SCHOOL OF DESIGN, ENGINEERING AND COMPUTING

BOURNEMOUTH UNIVERSITY.

# AN INVESTIGATION INTO THE IMPLEMENTATION ISSUES AND CHALLENGES OF SERVICE ORIENTED ARCHITECTURE.

A RESEARCH PROJECT SUBMITTED AS PART OF THE REQUIREMENT FOR MSC ADVANCED COMPUTING.

FAKOREDE OLUWATOYIN JOHNSON

SEPTEMBER 2007.

## Acknowledgement

I would like to thank my supervisors, Sheridan Jeary and Dr. Phalp Keith for their suggestions, feedback and input. Sheridan Jeary, my main supervisor, is a senior lecturer and VIDE project manager at Bournemouth University. Dr. Phalp Keith, my second supervisor, is a Reader in Software Engineering at Bournemouth University and joint director of the Software Systems and Modelling group (SoSyM) at Bournemouth University.

Many thanks to Michael Jones, a Principal Lecturer at Bournemouth University, for discussing preliminary project ideas with me. Also, I would like to appreciate Lindsay Main, Postgraduate Programme Administrator (DEC) at Bournemouth University, for her assistance throughout my year of study at Bournemouth University.

My gratitude also goes to members of the OWL-S discussion group and WSMO Studio discussion group for their suggestions.

## Abstract

Several literatures have been published about the semantic web services being the solution to interoperability challenges within the Service Oriented Architecture (SOA) framework. The aim of this dissertation was to find out, if the introduction of the semantic layer into the SOA infrastructure will actually solve these challenges.

In order to determine the existence of these challenges, a traditional web service built on XML technology was developed; first to understand the technology behind web services and secondly to demonstrate the limitations of the original SOA framework especially in the area of automatic service discovery and automatic service composition.

To further investigate how the Semantic layer could solve these limitations; a semantic web service was developed, to explore the tools and models available to develop semantic web services and the possible challenges that could arise from the inclusion of the semantic layer into the SOA infrastructure.

These two applications were evaluated and compared in terms of their capabilities and underlying technologies to find out if truly, the semantic web services could solve the interoperability challenges within the SOA infrastructure. Since semantic web services are built using ontologies, they have well described interfaces that allow for automatic web service discovery and invocation; it was found out that truly, they can solve the interoperability challenges in the SOA framework. However, there are a number of challenges that could impede the development of the Semantic SOA; such challenges were discussed in this paper.

Finally, this paper concludes by highlighting areas in which the work in this research could be extended.

# Table of Content

## Contents

# 1. General Introduction

## 1.1 Background

In today's world, businesses are striving for growth and competitiveness in order to be on the 'cutting-edge'; as a result business needs and processes change very frequently. In a recent survey, *"90% of CEOs expect to transform their enterprise to become more responsive, particularly to customer demand, within the next five years"* [72]. For an organization to retain its agility and adaptability, it must ensure that the supporting IT resources respond flexibly to the changing business needs. Basic business needs include reduced cost, quicker response to customer needs, integration across the enterprise, Business to Business (B2B) and Business to Consumer (B2C) integration and greater Return On Investment (ROI) [10].

In response to the changing business needs, companies face the challenge of building technologies from scratch, integrating applications across incompatible platforms and maintaining legacy systems. According to Microsoft [45], the core of these challenges is information sharing between legacy and line of business applications; although they perform optimally in their specific roles whether in finance, customer relationship management, marketing etc., it is not possible for them to exchange data either within the organization or across the organizations without some form of human intervention.

Human intervention could be by manually feeding one system's data into the other incompatible system or by development of a code that will act as an interface between the two incompatible applications in order to enable information sharing. Neither of these solutions is efficient because they could be error prone, expensive, difficult to maintain and could be a duplicate of process [45, 63].

However, today's retail business climate requires that business processes be automated and flexible through the alignment of technology with business objectives and processes. This will require developers to compose systems and integrate them over distributed networks [48]. Furthermore, systems must be developed across heterogeneous environments such that they can accommodate various operating systems, hardware, middleware, programming languages and databases [10, 16, 39, 40].

Hence, an architecture that allows for interoperable system-to-system communication irrespective of the platform, location and programming language across a loosely coupled network is required.

One solution that binds all these requirements together is **Service Oriented Architecture (SOA)** [16].

SOA has been regarded as an architecture that allows for interoperability between systems across diverse platforms [33, 36]. It has been highly rated as the next generation paradigm for developing loosely coupled applications and allowing the integration of complex systems at a reduced cost.

Although SOA promises to solve interoperability problems across heterogeneous systems, there are problems when services containing data and processes of different structures and definitions and could lead to failure in achieving true interoperability [33]. This is because SOA technologies such as the Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI) do not have the capability of providing the semantic meaning of data based on inference.

It has been proposed therefore, that introducing the Semantic layer into the SOA infrastructure will enhance true interoperability by providing automatic discovery of services and integration within the SOA framework [20, 53]

## 1.2 Purpose of Dissertation

This research will investigate the interoperability challenge in service discovery in the SOA framework and explore how service discovery could be automated by the semantic layer. It relies on information obtained from a combination of sources which entails an extensive review of the literature; drawn from both academic and commercial organisations.

The dissertation is organised into the following chapters.

- Chapter 2. Review of Literature

  This chapter gives an overview of services from different technological views such as Software as a Service, Software as a Product, Application Service Provider, and then goes ahead to discuss the origin of Service Orientation and how it achieves interoperability. Other sections of this chapter discusses early interoperability architecture, then defines Service Oriented Architecture (SOA) and its principle and technologies, its relationship with Web Services, benefits, limitations and challenges.

- Chapter 3. The Semantic Web Service

  This chapter extends the review and discusses the semantic web and its technologies such as Resource Description Framework (RDF), Ontology Web Language (OWL). Current development tools and real life applications of the semantic web services are equally explored.

- Chapter 4. Methodology

  The methodology of the research is discussed in this chapter. It includes the aims and objectives of the research, the project method, the implementation of the method and how the result will be analysed.

- Chapter 5. Practical Investigation I: The Traditional Web Service

This chapter describes two basic models for accessing web services data namely; SOAP and REST. It also describes Amazon's web services and discuses the implementation of a web service client that invokes the Amazon's web services to demonstrate interoperability.

- Chapter 6. Practical Investigation II: The Semantic Web Service

  This chapter discusses two semantic web services architectures such as the WSMX based and the OWL-S based. It provides a description of the OWL-S Editor tool and the implementation of a semantic web service client.

- Chapter 7. Evaluation and Discussion of Results

  This chapter evaluates and discusses the results of the two web services; it further describes the problems encountered during the implementation of the services.

- Chapter 8. Conclusion

  This chapter concludes by discussing how the aims and objectives were achieved, the limitations and general evaluation of the project and how the project could be extended.

# 2. Literature Review – Service Oriented Architecture (SOA)

## 2.1 Introducing Services

Services could be described as any non-material activities rendered to customers for the purpose of creating benefits [39]. They are carried out in the form of giving skills, expertise or experience based on the demand placed on the service to bring about human satisfaction. Services are delivered in virtually all sectors of the economy such as consulting services, customer services, transport, entertainment, information technology etc. In the field of Information Technology (IT), services support existing IT products, service an existing IT infrastructure or provide IT infrastructure to customers who need such services; a good example are Internet Service Providers (ISP) – they have invested time and resources into their infrastructure, then they advertise their services to people who need internet connections but do not have enough resources to acquire such infrastructure, hence they subscribe for such services.

Recently, the internet has changed the approach in which business is conducted as products and services could be sold and discharged over the internet. The software industry is also taking advantage of this opportunity, rather than having the conventional sale of software as a product; software components are now being sold as services to be used over the internet. This introduces us to Software as a Service.

## 2:2 Software as a Service (SaaS).

In the software industry, services are software components that exist independently over a network where their functionalities could be discovered based on the description of what they can do and how they can be used [45, 51].

When Software exists as a service, it allows for reuse of existing assets because they are loosely coupled, it also enhances interoperability between variety of applications and technologies. Another reason for reusability is that the software components are packaged as self-contained entities existing in modular form; this allows developers to manage each individual component without affecting other components of the software. Software as a service could be likened to as building blocks that are made available to develop different forms of new applications [7, 42, 51, 58].

SaaS [11] has been defined as a Software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the internet. Microsoft [15] simplifies it: "*Software deployed as a hosted service and accessed over the internet*".

SaaS is a concept different from the traditional software application for many reasons which shall be discussed further in this paper.

## 2:3 Software as a Service VS Software as a Product

The term software as a product is referred to as the traditional one-time licensing model of software applications, where software products are bought and installed on computers. Here, the consumer could assume responsibility over the implementation and management of the software. On the other hand, Software as a Service is a subscription model that could be referred to as "pay as you go" basis where consumers once made their subscription, could have immediate access to functionalities and features of the service [55]. The subscription could be a flat rate which gives unlimited access to all of the features of the application or varying charges depending on usage [15].

SaaS also known as Software on-demand has many benefits over software as a product both to the software service vendor and software service consumer. As it shifts responsibility of software management and maintenance to the vendor which in return favours the consumer as well as cutting the cost of having to set up the infrastructure and environment suitable to accommodate the licensing model of software applications. This invariably allows small businesses to easily subscribe to software as a service since it's cheaper and simpler to deploy. One source states that SaaS is 80% - 90% cheaper than traditional software as a product [11].

On the other hand SaaS benefits the vendor as it allows the developer to support many customers with a single version of the product [55]. Many enterprises prefer to purchase software solutions rather than developing them, this informs why software vendors are beginning to expose their software as services to be consumed over the network [52].

## 2:4 Software as a Service VS Application Service Provider

SaaS is closely related to Application Service Providers (ASP) but they are not the same. Application Service Providers are companies that sell enterprise/commercial software applications such as ERP (Enterprise Resource Planning) systems, CRM (Customer Relationship Management) systems, Supply Chain Management (SCM) systems and many others, to companies for deployment where the ASP host the data centres for each company [75]. Such software systems are customized to suit the business needs of the various customers. Since [41] one ASP is running an application customized into many versions to multiple companies, it was difficult to provide expertise in each of the applications. As a result many companies still have to employ in-house specialist to manage the software application which is an added cost.

In a 2006 survey, 80% of SaaS users are satisfied with their SaaS solutions and about 74% plan to acquire more SaaS solutions from their vendors [39].

The points below highlight the differences between Software as a Service (SaaS) and Application Service Providers (ASP):

1. Multi-Tenant Scalability: SaaS vendors develop applications from scratch and in a multi-tenant environment where the applications are customized for each customer yet one source code is maintained and deployed for the benefit of every one. On the other hand, ASP applications are not developed for multi-use environment, each customer's application is maintained within the ASP environment which makes it more expensive [75].
2. Upgrade and Enhancement: SaaS is easily enhanced since it is single software-to-many consumers; enhancement could easily be implemented at the data centre of the vendor and made available to users via the internet. The user may decide to accept or reject the changes depending on their needs.

   ASP applications upgrades are usually done when the core application supplier issues them. It is difficult to have frequent upgrades due to the fact that there are multiple customized versions of the software unlike SaaS that has a single source code for its numerous subscribers [71, 75].
3. Integration and IT Support: For SaaS, integration is done via Web Services and XML while IT support is not required in the sense that the vendor assumes responsibility for that. ASP integration is usually expensive because the applications are tightly coupled and platform dependent. It also requires internal monitoring and maintenance depending on the degree of customization [71].
4. Time to Implement: SaaS is implemented immediately while ASP usually takes a longer period to customize and implement applications [71, 75].

## 2:5 Service Orientations – Origin

The IT industry has been experiencing the evolution of design models from object-oriented models to component-based development and currently service orientation.

The origin of SOA style of design is unknown but it was first used to describe the architectural principle that could enable the integration of the CORBA and DCOM programming model in an enterprise [16, 34].

While the origin may not be known it may well stem from the software engineering principle called the "separation of concerns". The principle is about breaking down a given problem/application into smaller concerns or functions where each functions are invoked remotely using a technology called Remote Procedure Call (RPC) such that it becomes easier to solve. RPC enhances solutions by shielding the developer from considering the differences in programming language, operating system, network protocols and other platform dependent components of the application (interoperability) [3, 9]. This principle (separation of concerns) is well embraced by Object Orientation and Component Based Software Development [21]. Service orientation is unique in its approach as it blends the benefits of component-based development such as encapsulation, dynamic discovery, self-description and loading with object orientation in terms of modularity but the differs in operation by passing messages between services instead of the object orientation way of invoking methods on objects. Service Orientation makes use of Schemas - a system which describes and set policies/patterns for messages sent across services to ensure standardized service semantics. This yields interoperability in such a way that concern is not given to how the messages are sent from one service to another because a standard format is in place and messages are well described [48, 78]. This will be explained further in SOA principles and implementation.

The term interoperability has its roots from the idea that systems existing in a distributed environment need to communicate to solve a particular task. For example, a UNIX system may want to retrieve customer profile from customer database existing in a mainframe computer running an operating system and application different from that existing in the UNIX system. An incompatibility problem arises where it becomes difficult for the two systems to communicate effectively, hence a need to develop interface 'software' or a middleware that processes both the requests and responses from both systems in a way they can both understand. This is illustrated in the figure below [63].

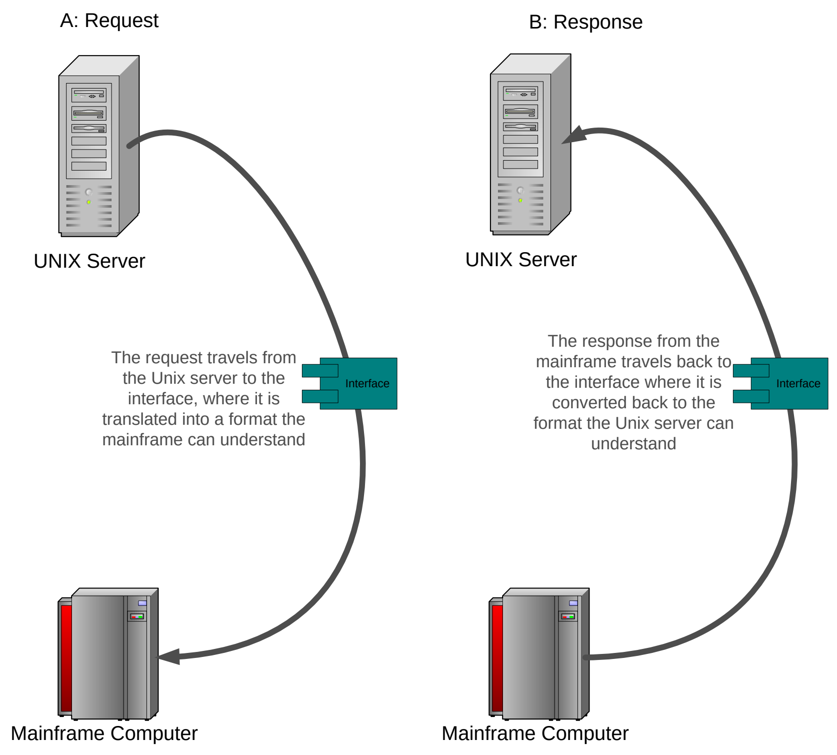

A: Request
B: Response
UNIX Server
UNIX Server
The request travels from the Unix server to the interface, where it is translated into a format the mainframe can understand
Interface
The response from the mainframe travels back to the interface where it is converted back to the format the Unix server can understand
Interface
Mainframe Computer
Mainframe Computer

**Figure 1: Communication between a Mainframe and a UNIX Server [63].**

This interface introduced does not solve the problem, but increases the complexity because for every system added to the network a need arises to add more interfaces leading to a more complex network [63]. Furthermore, any attempt to change the systems will require that the interfaces be changed which is expensive and time consuming.

## 2:6 Early Interoperable Architectures – CORBA

To solve the problem of interoperability the Object management Group[1] (OMG) developed an architecture known as Common Object Request Broker Architecture (CORBA) that enables systems to communicate in form of objects. In CORBA, an interface is created between the services provided by an object and other systems, that is, a system through the interface created usually referred to as "broker" could request for the services of an object in a distributed computing environment [32, 36]. A Distributed Computing Environment is a system of computing where both hardware and software resources existing on a network could communicate and coordinate their actions via exchanging messages [70]. The intention of creating CORBA was to have a language that could serve as an independent standard middleware [29]. CORBA operates similar to SOA but focuses on objects to enhance the communication of systems across a distributed computing environment while SOA concentrates on business process interoperability and documents [79].

CORBA raised expectations as a technology that could solve the challenge of Interoperability but is believed to have failed because:

1. Complexity: Developing CORBA applications was difficult because the APIs were complex, difficult to understand and inconsistent. As a result of its complexity, CORBA applications development is time consuming and usually contains bugs. CORBA applications are tightly coupled which makes it difficult to achieve reusability and easily implement change. (This gives SOA a major edge over CORBA) [32].
2. Cost: Implementing CORBA was expensive coupled with the fact that runtime royalties had to be paid whenever an application is deployed leading to "vendor lock-in" [66].

[1] Object Management Group (OMG) was founded in 1989 to promote the adoption of object-oriented technology and reusable software components)

3. Missing Features: A major flaw in CORBA was the issue of security. Due to a lack of encryption in CORBA messaging, messages under transition were susceptible to eavesdropping and man-in-the-middle attack. The risk of having to open a port in the corporate firewall for every service was a bridge of corporate security policies. This gives SOAP (a message protocol used in SOA infrastructure to send messages across distributed computing environment) an advantage over CORBA as SOAP is only required to pass through port 80 [32].

There are many other issues with CORBA leading to its decline such as lack of mapping between C#, Visual Basic and .Net, inability to handle threaded applications, not suitable for e-commerce applications, political issues et cetera.

## 2:7 Other Early Interoperable Architectures – DCOM and DSOM

Microsoft used Distributed Component Object Model (DCOM) as a way of simplifying the process of using objects but was only limited to their platform. Other platforms such as Linux or a mainframe computer could not use DCOM. In the process of implementing interoperability, IBM released Distributed System Object Model (DSOM). DSOM had its own limitation as well; it was only suitable for IBM OS/2 operating system although had the many features when compared with DCOM and CORBA [32, 64].

None of the three early interoperable architectures mentioned above could really achieve interoperability so to say because they all required custom or proprietary integration across systems/applications and applications were tightly coupled.

To achieve real interoperability, technologies have to be built upon open standards, which is why Service Oriented Architecture is believed to be one solution that could achieve true interoperability.

## 2:8 Service Oriented Architecture – Defined

Service Oriented Architecture (SOA) is a design approach to developing a network of integrated, simplified and highly flexible IT resources from a variety of distributed, complex systems and applications in a cost effective way. This is done by making information resources, applications and business processes accessible through standard service interfaces without bothering about the internal workings of the system [26, 44, 52]. SOA enables software systems to be designed in a way to provide services that are published to a directory and could be invoked over a network through a discoverable interface [10, 27].

With the SOA paradigm, organizations could easily unite customer or product data through data integration across the enterprise as well as application integration such that services or tasks could be invoked across departments within the organization or between business partners.

With the help of SOA data could be available in near real-time, for example; the supplier of a B2B company may check the company's inventory and know when and what to supply, a traveller may get real time information from a travel agent about flight schedule, hotel accommodations and other travel aggregators all in a single package through the use of Web Services [39, 56].

Service Oriented Architecture stems from the idea of service orientation: an idea that allows service to be readily available "on-demand". In other words, SOA is a framework that allows services to be accessed and consumed over a network [7, 33].

## 2.9 Principles of Service Oriented Architecture

So much has been written about service orientation and there are many misconceptions about SOA. The reason is that SOA is not a technology but a design philosophy/principle not dependent on any platform, programming language and not owned by anyone as a product [48]. A question that comes to mind is; how do we determine if a company is really implementing SOA? The answer is, there are SOA principles such that when followed effectively SOA will be realized. Eight principles [21] of service orientation are summarized below:

1. Service Contract: The idea of Service Oriented Architecture is to be able to find and use services that exist anywhere in a network not minding the program or language used in locating the service. The service is consumed based on its interface description. In implementing SOA using Web Services, services are published to a service registry or registries, when the service is found, it is invoked by the service requestor as long as the terms and conditions stated in the service description are met by the service requestor. To establish a service contract, the following roles and operations must be established:
    a) Service Provider: The service provider creates and publishes a service description using a Universal Discovery, Description, and Integration (UDDI) directory as the general standard for services to be discoverable by its requestors over the network [49, 63]. The UDDI provides a method for publishing and finding service descriptions, if a document (usually from a service requestor) conform to the description given in any directory stored in the UDDI, that service is invoked. A service provider could be any company that hosts a web service made available on a network or could be referred to as the *server side* of a *client-server* relationship while the service requester could be referred to as the *client side* [27].
    b) Service Requester: The service requester could be referred to as service consumer/requestor that finds a service description stored in a UDDI by a service provider and binds or invokes the service when it is found using SOAP (Simple Object Access Protocol) messages. The UDDI

alongside with WSDL (Web Service Description Language) document does not only describe what the service does but also describes how to invoke the service [3, 49].

c) Service Registry: This serves as a middle-man between the service provider and the service requester by linking the two services together. The service registry - UDDI is the directory where the service provider publishes its service description [10, 27].

The services mentioned above share the contract (standard & conditions that must be complied with), therefore its design is very important. The diagram below illustrates the relationship between the three entities. (Note: UDDI, SOAP and WSDL shall be explained in later sections).

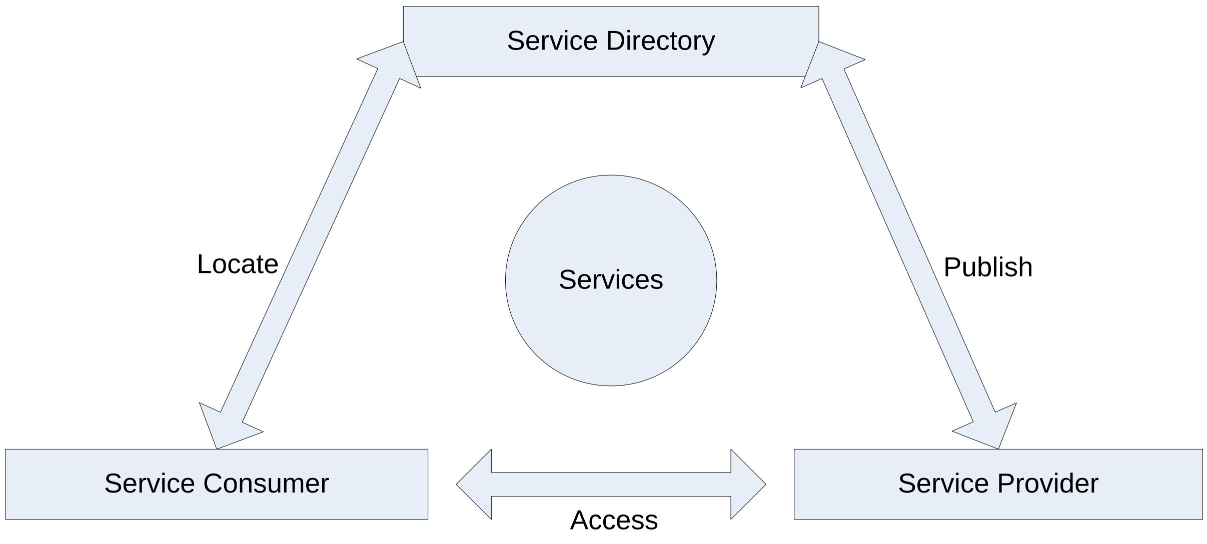


**Figure 2: Relationship between Service Provider, Service Consumer and Service Directory**

2. Loose Coupling: One of the ills of the traditional ways of achieving interoperability in a distributed computing environment was that systems were tightly coupled; this led to the inability of IT systems to respond to change very quickly [48]. The term "Loose Coupling" has to do with the extent to which systems/programs are dependent on themselves. The higher

the dependency, the tighter the coupling. To achieve loose coupling, software services have to be developed a model where a web service and its requestor are independent of one another such that any requestor whatsoever can invoke the web service [21]. Once a software program is available as a web service, it can be accessed through any service requestor in variety of ways.

