol. 10. (2009), pp. 2935-2962.

[33]. Guo, R. and Chen, D.and Le, J. (2005). Matching Semantic Web Services across Heterogeneous Ontologies CIT, pp. 264-268, IEEE Computer Society, 2005

[34]. Haas, H. and Brown, A. (2004, August). Web Services Glossary W3C Working Group Note 11 February 2004. Retrieved from http://www.w3.org/TR/ws-gloss/

[35]. Helal, S. Desai, N. Verma, V. and Lee, C., (2003). "Konark: A Service Discovery and Delivery Protocol for Ad Hoc Networks," Wireless Comm. and Networking, vol. 3, 2003, pp. 2107–2113.

[36]. Hu, R., Liu, J. Liao, Z. and Liu, J. (2008). A Web Service Matchmaking Algorithm Based on an Extended QoS Model ICNSC, pp. 1565-1570, IEEE, 2008.

[37]. Jaeger, M.C., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., and Geihs, K., (2005). Ranked Matching for Service Descriptions Using OWL-S, KiVS, pp. 91--102.

[38]. Jeckle M., and Zengler B., (2003) "Active UDDI - an Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation". Revised papers from NODe 2002 Web and Database related workshops on web, Web-Service and Database systems, LNCS. 91-99.

[39]. Jeckle, M. and Zengler, B. (2002). *Active UDDI - an Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation (2002)*. International Workshop Web Services – Research, Standardization, and Deployment (WS-RSD'02.

[40]. Kacholia, V., Pandit, S., Chakrabarti, S., Desai, S. S. R. and Karambelkar, H. (2005). Bidirectional Expansion For Keyword Search on Graph Databases. Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

[41]. KÄuster, U., Lausen, H. and KÄonig-Ries, B. (2008). "Evaluation of Semantic Service Discovery -A Survey and Directions for Future Research", Whitestein Series in Software Agent Technologies and Autonomic Computing, Emerging Web Services Technology, Volume II, pp 42-58, Birkhärequester Basel,.

[42]. Khalid, M.A, and Kapahnke, P., (2008). Owls- tc version 2.2 revision 2, March 2008. Retrieved 12 August 2008 at http://www.semwebcentral.org/projects/owls-tc/

[43]. Kirsch-Pinheiro, M., Vanrompay, Y., and Berbers, Y., (2008). Context-aware service selection using graph matching, 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'08), at ECOWS 2008, Dublin, Ireland November 12, 2008. CEUR Workshop proceedings, vol. 411.

[44]. Klan, F. (2006). Context-aware service discovery, selection and usage. Institute of Computer Science, Martin-Luther-University.

[45]. Klein M. and Bernstein A. (2004). Towards a high precision service retrieval. IEEE Internet Computing.

[46]. Klein, M. and König-Ries, B. (2002). "Multi-layer clusters in ad hoc networks - an approach to service discovery," Proc. 1st International Workshop on Peer-to-Peer Computing (Co-Located with Networking), Pisa, Italy. pp. 187–201.

[47]. Klein, M. Konig-Ries, B. and Obreiter, P. (2003a). "Lanes: A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Networks," 3rd Workshop Applications and Services in Wireless Networks (ASWN 03), July 2003; http://hnsp.inf-bb.uni-jena.de/DIANE/docs/ASWN2003.pdf.

[48]. Klein, M. Konig-Ries, B. and Obreiter, P. (2003b). “Service Rings: A Semantic Overlay for Service Discovery in Ad Hoc Networks,” Proc. 14th Int’l Conf. Database and Expert Systems Applications (DEXA 03), IEEE CS Press, 2003, pp. 180–185.

[49]. Klusch, M., Fries, B. and Sycara, K., (2006). 'Automated semantic web service discovery with OWLS-MX', in AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems , pp. 915-922. New York, NY, USA: ACM Press.

[50]. Klusch, M., Fries, B. and Khalid, M. (2005). Owls-mx: Hybrid OWL-S service matchmaking , 2005.

[51]. Kozat, U. and Tassiulas, L. (2003). “Network Layer Support for Service Discovery in Mobile Ad Hoc Networks,” Proc. IEEE Conf. Computer Comm. (INFOCOM 03), IEEE Press, pp. 1965–1975.

[52]. Kritikos, K. and Plexousakis, D. (2008). Evaluation of QoS-Based Web Service Matchmaking Algorithms. IEEE Congress on Services- Part I

[53]. Kuhn H.W.,(1955). The Hungarian method for the assignment problem, Naval ResearchLogistics Quarterly 2 (1955) 83-97.