3. Service Abstraction: One of benefits of SOA is the ability to establish reuse of legacy systems [35]. Legacy systems (e.g. mainframe systems) are systems that have been acquired for a long time and considered not to be replaced because it is cheaper and less complex to keep than to replace them. Companies spend between 70-80% of their IT budget on mainframe maintenance [56]. To enable reuse, the systems' functionality is abstracted on the interface-level and underlying the details is hidden from the surface. With the help of the service layer provided on top of the legacy application, the system is able to receive and respond to SOAP messages that request for its functionality without minding the internal workings of the web service and the system [63]. This is illustrated in Figure below:

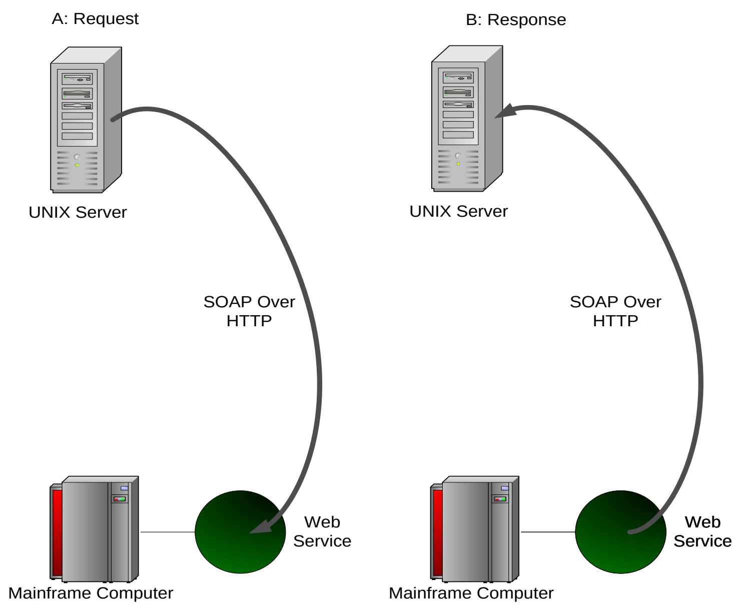


**Figure 3: Communication between a Mainframe computer exposed as a web service and a UNIX Server [63].**

The web service layer in figure 3 replaces the interface described in figure 1.

4. Service Discoverability: The principle of discoverability embodies the ability of a service to describe its purpose and its whole functionality [78]. This employs the use of the WSDL to describe the service to be as self discoverable as possible, so that it becomes easy for service requestors to locate and consume them [27].
5. Service Reusability and Concurrency: One of the reasons many companies are excited about SOA is its reusability, hence services must be designed such that its internal logic support reuse when exposed to the outside world. Moreso, services are bound to be accessed concurrently by service consumers, hence the need for services to provide excellent and reliable performance. Services are to be designed to respond to messages promptly so that other concurrent consumers will not be hindered [21].

## 2:9:1 SOA and Web Services

Although, Web services are not the only way to implement SOA, but has been identified as the best technology that fully implements SOA potentials. In the late '90s, Sun Microsystems actually implemented SOA over a private network to describe Jini technology (an environment created where services could be discovered dynamically and used over a network.) but what is outstanding about Web Services is that it uses open standards such as XML, WSDL, UDDI and SOAP to implement SOA over the internet [27, 43, 52, 58, 59]. It is predicted that 75% of SOA and Web Services will be implemented together in more than 75 percent new SOA and Web Services projects [52].

The Web Services Architecture Group at the W3C [27] gives a very clear definition of Web service:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process-able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web related standards.*

Web Services are based on three XML-based technologies, which have been adopted as standards[2] for interoperability:

I. Simple Object Access Protocol (SOAP): With SOAP messages in XML format, machines could easily understand the data being sent from a service requestor and respond by sending SOAP messages. A message could be any form of request such as a request for exchange rate for a particular currency, hotel bookings, purchase order, a search engine query. SOAP relies on the XML Schema and XML Namespaces for its definition and function. Applications/systems communicate basically using SOAP messages (9, 67].

II. Web Services Description Language (WSDL): The WSDL document also designed in XML based on standards specified by the W3C. The WSDL describes what the service does, how it is to be accessed and where it located (i.e. it's URL) [27].

III. Universal Discovery, Description and Integration (UDDI): The UDDI was created for the purpose of discovering services published to the registry. The UDDI serve as pointer to where to locate a particular Web service [58, 63].

[2] Web Services standards such as XML, SOAP, UDDI, WSDL are developed by organizations such as the W3C (World Wide Web Consortium), OASIS (Organization for the Advancement of Structured Information Standards), WS-I (Web Services Interoperability) to establish a common format for developers.

## 2:10 Benefits of SOA:

The main challenge of businesses over time has been integration of applications simply because they were built on '*siloed*' and tightly coupled applications but SOA implements loosely coupled integration which makes integration easier to accomplish and also reduces cost in a distributed computing environment [48]. The loosely coupled nature of SOA allows software applications to be reusable; this saves time and allows for flexibility in applications development, it also gives room for easy upgrades of software products.

In a fast growing business of our world today, where agility and flexibility is required in businesses, SOA allows IT infrastructure to be easily aligned to changing business requirements and companies are able to respond faster to customer needs. For example, in a supply chain environment businesses can build stronger relationship with their trading partners by creating dynamic services between them and real time data could be accessed to make quick decisions. A source reported how a business reduced time to market a product from two years to six months while improving business users' satisfaction [45, 52, 56].

## 2:11 SOA Limitations and Challenges

SOA as a paradigm has a number of limitations but standard bodies such as OASIS (Organization for Advancement of Structured Information Standards), W3C (World Wide Consortium), WS-I (Web Services Interoperability Organization) are currently looking at Web Services standards with the intension of improving SOA.

There are a number of SOA challenges such as Security, Performance Requirements, Semantics and Governance;

a. Security: SOA is built on open standards such as XML, WSDL, SOAP, UDDI and these standards do not have security of their own. This makes services and systems vulnerable to attacks. A number of authorities on SOA have stated [51, 68] top ten most critical Web Application Security Vulnerabilities were identified by Open Web Application Security Project (OWASP), the same security flaws were identified by SOA Software which are summarized below:

i. Lack of encryption: Messages on transit could be intercepted by the 'man-in-the-middle' if SOA is not secured.
ii. Denial of Service Attack: Unauthorized users could feed the service with unsolicited requests such that it denies other genuine users access to the service
iii. Lack of Logging feature: SOA has no logging facility, this makes it difficult to track whoever has used the service and where the request came from be in case there are breaches in security.
iv. Machine-to-Machine communication: One of the reasons for SOA is to automate processes and avoid human interactions, limiting human-to-machine communication. With this, it's easier for an unauthorized user to enter SOA domain as compared to traditional architectures
v. Eavesdropping: SOA messages travel over the internet unlike traditional architectures that make use of private networks. SOA messages if unsecured are subject to "unwanted Listening".

One of the main solutions to security is XML encryption and XML digital signing [27, 50, 67] but there are limitations as only IBM Web Services Toolkit and Microsoft .Net web services support SOAP messages digital signing and encryption though they support the same standards, they have different methods of implementation.

These toolkits from these two companies are not compatible with one another [67].

(b) Governance: Governance is an act of establishing policies, rules, and controls that stipulates how organizational assets are managed and utilized [38]. SOA Governance is concerned with establishing policies, controls and enforcement mechanism that contributes to the success of implementing SOA. SOA Governance will mark the difference between the success and failure of SOA.

In a recent research [76], most common failure of SOA projects in medium to large size enterprise were attributed to lack of working governance. SOA Governance has a lot to do with SOA management as it determines the roles of IT professionals, standards within the organization, ownership, allocation of resources and project delivery lifecycles. Negligence of these policies will definitely lead to SOA project failure [22]. Defining WS-Policy could pose a challenge where rendering too much could jeopardize the agility and the benefits of SOA [22, 37, 38]. Lack of SOA governance will lead to the following flaws [34, 76]:

I. A fragile and brittle SOA implementation
II. Siloed applications across enterprise, or similar services being provided by multiple applications
III. Services that cannot be reused easily because they are unknown to developers or because they were not designed with reuse in mind.
IV. Unpredictable performance of systems
V. Security breaches that cannot be easily traced
VI. Enterprise is unable to leverage new IT technologies or upgrade to meet new business processes
VII. IT is locked into vendor technology.

Hence a need to critically look into the issue of SOA governance as every other SOA issues could be managed by it [34].

(c) Lack of Understanding of Performance Requirements: When building web services, the performance environment has to be put into consideration. As mentioned in the SOA principles earlier, Services are subject to concurrent users as they grow. This call for an understanding of the performance requirements so as to build a service that responds promptly to response, if not we would have a web service that is non-responsive [22].

(d) Semantics: SOA tends to solve the problem of interoperability across diverse, distributed computing environment but there exist some semantic limitations or differences. This is because each services has a description of its functionalities and how it is to be invoked, a problem arises when developing services where it must correspond to the consumer's operations and data structures, especially when the services use different data structures. This is where the Semantic layer of service-based infrastructure comes to play [50, 60, 78].

The semantic layer allows data to be described as concepts, relations and entities representing an abstraction of the real world. This would enable data to be well understood by service consumers and providers. Dealing with Semantics in SOA is a very difficult challenge; currently researches are being conducted to investigate more in this area.

The next chapter provides a review of the semantic web, its technologies, application areas, benefits and challenges.

# 3. The Semantic Web

In 1989, the World Wide Web idea was birthed by Tim Berners-Lee which comprises of documents (web pages), links to documents and a means of transferring documents over HTTP for information sharing. In 1992, he further established the World Wide Web Consortium (W3C) group to develop standards and further extends the web [17].

The web was primarily designed for display over the internet and to be read and understood by humans using HTML (Hypertext Mark-up Language); however, it was difficult for machines to interpret the meaning of the information published to the web. This posed a major limitation to the web as it was not possible to achieve interoperability and integration between systems as HTML is not sophisticated enough to present data in such a way that it is easily processed by machines.

With the introduction of the semantic web, information could be organised, processed and found based on meaning and not just text. The semantic web has the capability of bringing structure to the web and provides a well defined content which could be easily processed by machines, for example when searching for the word 'SOAP' in the context of Web technologies, the system can ignore references to bathing or washing soap and locate information based of Simple Object Access Protocol (SOAP) [17].

According to Tim Berners-Lee, "*The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation* " [8]. This means that with the help of the semantic web, information could be merged based on relevance to one another thereby creating relationships between documents. This enables computers to reason and draw inference from existing knowledge to make decisions.

## 3.1 Resource Description Framework (RDF)

To enable the features of the semantic web, richer mark-up languages are being developed by the W3C, in conjunction with industrial partners and academic researchers to define specifications for the semantic web. Resource Description Framework (RDF) is one of the specifications of the semantic web; it is an XML based mark-up language based on knowledge modelling such that the model is comprised of a number of triplets.

The triplets are described as follows:

a. Subject: This is the resource (person, place or thing) that the statement describes. The resource is usually identified by a URI (Universal Resource Identifier

b. Predicate: This describes the property (name, colour, shape) of the subject.

c. Object: The value (Mark, Orange, triangle) for the property of the subject.

An RDF triplet example – The Person (subject) has a name (predicate) Mark (object).

Each triplet entails a subject, predicate and object where each RDF triplets exist independently. With the structure of the RDF, it is possible for two systems to communicate and process XML data using different syntax but with the same concept of equivalence i.e. it could be inferred based on the specification that a “car” is an “automobile”. In other words, RDF helps to define a way in which knowledge could be represented and provides a way of using formal logic to generate new knowledge. The process of representing and synthesizing knowledge is called ontology [1] which will be discussed in the next section.

RDF triplets must have the following properties [1]:

a) Each RDF triple is comprises of subject, predicate and object.

b) Each RDF triple is a complete and unique fact.

c) Each RDF triple is a 3-tuple that the subject is either a uriref or a Bnode, the predicate is a uriref, and the object is a uriref, bnode, or literal.

d) Each RDF triple can be joined to other RDF triples, but still retains its own unique meaning, regardless of complexity.

## 3.2 Ontology

Imagine that a bucket of 10 litres capacity is filled up to 5 litres capacity. An individual could say that the bucket is half filled with water; another person could say the bucket is half-empty and others could say its half filled to its capacity. The idea here is that, there are different ways of representing knowledge, but knowledge could be represented in a general acceptable format. According to Davies et al [17], the term Knowledge Representation (KR) is a medium of human expression that models the world with respect to a particular domain, a problem; which allows for automatic reasoning and interpretation. KR models are built using ontologies; to provide semantic meanings to information to allow for machine processing. Ontology is an explicit and formal specification of a conceptualization of a domain of interest [30].

An ontology could also be described [17] using 4-tuple (C, R, I, A), where C is a class of concepts we wish to reason about in a specific domain (e.g. Book Selling domain) for example; title, author, publisher, prices. R is a set of Relations between those classes (Book hasAuthor author); I is a set of Instances, where each instance can be made up of one or more classes and can be linked to other instances by relations for example, BookA isFrom Publisher X; BookB hasPrice £43; and A is a set of axioms for example if a book has a price greater than £15, shipping is free.

Ontologies could be classified according to their intended purpose, generality of the conceptualization behind them and their coverage. Such classifications include:

a) Upper-Level ontologies: This is a general model of the world, having many application areas.

b) Domain ontologies: This is a model of a specific domain, for example law, medicine, air transportation.

c) Application and task ontologies: These are ontologies suitable for specific ranges of applications and tasks.

## 3.3 Web Ontology Language (OWL)

Several efforts have been made to develop a standard language for writing ontologies, languages such as Simple HTML Ontology Extensions (SHOE), DARPA Agent Markup Language, (DAML), DAML-ONT, Ontology Inference Layer (OIL), and DAML+ OIL. In 2003, a breakthrough finally came through the W3C (World Wide Web Consortium) in unifying disparate ontology efforts into a standardized ontology called the Web Ontology Language (OWL). OWL builds on the limitations of RDF as RDF is not capable enough to define the properties of properties and does not give sufficient conditions for class membership, now with OWL; additional vocabulary could be defined to facilitate greater machine readability of the Web [1a].

The OWL language has the capability of representing all the components of ontology namely; Concepts, Relations, Instances and Axioms [17].

The OWL comes in three different versions which are:

I. OWL Lite: This offers limited features but still sufficient to develop many applications.

II. OWL DL (Description Logic): This supports maximum expressiveness of the OWL language, also includes the full constructs of the OWL Full under certain restrictions.

III. OWL Full: This supports maximum expressiveness and syntactic freedom of RDF but with computational limitations.

With Semantic web technologies data could be described semantically thereby enhancing automation of processes, for example in web services where the functions and context of the web services are semantically described, this could enhance automatic discovery of the web services and then consumed by a web service client.

Although SOA uses the UDDI (Universal Discovery, Description and Integration), to store services published by providers (as described in the literature review) in a directory. To locate and bind such services, service requesters still have to use keyword-based search to find the relevant services [31, 61]. With the semantic web services, such service discoveries could be automated rather than using keyword-based search.

Semantics in web services could be classified into the following categories [4]:

I. Functional Semantic: Semantic description of what the service does.

II. Discovery Semantic: Semantic description of service capabilities to enhance service discovery.

III. Data Semantic: Semantic description of both inputs and outputs of the web service providing a relating mapping to existing ontologies.

IV. Execution Semantic: Semantic describing how the service may be verified and executed

V. Selection Semantic: Quality of Service (QoS) and other non-functional parameters associated with the service that may be used for selecting the particular service among other similar service.

Each of these semantic descriptions could be implemented using different approaches but for this project focus will be placed on the Discovery Semantic and every other discussion will be made in this regard.

To develop a fully fledged semantic web service, a well described mark up language easily understandable by computer must be used. A web service that is semantically described has the capability to perform the following tasks [1]:

1. Dynamic Discovery: A relevant web service should be discovered automatically by a program. Currently, the UDDI and WSDL do not have the capability of automatically discovering a web service because; (a) the challenge involved in locating the right UDDI to search for a web service, (b) the challenge of locating the most appropriate web service to search in the UDDI out of numerous web services registered [76]. Although the WSDL helps in service discovery but only on the syntactic level which is limited in performance. The semantic web service, enabled with its ontology has the capability to discover a web service based on the description of its properties and capabilities.
2. Dynamic Invocation: A semantic web service must be able to determine how to invoke/bind a service automatically.
3. Composition: A semantic web service should be able to select and combine multiple web services to accomplish a certain task.
4. Monitoring: A service agent has to be able to monitor the properties of a service while in operation.

With all the tasks mentioned above, the semantic web agent has many application areas in the industry. The next section discusses some applicable areas of the semantic web service.

## 3.4 Application Areas of the Semantic Web Service

The semantic web service has the promise of changing the way business services and information is being provided over the internet. Web services with semantic description have the potential of improving interoperability and integration of disparate IT infrastructures. The semantic web service is necessary as many companies face challenges of data integration and some organisations seek to improve their competitiveness by making real time data available to their clients automatically. This section aims to highlight some industrial applications of semantic web services such as the telecoms, banking, oil industries, search engines and digital libraries.

a) Telecoms Industry: One of the challenges being faced by the Telecommunications industry is lack of automation and integrated support systems which have resulted into poor customer service and 'siloed' business processes. As a result of increase in market demands between their partners, it becomes necessary for the telecoms industry to expose their processes and business interfaces to enable their partners to integrate their systems with them. For example, the British Telecommunications (BT) Call Centres were established as a platform to allow companies to communicate with their customers, due to increased customer base, so much is being spent to improve customer service. With the semantic web services, the call centres could be exposed as semantic web services where customers through the semantic web agents could locate relevant information for their needs. This would improve customer service and save operational cost [19].

b) Banking Industry: e-Banking solutions could be built using semantic web technologies. The e-banking solutions will support mortgage applications and stock-brokering, especially in the area of on-line loan applications. For example, a user could enter the following parameters to the e-mortgage semantic agent, 'find an unsecured loan of £10,000 within the United

Kingdom, payable over a period of 5 years. The semantic would then bring as output the relevant loan services [19].

c) Oil Industry: As data is the epicentre of modern day business, having the ability to control the flow of information across businesses and partners is very essential. With semantic web technologies, oil companies could integrate their business processes and modern oil monitoring technologies with their business partners' supply chain automated systems. This will help make data such as oil tanks temperatures, fluid composition, pump performance, status enquires/deliveries, daily deliveries etc. available and accessible automatically in real time to business partners across heterogeneous platforms [33].

d) Search Engines: Google and Yahoo! search engines have become house hold names, as almost all users of the internet have made search using Google or Yahoo! search engines. However, the effectiveness of search engines could be optimized with semantic search capabilities as searches are not based on keywords only but on concepts and logical relationships. With semantic web technologies, search engines could be composed to have a better understanding of search requests [14].

e) Digital Libraries: Digital Libraries have been able to offer access to books, journals, articles, conference papers to users. This has reduced human effort in search of physical libraries for books and journals. Also, academic research has been made much easier through digital libraries. To achieve the full benefits of digital libraries, many libraries share their databases to allow for a resourceful share of information. However, interoperability challenge occurs due to different user interfaces, different schema types, heterogeneous data sets etc. With semantic web services, this challenge could be solved by introducing semantic information access to improve ways to search for and browse for information [17].

## 3.5 Benefits and Challenges

Traditional web services are described using XML based technologies such as WSDL and UDDI, but semantically enabled web services are described using ontologies which improves web service discovery. Rather than using keyword based search, the semantic web allows search results to provide detailed information about its context such as attributes, elements and relationship with other information. Enhanced service discovery could also be easily understood and applied; leading to reduced effort in creating repeated services to perform the same function. On the other hand, XML based web services are numerous and there are possibilities that some services could be performing the same function simply because there are no rich descriptions of such services [17].

Semantically described services will allow a specific searchable knowledge base to be easily accessible for developers. This would enhance more directed query to determine the history or destination of a particular data as well as give complex query to be executed on disparate platforms. For example, if a search query is executed to find the number of students in a particular postcode. In this situation, there could be multiple data bases containing the results of this query, hence it is difficult for a manual kind of query to find such result. With a unifying ontology, the semantic web could access the information and retrieve the data for the requirements [17].

The core activities of the semantic web are based on ontologies which define the meaning of words and concepts. Due to lack of ontologies, it is difficult to create semantic web content. This has been a major challenge in the development of the semantic web; there is a need for a central ontology to be used by all domains to enable easy mappings between ontologies across different domains. Ontology process development activities such as ontology alignment and mapping; ontology integration, ontology translations and re-engineering tools are still at their infant stage. Such processes need to be given methodological and technological support to allow the semantic web to come to maturity [1, 17].

Another challenge hindering the development of the semantic web services is the availability of tools to support semantic annotation of web content. Presently, the available tools only annotate static pages while very few address dynamic web documents. Also, most of the tools are still being developed, so they are error prone. Currently, researchers and developers are working on semantic web tools to aid and simplify the development of semantic web content.

Security is an important requirement in building commercial applications, hence needs to be considered when build semantic web applications. It is very essential to proof and trust the other party requesting to consume a particular service. Digital Signatures and Encryption could play an active role in ensuring Proof and Trust but at the moment, they are yet to be integrated into the inference engines of the Semantic Web [17].

This Chapter has been able to describe the semantic web services; combining web services technologies with the semantic web. Also, the real world application areas were discussed as well as the benefits and limitations. The following chapter proceeds to describe the project methodology, the aims and objectives of the project, the implementation, testing and results.

# 4. Project Methodology

## 4.1 Overview

Much has been published about the success of Service Oriented Architecture (SOA) implementation especially in the area of the integration of complex systems and automation of processes [20]. SOA has been proclaimed to be a technology that offers a complete solution to interoperability issues as if there are no limitations or challenges. In the past, the same was said about CORBA (as described in the literature review) but CORBA had implementation issues.

Numerous SOA implementations have also resulted in varying degrees of disappointment [20, 60]. Many companies are beginning to realize that SOA is more complex than expected. Some questions come to mind like; can we be so sure that SOA will be a lasting technology as is being proclaimed at the moment? Does SOA implementation actually deliver the promised features? For example, there are some interoperability issues within the SOA framework such as data mismatch in structure and meaning within a complex communication pattern [31]. Although, XML Schema was designed to solve mismatch problems on the syntactical and structural level (which can be handled by the traditional web service), on the semantic level it is normally done by on a case-by-case level according to Haller [31], hence, the need to introduce the semantic layer.