[54]. Kutz, O. and Mossakowski, T. (2008). Conservativity in Structured Ontologies. ECAI 2008: 89-93

[55]. Lam, W., Chan S. and Huang, R. (2007, February). Named entity translation matching and learning: With application for mining unseen translations ACM Transactions on Information Systems, 25(1), p. 2:1-2.

[56]. Le, D. N., Goh, A.E.S. and Cao T.H., (2007). A Survey of Web Service Discovery Systems. International Journal of Information Technology and Web Engineering, Volume 2, Issue 2. Igi publishing, 701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA

[57]. Le, J., Guo, R. and Chen, D. (2005). Matching semantic web services across heterogenous ontologies. CIT 05: Proceedings of the Fifth international conference on computer and information technology.

[58]. Lee, C. and Helal, S. (2003). Context Attributes: An Approach to Enable Context-awareness for Service Discovery," saint, pp.22, 2003 Symposium on Applications and the Internet (SAINT'03.

[59]. Lenders, V., May, M., and Plattner, B. (2005). “Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach,” Proc. 6th IEEE Int’l Symp. World of Wireless Mobile and Multimedia Networks (WoWMoM 05), IEEE Press, vol. 1, 2005, pp. 120–130.

[60]. Li, W. and Guo, W. (2008). Semantic-based Web Service Matchmaking Algorithm in Biomedicine BMEI (1), pp. 648-652, IEEE Computer Society, 2008.

[61]. Ludwig, S.A., and Reyhani, S. M. S., (2005). 'Semantic Approach to Service Discovery in a Grid Environment', Journal of Web Semantics, vol. 3, no. 4 (2005).

[62]. Mahmud, U., Iltaf, N., Rehman, A. and Kamran, F. (2007). *Context-aware paradigm for a pervasive computing environment (capp).* IADIS International Conference WWW/Internet 2007.

[63]. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and Sycara, K. (2004). OWL-S: Semantic Markup for Web Services W3C Member Submission 22 November 2004.

[64]. McBride, B. (2004). Resource Description Framework, http://www.w3.org/RDF.

[65]. Meshkova, E., Riihijärvi, J., Petrova, M., and Mähönen, P. (2008). A survey on resource discovery mechanisms, peer-to-peer and servicediscovery frameworks, Computer Networks 52. pp. 2097–2128

[66]. Mian, A.N. Baldoni, R. and Beraldi, R. (2009). A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks. Pervasive computing Published by the IEEE CS

[67]. Miles S., Papay J., Dialani V., Luck M., Decker K., Payne T., and Moreau L., (2003). "Personalized grid service discovery". In S. A. Jarvis, editor, Performance Engineering. 19th Annual UK Performance Engineering Workshop (UKPEW, 2003).

[68]. Mokhtar, S. B. Preuveneers, D. Georgantas, N. Issarny, V. and Berbers, Y. (2007). EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. The Journal of Systems and Software.

[69]. Moore Gordon E., (1965). Cramming More Components onto integrated Circuits, Electronics, Volume 38, Number 8, April 19, pp 114-117.

[70]. Moreno-Vozmediano, R.(2008, February). A hybrid mechanism for resource/service discovery in ad-hoc grids, Dept. de Arquitectura de Computadores y Autom´atica, Universidad Complutense de Madrid, 28040 - Madrid, SpainReceived 22 January 2007; received in revised form 8 October 2007; accepted 16 February 2008.

[71]. Motegi, S. Yoshihara, K. and Horiuchi, H. (2002). "Service discovery for wireless ad hoc networks," Proc. 5th International Symposium on Wireless Personal Multimedia Communications, vol. 1, pp. 232 -236.

[72]. Motik, B., Patel-Schneider, P.F. and Horrocks, I. (2007). The Web Ontology Language Structural Specification and Functional-Style Syntax, Editor's Draft of 23 May 2007.

[73]. Oliver Kutz and Till Mossakowski. (2008). Conservativity in Structured and Heterogeneous Ontologies Technical Report. Not published

[74]. Ort, E. (2005). Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools. Retrieved January, 2009, from http://java.sun.com/developer/technicalArticles/WebServices/soa2/SOATerms.html

[75]. Papazoglou, M.P. and W.J. van den Heuvel (2005): Service Oriented Architectures. to appear in VLDB Journal. Retrieved from http://infolab.uvt.nl/pub/papazogloump-2005-81.pdf

[76]. Papazoglou, M.P., Traverso, P., Dustdar, S. and Leymann, F. (2007). "Service-Oriented Computing: State of the Art and Research Challenges", Computer Volume 40, Issue 11, Nov. 2007 Page(s):38 – 45

[77]. Payne, T., Paolucci, M., Kawmura, T. and Sycara, K. (2002). Semantic matching of web service capabilities. Springer Verlag, LNCS, Proceedings of the International Semantic Web Conference, 2002.

[78]. Pearl, J. (1984). Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley.

[79]. Ratsimor, O., Chakraborty, D., Joshi, A. and Finin, T., (2002). "Allia: Alliance-Based Service Discovery for Ad Hoc Environments," Proc. 2nd Int'l Workshop Mobile Commerce (WMC 02), ACM Press, 2002, pp. 1–9.

[80]. Resnik, P. (1999). Semantic Similarity in a Taxonomy: An Information- Based Measure and its Application to Problems of Ambiguity in Natural Language [J], Journal of Articial Intelligence Research 1999, 95-130.

[81]. Riesen, K., Neuhaus, M. and Bunke, H. (2007). Bipartite Graph Matching for Computing the Edit Distance of Graphs Graph Based Representation for Pattern Recognition, pp. 1-12, 2007.

[82]. Roman, D., Lausen, H. and Keller, U. (2006). D2v1.3. Web Service Ontology (WSMO) WSMO Final Draft 21 October 2006.

[83]. Sailhan, F. and Issarny, V. (2005). " Scalable Service Discovery for MANET," Proc. 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005), Kauai Island, Hawaii 8–12.

[84]. Sbihi, A. (2007). A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem. J Comb Optim (2007) 13:337–351. Springer Science+Business Media, LLC 2007

[85]. Schilit, B., Adams, N., and Want, R. (1994). Context-Aware Computing Applications. Mobile Computing Systems and Applications, IEEE Workshop on, IEEE Computer Society , 85-90.

[86]. Seaborne, A. (2004). RDQL - A Query Language for RDF, W3C Member Submission 9 January 2004.

[87]. Seno, S.A.H., Budiarto, R. and Wna T., (2007). Survey and new approach in discoveyr and advertisement for Mobile Adhoc Networks.

[88]. ShaikhAli, A., Rana, O, Al-Ali R, and Walker, D. (2003). "UDDIe: An extended registry for web services," in Proceedings of Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT 2003.

[89]. Sinclair, B., Goscinski, A. and Dew, R. (2005). "Enhancing UDDI for Grid Service Discovery by Using Dynamic Parameters", in Lecture Notes in Computer Science, LNCS 3482, pp. 49-59, Springer-Verlag, Germany , 2005.

[90]. Sotomayor, B. and Childers, L. (2006) Globus 4 Toolkit: Programming Java Services. Morgan Kaufmann. San Francisco, CA.

[91]. Sriharee, N. and Senivongse, T. (2006, October). Matchmaking and ranking of semantic web services using integrated service profile Int. J. of Metadata and Semantics and Ontologies, Vol. 1, pp. 100-118, Inderscience Publishers, October 03 2006.

[92]. Srivasan, L. and Treadwell, J., (2005). An Overview of Service-Oriented Architecture, Web Services and Grid Computing. HP Software Global Business Unit, November 3, 2005. http://h71028.www7.hp.com/ERC/downloads/SOA-Grid-HP-WhitePaper.pdf, date accessed: 1 June 2008.

[93]. Strang, T. and Linnhof-Popein, C. (2004). A Context Survey, n: Workshop on Advanced Context Modeling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, 2004.

[94]. Suraci, V., Inzerilli, T. and Mignanti, S. (2005). *Design and Implementation of a Service Discovery Architecture in Pervasive Systems*, IST Mobile Summit 2005.

[95]. Suraci, V., Mignanti, S. and Aiuto, A., (2007). Context-aware Semantic Service Discovery, 16th IST Mobile and Wireless Communications Summit, 2007, 1-5.

[96]. Tchakarov, J. and Vaidya, N. (2004, January) "Efficient Content Location in Wireless Ad Hoc Networks," IEEE International Conference on Mobile Data Management (MDM).

[97]. Toma, I., Iqbal, K., Moran, M., Roman, T., Strang, T., Fensel, D. (2005).An Evaluation of Discovery approaches in Grid and Web services Environments. 2nd International Conference on Grid service Engineering and Management (GSEM 2005), Erfurt, Germany.

[98]. Toninelli, A., Corradi, A., and Montanari, R. (2008). "Semantic-based discovery to support mobile context-aware Service Access" Comput. Commun., doi:10.1016/j.comcom.2007.12.026.

[99]. Tyan, J. and Mahmoud, Q. (2004). "A Network Layer Based Architecture for Service Discovery in Mobile Ad Hoc Networks," Proc. 17th Ann. IEEE Canadian

Conf. Electrical and Computer Eng. (CCECE 04), IEEE Press, vol. 3, , pp. 1379–1384.