SOA needs to be augmented with semantic web services to allow for automated service discovery and integration but the question remains, does the semantic web services actually solves the interoperability challenge within the SOA framework or should we expect another technology? If the semantic web service is the solution, does it not make the whole SOA infrastructure really complex and difficult to maintain?

These thoughts are the motivation behind the research question: Is the introduction of the semantic layer to the SOA infrastructure of any benefit at all?

## 4.2 The aims of the Project

The purpose of the project is to demonstrate and investigate how the interoperability challenge in the service discovery of the SOA infrastructure could be solved by introducing the semantic layer. The approach to this research is to develop two web services; a traditional web service and a semantic web service. There are five (5) categories of semantics in web services (as discussed in the chapter 3) but for the aim of this project, focus shall be on discovery semantics.

Also, discovery of service shall be based on simple semantic description of services and will not consider description logics involved in service discovery [14]. Other complexities such as abbreviations, spellings, punctuations like colon will not be considered as well.

The aims are;

- Investigate and demonstrate the limitations of dynamic binding of services in an SOA infrastructure
- Investigate how the dynamic binding of services could be enhanced by introducing the semantic layer to the SOA infrastructure.
- Investigate the possible challenges that could arise from the inclusion of the semantic layer into the SOA infrastructure
- Evaluate and compare the performance of the SOA infrastructure and the Semantic SOA infrastructure.

The SOA infrastructure uses the UDDI, where services are stored and to be discovered based on their description. This is done manually by using a humanreadable description which can only be discovered by searching using key words since a computer cannot understand the description of the service [14, 20]. Hence dynamic binding is limited. This will be demonstrated by building a XML based web services.

With the inclusion of a Semantic layer, it is expected that the binding is done automatically as searches are based on ontological concepts rather than keywords, which is more understandable by computer. This will enable service requesters to discover the most relevant services quicker [20, 69]. This will be investigated and demonstrated by developing a semantic web service client.

There are many technologies involved in the development of a semantic web service such as Web Service Modelling Ontology (WSMO), Web Service Modelling Language (WSML) and Web Service Execution Environment (WSMX), Web Ontology Language for Services Editor (OWL-S Editor) coupled with the web service technologies (SOAP and WSDL). All of these are standards provided by the OASIS (Organization for the Advancement of Structured Information Standards) which could be sourced from their web site [53].

Lastly, the two Web Services will be evaluated and compared based on some properties such as the speed of the web services, complexities, automated service discovery, semantic interpretation of search criteria etc.

This study will not involve other issues like XML schema validation, Web services security and variation in the tools such as Microsoft .Net, Java tool kit and others used in developing web services. The study is limited to evaluating the benefit of the semantic layer in SOA infrastructure.

## 4.3 Implementation:

### 4.3.1 Traditional Web Service:

There are several tools for building traditional web services such as PHP, Perl, Cold Fusion, Microsoft .Net and J2EE [1], but for this project the web services will be implemented using Microsoft Visual Studio C# because it is a more familiar tool and has a better user design interface. The traditional web service will involve two approaches to investigate the mechanism behind web services technology.

a) The first approach will involve developing a web service client called BricoBooks Service using Microsoft Visual Studio C# that will consume the Amazon's E-commerce web services. Then create a proxy application which is created from the Amazon's WSDL document of the web service. Through the proxy application the web service client (BricoBooks Service) will communicate with the Amazon's web service to consume it and retrieve some data based on the search parameters, specifically title of books. The proxy serves as an interface between the BricoBooks Service and Amazon's web services.

   Since the BricoBooks service will be consuming Amazon's web service, it would require Amazon's web services key ID which could be acquired by signing up for as a member of Amazon's web services [5].

   On completion of the web service application and the proxy file, the proxy file should be able to convert the C# data types into XML, and then send them as SOAP messages to the Amazon's web server. After the data is retrieved from the web service, it is sent back as SOAP messages back to the proxy file, then the proxy file converts it back to raw data and it is displayed on the browser.

Amazon's WSDL document could be found at [6], since the WSDL could be easily accessed, it does require searching through a UDDI.

b) The second approach will employ the use of Amazon's E-Commerce web services C# based API to develop a web service client that connects to Amazon's web services to retrieve data and displays the result on the console. The API will allow access to data on any items sold at Amazon to be queried depending on the access key ID. The API uses the Amazon's WSDL as its web reference to create a proxy file. This approach uses the same principle as the first approach only that it's a more functional application. The purpose of the two approaches is to further investigate the workings of web services.

### 4.3.2 The Semantic Web Service

The Semantic Web Service will be implemented using OWL-S Editor. OWL-S Editor is an ontology based mark-up language for describing the features and capabilities of web services such that computers can intelligently interact with data based on inference. The OWL-S Editor was developed by the Web-Ontology Working Group at the World Wide Web Consortium [65].

Other implementation tools for semantic web services are available such as JENA INFRAWEBS SWS, WSMO Studio, ODE SWS but OWL-S Editor will be used because it has more features, is more user friendly and there are developer discussion groups which could help in the research project. Another feature of the OWL-S Editor is that it hides the developer from the complexities of the semantic web service. OWL-S Editor enables a service description to be done in four models namely;

a. The Process Model

b. The Profile Description

c. The Grounding

d. The Service

These models will be described in chapter six on OWL-S Editor tool description.

Since OWL-S Editor is a plug-in to protégé, protégé will be installed and configured, then add OWL-S Editor to it as a plug-in.

The OWL-S Editor will be used to develop a semantically annotated dictionary service. The service is intended to consume a dictionary web service from MindSwap web server. This will require that the profile, process and grounding descriptions be made, stating the input, output, conditional parameters and then map the service unto the WSDL document of the dictionary service located at [46].

To further investigate the semantic web service, the web service composer tool will be used to implement a temperature conversion service. The service is expected to convert a Celsius parameter to Fahrenheit.

## 4.4 Testing:

The testing phase will be done by testing the two web services on the personal computer using the development tools since they have interfaces for testing applications; Microsoft C# for the traditional Web Services, OWL-S Editor for the dictionary services and web service composer for the temperature conversion service. The C# web service client will be executed to consume the Amazon's web service while the semantic web service clients will be executed to consume Mindswap's dictionary service and temperature conversion service to consume developersday.com temperature service. The features of the two web services will be evaluated and discussed.

## 4.5 Results and Discussion:

On completion of the implementation and testing, the results shall be evaluated and discussed. The results will be evaluated against the following criteria:

**Dynamic Discovery and Invocation**: if for example, the name of the book is specified, the semantic web should be able to locate the book, connect to another web service that will provide description of the book and connect to another web service to provide the book price; all of these are dependent on the semantic description of the web service. On the other hand, the traditional web service will involve human intervention to connect to the other web services.

**Semantic Description**: If for example, the semantic web is able to separate between two web services that look the same but are really not the same e.g books that have similar titles like; SOAP is good. SOAP is good could mean bathing soap is good or Simple Object Access Protocol (SOAP) is good. If the semantic web is able to handle these differences, then the desired aims are accomplished.

Also, to be discussed are the challenges faced in the process and measures taken to overcome the challenges where possible and reasons if otherwise.

Other sections to be discussed are how the project could be executed in a real-life scenario and potential challenges of implementing the semantic Web Services.

# 5. Practical Investigation – Online Book Service

This chapter describes how the traditional web service was implemented but first of all gives a description of two available approaches of sending requests to a web service provider namely; SOAP and REST. Then goes ahead to describe the Web Service Provider namely: Amazon's Web Services and then a description on how the web service client – BricoBooks was developed.

To develop the web service client, there are two models in accessing web services namely; SOAP (Simple Object Access Protocol) and REST (REpresentational State Transfer).

REpresentational state transfer (REST) is intended to evoke an image of how a well-designed web application behaves: a network of web pages (a virtual state machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use [25].

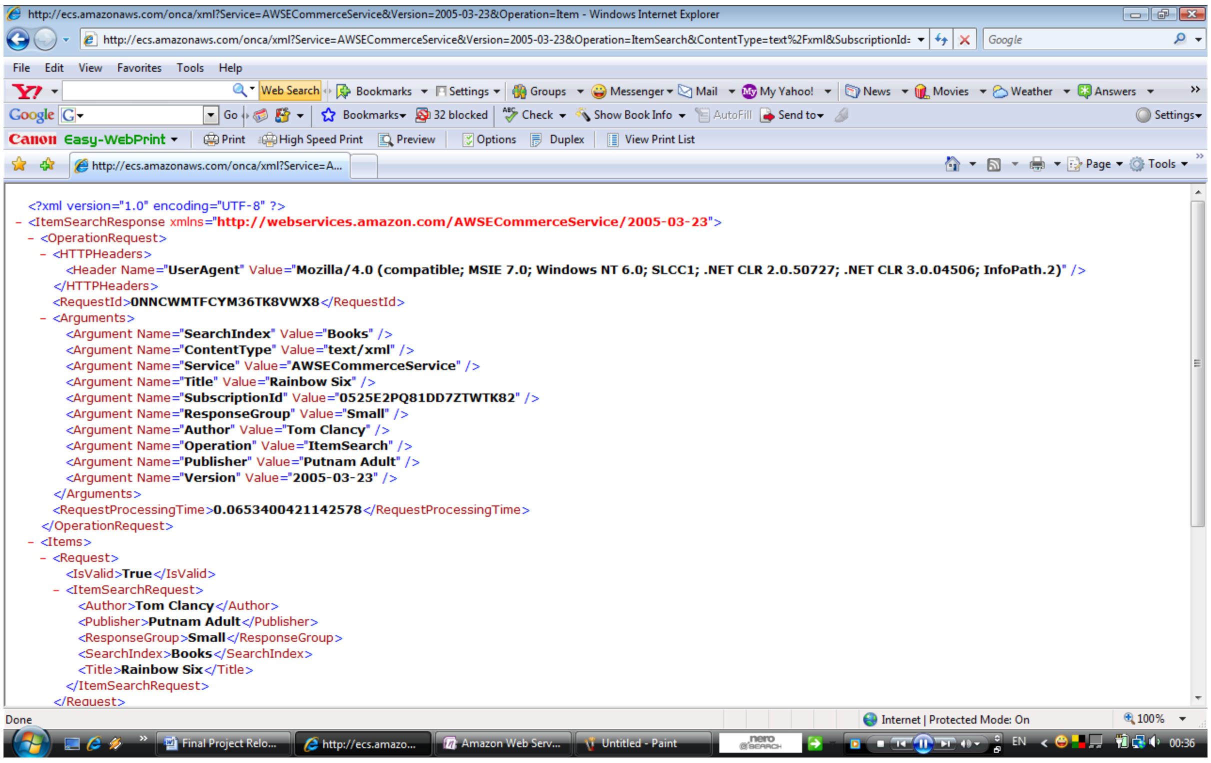
REST could be seen as a resource represented in different states through the selection of several links. A resource could be a document, an image, service (e.g weather in Bournemouth) or a non-virtual object (a person). Each of these resources needs to be identified in order to be accessed using a standard syntax called Universal Resource Identifier (URI) to locate a resource on the web.

The URI of a particular resource is represented in states, in other words, as a user moves from one URI to another, there is a state transition (transfer) being presented to the user for their consumption.

REST architecture makes use of some sets of verbs (e.g., the GET, POST, and DELETE) to act on URIs to retrieve the content of a URL (Universal Resource Locator) that is specified. The output of the data retrieved is in XML format.

Whenever a URL is entered into a browser, HTTP GET is being put to work to get some response [2]. For example the Amazon E-Commerce Service (Amazon ECS) could be accessed using REST by entering the following link on the browser - http://ecs.amazonaws.com/onca/xml?Service=AWSECommerceService&Version=2005-03-23&Operation=ItemSearch&ContentType=text%2Fxml&SubscriptionId=0525E2PQ81DD7ZTWTK82&SearchIndex=Books&Author=Tom%20Clancy&Publisher=Putnam%20Adult&Title=Rainbow%20Six&ResponseGroup=Small.

This triggers the browser to perform an HTTP GET operation and retrieves the content of the URL by entering the search parameters as part of the URL; the result is a XML data containing all books written by Tom Clancy, published by Putnam and titled Rainbow as shown in the figure below:



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <ItemSearchResponse xmlns="http://webservices.amazon.com/AWSECommerceService/2005-03-23">
  - <OperationRequest>
    - <HTTPHeaders>
        <Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.04506; InfoPath.2)" />
      </HTTPHeaders>
      <RequestId>0NNCWMTFCYM36TK8VWX8</RequestId>
    - <Arguments>
        <Argument Name="SearchIndex" Value="Books" />
        <Argument Name="ContentType" Value="text/xml" />
        <Argument Name="Service" Value="AWSECommerceService" />
        <Argument Name="Title" Value="Rainbow Six" />
        <Argument Name="SubscriptionId" Value="0525E2PQ81DD7ZTWTK82" />
        <Argument Name="ResponseGroup" Value="Small" />
        <Argument Name="Author" Value="Tom Clancy" />
        <Argument Name="Operation" Value="ItemSearch" />
        <Argument Name="Publisher" Value="Putnam Adult" />
        <Argument Name="Version" Value="2005-03-23" />
      </Arguments>
      <RequestProcessingTime>0.0653400421142578</RequestProcessingTime>
    </OperationRequest>
  - <Items>
    - <Request>
        <IsValid>True</IsValid>
      - <ItemSearchRequest>
          <Author>Tom Clancy</Author>
          <Publisher>Putnam Adult</Publisher>
          <ResponseGroup>Small</ResponseGroup>
          <SearchIndex>Books</SearchIndex>
          <Title>Rainbow Six</Title>
        </ItemSearchRequest>
      </Request>
```

**Figure 4: The search result of a REST request from a web service [5].**

Using SOAP messaging protocol, the same result would be retrieved except that a SOAP message which contains the search parameters has to be sent to the WSDL document to retrieve the desired data. The location of the Amazon.com E-Commerce web service description (WSDL) can be located at [6].

For the purpose of this research project, the online Book service - BricoBooks is the service consumer while the service provider is Amazon.com. The SOAP/HTTP model would be adopted for creating the web service client.

The next section will provide a description of the Amazon's web services since its going to be the service provider for BricoBooks Service.

## 5.1 Amazon Web Service

Amazon.com is growing to become one of the leaders in the industry of Software as a Service (SAAS) and robust web services development. Amazon Web Service is a fast growing and strategic division of amazon.com. Amazon has invested time, money and technical resources in building a world class web-scale computing platform. Over $ 2 billion and time scale of 11 years has been invested in building the Amazon's web service infrastructure [5].

Amazon Web Service (AWS) provides developers and businesses to build reliable and inexpensive web applications on its platform, enabling business managers to invest their energy on other business issues rather than infrastructure and technical issues.

Amazon Web Service was launched in July 2002 as a result of the high demand of Amazon's data by their customers. Over time the Web Services have evolved to become a huge platform for e-commerce applications [5].

To access the Amazon Web Services, users must create a web services account which automatically generates an Access Key ID and Secret Access Key. This enables them to identify users when they make transactions as well as make payments to users who subscribe to become Amazon's Web Services Associate.

There are a number of categories of web services provided by Amazon [5]; but this project will focus on Amazon E-Commerce Service (Amazon ECS) because it's the service that exposes Amazon's products data and functionality which could be deployed by developers to build their applications.

## 5.2 The Online Book Service – C# Web Service Client

In the course of developing the web service client to the Amazon's E-Commerce Service, the Microsoft .Net framework was explored to understand the mechanism behind the creation of a web service using C#. The .Net web service hides developers from the task of having to create the WSDL and SOAP files; instead these files are generated automatically. C# uses the WebMethod attribute to expose the function to be consumed as a web service, when the web service application is compiled a WSDL file is generated which specifies how the web service would be interacted with. A client application finds the web service WSDL from the UDDI and is added as a web reference on the Visual Studio.Net project.

A proxy class is generated after the WSDL document is added as a web reference. The purpose of the proxy class is to convert the user input parameters of the web service client to a SOAP message and sends it to the web service. The web service returns the data in form of SOAP messages as well and is received by the proxy class, and then the client application displays the data retrieved.

This is illustrated from the diagram below:

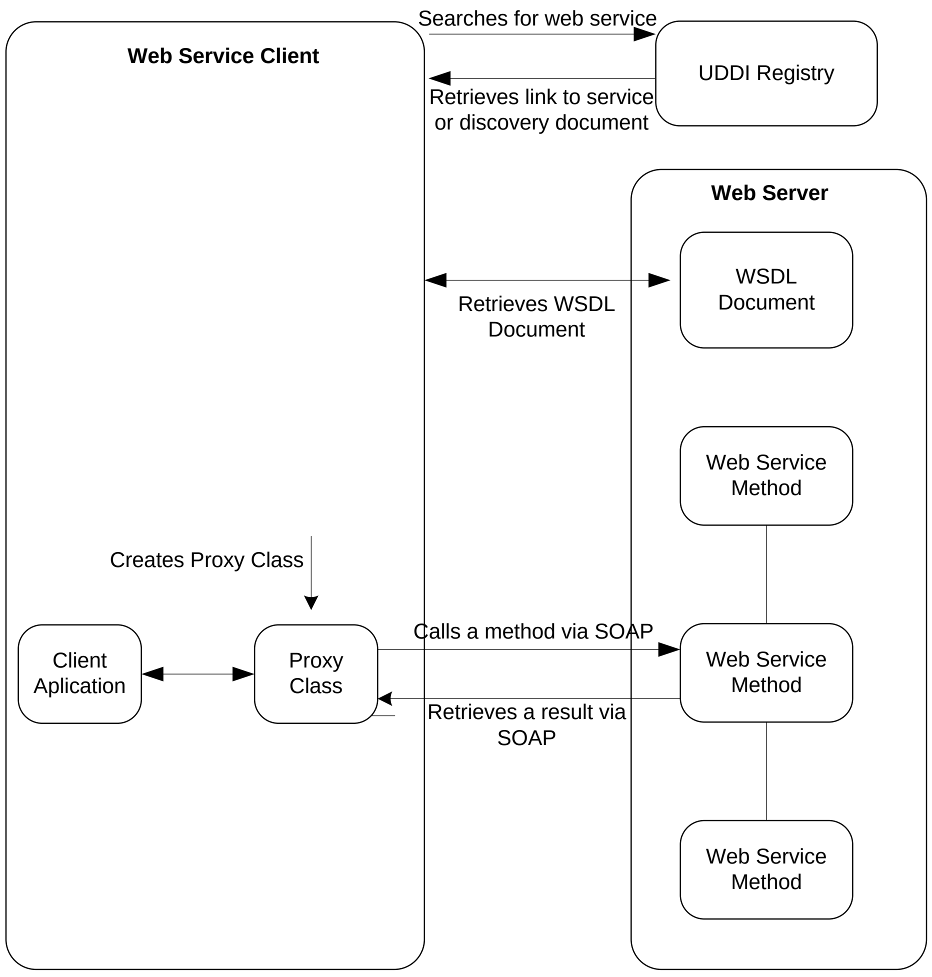


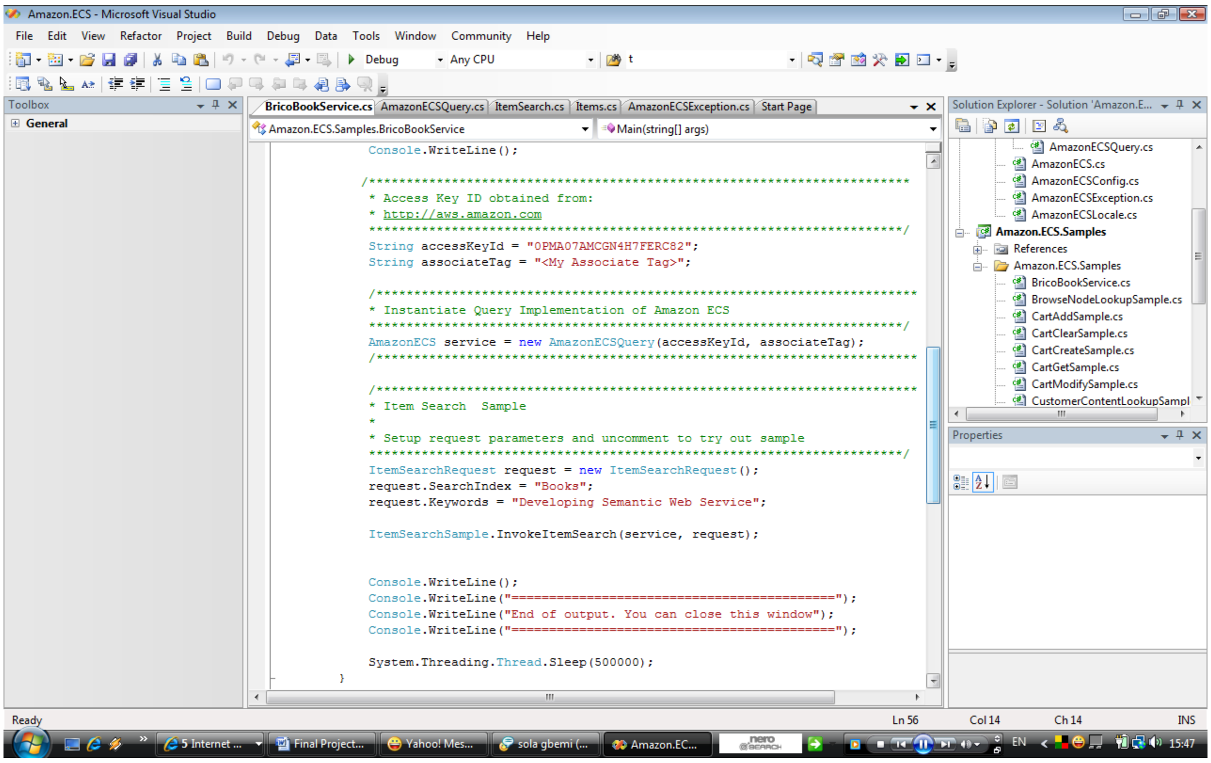
**Figure 5: A client Application interacting with a Web Service [2].**

On the account of Amazon Ecommerce Web Services, two web services client were developed to consume the web services.

1. The first application made use of APIs (Application Programming Interfaces) provided by Amazon to access their web services.

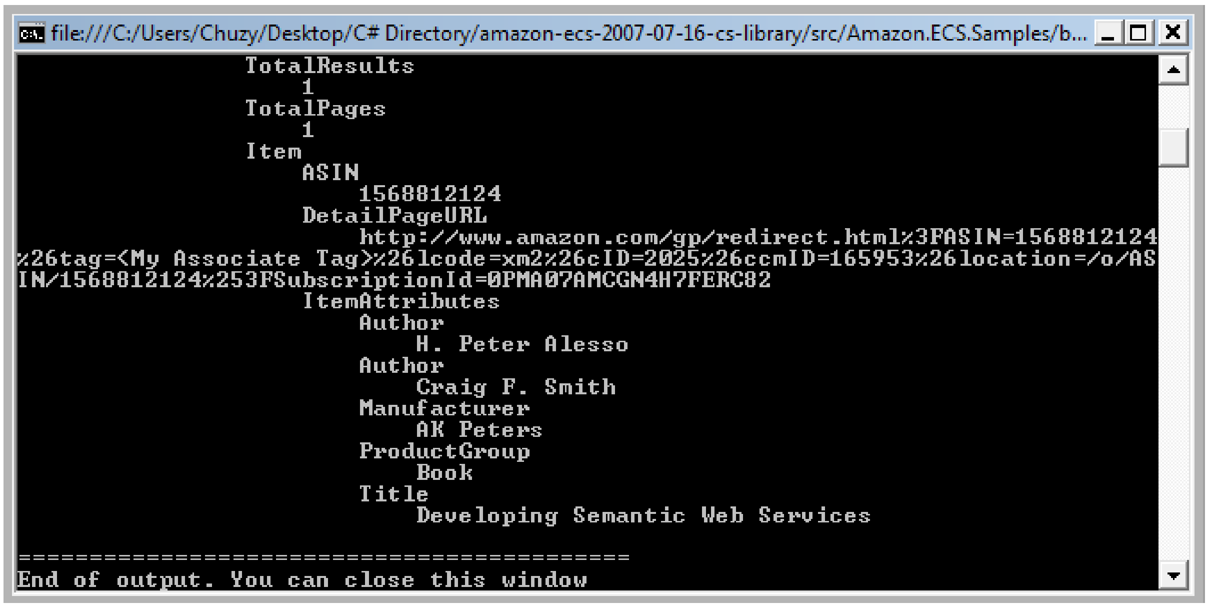
With the API, the web service client was built having the capacity to consume Amazon Ecommerce Web Service based on the search parameters and the author's personal AWSAccess Key ID (Amazon Web Services Access Key ID).

For example, if the search index was 'Books' and the request keyword was 'Developing Semantic web Services'. The retrieved data would include the attributes of the item searched such as the authors of the book, the title, the publisher and the manufacturer. The result of this application is displayed on the console.


Amazon.ECS - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Window Community Help
Debug
Any CPU
Toolbox
General
BricoBookService.cs AmazonECSQuery.cs ItemSearch.cs Items.cs AmazonECSException.cs Start Page
Amazon.ECS.Samples.BricoBookService
Main(string[] args)
Solution Explorer - Solution 'Amazon.E...
AmazonECSQuery.cs
AmazonECS.cs
AmazonECSConfig.cs
AmazonECSException.cs
AmazonECSLocale.cs
Amazon.ECS.Samples
References
Amazon.ECS.Samples
BricoBookService.cs
BrowseNodeLookupSample.cs
CartAddSample.cs
CartClearSample.cs
CartCreateSample.cs
CartGetSample.cs
CartModifySample.cs
CustomerContentLookupSampl
Properties
Ready
Ln 56 Col 14 Ch 14 INS

```
            Console.WriteLine();

           /**********************************************************************
            * Access Key ID obtained from:
            * http://aws.amazon.com
            **********************************************************************/
            String accessKeyId = "0PMA07AMCGN4H7FERC82";
            String associateTag = "<My Associate Tag>";

            /**********************************************************************
            * Instantiate Query Implementation of Amazon ECS
            **********************************************************************/
            AmazonECS service = new AmazonECSQuery(accessKeyId, associateTag);
            /**********************************************************************

            /**********************************************************************
            * Item Search  Sample
            *
            * Setup request parameters and uncomment to try out sample
            **********************************************************************/
            ItemSearchRequest request = new ItemSearchRequest();
            request.SearchIndex = "Books";
            request.Keywords = "Developing Semantic Web Service";

            ItemSearchSample.InvokeItemSearch(service, request);


            Console.WriteLine();
            Console.WriteLine("==========================================");
            Console.WriteLine("End of output. You can close this window");
            Console.WriteLine("==========================================");

            System.Threading.Thread.Sleep(500000);
        }
```

**Figure 6: Code Interface showing the search index and request keyword.**


file:///C:/Users/Chuzy/Desktop/C# Directory/amazon-ecs-2007-07-16-cs-library/src/Amazon.ECS.Samples/b...

```
                TotalResults
                    1
                TotalPages
                    1
                Item
                    ASIN
                        1568812124
                    DetailPageURL
                        http://www.amazon.com/gp/redirect.html%3FASIN=1568812124
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/1568812124%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                    ItemAttributes
                        Author
                            H. Peter Alesso
                        Author
                            Craig F. Smith
                        Manufacturer
                            AK Peters
                        ProductGroup
                            Book
                        Title
                            Developing Semantic Web Services

===========================================
End of output. You can close this window
```

**Figure 7: The console showing the result of the request.**

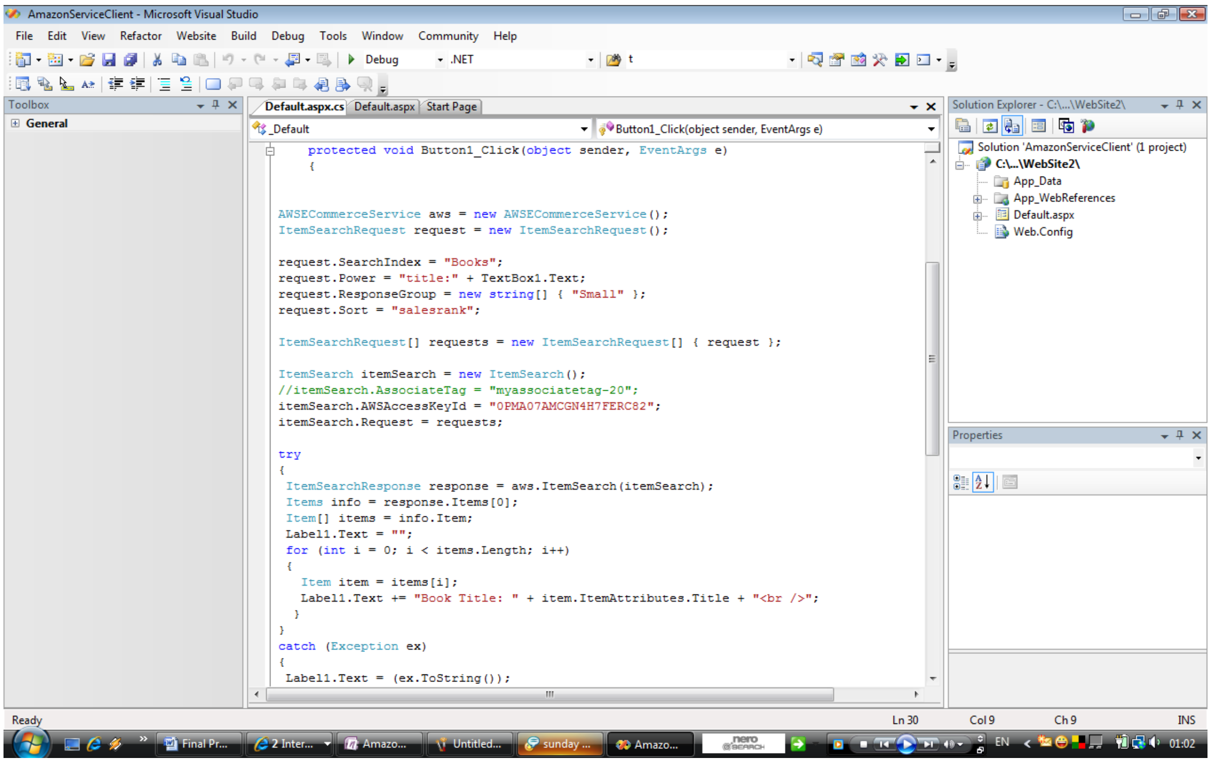
2. The second web service client application was developed from the scratch to further understand the mechanism behind the workings of web services. This

application was specifically designed to retrieve information (such as Book title and Editions based on the input parameters) on books from the Amazon's web services. On the application code, the SearchIndex specifies which catalogue will be searched from the Amazon's web services. There are many catalogues such as Books, DVD, Cloths, Electronics etc but for the application, 'Books' was used.

The power attribute is Amazon's power search syntax which allows for in-depth searching of items. In this case, the power syntax was set to 'title'.

The ResponseGroup attribute allows the search to retrieve only the data relevant to the request; this was set to 'small' in the application. Amazon has a number of ResponseGroup.

The SalesRank attribute allows the search output to be based Amazon's ranking of the item; that is the most popular item will be displayed first. This is shown on the figure below:


AmazonServiceClient - Microsoft Visual Studio
File Edit View Refactor Website Build Debug Tools Window Community Help
Debug .NET
Toolbox
General
Default.aspx.cs Default.aspx Start Page
_Default
Button1_Click(object sender, EventArgs e)
Solution Explorer - C:\...\WebSite2\
Solution 'AmazonServiceClient' (1 project)
C:\...\WebSite2\
App_Data
App_WebReferences
Default.aspx
Web.Config
Properties
Ready
Ln 30 Col 9 Ch 9 INS

```
    protected void Button1_Click(object sender, EventArgs e)
    {


AWSECommerceService aws = new AWSECommerceService();
ItemSearchRequest request = new ItemSearchRequest();

request.SearchIndex = "Books";
request.Power = "title:" + TextBox1.Text;
request.ResponseGroup = new string[] { "Small" };
request.Sort = "salesrank";

ItemSearchRequest[] requests = new ItemSearchRequest[] { request };

ItemSearch itemSearch = new ItemSearch();
//itemSearch.AssociateTag = "myassociatetag-20";
itemSearch.AWSAccessKeyId = "0PMA07AMCGN4H7FERC82";
itemSearch.Request = requests;

try
{
 ItemSearchResponse response = aws.ItemSearch(itemSearch);
 Items info = response.Items[0];
 Item[] items = info.Item;
 Label1.Text = "";
 for (int i = 0; i < items.Length; i++)
 {
   Item item = items[i];
   Label1.Text += "Book Title: " + item.ItemAttributes.Title + "<br />";
  }
}
catch (Exception ex)
{
 Label1.Text = (ex.ToString());
```

**Figure 8: The code interface of the web service client.**

The two web services clients successfully consumed Amazon's web services using standard technologies.

The next chapter takes a look the semantic web services, the tools and the development while chapter seven will concentrate on the evaluation between the traditional web service and the semantic web service.

# 6. Practical Investigation - The Semantic Web Services

## 6.1 A Decision – OWL-S or WSMO

The Semantic Web Services (SWS) is required to enable automated web service discovery, composition and execution as a result of rich semantic description of data which allows machines to process the data with little or no human intervention [17].

In order to provide the basis for Semantic Web Services, an established framework needs to be provided to ease development and execution of such services. There are two models developed to realise the aims of the semantic web services which are: Web Services Modelling Ontology (WSMO) and Ontology Web Language for Services (OWL-S). The WSMO model [23] is a product of the European Semantic Systems Initiative (ESSI) working group with the aim of standardizing a unifying framework for European research projects on semantic web services area [24]. Several tools have been developed to realize these conceptual models, for example; the Web Services Execution Environment (WSMX), Web Services Modelling Language (WSML), Web Services Modelling Toolkit (WSMT) all work together to make up the WSMO.

The figure below describes the WSMX Architecture

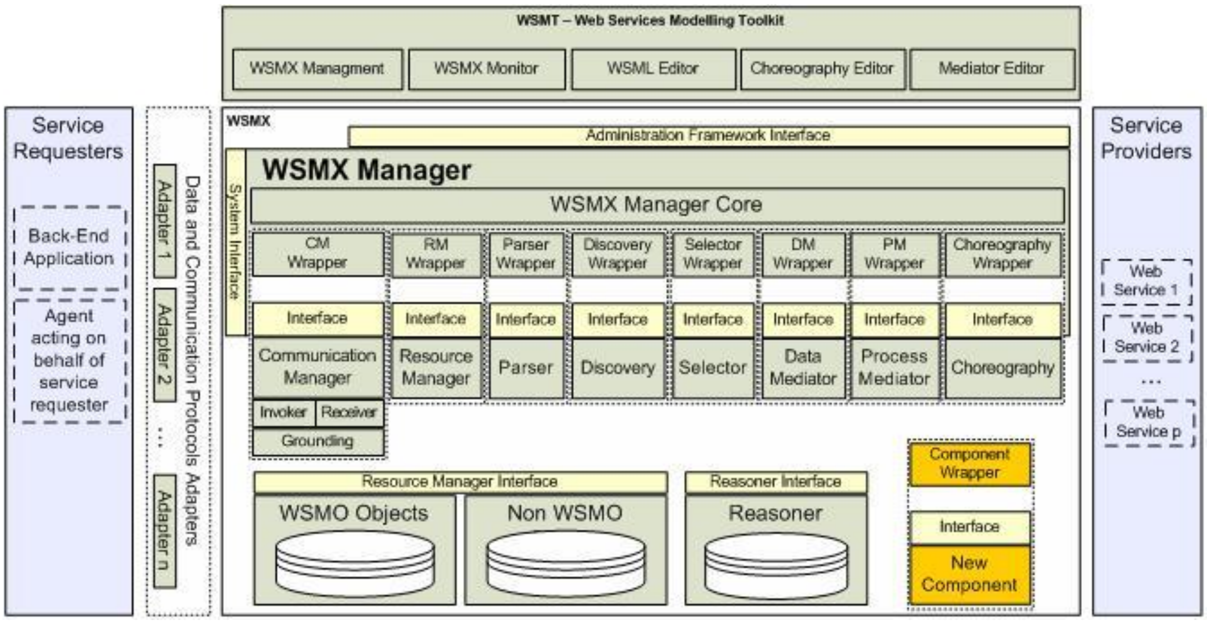


**Figure 9: The WSMX Architecture [24].**

The WSMX architecture provides descriptions of the external interfaces and the behaviours of all the components and for the system as a whole. This allows the system's overall functionality to be separated from the implementation of particular components. The WSMX accepts WSML messages as inputs and returns the results as WSML outputs. The role of the Adapter framework is to transform between a particular representation to WSML messages and vice versa [17].

### 6.1.1 OWL-S

OWL-S is a part of DARPA Agent Mark-up Language (DAML) , is an OWL- based Web Service Ontology which allows services to be described using OWL. OWL-S model tools such as OWL-S Editor, OWL-S Virtual Machine, OWL-S IDE, OWL-S Matchmaker, WSDL2OWL-S Converter, OWL-S2UDDI Converter, Web Service Composer, Semantic Web Author, SMORE – Semantic Mark-up, Ontology and RDF Editor have been developed to work individually and complimentarily to realise the OWL-S model.

The OWL-S ontology has four interrelated sub-ontologies such as profile, process model, grounding and service.

The profile provides a general description of the web service for the purpose of advertising, constructing service request and service discovery. The profile details both functional properties (such as the inputs, outputs, preconditions and results) and non-functional properties (service name, text, service category, service parameters etc.)

The process model describes how the service works to enable invocation, composition, monitoring and recovery. This description details the inputs, outputs (including a specification of the condition under which various outputs will occur), preconditions and results. The process model comprises of the atomic, simple and composite processes

The grounding specifies how the service is invoked. OWL-S uses WSDL grounding to map between any web service and semantic web services.

The service model binds the other parts together to form a service that could be published and executed. The relationship between service components are designed using properties such as presents (service–to-Profile), describedBy (Service-to-Process Model), and supports (Service-to-Grounding) [65]. This is illustrated in figure 10:

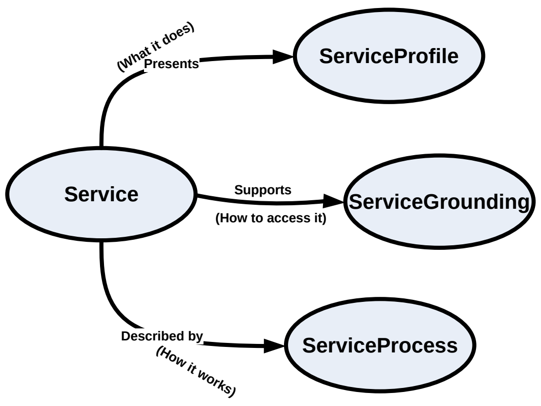


**Figure 10: The components of the OWL-S model [1].**

### 6.1.2 Comparison between WSMX and OWL-S Tools

This section describes the comparison between WSMO and OWL-S based on certain perspectives [65];

| Perspective | OWL-S Tools | WSMX |
|---|---|---|
| Reference Implementation | Reference to OWL-S | Reference to WSMO |
| Mediation | No support for mediation but based on rich description of the process and WSDL mapping could replace mediation | Provides tool for mediation |
| Data Storage | OWL-S makes use of UDDI and/or RDF Repositories which makes it simple to manage. | Data Storage is supported to store both at run time and design time entities like ontologies, web service descriptions, component APIs, e.t.c |
| Service Discovery | OWL-S matchmaker discovery is based on computer syntactical and semantic descriptions in terms of namespace, word frequency, ontology, ontology consumption and constrain matching, AI techniques and information retrieval | Discovery is based on simple description and keyword search while full support for semantic description is yet to be implemented |
| Grounding | Tools are available for grounding. The | Grounding is only on the conceptual stage. Not yet |

| | | |
|---|---|---|
| | WSDL2OWL-S tool maps from WSDL to OWL. Also the OWL-S2UDDI maps and publishes OWL-S descriptions to UDDI registry. | implemented. |
| End User Support Tools | Tools are available to end users for support such as to create, validate and visualize OWL-S descriptions | Tool such as WSMT (Web Services Modelling Toolkit is available |
| Support APIs | Programmable APIs are available such as OWL-S API to create, read and execute service descriptions | WSMO4J API is available read, write, execute WSMO elements |
| Security | Does not exist | Does not exist |
| General Complexity | Less complex than WSMX | More complex than OWL-S |
| Plug-in | Exist as a plug-in to Protege | Exist as a plug-in to Eclipse |

**Table 1: Comparison between OWL-S execution and WSMX execution environments [65].**

## 6.2 Modification to Methodology

The original intension of this project was to use WSMO Studio to implement the semantic web service but in the course of this research, it was discovered that WSMO Studio is WSML (Web Services Modelling Language) based [17, 65]. While the project intends to implement an OWL-S based semantic description of service. More so, the semantic web Client is intended to consume Amazon's web services which would require that Amazon's WSDL be grounded into the semantic web service. WSMO studio lacks WSDL grounding which makes it difficult to establish grounding between the semantic web service and Amazon's web services [65]. Lastly, WSMO Studio has a complex architecture which could be difficult to implement within the time frame of the project, hence a need to consider the option of using another tool – OWL-S Editor.

## 6.3 OWL-S Editor Tool - Description

OWL-S Editor was developed as open source software at Semwebcentral [57], designed to work as a plug-in to Protégé OWL software developed by Standford Medical Informatics [62]. The tool is configured as a tab widget plug-in for protégé. The tab provides interfaces for editing the 4 main sub-ontologies of OWL-S namely; Service, Profile, Process, Grounding.

### 6.3.1 Installation of OWL-S Editor

Before installing OWL-S plug-in, the following requirements were put in place:

i. Installed Java Virtual Machine

ii. Protégé 3.3.1 was installed. The files are available from [62].

iii. OWL plug-in was downloaded [57] and extracted to the plug-in directory of Protégé i.e. C/program files/protege3.3.1/plug-ins.

iv. GraphViz was downloaded from [28] and installed. This enables the process models to be displayed in graphical form. GraphViz is optional; OWL-S Editor could work without it.

### 6.3.2 Features of OWL-S Editor

Input/output/Precondition/Result (IOPR) Manager: OWL-S Editor has a tab for monitoring consistency between the input, output, preconditions and results of the service description. The IOPR manager shows the IOPR relationships between all the sub-ontologies – process, profile, and the WSDL grounding. For instance, if an output parameter exists in the profile ontology and does not exist in the corresponding process, this will generate an error. The IOPR manager checks all these complexities.

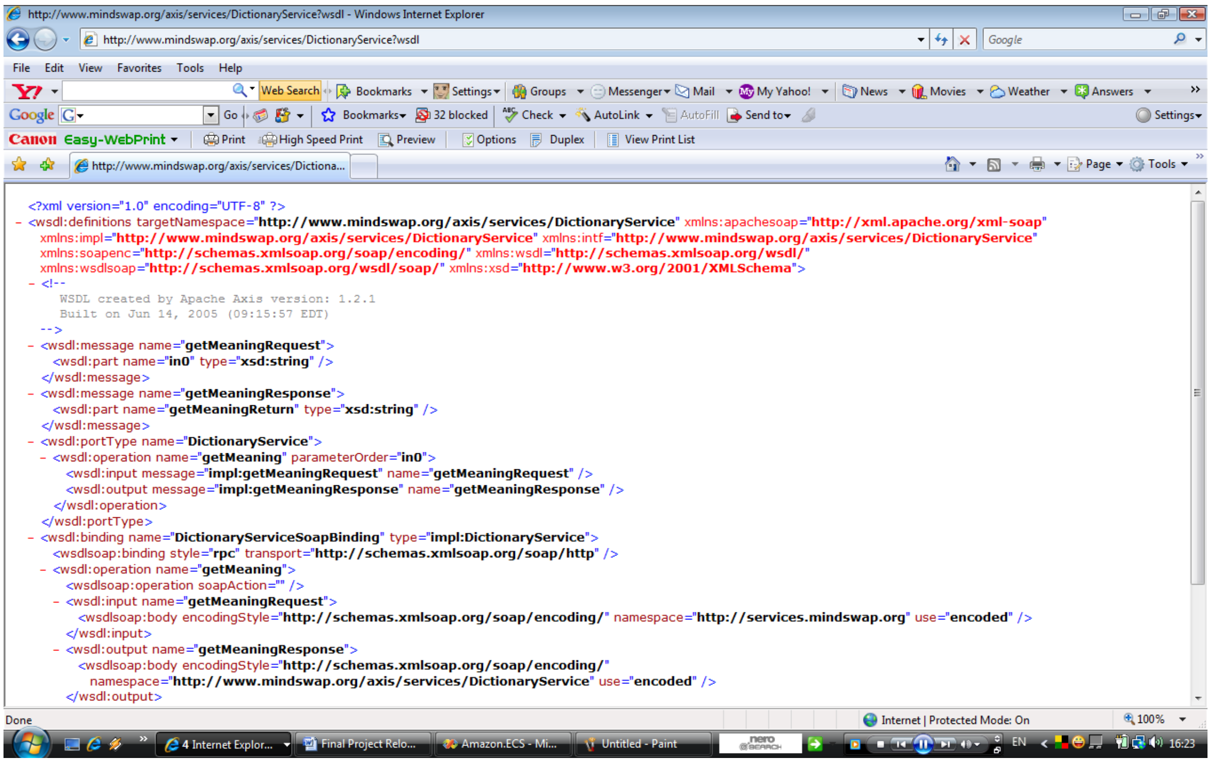
WSDL Support: The WSDL2OWL tool developed at Mindswap is integrated into the OWL-S editor. This enables OWL-S Editor to load WSDL files and then generate OWL-S service with the necessary profile, process and grounding descriptions.

Graphical Editing: This helps in structuring composite processes with all the control constructs such as sequence, split, If-Then-Else e.t.c. describing the control flow of the processes.

## 6.4 Implementing OWL-S based Web Service

In order to demonstrate the OWL-S based web service, two semantic web services were implemented. The first service was implemented using OWL-S Editor - a dictionary service while the second was implemented using Web Service Composer tool – a temperature converter service.

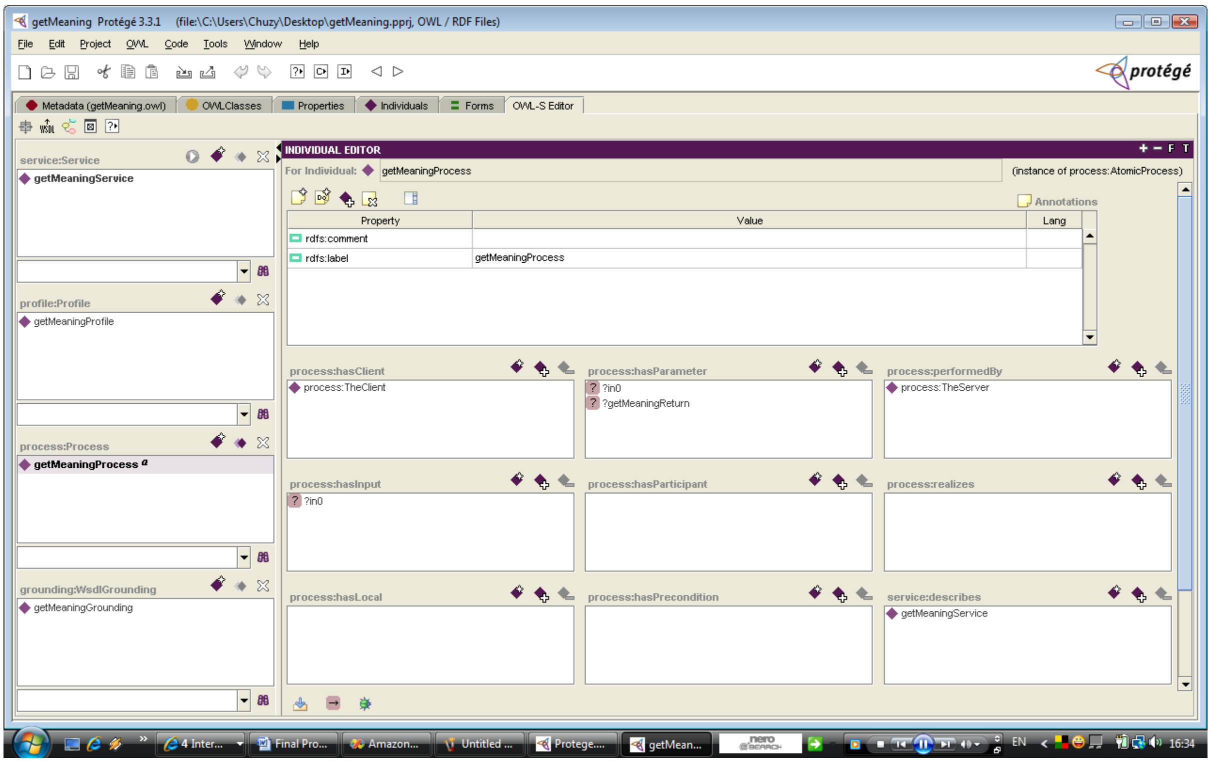
1. The semantic dictionary web service was implemented using the WSDL2OWL function of the OWL-S Editor to generate owl files from a dictionary service WSDL document located at [46].