[100]. Umesh, B., Harin, V. and Amit, G. (2008). Semantic Matchmaking Algorithms, Advances in Greedy Algorithms, Book edited by: Witold Bednorz, ISBN 978-953-7619-27-5, pp. 586, November 2008, I-Tech, Vienna, Austria

[101]. van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. and Stein, L.A. (2004). OWL Web ontology language reference, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-ref/ (last accessed 2009).

[102]. Vanrompay, Y., Kirsch-Pinheiro, M. and Berbers, Y. (2009). *Context-Aware Service Selection with Uncertain Context Information.* Proceedings of the Second International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2009).

[103]. Varshavsky, A. Reid, B. and de Lara, E. (2005). "A Cross Layer Approach to Service Discovery and Selection in Manets," Proc. 2nd Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS 05), IEEE Press, 2005, doi:10.1109/MAHSS.2005.1542832.

[104]. Ververidis, C. N., and Polyzos, G.C. (2007). Service D iscovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques, 2007. Retrieved April, 09, 2010 from http://pages.cs.aueb.gr/~chris/COMST-00012-2007.pdf

[105]. Vu, F. P. L., Hauswirth, M. and Aberer, K. (2006). A search engine for qos-enabled discovery of semantic web services. International Journal of Business Process Integration and Management 2006 - Vol. 1, No.4, pages 244– 255, 2006.

[106]. Vu,Â L., Porto,Â F.andÂ Hauswirth, M.Â andÂ Aberer, K. (2007). "An Extensible and Personalized Approach to QoS-enabled Service Discovery," ideas, pp.37-45, 11th International Database Engineering and Applications Symposium (IDEAS 2007).

[107]. Yonglei, Y., Sen, S. and Fangchun, Y., (2006). Service Matching Based on Semantic Descriptions, Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services, p.126, February 19-25, 2006.

[108]. Yu, H.Q. and Reiff-Marganiec, S. (2008). "Non-functional Property based service selection: A survey and classification of approaches", Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop co-located with The 6th IEEE European Conference on Web Services , November 12, 2008, Ireland, Dublin.

[109]. Yu, H.Q. and Reiff-Marganiec, S. (2009). Automated Context-aware Service Selection for Collaborative Systems. In P. van Eck, J. Gordijn, R. Wieringa (eds.): Advanced Information Systems Engineering CAiSE 2009, LNCS 5565, pp 261-274, DOI, 2009.

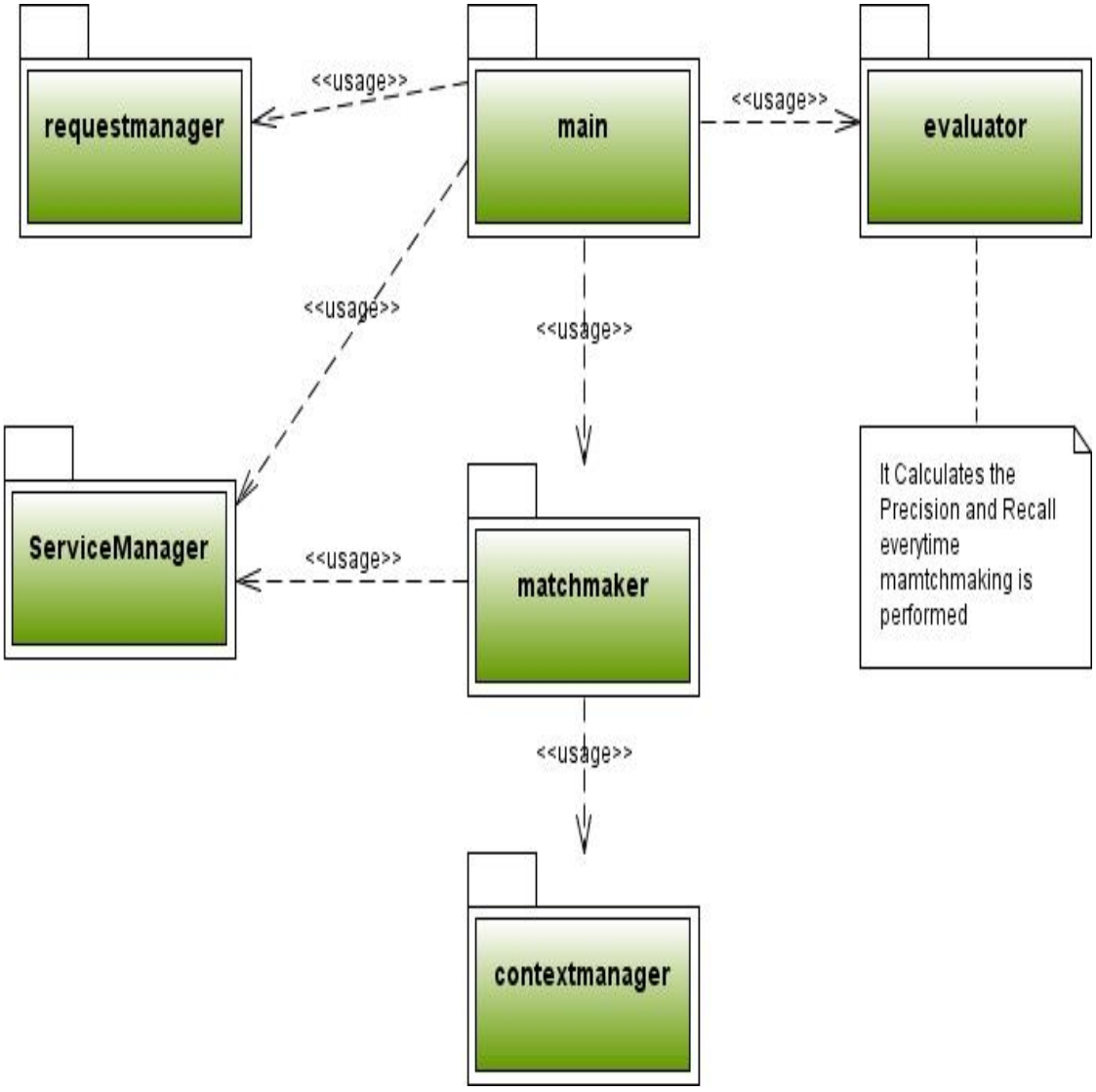
[110]. Yu, S., Ge, X., Zhang, J. and Wu, G. (2003). Web Service Discovery in Large Distributed System Incorporating Semantic Annotations, 2003. Unpublished.

[111]. Zhang, Y. Liu, B. and Wang, H. (2009). A Method of Web Service Discovery based on Semantic Message Bipartite Matchin`g for Remote Medical System. Journal of Theoretical and Applied Electronic Commerce Research ISSN 0718–1876 Electronic Version Vol 4 / Issue 2 / August 2009 / 79-87.

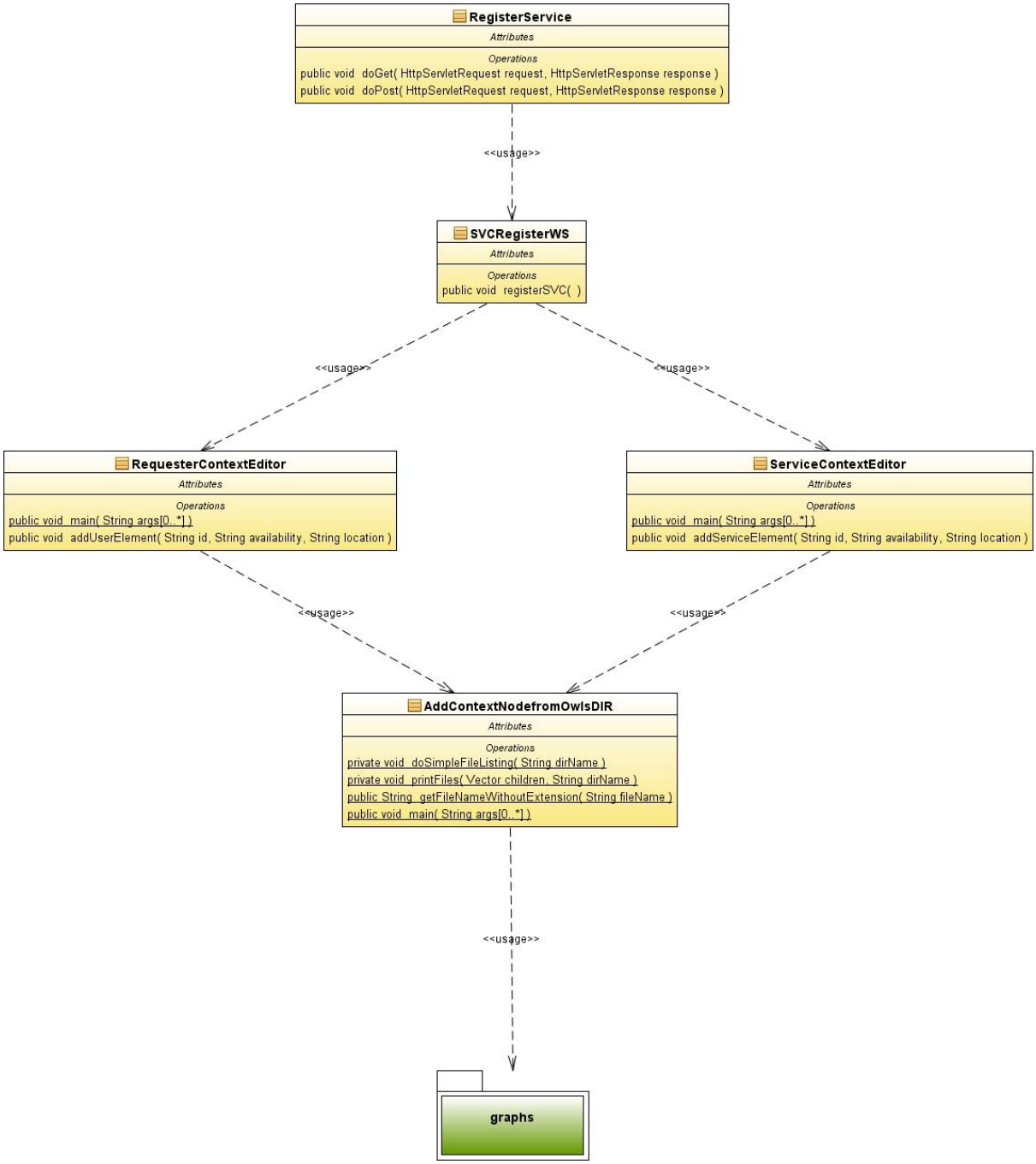
[112]. Zilin, S. Weihua, A. Yi, W. and Liang, W. (2006). Service Search Strategy Based on Graph in Grid Environment. Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKG'06).