```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://www.mindswap.org/axis/services/DictionaryService" xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:impl="http://www.mindswap.org/axis/services/DictionaryService" xmlns:intf="http://www.mindswap.org/axis/services/DictionaryService"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 - <!--
     WSDL created by Apache Axis version: 1.2.1
     Built on Jun 14, 2005 (09:15:57 EDT)
   -->
 - <wsdl:message name="getMeaningRequest">
     <wsdl:part name="in0" type="xsd:string" />
   </wsdl:message>
 - <wsdl:message name="getMeaningResponse">
     <wsdl:part name="getMeaningReturn" type="xsd:string" />
   </wsdl:message>
 - <wsdl:portType name="DictionaryService">
   - <wsdl:operation name="getMeaning" parameterOrder="in0">
       <wsdl:input message="impl:getMeaningRequest" name="getMeaningRequest" />
       <wsdl:output message="impl:getMeaningResponse" name="getMeaningResponse" />
     </wsdl:operation>
   </wsdl:portType>
 - <wsdl:binding name="DictionaryServiceSoapBinding" type="impl:DictionaryService">
     <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
   - <wsdl:operation name="getMeaning">
       <wsdlsoap:operation soapAction="" />
     - <wsdl:input name="getMeaningRequest">
         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://services.mindswap.org" use="encoded" />
       </wsdl:input>
     - <wsdl:output name="getMeaningResponse">
         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://www.mindswap.org/axis/services/DictionaryService" use="encoded" />
       </wsdl:output>
```

**Figure 11: Dictionary service WSDL document [46].**

The file getMeaning.owl were generated with their corresponding descriptions - getMeaningProfile, process - getMeaningProcess, grounding - getMeaningGrounding and service – getMeaningService as shown below. Further descriptions could be made by clicking on each descriptions (see figure 12), then a dialogue window opens to allow for more descriptions.

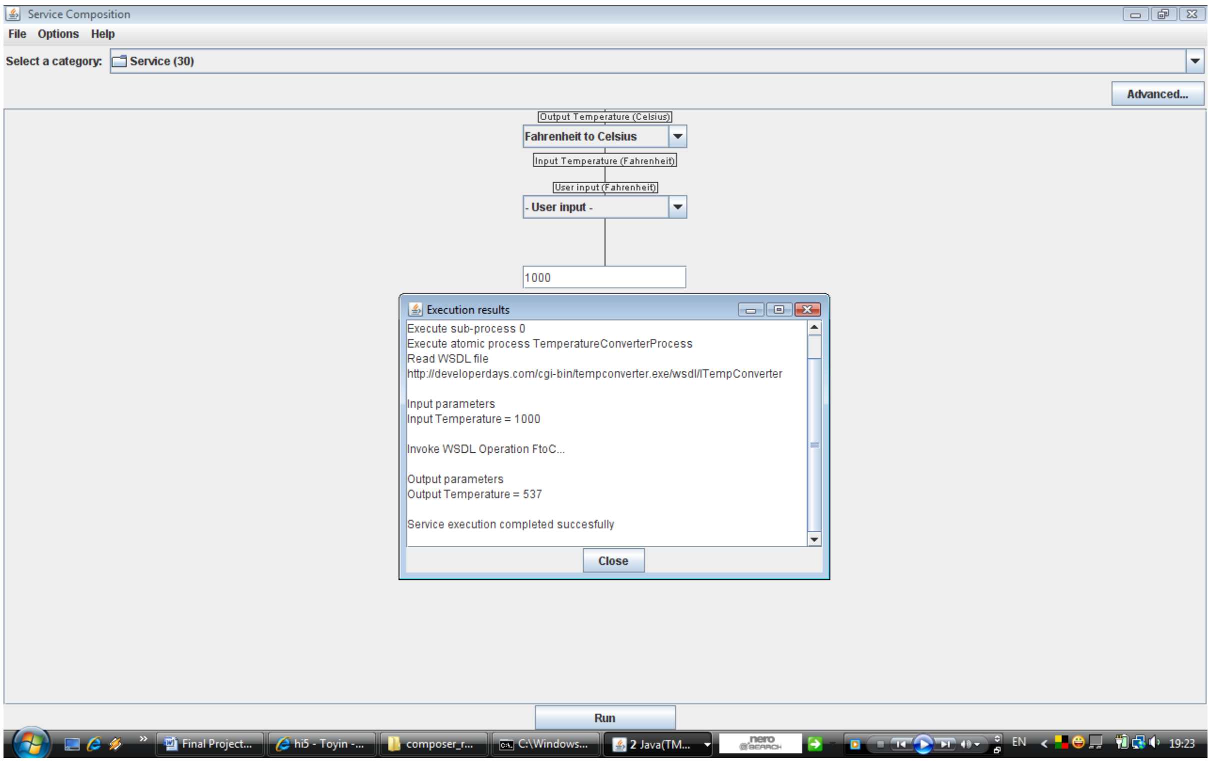


| Property | Value | Lang |
|---|---|---|
| rdfs:comment | | |
| rdfs:label | getMeaningProcess | |

**Figure 12: OWL-S Editor interface showing the dictionary service components.**

When the getMeaningService is executed, the input parameter is entered which could be any dictionary word (subject to availability of the word on the dictionary database server at Mindswap.org), then the service connects to the Mindswap.org server based on the WSDL grounding to retrieve the data: the meaning of the word entered as input parameter. The source code for the getMeaningService is located at (Appendix F).

2. The temperature web service was implemented using Web Service composer tool developed by Ervin Sirin of Mindswap.org available at [47]. This tool has already built in services that could be implemented. The temperature conversion service converts from Fahrenheit to Celsius.



```
Execute sub-process 0
Execute atomic process TemperatureConverterProcess
Read WSDL file
http://developerdays.com/cgi-bin/tempconverter.exe/wsdl/ITempConverter

Input parameters
Input Temperature = 1000

Invoke WSDL Operation FtoC...

Output parameters
Output Temperature = 537

Service execution completed succesfully
```

**Figure 13: Web Service Composer tool executing the temperature conversion service.**

The WSDL document is found at [74].

These two semantic web services were executed successfully to demonstrate the capabilities of the semantic web services technology.

# 7. Evaluation of Results – Semantic and C# Web Services.

Web Services have been used as systems to communicate and exchange data over the network irrespective of the platforms and this has been demonstrated through both applications implemented in the course of this project. The C# web service client was able to retrieve data from Amazon's Web server without bothering about the kind of web server whether UNIX based, Microsoft Operating System based or the kind of data base system used in developing the data base. Also, the semantic web service was able to retrieve the meaning of a word entered as input parameter from the Mindswap web server never minding the platform of the server.

Whilst, the two services seem similar, there are differences between them in terms of service description, service invocation and capabilities of the tools and technologies.

This section will evaluate the two services under the following criteria:

A. Service Description based on the capabilities of the two technologies

B. Service Invocation based on the capabilities of the two technologies

Service Description: The semantic web service technology is rich enough to describe the service to ease automatic discovery, invocation, composition and interoperation. With the OWL-S based semantic web service, the profile ontology gives a description of service in such a way that it enhances advertisement of the service and describes what the service does. The process ontology describes how the service is invoked, it also describes the inputs and the outputs parameters which help to streamline the kind of service it will consume. For example, a search for 'SOAP' (organic soap) would require that the input and output parameters are described in such a way that when the service is executed, results on bathing soap would be retrieved and would not be mixed with SOAP protocol. The description of the inputs and outputs enables automation of web service invocation.

The grounding description specifies the WSDL input message URI and output message URI. This helps to specify the attribute of the item to retrieve, hence making a more direct request.

With the experience on OWL-S Editor, it has been investigated that these descriptions could be made, the only short coming was that, it couldn't be implemented because while the Amazon's WSDL was converted to OWL-S file (using WSDL2OWL feature of the OWL-S Editor), it needed some more hard coding because of the complex nature of the WSDL.

Secondly, the input parameters required Amazon's Web Services Subscription ID (apart from the Access Key ID) and MarketplaceDomain which could only be acquired on commercial basis as well as a real e-commerce website [5]. This is shown in the figure:

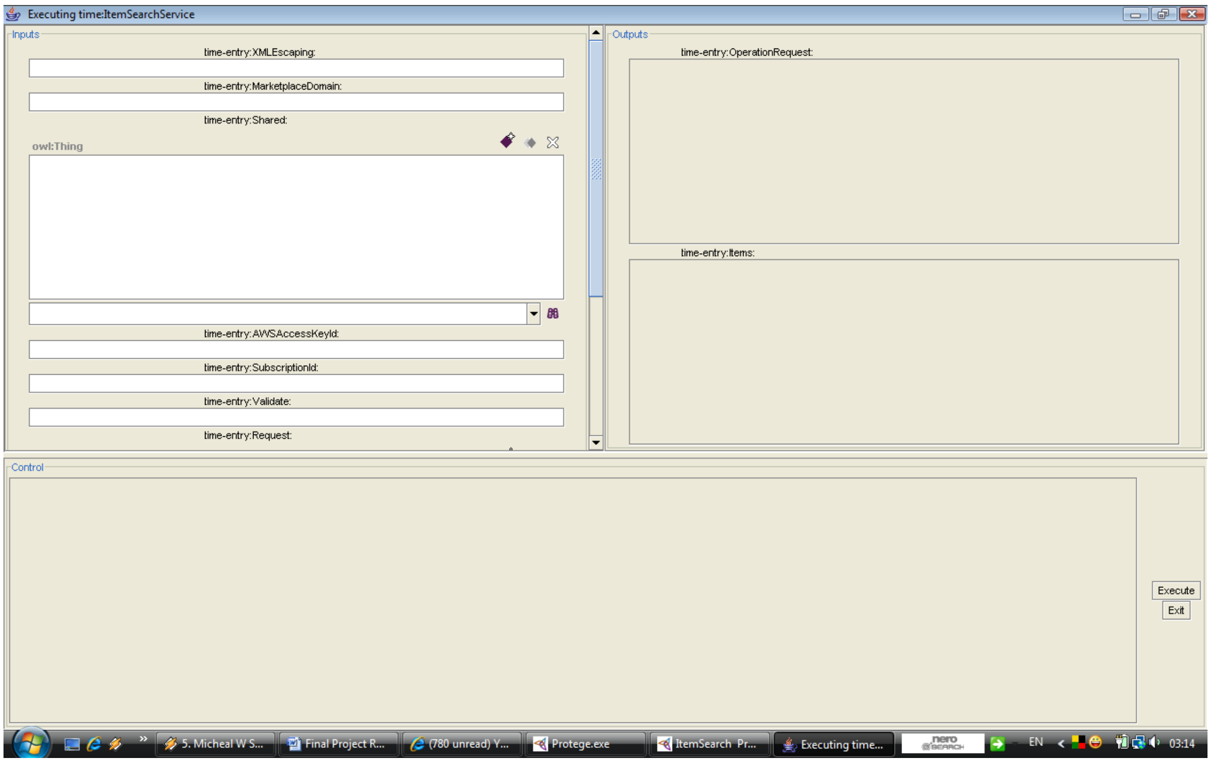

Executing time:ItemSearchService
Inputs
time-entry:XMLEscaping:
time-entry:MarketplaceDomain:
time-entry:Shared:
owl:Thing
time-entry:AWSAccessKeyId:
time-entry:SubscriptionId:
time-entry:Validate:
time-entry:Request:
Outputs
time-entry:OperationRequest:
time-entry:Items:
Control
Execute
Exit

**Figure 14: Input interface of the ItemSearch.owl service.**

However, the principle of operation was well understood and would be a good place to extend this work.

The dictionary service was successfully implemented having fully described the Service, Profile, Process and Grounding ontologies as; getMeaningService, getMeaningProfile, getMeaningProcess and getmeaningGrounding (see figure 15 and 16).

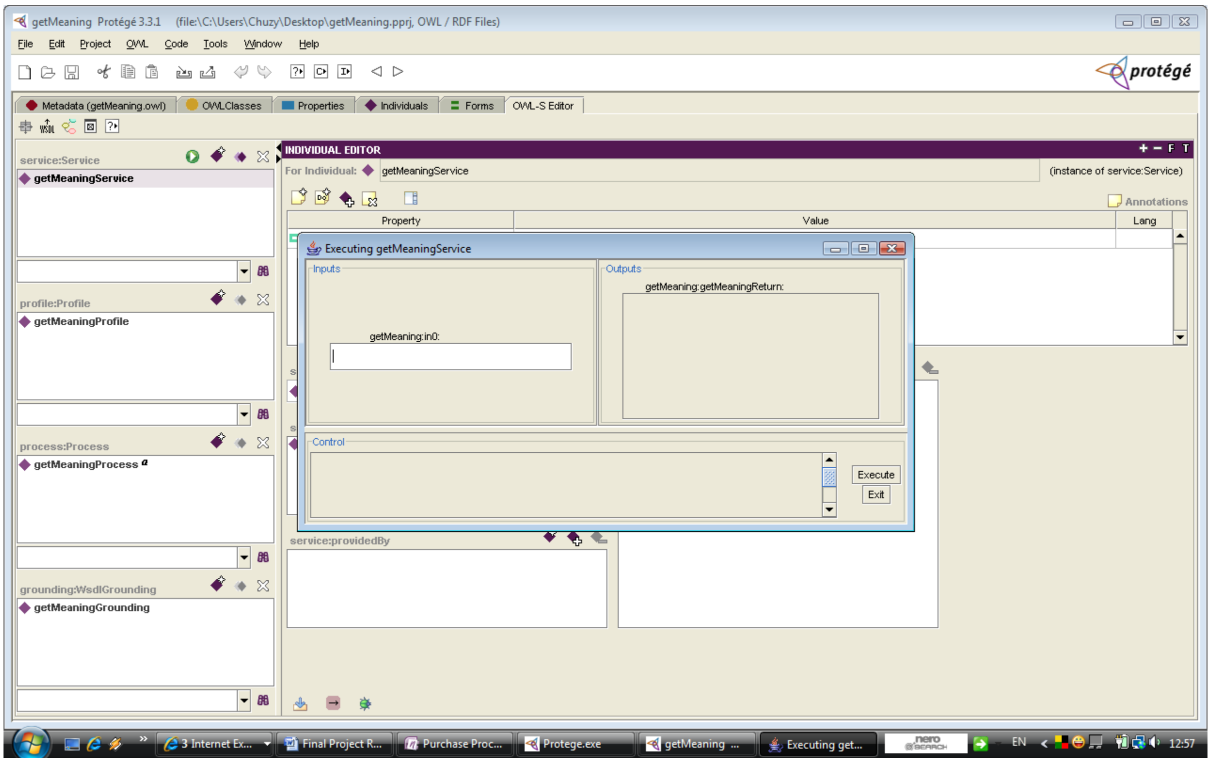
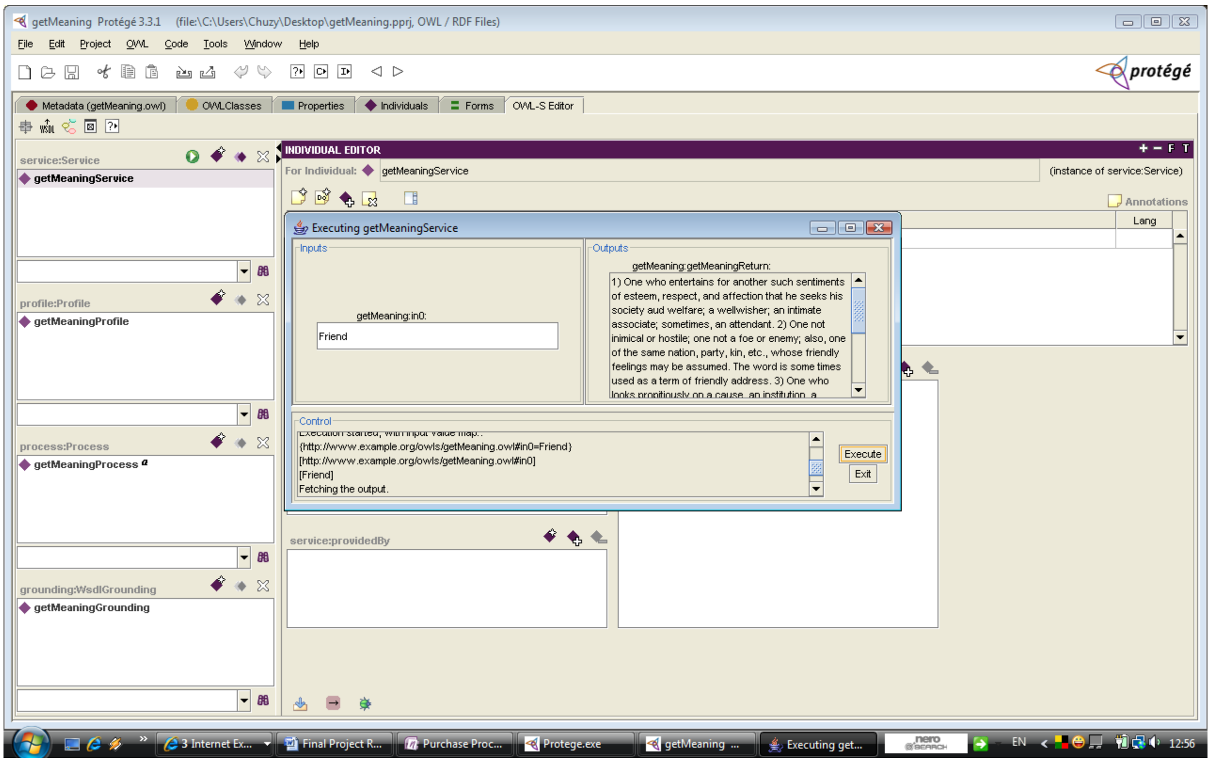


Figure 15: Input interface of the dictionary service – getMeaning.owl service.


getMeaning Protégé 3.3.1 (file:\C:\Users\Chuzy\Desktop\getMeaning.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Metadata (getMeaning.owl) OWLClasses Properties Individuals Forms OWL-S Editor
service:Service
getMeaningService
profile:Profile
getMeaningProfile
process:Process
getMeaningProcess
grounding:WsdlGrounding
getMeaningGrounding
INDIVIDUAL EDITOR
For Individual: getMeaningService
(instance of service:Service)
Annotations
Lang
service:providedBy
Executing getMeaningService
Inputs
getMeaning:in0:
Friend
Outputs
getMeaning:getMeaningReturn:
Control
Execute
Exit

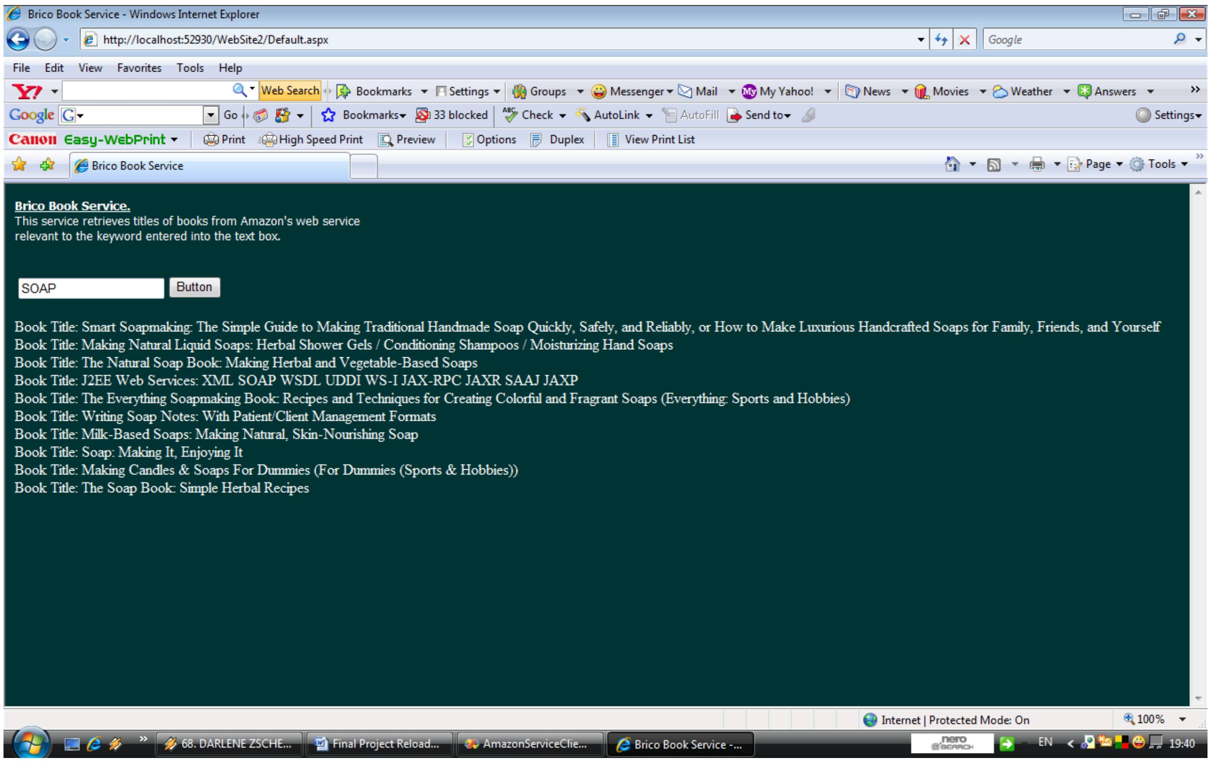
1) One who entertains for another such sentiments of esteem, respect, and affection that he seeks his society aud welfare; a wellwisher; an intimate associate; sometimes, an attendant. 2) One not inimical or hostile; one not a foe or enemy; also, one of the same nation, party, kin, etc., whose friendly feelings may be assumed. The word is some times used as a term of friendly address. 3) One who looks propitiously on a cause, an institution, a

```
Execution started, with input value map.:
{http://www.example.org/owls/getMeaning.owl#in0=Friend}
[http://www.example.org/owls/getMeaning.owl#in0]
[Friend]
Fetching the output.
```

**Figure 16: The meaning of the word 'Friend' fetched successfully.**

Figure 16 shows the output of the dictionary service when the input parameter was fed with the word 'Friend'. The service was able to retrieve the meaning from the web service as output.

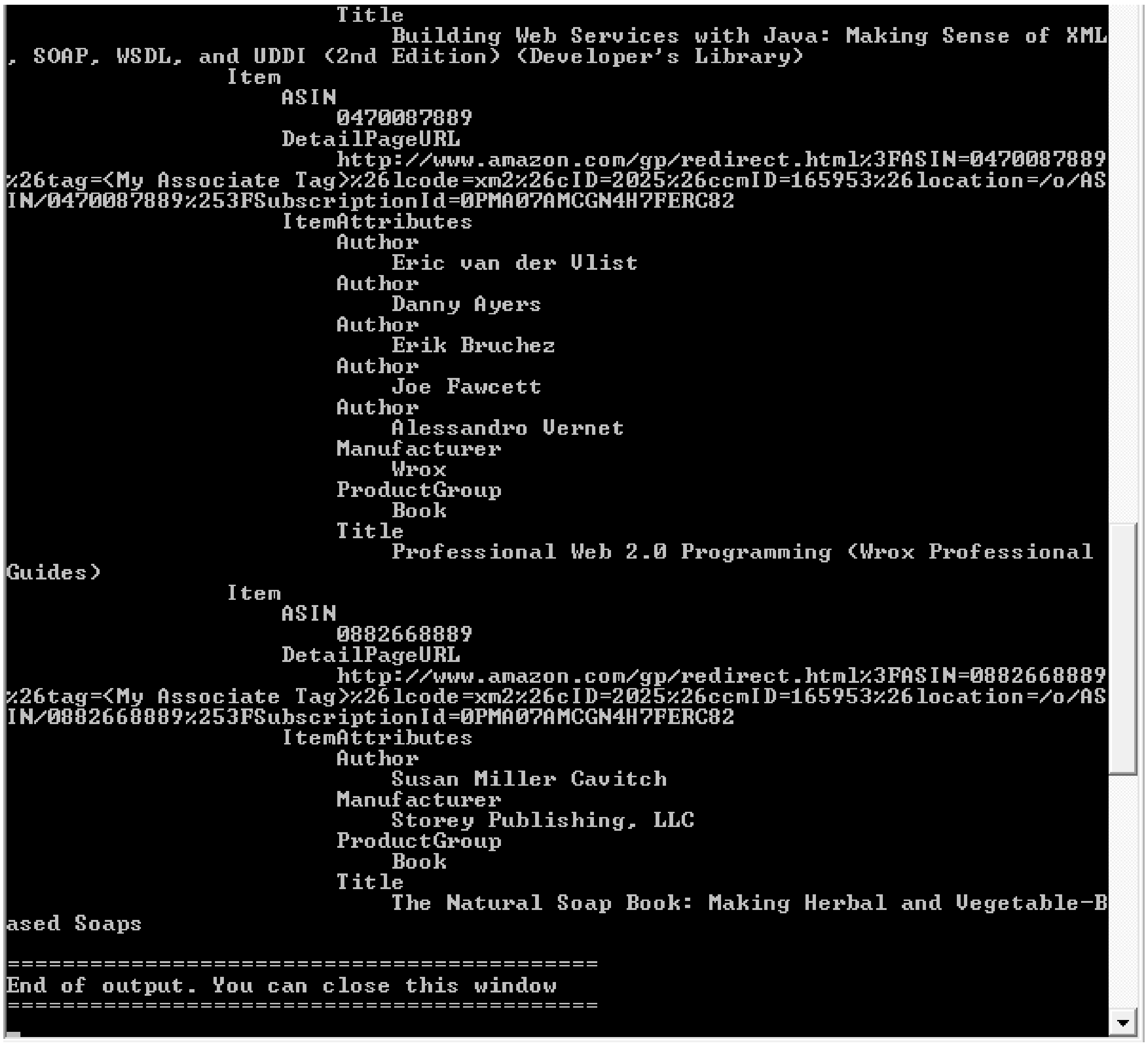
On the other hand, the C# web service technology is limited in providing support in service description. Hence, does not have the capacity to make more direct request. This was demonstrated using 'SOAP' as the keyword search request. The response was a mixture of SOAP technology and organic soap. This is shown in figure 17:


Brico Book Service - Windows Internet Explorer
http://localhost:52930/WebSite2/Default.aspx
File Edit View Favorites Tools Help
Brico Book Service
SOAP
Button
Internet | Protected Mode: On
100%
19:40

**Brico Book Service.**
This service retrieves titles of books from Amazon's web service
relevant to the keyword entered into the text box.

```
Book Title: Smart Soapmaking: The Simple Guide to Making Traditional Handmade Soap Quickly, Safely, and Reliably, or How to Make Luxurious Handcrafted Soaps for Family, Friends, and Yourself
Book Title: Making Natural Liquid Soaps: Herbal Shower Gels / Conditioning Shampoos / Moisturizing Hand Soaps
Book Title: The Natural Soap Book: Making Herbal and Vegetable-Based Soaps
Book Title: J2EE Web Services: XML SOAP WSDL UDDI WS-I JAX-RPC JAXR SAAJ JAXP
Book Title: The Everything Soapmaking Book: Recipes and Techniques for Creating Colorful and Fragrant Soaps (Everything: Sports and Hobbies)
Book Title: Writing Soap Notes: With Patient/Client Management Formats
Book Title: Milk-Based Soaps: Making Natural, Skin-Nourishing Soap
Book Title: Soap: Making It, Enjoying It
Book Title: Making Candles & Soaps For Dummies (For Dummies (Sports & Hobbies))
Book Title: The Soap Book: Simple Herbal Recipes
```

**Figure 17: Output of the search keyword 'SOAP' using the web service client application.**



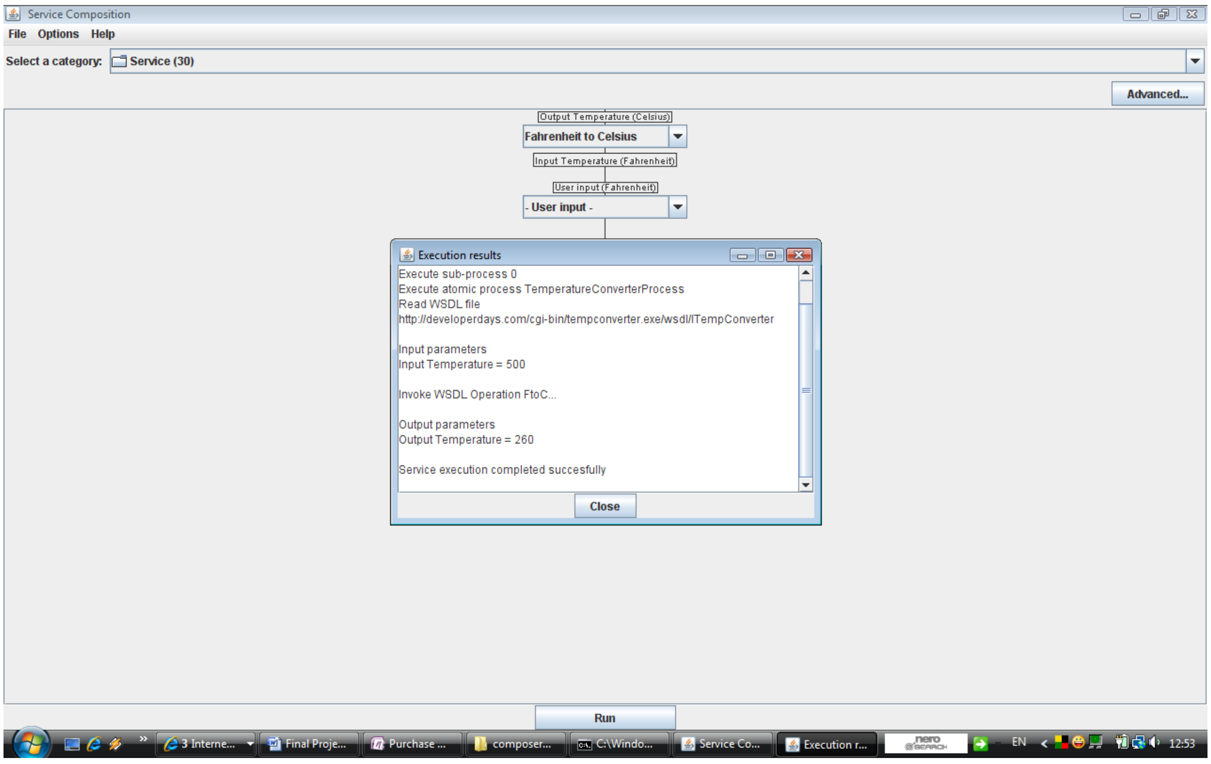
```
                        Title
                            Building Web Services with Java: Making Sense of XML
, SOAP, WSDL, and UDDI (2nd Edition) (Developer's Library)
                Item
                    ASIN
                        0470087889
                    DetailPageURL
                        http://www.amazon.com/gp/redirect.html%3FASIN=0470087889
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/0470087889%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                    ItemAttributes
                        Author
                            Eric van der Vlist
                        Author
                            Danny Ayers
                        Author
                            Erik Bruchez
                        Author
                            Joe Fawcett
                        Author
                            Alessandro Vernet
                        Manufacturer
                            Wrox
                        ProductGroup
                            Book
                        Title
                            Professional Web 2.0 Programming (Wrox Professional
Guides)
                Item
                    ASIN
                        0882668889
                    DetailPageURL
                        http://www.amazon.com/gp/redirect.html%3FASIN=0882668889
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/0882668889%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                    ItemAttributes
                        Author
                            Susan Miller Cavitch
                        Manufacturer
                            Storey Publishing, LLC
                        ProductGroup
                            Book
                        Title
                            The Natural Soap Book: Making Herbal and Vegetable-B
ased Soaps

===========================================
End of output. You can close this window
===========================================
```

**Figure 18: The output of the search keyword 'SOAP' using Amazon's Web Services API.**

Service Invocation: Due to the rich description of services in the semantic web technology, services could be invoked automatically. The composite process ontology describes how different atomic processes (where each atomic process could be an individual web service) are composed together using control constructs such as sequence, if-then-else etc; this determines the ordering and conditional execution of processes in the composition. The bulk of the automatic service invocation lies on the service process and grounding ontology. The process ontology also has preconditions which specify what must be true in order for an agent to execute a service. For example, a composite process could be – Buying a book which comprises of three atomic processes such as; locate book, payment method and delivery details.

So when the service, BookPurchase is executed, each atomic process is executed automatically. As earlier mentioned, the service could not be executed due to the limitations described earlier but it is clear that OWL-S Editor could enable these descriptions and allow automatic service invocation.

The web service composer tool was used to execute the temperature conversion web service. The service consumes the service of developersday.com (see figure 19).



```
Execute sub-process 0
Execute atomic process TemperatureConverterProcess
Read WSDL file
http://developerdays.com/cgi-bin/tempconverter.exe/wsdl/ITempConverter

Input parameters
Input Temperature = 500

Invoke WSDL Operation FtoC...