APPENDIX A

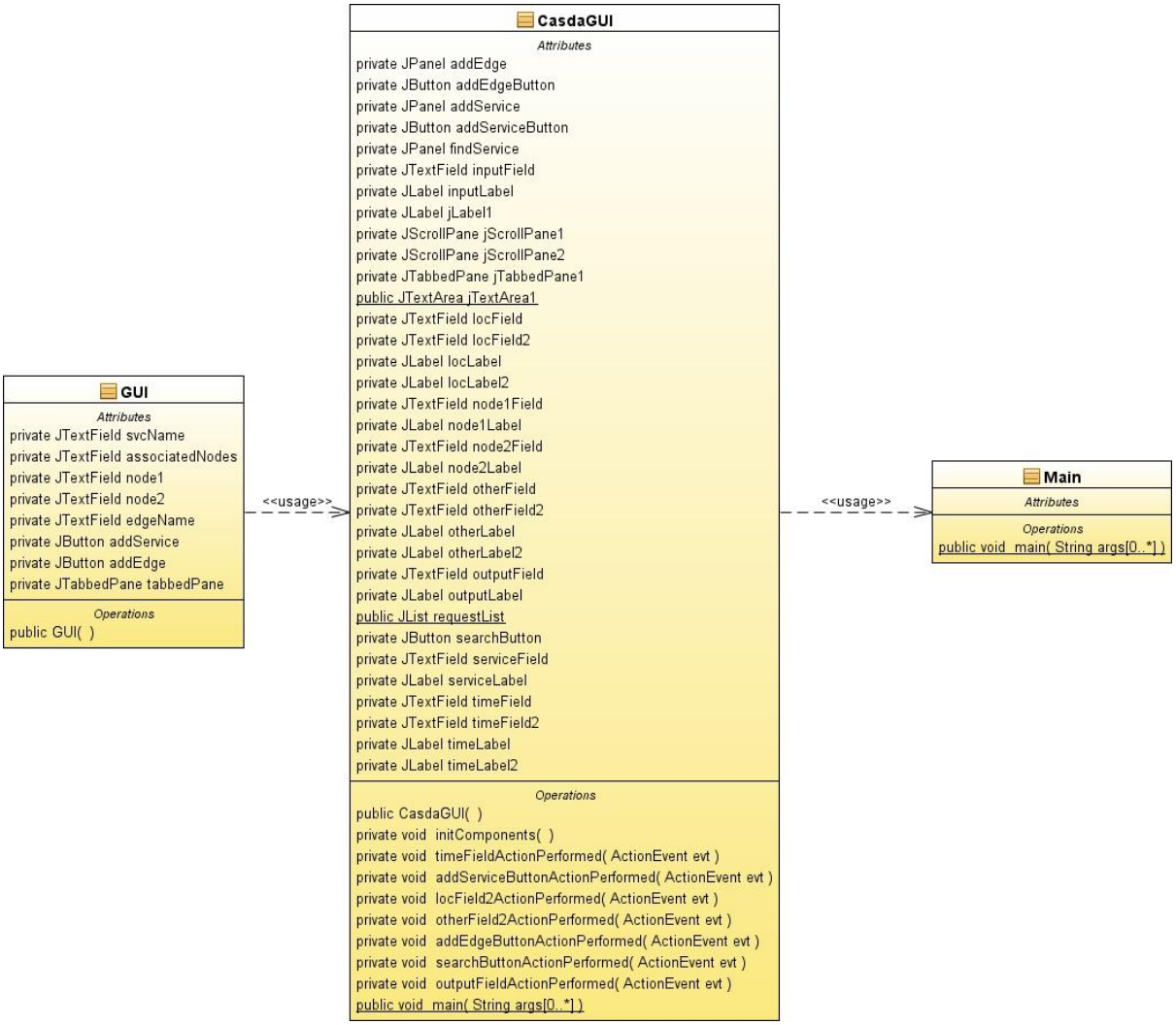
I. CASDA PACKAGE DIAGRAM



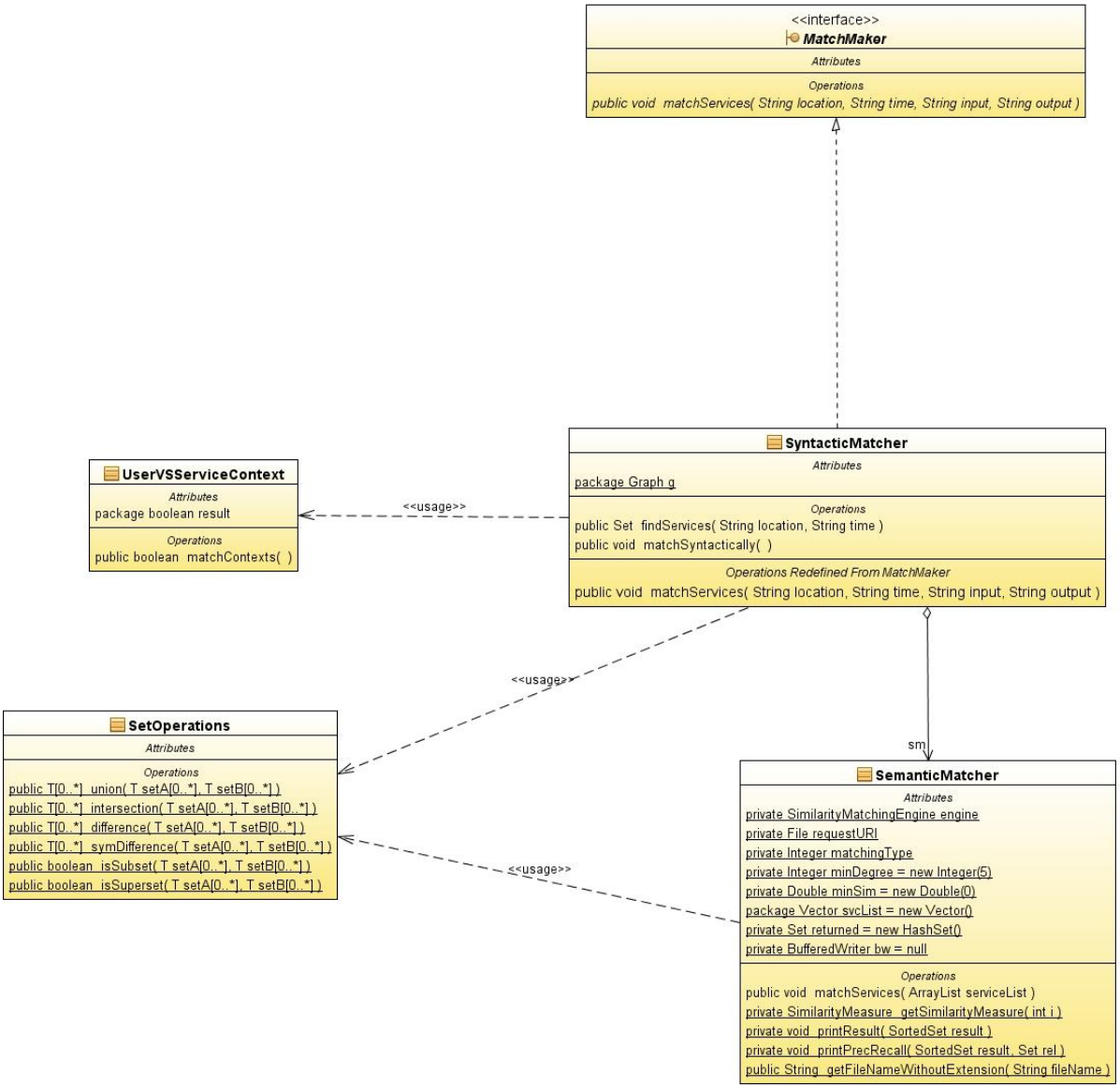
II. SERVICE MANAGER CLASS DIAGRAM



III. CASDA MAIN PACKAGE CLASS DIAGRAM



IV. MATCHMAKER CLASS DIAGRAM



APPENDIX B

I. OWLS-MX variants

The first variant of OWLS-MX, **OWLS-M0** utilizes the logic-based semantic filters Exact, Plug-in, and Subsumes which are described in section 2.7, whereas the hybrid filter Subsumed-By is utilized without checking the syntactic similarity constraint.

The other variants of OWLS-MX **OWLS-M1**, **OWLS-M2, OWLS-M3**, and **OWLSM4** compute the syntactic similarity value $Sim_{IR}\ (out_S, \mathrm{out_R})$ by use of the loss-of-information measure, extended Jacquard similarity coefficient, the cosine similarity value, and the Jensen-Shannon information divergence based similarity value, respectively. These symmetric token-based string similarity measures are defined as follows.

→ The Cosine Similarity metric

$$Sim_{cos}(S,R) = \frac{\vec{R}.\vec{S}}{\|\vec{R}\|_2^2 \cdot \|\vec{S}\|_2^2} \quad \text{B1}$$