Output parameters
Output Temperature = 260

Service execution completed succesfully
```

**Figure 19: Web Service Composer tool executing the temperature conversion service.**

The tool also shows different atomic processes forming a composite process. It could be used to generate a workflow of web services where the inputs are entered manually and executed to generate an output. For example, a CVS Pharmacy Locator Service would allow a user to state the illness symptoms; enter his/her zip and state code. Then the Web Service will find a medicine suitable for the symptoms and locate the nearest pharmacy where the medicine is available. This is shown in the figure below:

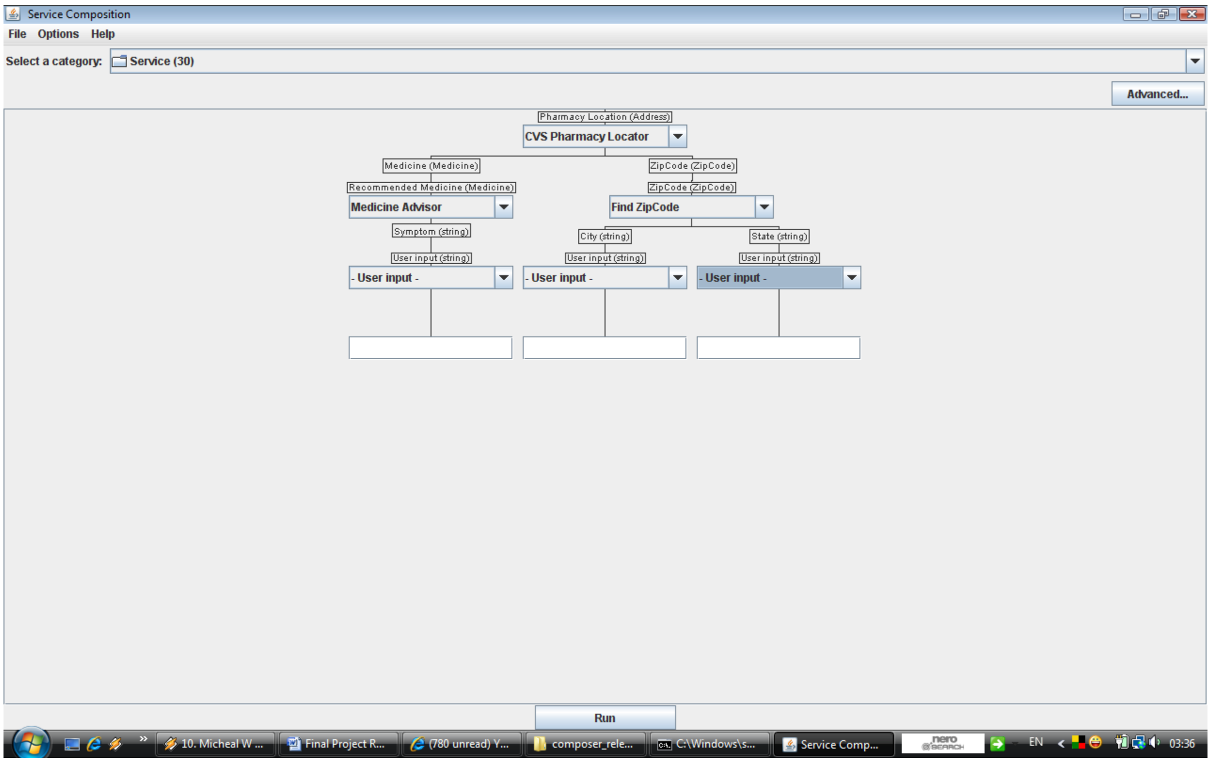


**Figure 20: Web Service Composer tool demonstrating composite process of a CVS Pharmacy Locator Service.**

However, the C# web service do not have the capability to carry out automatic service invocation, as human intervention is involved in each of the processes like, locating the book, method of payment and delivery details.

## 7.2 Limitations of the Semantic Web Service

The major challenge in implementing the semantic web services is the availability of working ontologies that could be used as examples; this made it really difficult to test the effectiveness of the OWL-S Editor. Most of the available OWL-S descriptions did not execute successfully when loaded into the OWL-S Editor. A good example was the CVS Pharmacy Locator Service; the execution was not successful, due to non-existence of real data. This agrees to what was discussed in the literature review; lack of ontologies for specific domains. In order to get the CVS Pharmacy Locator

Service working, it must be mapped unto central pharmacy ontology to allow for intelligent reasoning.

Another limitation was due to the 'skeletal' ontologies generated after importing WSDL files to OWL-S Editor; while executing the service ontology, errors were generated because the ontologies needed further descriptions (coding) which could be very difficult [1].

The OWL-S Editor is relatively new and still needs some further developments, although the tool has features; when fully developed it will be helpful in developing sophisticated semantic web services.

One of the reasons why OWL-S Editor is not yet embraced as a standard tool is because of its instability, as changes are being made to improve the tool. For example, the change from DAML-S 0.9 to OWL-S 1.0, OWL-S 1.0 to OWL-S 1.1. It is expected that, by the time OWL-S Editor matures, it will have the capability to realize the potentials of the semantic web services.

# 8. Conclusion

The motivation behind this research was to investigate the benefit of the semantic layer in the SOA infrastructure, hence it is necessary to review the aims of the research project and see if they have been addressed. The overall goal was to demonstrate and investigate how the interoperability challenge in the service discovery of the SOA infrastructure could be solved by introducing the semantic layer. The first objective was to investigate if the interoperability challenge actually exists. This objective was achieved by examining the limitations of XML based web service standards such as SOAP, WSDL and UDDI.

For true interoperability to be achieved there has to be automatic web service execution which involves an agent that automatically executes a discovered web service based on the semantic description of the web service. XML based web services do not have the required mark-up language capability that is rich enough to tell the agent the required input, the expected output and how the service would be executed automatically.

Another measure for true interoperability is automatic web service composition; this entails the automatic mapping and composition of an appropriate web service to perform a specific task. This information required to perform this task is described in the web service.

Currently, XML web services standards is not rich enough to give these descriptions, this was well backed up with literatures as well as demonstrated by implementing an XML web service using Microsoft C#. It is therefore fair to say, that this objective was accomplished.

The second objective was to investigate how automatic service composition and execution is enhanced by the semantic web service. This was accomplished by exploring the OWL-S Editor tool and its capabilities. It was discovered that, the tool has features that could enable a well described service using its service profile, service process and service grounding sections to describe a service and how the service could be automatically invoked. This was well investigated and it was found that, it was difficult to develop a multi-stage web service to demonstrate automatic service composition.

The third aim was to find out the possible challenge that could arise from the inclusion of the semantic layer into the SOA infrastructure. With Semantic SOA, web services are described in terms of: what kind of service it is; the kind of inputs it requires; the outputs it provides; the pre-conditions; the post-conditions and the data it consumes and provides. All these descriptions are made using ontologies, therefore it is right to say that the relationship between web services and ontologies is the 'backbone' of semantic web services.

However, in the course of the research, it was discovered that there are very few ontologies and limited networked ontologies which is a challenge in realizing the expectations of the semantic web service; as more ontologies in different domains have to be developed and well mapped. Even if web services are semantically annotated, and there are no corresponding ontologies to relate with, the services will not execute successfully – that was the case with most of the semantic web services that tested failed during the research. This aim was well addressed.

The last aim was to evaluate and compare the performance of the SOA infrastructure and the Semantic SOA infrastructure. This was achieved by comparing the two web services using service description and automatic service invocation as criteria. With the aid of the tools:  OWL-S Editor and web service composer, web services could be well described to aid automatic service discovery and invocation. The capabilities of the tools were explored and confirmed true that they could handle such descriptions. The key to the success of the semantic web services lies on the tools, although the tools are still in their developmental stage but when fully developed, the benefits of the semantic web services will be realised. On the other hand, the traditional web service (which was used to demonstrate the capabilities of the SOA infrastructure) is not technologically empowered to make semantic description of elements and attributes within the SOA infrastructure; this was demonstrated by executing a search on 'SOAP' with the intension of finding results on organic SOAP, meanwhile the output was a mixture of SOAP technology and organic SOAP. This aim was accomplished.

## 8.1 Personal Reflection

The two web services; the traditional and the semantic web service were developed to investigate the limitations of the traditional XML web service and the benefits of the semantic web service over the traditional web service. This was done to find out if what has been written in literatures about them is actually true. The applications helped to further understand the technologies behind the two frameworks. At first, a miniature XML C# web service was developed with a SQL database to find out more about the technology before the actual web service client (BricoBooks) was developed to consume the Amazon's web services.

The implementation of the semantic web service was a major challenge because there are no clear instructions as to which tool is best for the development. The first approach was to explore WSMO studio; it was installed as a plug-in to eclipse but it was later realised that WSMO model was complex because of a high learning curve. OWL-S Editor was found to be a better tool and more straight forward, although there were doubts about the efficiency of the tool because it was hard to find working examples of semantic web service to test the tool. More so, up till the point of concluding this work, there was only one tutorial available for OWL-S Editor and the examples that were used in the tutorials were not real examples, so it was hard to test if the tool was suitable or not.

However, despite the challenges faced in the implementations, the OWL-S tool was found to possess all the features (described in literatures) to semantically describe a web service and would be recommended for semantic web service deployment when the tool is fully developed. The only limitation would be availability of relevant ontologies as mentioned earlier.

With regard to the overall aim, this paper has been able to explore the answer to the research question - Is the introduction of the semantic layer to the SOA infrastructure of any benefit at all? The author now believes that it is worth introducing the semantic layer into the SOA infrastructure, although there are evident challenges which are common to every evolving technology but with time and efforts being currently invested by researchers and developers, the benefits of the Semantic SOA will be realised.

## 8.2 Further Work

There are areas for further implementation. A good place to start is to implement the Amazon's Semantic Web Service client to demonstrate a more concise search. The approach would be to first acquire Amazon's subscription ID; the main requirement in acquiring this is to have a real e-commerce web site that could be routed to Amazon's web site, then a market place domain will be given to the user. This service would help demonstrate the difference between traditional web service 'SOAP' search and semantic web service 'SOAP' search.

Since it has been confirmed that OWL-S Editor is suitable for semantic annotation of web services, further descriptions could be made to itemSearch.owl file generated from Amazon's WSDL document.

Another area worth researching is the evaluation of semantic web technology Models – OWL-S and WSMX. This will involve more research effort in exploring WSMO studio because despite its complexities, it has rich components that could help developing sophisticated semantic web services.

# Bibliography

[1] ALESSO, P., SMITH, C. Developing Semantic Web Services. 2005. A K Peters, Ltd. Massachusetts.

[2] ALEX, F., MACDONALD, M. Programming .Net Web Services. 2002. O'Reilly & Associates. United States Of America.

[3] AGRAWAL, R., BAYARDO, R., J., GRUHL, D., PAPADIDIMITRIOU, S. (2002). Computer Networks. Vinci: A Service-Oriented Architecture for Rapid Development of Web Applications [Online], 39 (5).

Available from: http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VRG-45KNFVW-3&_user=1682380&_coverDate=08%2F05%2F2002&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000011378&_version=1&_urlVersion=0&_userid=1682380&md5=d61874f0d07fa1cdcc00471f54fcfd23

[Accessed July 15th 2007]

[4] AGRE, G., ATANASOVA, T., NERN, J. Migrating from Web Services to Semantic Web Services, INFRAWEBS APPROACH. [Online]

Available from: http://www.infrawebs-eu.org/get_file.php?id=248

[Accessed July 18th 2007]

[5] Amazon Web Services. [online]

Available from: http://www.amazon.com/exec/obidos/tg/browse/-/3435361/sr=53-1/qid=1190915296/ref=tr_105761/105-0508407-5958863

[Accessed June 15th 2007]

[6] Amazon Web Services WSDL URL. [online]

Available from: http://ecs.amazonaws.com/AWSECommerceService/2007-07-16/AWSECommerceService.wsdl

[Accessed June 17th 2007]

[7] AOYAMA, M., WEERANWARANA, S., MARUYAMA, H., SZYPERSKI, C., SULLIVAH, K., LEA, D. (2002). Web Services Engineering: Promises and Challenges. Proceedings of the International Conference of Software Engineering, Orlando Florida. New York, USA: ACM Press, 647 - 648

[8] BERNERS-LEE, T., HENDLER, J., LASSILA, O. (2001). Scientific American. The Semantic Web [online].

Available from: http://www.sciam.com/article.cfm?articleID=00048144-10D2-

1C70-84A9809EC588EF21&pageNumber=2&catID=2

[Accessed July 20th 2007]

[9] BOX, D. (2000). MSDN Magazine. A Young Person's Guide to the Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages [online], 15(3).

Available from: http://msdn.microsoft.com/msdnmag/issues/0300/soap/

[Accessed July 27th 2007]

[10] BIH, J. (2006). Ubiquity: Service Oriented Architecture; A New Paradigm to Implement Dynamic E-business Solutions. [Online], 7(30).

Available From: http://www.acm.org/ubiquity/views/pf/v7i30_soa.pdf

[Accessed July 12th 2007]

[11] BITPIP. (2007). Software As A Service. [Online]

Available From: http://www.bitpipe.com/tlist/Software-as-a-Service.html

[Accessed July 17th 2007]

[12] BRITT, P., (2007). KMWorld Magazine: Service Oriented Architecture. [online], 16(2)

Available From: http://www.kmworld.com/Articles/ReadArticle.aspx?ArticleID=19134

[Accessed July 13th 2007]

[13] CANNATARO, M., TALIA, D. (2004). E-Science; Semantics and Knowledge Grids: Building the Next-Generation Grid. IEE Intelligent Systems. [Online]

Available from: http://ieeexplore.ieee.org/iel5/5254/28315/01265886.pdf

[Accessed August 16th 2007]

[14] CARDOSO, J. Semantic Web Services; Theory, Tools and Applications. [Online]

Available from: http://dme.uma.pt/jcardoso/Books/IDEA-SWTTA/index.php?file=17

[Accessed August 18th 2007]

[15] CHONG, F., CARRORO, G. 2006. Software As A Service (SAAS): An Enterprise Perspective [Online]. Microsoft Corporation.

Available From: http://msdn2.microsoft.com/en-us/architecture/aa905332.aspx

[Accessed July 19th 2007]

[16] CRAWFORD, C., H., BATE, G., P., CHERBAKOV, L., HOLLEY, K., TSOCANOS, C. (2005). IBM Systems Journal. Toward an on Demand Service-Oriented Architecture [online], 44(1).

Available from: http://www.research.ibm.com/journal/sj/441/crawford.html

[Accessed July 27th 2007]

[17] DAVIES, J., STUDER, R., WARREN, P. Semantic Web Technologies. Trends and Research in Ontology-based systems. (2006). John Wiley & Sons, Ltd. England.

[18] DIMITOV, M., SIMOV, A., KONSTANTINOV, M., CEKOV, L., MOMTCHEV, V. (2007). WSMO Studio User Guide V 1.25. [Online]

Available from: http://www.wsmostudio.org/doc/wsmo-studio-ug.pdf

[Accessed August 18th 2007]

[19] Data, Information and Process Integration with semantic Web Service (DIP). [Online]

Available from: http://dip.semanticweb.org/Impact.html

[Accessed September 18th 2007]

[20] DUKE, A., DAVIES, J., RICHARDSON, M. (2005). BT Technology Journal: Enabling a Scalable Service –Oriented Architecture with Semantic Web Services. 23[3]. [Online]

Available from:
http://www.springerlink.com/content/qn515r355585226t/fulltext.pdf

[Accessed 12th August, 2007]

[21] ERL, T. (2006). The Principles of Service-Orientation part 1 of 6: Introduction to service-orientation. [Online]

Available from:
http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1165286,00.html

[Accessed July 15th 2007]

[22] ERL, T. (2006). The Top 5 SOA Adoption Pitfalls. [Online]

Available from:

http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1195738,00.html

[Accessed July 15th 2007]

[23] European Semantic System Initiative (ESSI). [Online]

Available from: www.sdk-cluster.org

[Accessed September 15th 2007]

[24] ESSI WSMO Working Group. [Online]

Available from: www.wsmo.org

[Accessed September 15th 2007]

[25] FIELDING, R, T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine. [Online].

Available from:
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[Accessed August 25th 2007]

[26] FREIVALD, J., (2006). iWay SOA Middleware: An Agile Framework for Fast, Flexible, Low-Risk Service Deployments. [Online]

Available from:
http://www.iwaysoftware.com/products/whitepapers/pdf/iWaySOA_wp.pdf

[Accessed on July 12th 2007]

[27] GRAHAM, S., DAVIS, D., SIMEONOV, S., DANIELS, G., BRITTENHAM, P., NAKAMURA, Y., FREMANTLE, P., KONIG, D., ZENTNER, C. Building Web Services with Java – Making sense of XML, SOAP, WSDL, and UDDI. 2nd ed.2005. Sams Publishing. United States

[28] GRAPHVIZ Download Page. [Online]

Available from: http://www.graphviz.org/Download..php

[Accessed August 15th 2007]

[29] GROSSO, W. (2003). CORBA, I Loved Thee Well. [Online]

Available from: http://www.oreillynet.com/onjava/blog/2003/04/corba_i_loved_thee_well.htm

[Accessed July 3rd 2007]

[30] GRUBER, T. (1993). International Journal Human-Computer Studies. Towards Principles for the design of ontologies used for knowledge sharing. [Online]

Available from: http://tomgruber.org/writing/onto-design.pdf

[Accessed August 20th 2007]

[31] HALLER, A., CIMPIAN, E., MOCAN, A., OREN, E., BUSSLER, C. (2005). Digital Enterprise Research Institute (DERI); WSMX – A Semantic Service-Oriented Architecture. [online]

Available from: http://www.deri.ie/fileadmin/documents/wsmx-icws2005.pdf

[Accessed 12th August 2007]

[32] HENNING, M. (2006). The Rise and Fall of CORBA. [Online]

Available From: http://www.zeroc.com/documents/riseAndFallOfCorba.pdf

[Accessed August 1st 2007]

[33] IBM EXECUTIVE BRIEF (2004). Meeting the Challenges of Today's Oil and Gas Exploration and Production Industry.

Available from: http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-3882-meeting-challenges-oil-gas-exploration.pdf

[Accessed August 10th 2007]

[34] IBM CORPORATION (2005). Introduction to the Value and Governance Model of Service-Oriented Architecture

Available from: https://www6.software.ibm.com/developerworks/education/websphere/wbt/sw717/htmls/SW717/pdfs/SW717Topic4.pdf

[Accessed July 28th 2007]

[35] IBM PRESS ROOM. (2006). IBM Unveils Software and ISV Initiatives to Handle Surge in Mainframe Transactions [Online]

Available from: http://www-03.ibm.com/press/us/en/pressrelease/19620.wss

[Accessed August 25th 2007]

[36] JON, S., (1996). CORBA Fundamentals and Programming: John Wiley & Sons, Inc: Canada.

[37] JASON, B. (2006). SOA Governance and the Butterfly Effect. [Online]

Available From:

http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1161716_tax305384,00.html

[Accessed August 11th 2007]

[38] JASON, B. (2006). Making Sense of SOA Governance, Service Lifecycle Management, Registries and Repositories. Zapthink White Paper. [online]

Available from: http://www.zapthink.com/report.html?id=WP-0139

[Accessed August 11th 2007]

[39] KAPLAN, J., F. (2006). Sourcing and Vendor Relationship Advisory Service: SaaS Appeal Growing. [Online], 7(21).

Available from: http://www.cutter.com/promotions/srcu0621/srcu0621.pdf

[Accessed August 24th 2007]

[40] KELLY, A., D (2006). What You Need To Know About SOA. [Online]

Available from: http://www.oracle.com/technologies/soa/transitioning-to-soa.pdf

[Accessed July 25th 2007]

[41] KINCORA, M. 2006. IT/BUSINESS STRATEGIES**:** ASP vs. SaaS: What's the difference? [Online]

Available From:
http://searchcio.techtarget.com/tip/0,289483,sid19_gci1216679,00.html

[Accessed July 24th 2007]

[42] LIU, A. and GORTON, I. (2005). The Architecture Journal: Process and Criteria for Evaluating Services-Based Integration Technologies. Journal [5] [online].

Available from: http://msdn2.microsoft.com/en-us/arcjournal/aa480046.aspx

[Accessed August 27th 2007]

[43] MAHMOUD, Q., H. (2005).Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI). [Online]

Available From:

http://java.sun.com/developer/technicalArticles/WebServices/soa/

[Accessed August 12th 2007]

[44] MAKOLA, D., SIM, Y., W., WANG, C., GILBERT, L., GRANGE, S., WILLS, G. (2006). A Service-Oriented Architecture for a Collaborative Orthopaedic Research Environment. University of Southampton [online]. Available from: http://eprints.ecs.soton.ac.uk/12898/01/WWW2006Core.pdf

[Accessed August 28th 2007]

[45] MICROSOFT. (2006) Enabling "Real World SOA" through the Microsoft Platform [Online]. Microsoft White Paper.

Available From:

http://download.microsoft.com/download/b/4/d/b4db580a-0361-4907-9a6e-9d2866d8b581/Real%20World%20SOA.doc

[Accessed 15 July 2007]

[46] MINDSWAP Dictionary Service WSDL URL. [Online].

Available from:
http://www.mindswap.org/axis/services/DictionaryService?wsdl

[Accessed September 10th 2007]

[47] MINDSWAP Service Composer. [Online]

Available from: http://www.mindswap.org/~evren/composer/

[Accessed September 2nd 2007]

[48] MOINUDDIN, M., (2007) An Overview of Service–Oriented Architecture in Retail [Online]. Microsoft Corporation

Available From: http://msdn2.microsoft.com/en-us/architecture/bb264584.aspx

[Accessed on July 10th 2007]

[49] MONDAY, B., P. Web Service Patterns: Java Edition. 2003. United States.

Apress.

[50] NADHAN, E., G. (2004). The Architecture Journal: Service Oriented Architecture, Implementation Challenges. Journal [2] [online].

Available from: http://msdn2.microsoft.com/en-us/library/aa480029.aspx

[Accessed August 17th 2007]

[51] NASCIO. (2006). Service Oriented Architecture: An Enabler of the Agile Enterprise in State Government. [Online]. NASCIO Research Brief

Available From: http://colab.cim3.net/file/work/Expedition_Workshop/2007-01-23_CollaborativeOrganizingWorkshopToPlanFutureWorkshops/Sweden_SOA_ResearchBrief__2007_01_23.pdf

[Accessed July 25th 2007]

[52] NATIS, Y., V. (2003). Service Oriented Architecture Scenario. [online]

Gartner Research Publication. Available from: http://www.gartner.com/resources/114300/114358/114358.pdf

[Accessed August 27th 2007]

[53] OASIS REFERENCE MODEL FOR SEMANTIC SERVICE ORIENTED ARCHITECTURE. (2007)

Available from: http://66.102.9.104/search?q=cache:XwUHvVuGT6sJ:www.oasis-open.org/committees/download.php/23108/SEE%2520SemanticSOA%2520Reference%2520Model.doc+REFERENCE+MODEL+FOR+SEMANTIC+SERVICE+ORIENTED+ARCHITECTURE+working+draft+0.1,+21+March+2007&hl=en&ct=clnk&cd=1&client=opera

[Accessed August 6th 2007]

[54] ODE SWS Tools [Online]

Available from: http://kw.dia.fi.upm.es/odesws/content.htm

[Accessed August 19th 2007]

[55] OPSOURCE. (2007) [Online]

Available From: http://www.opsource.net/saas/whatisit.shtml

[Accessed July 19th 2007]

[56] ORACLE. 2006. Bringing SOA Value Patterns to Life. [Online]

Available from: http://www.oracle.com/technologies/soa/soa-value-patterns.pdf

[Accessed July 30th 2007]

[57] OWL-S Editor Tools [Online]

Available from: http://www.daml.org/services/owl-s/tools.html

[Accessed July 19th 2007]

[58] PAPAZOGLOU, M. P and GEORGAKOPOULOS, D. (2003). Communications of the ACM: Service Oriented Computing [Online]. 46(10). Available from: http://steelhead.seattleu.edu/bkim/courses/ecis469sp05/LectureNotes/socIntro.pdf

[Accessed July 27th 2007)

[59] PARASTATIDIS, S. and WEBBER, J. (2005). The Architecture Journal: Service-Oriented, Distributed, High-Performance Computing [online]. Journal [5].

Available from: http://msdn2.microsoft.com/en-us/arcjournal/aa480054.aspx

[Accessed July 27th 2007]

[60] PEDRINACI, C., MORAN, M., NORTON, B. (2006). Towards a Semantic Event-Based Service-Oriented Architecture [Online].

Available from: http://kmi.open.ac.uk/projects/dip/resources/ISWC2006/swese2006Final.pdf

[Accessed August 28th 2007]

[61] PETINOT, Y., GILES, C., L., BHATNAGAR, V., PRADEEP, B., HAN, H., COUNCIL, I. (2004). A Service –Oriented Architecture for Digital Libraries.

Available from: http://clgiles.ist.psu.edu/papers/ICSOS-2004-service-oriented-arch.pdf

[Accessed August 10th 2007]

[62] PROTEGE Download Page. [Online]

Available from: http://protege.stanford.edu/download/registered.html

[Accessed August 10th 2007]

[63] PULIER, E., TAYLOR, H. (2006). Understanding Enterprise SOA (Excerpt)

[Online]

Available From: http://www.soasoftpressroom.com/UnderstandingEnterpriseSOAexcerpt.pdf

[Accessed July 25th 2007]

[64] PULIER, E., TAYLOR, H. (2006). Understanding Enterprise SOA (Sample Chapter) [Online].

Available From: http://www.manning-source.com/books/pulier/pulier_ch09.pdf

[Accessed July 27th 2007]

[65] SAADATI, S., DENKER, G. OWL-S Editor [online].

Available from: http://owlseditor.semwebcentral.org/documents/tutorial.pdf

[Accessed August 2nd 2007]

[66] SESSIONS, R. (2003). What is a Service-Oriented Architecture? ObjectWatch Newsletter (45) [Online]

Available from: http://www.objectwatch.com/newsletters/issue_45.htm

[Accessed April 24th 2007]

[67] SNELL, J., TIDWELL, D., KULCHENKO, P. Programming Web Services with SOAP. 2002 O'Reilly & Associates, Inc., United States

[68] SOA SOFTWARE, 2005. Security Issues in the SOA. [Online]

Available from: http://www.soasoftpressroom.com/SOASoft_Security_Issues_in_SOA_Whitepaper.pdf

[Accessed August 12th 2007]

[69] SOLLAZZO, T., HANDSCHUH, S., STAAB, S., FRANK, M. Semantic Web Service Architecture – Evolving Web Service Standards towards the Semantic Web. [Online]

Available from: http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/sub-flairs2002.pdf

[Accessed August 17th 2007]

[70] SORENSEN, C. Object-Oriented Systems: A Comparison of Distributed Object

Technologies. [Online].

Available from: http://www.idi.ntnu.no/emner/dt8100/Essay2001/essay2001-calle.pdf

[Accessed July 29th 2007]

[71] SPRINGCM. (2006). How Do SAAS and ASP Models Differs? [Online]

Available From: http://images.vertmarkets.com/CRLive/files/downloads/64f2ad4a-4101-49a5-9c1a-dc206e59c2e3/ASPsandSAAS.pdf

[Accessed July 20th 2007]

[72] SPROTT, D. (2005). Service Oriented Architecture: An Introduction for Managers [Online]. CBDI Report.

Available From: http://www-935.ibm.com/services/us/gbs/bus/pdf/soa-cbdi-report-2004-july.pdf

[Accessed July 10th 2007]

[73] STANTCHEV, V., SCHERZ, M. (2003). Challenges of Service–Oriented Architecture. [Online]

Available from: http://www2.informatik.hu-berlin.de/~vstantch/pubs/stantchev_scherz_soa_1104.pdf

[Accessed July 14th 2007]

[74] Temperature Converter Web Service WSDL Document. [Online]

Available from: http://developerdays.com/cgi-bin/tempconverter.exe/wsdl/ITempConverter

[Accessed September 15th 2007]

[75] TERRAR, D., (2006). SAAS IS NOT ASP REBRANDED. [Online]

Available from: http://biztwozero.com/btz/2006/08/04/saas-is-not-asp-rebranded/

[Accessed July 23rd 2007]

[76] WEBMETHODS. (2006). SOA Governance: Enabling Sustainable Success with SOA [Online]

Available from:

http://www1.webmethods.com/PDF/whitepapers/SOA_Governance.pdf

[Accessed July 15th 2007]

[77] VERMA, K., SIVASHANMUGAM, K., SHETH, A., PATIL, A., OUNDHAKAR, S., MILLER, J. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publications and Discovery of Web Services [online].

Available from: http://lsdis.cs.uga.edu/lib/download/VSS+03-TM06-003-METEOR-S-WSDI.pdf

[Accessed August 20th 2007]

[78] VETERE, G and LENZERINI, M. (2005). IBM Systems Journal. Models for Semantic Interoperability in Service-Oriented Architectures [online]. 44(4)

Available from: http://www.research.ibm.com/journal/sj/444/vetere.html

[Accessed July 28th 2007]

[79] WEST, K. (2004). SOA VS CORBA: Not Your Father's Architecture. [Online]

Available From:
http://microsites.cmp.com/documents/s=9063/ilc1086397013472/

[Accessed July 1st 2007]

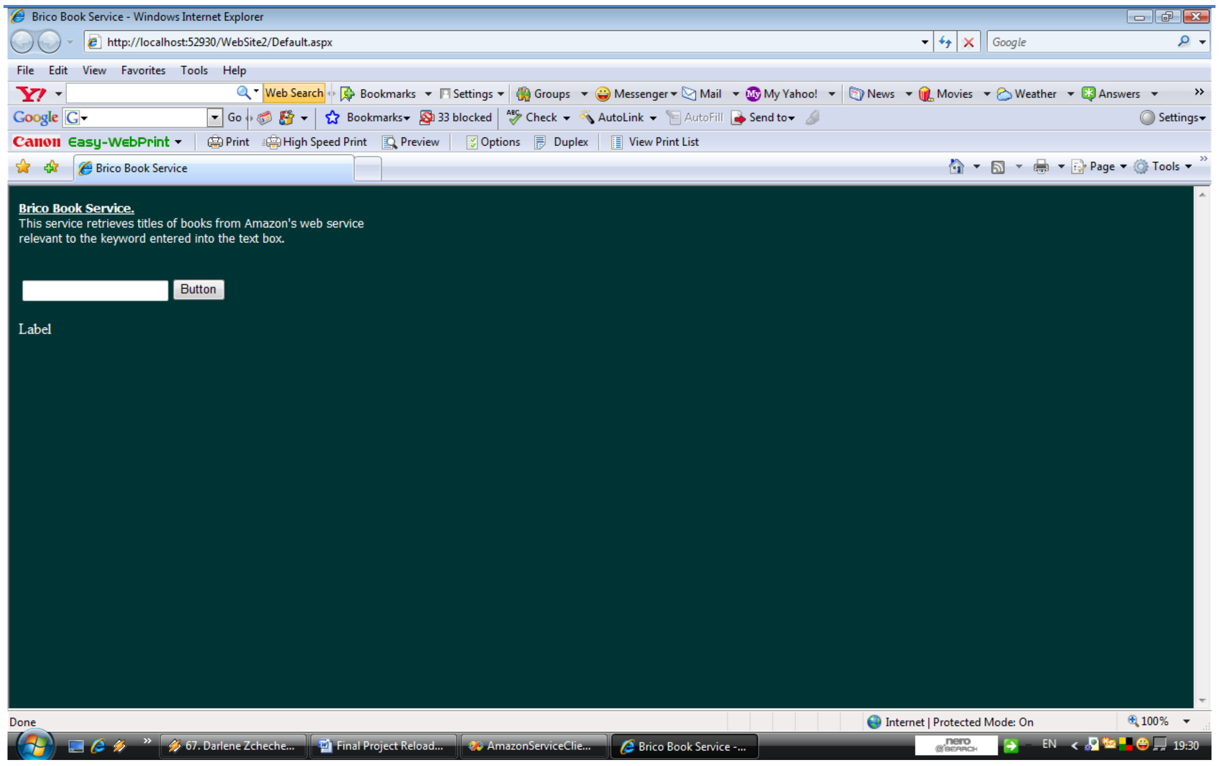
[80] WSMO Studio Download Page. [Online]

Available from: http://sourceforge.net/project/showfiles.php?group_id=119791

[Accessed July 19th 2007]

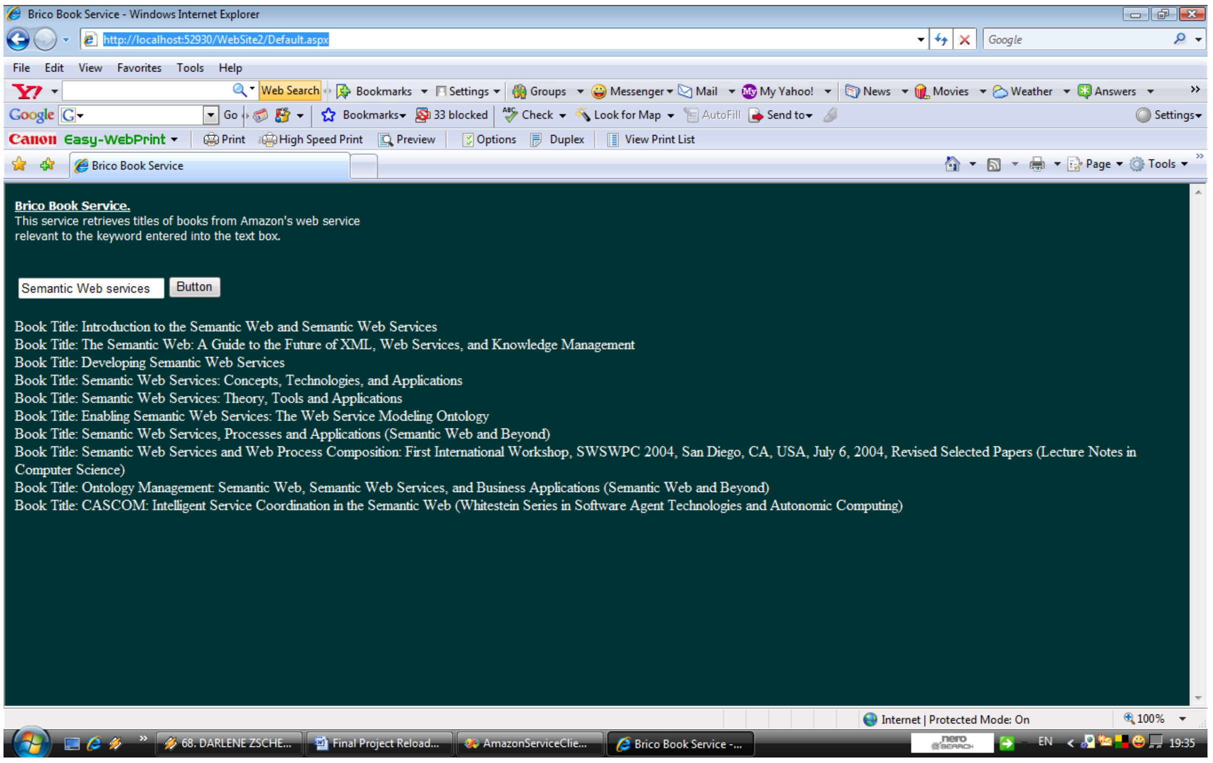
# Index of Figures

# Appendix A: C# Web Service Interface


Brico Book Service - Windows Internet Explorer
http://localhost:52930/WebSite2/Default.aspx
Button
Label
Done
Internet | Protected Mode: On

**Brico Book Service.**
This service retrieves titles of books from Amazon's web service relevant to the keyword entered into the text box.

1. The user could search books available on any topic he/she desires to find.

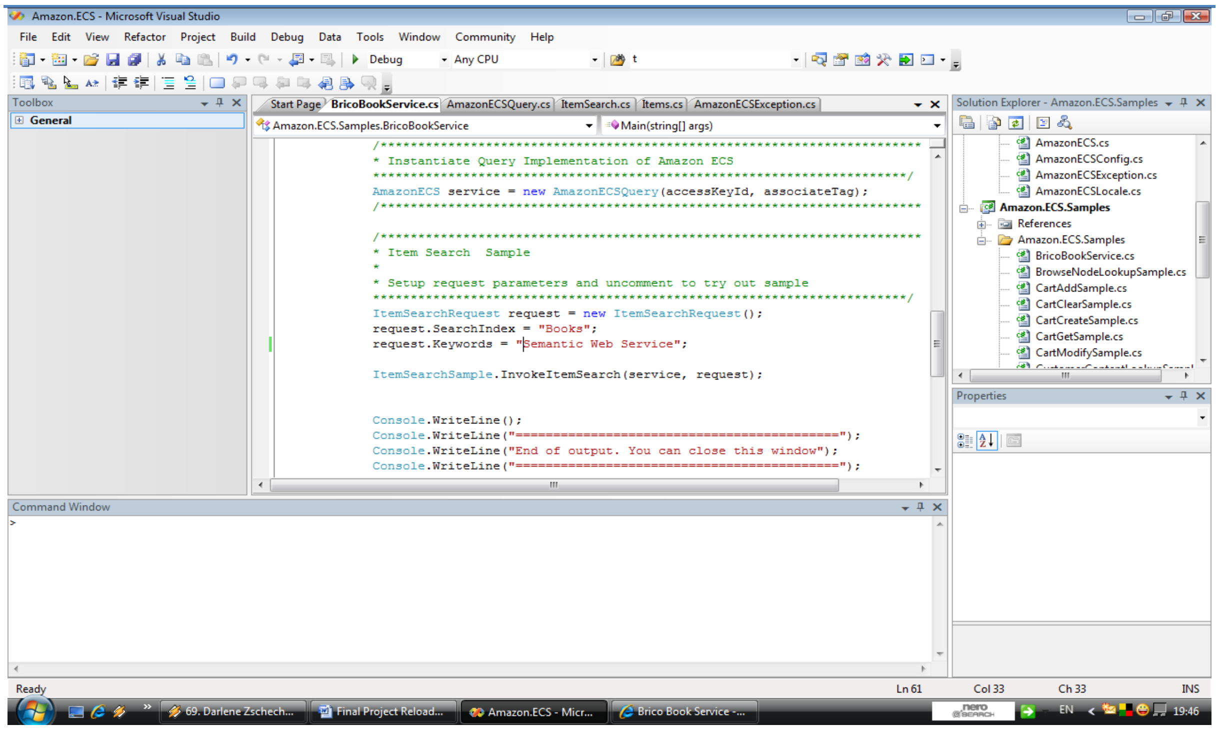

Brico Book Service - Windows Internet Explorer
http://localhost:52930/WebSite2/Default.aspx
Semantic Web services
Button
Internet | Protected Mode: On

**Brico Book Service.**
This service retrieves titles of books from Amazon's web service relevant to the keyword entered into the text box.