With standard TFIDF term weighting scheme and the unfolded concept expressions of request R and service S are represented as n-dimensional weight index term vector $\vec{R}$ and $\vec{S}$ respectively. $\vec{R} \cdot \vec{S} = \sum_{i=1}^{n} \omega_{i,R} \times \omega_{i,S}$, $\|X\|_2 = \sqrt{\sum_i^n \omega_{i,X}^2}$, and $\omega_{i,X}$ denotes the weight of the i-th index term in vector X.

→ The extended Jacquard Similarity Metric

$$Sim_{EJ}(S,R) = \frac{\vec{R}.\vec{S}}{\|\vec{R}\|_2^2 + \|\vec{S}\|_2^2 - \vec{R}.\vec{S}} \quad \text{B2}$$

With standard TFIDF term weighting scheme.

→ The Intentional Loss of information

$$Sim_{LOI}(S,R) = 1 - \frac{LOI_{IN}(R,S) + LOI_{OUT}(R,S)}{2} \quad \text{B3}$$

$$LOI_x(R,S) = \frac{|PC_{R,x} \cup PC_{S,x}| - |PC_{R,x} \cap PC_{S,x}|}{|PC_{R,x}| + |PC_{S,x}|}$$

With $x \epsilon \{IN, OUT\}$, $PC_{R,x}$ and $PC_{S,x}$ set of primitive components in unfolded logical input/output concept expression of request R and service S

→ The Jansen-Shannon Information Divergence

$$Sim_{JS}(S,R) = log\ 2 - JS(S,\mathrm{R})$$

$$\therefore \mathrm{Sim_{JS}(S,R)} = \frac{1}{2\log 2}\sum_{i=1}^{n} \mathrm{h(p_i,s) + h(p_i,s) - h(p_{i,R} + p_{i,S})} \ldots .. \text{B4}$$

With probability term frequency weighting scheme, e.g., $\mathrm{p_{i,R}}$ denoting the probability of i-th index term occurrence in request R, and $\mathrm{h(x) = -x\log x}$

II. Ranking tree

Ranking tree (Bellur et. al., 2008)

a ranking tree is defined as follows:

1. for r ∈[0 ; 1] (r) is a ranking tree.

2. let r ∈ [0 ; 1] and t1,t2 .. tn be ranking trees with n ≥ 1, then (r,t1,t2,...,tn) is a ranking tree.

for example t1:= (0,(1), (0,(1),(0))) is ranking tree.

Ordering on ranking tree Let a = (*ra*, *a*1, *a*2, *a*3....*an*) and b = (*rb*, *b*1, *b*2, *b*3.....*bn*) where a1,a2...an and b1,b2 ... bn are ranking trees.

Let a < b⇔*ra* < *rb* now, a ⋪ b iff

1. a < b or

2. *ra* = *rb* and ∃*i* : *ai* < *bi* and ∀1 ≤ *j* ≤ *n* : *bj* ≮ *aj* or

3. *ra* = *rb* and ∀1 ≤ *i* ≤ *n* : *ai* ⋪ *bi*