```
Book Title: Introduction to the Semantic Web and Semantic Web Services
Book Title: The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management
Book Title: Developing Semantic Web Services
Book Title: Semantic Web Services: Concepts, Technologies, and Applications
Book Title: Semantic Web Services: Theory, Tools and Applications
Book Title: Enabling Semantic Web Services: The Web Service Modeling Ontology
Book Title: Semantic Web Services, Processes and Applications (Semantic Web and Beyond)
Book Title: Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004, Revised Selected Papers (Lecture Notes in
Computer Science)
Book Title: Ontology Management: Semantic Web, Semantic Web Services, and Business Applications (Semantic Web and Beyond)
Book Title: CASCOM: Intelligent Service Coordination in the Semantic Web (Whitestein Series in Software Agent Technologies and Autonomic Computing)
```

2. The user decides to search books available on Semantic Web Services and the available books were shown.

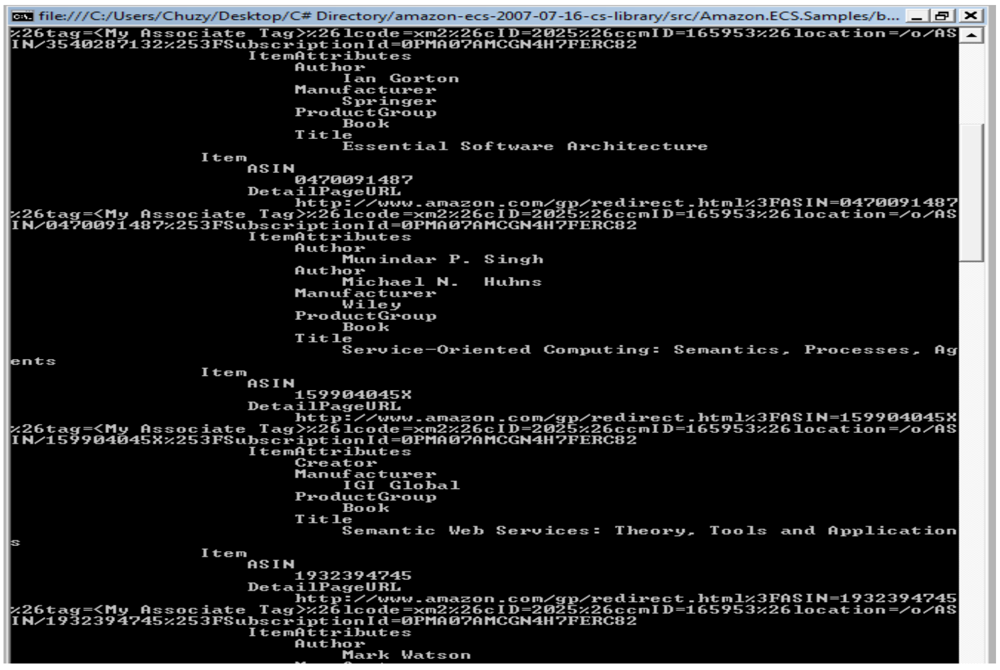
# Appendix B: Amazon's API Interface for C# Web Service



```
/************************************************************************
* Instantiate Query Implementation of Amazon ECS
***********************************************************************/
AmazonECS service = new AmazonECSQuery(accessKeyId, associateTag);
/************************************************************************

/************************************************************************
* Item Search  Sample
*
* Setup request parameters and uncomment to try out sample
***********************************************************************/
ItemSearchRequest request = new ItemSearchRequest();
request.SearchIndex = "Books";
request.Keywords = "Semantic Web Service";

ItemSearchSample.InvokeItemSearch(service, request);


Console.WriteLine();
Console.WriteLine("==========================================");
Console.WriteLine("End of output. You can close this window");
Console.WriteLine("==========================================");
```

1. The user enters the topic, say 'Semantic Web Service' he/she wishes to search for available books.

-------------------------------------------------------------------------------------------------------------



```
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/354028713 2%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                ItemAttributes
                    Author
                        Ian Gorton
                    Manufacturer
                        Springer
                    ProductGroup
                        Book
                    Title
                        Essential Software Architecture
            Item
                ASIN
                    0470091487
                DetailPageURL
                    http://www.amazon.com/gp/redirect.html%3FASIN=0470091487
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/0470091487%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                ItemAttributes
                    Author
                        Munindar P. Singh
                    Author
                        Michael N.  Huhns
                    Manufacturer
                        Wiley
                    ProductGroup
                        Book
                    Title
                        Service-Oriented Computing: Semantics, Processes, Ag
ents
            Item
                ASIN
                    159904045X
                DetailPageURL
                    http://www.amazon.com/gp/redirect.html%3FASIN=159904045X
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/159904045X%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                ItemAttributes
                    Creator
                    Manufacturer
                        IGI Global
                    ProductGroup
                        Book
                    Title
                        Semantic Web Services: Theory, Tools and Application
s
            Item
                ASIN
                    1932394745
                DetailPageURL
                    http://www.amazon.com/gp/redirect.html%3FASIN=1932394745
%26tag=<My Associate Tag>%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/AS
IN/1932394745%253FSubscriptionId=0PMA07AMCGN4H7FERC82
                ItemAttributes
                    Author
                        Mark Watson
```

2. The data on available books are retrieved.

# Appendix C: Amazon's API Web Service Source Code

**BricoBookService.cs**

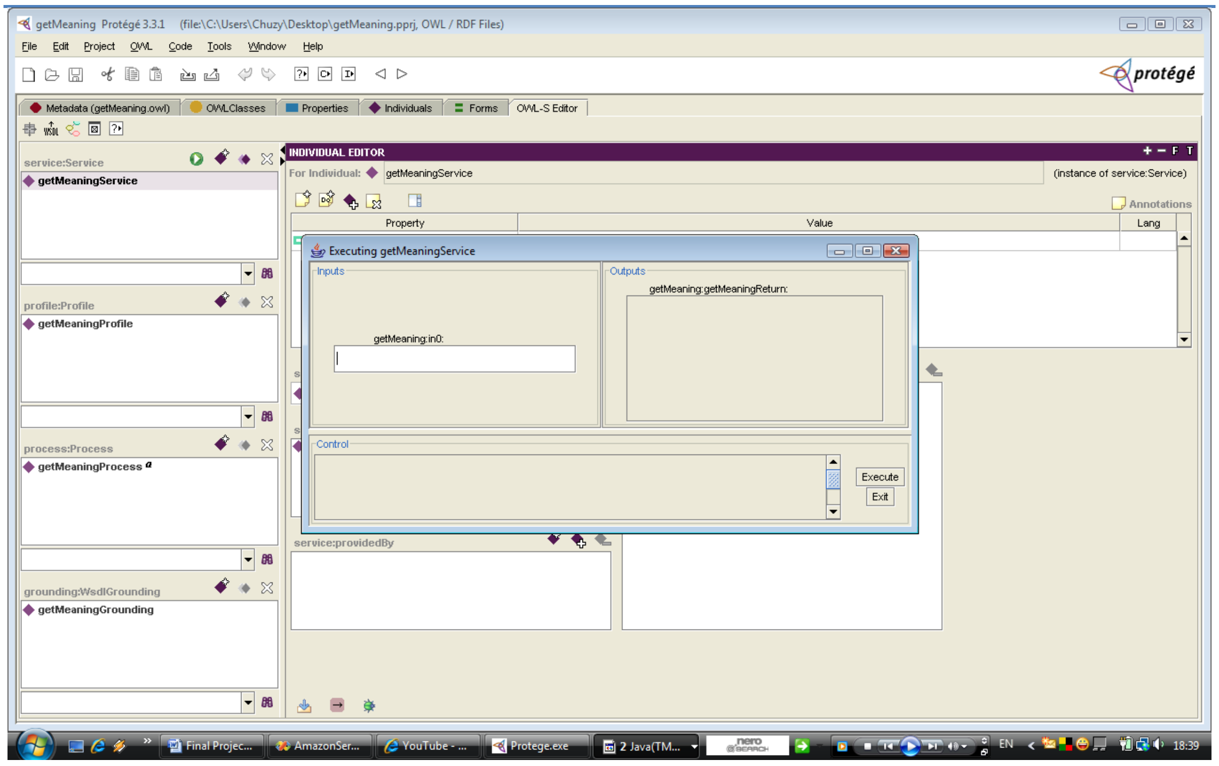
```
using System;
using System.Xml.Serialization;
using System.Collections.Generic;
using Amazon.ECS;
using Amazon.ECS.Query;
using Amazon.ECS.Mock;
using Amazon.ECS.Model;

namespace Amazon.ECS.Samples
{

    /// <summary>
    /// Amazon ECS  Samples
    /// </summary>
    public class BricoBookService
    {

       /**
        * Samples for Amazon ECS functionality
        */
        public static void Main(string [] args)
        {

Console.WriteLine("===========================================");
            Console.WriteLine("Welcome to Amazon E-Commerce Service ");
            Console.WriteLine("Demonstration for Brico Book Service!");

Console.WriteLine("===========================================");
            Console.WriteLine(" - This Web Service Client will consume the
Amazon Web Service");
            Console.WriteLine(" - using AWS Access Key ID ");

            Console.WriteLine(" - This Web Service Client will retrieve
information ");
            Console.WriteLine(" - based on the Search parameters contained
in this code");
            Console.WriteLine(" - The information will be displayed on the
console.");

Console.WriteLine("===========================================");
            Console.WriteLine(" This service is only for demonstration
purposes");

Console.WriteLine("===========================================");
            Console.WriteLine("Samples Output");

Console.WriteLine("===========================================");
            Console.WriteLine();


/***********************************************************************
            * Access Key ID obtained from:
            * http://aws.amazon.com

***********************************************************************/
            String accessKeyId = "0PMA07AMCGN4H7FERC82";
            String associateTag = "<My Associate Tag>";


/*************************************************************************
            * Instantiate Query Implementation of Amazon ECS

************************************************************************/
            AmazonECS    service    =    new    AmazonECSQuery(accessKeyId,
associateTag);

/*************************************************************************


/*************************************************************************
            * Item Search  Sample
            *
            * Setup request parameters and uncomment to try out sample

************************************************************************/
            ItemSearchRequest request = new ItemSearchRequest();
            request.SearchIndex = "Books";
            request.Keywords = "Semantic Web Service";

            ItemSearchSample.InvokeItemSearch(service, request);


            Console.WriteLine();

Console.WriteLine("===========================================");
            Console.WriteLine("End of output. You can close this window");

Console.WriteLine("===========================================");

            System.Threading.Thread.Sleep(500000);
        }

    }
}
```

# Appendix D: C# Web Service Client Source Code

**BricoBooks.aspx.cs**

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using com.amazon.webservices;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {


AWSECommerceService aws = new AWSECommerceService();
ItemSearchRequest request = new ItemSearchRequest();

request.SearchIndex = "Books";
request.Power = "title:" + TextBox1.Text;
request.ResponseGroup = new string[] { "Small" };
request.Sort = "salesrank";

ItemSearchRequest[] requests = new ItemSearchRequest[] { request };

ItemSearch itemSearch = new ItemSearch();
//itemSearch.AssociateTag = "myassociatetag-20";
itemSearch.AWSAccessKeyId = "0PMA07AMCGN4H7FERC82";
itemSearch.Request = requests;

try
{
 ItemSearchResponse response = aws.ItemSearch(itemSearch);
 Items info = response.Items[0];
 Item[] items = info.Item;
 Label1.Text = "";
 for (int i = 0; i < items.Length; i++)
 {
   Item item = items[i];
   Label1.Text += "Book Title: " + item.ItemAttributes.Title + "<br />";
  }
}
catch (Exception ex)
{
 Label1.Text = (ex.ToString());
}
    }
}
```

# Appendix E: Semantic Web Service User Interface


getMeaning Protégé 3.3.1 (file:\C:\Users\Chuzy\Desktop\getMeaning.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Metadata (getMeaning.owl) OWLClasses Properties Individuals Forms OWL-S Editor
service:Service
profile:Profile
process:Process
grounding:WsdlGrounding
INDIVIDUAL EDITOR
Property Value Lang
Annotations
service:providedBy
Inputs
Outputs
Control
Execute
Exit
Final Projec... AmazonSer... YouTube - ... Protege.exe 2 Java(TM... EN 18:39

For Individual: getMeaningService (instance of service:Service)

getMeaningService

getMeaningProfile

getMeaningProcess

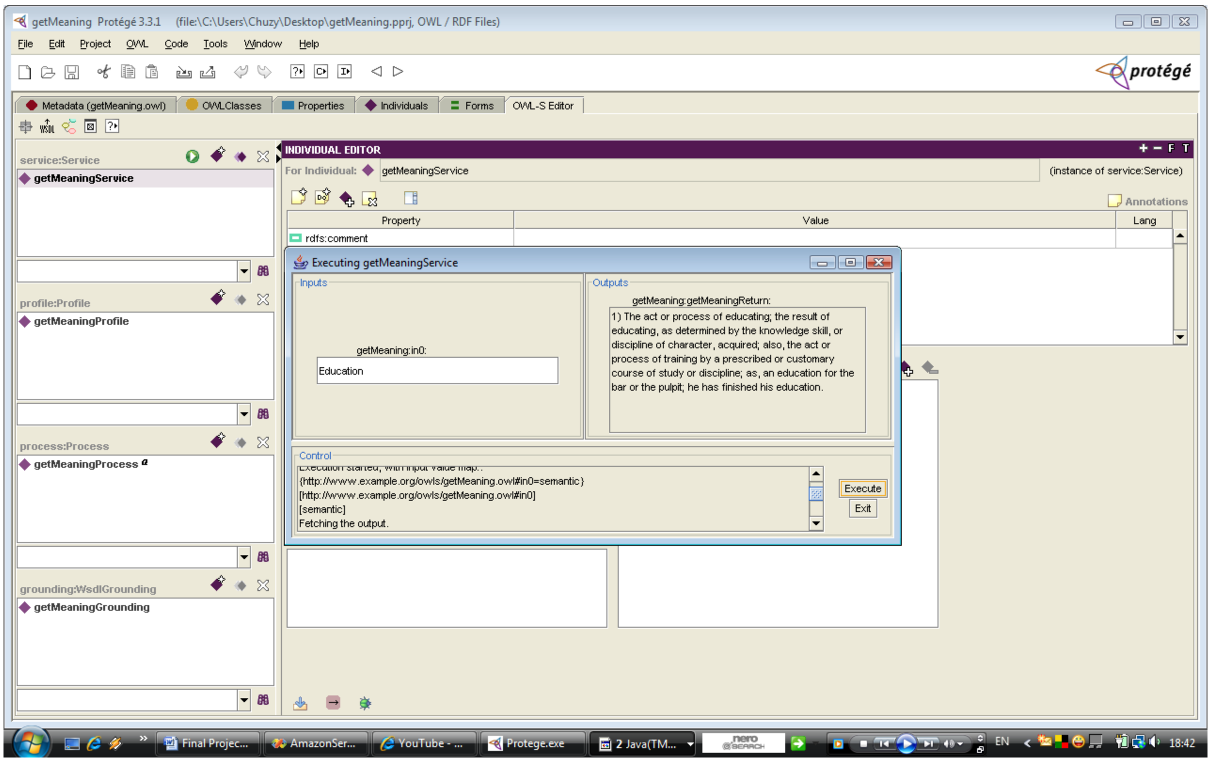
getMeaningGrounding

Executing getMeaningService

getMeaning:in0:

getMeaning:getMeaningReturn:

1. The user enters the word he/she wishes to search the meaning


getMeaning Protégé 3.3.1 (file:\C:\Users\Chuzy\Desktop\getMeaning.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Metadata (getMeaning.owl) OWLClasses Properties Individuals Forms OWL-S Editor
service:Service
profile:Profile
process:Process
grounding:WsdlGrounding
INDIVIDUAL EDITOR
Property Value Lang
Annotations
Inputs
Outputs
Control
Execute
Exit
Final Projec... AmazonSer... YouTube - ... Protege.exe 2 Java(TM... EN 18:42

For Individual: getMeaningService (instance of service:Service)

rdfs:comment

Executing getMeaningService

getMeaning:in0: Education

getMeaning:getMeaningReturn:

1) The act or process of educating; the result of educating, as determined by the knowledge skill, or discipline of character, acquired; also, the act or process of training by a prescribed or customary course of study or discipline; as, an education for the bar or the pulpit; he has finished his education.

```
Execution started, with input value map.:
{http://www.example.org/owls/getMeaning.owl#in0=semantic}
[http://www.example.org/owls/getMeaning.owl#in0]
[semantic]
Fetching the output.
```

2. The user enters the word ‘Education’ and the meaning is retrieved.

# Appendix F: Semantic Web Service Source Code

getMeaning.owl

```
<?xml version="1.0"?>

<rdf:RDF

    xmlns:service="http://www.daml.org/services/owl-s/1.2/Service.owl#"

    xmlns:swrl="http://www.w3.org/2003/11/swrl#"

    xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"

    xmlns:list="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

    xmlns:owl="http://www.w3.org/2002/07/owl#"

    xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#"

    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"

    xmlns:profile="http://www.daml.org/services/owl-s/1.2/Profile.owl#"

    xmlns="http://www.example.org/owls/getMeaning.owl#"

    xmlns:process="http://www.daml.org/services/owl-s/1.2/Process.owl#"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:grounding="http://www.daml.org/services/owl-s/1.2/Grounding.owl#"

    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"

  xml:base="http://www.example.org/owls/getMeaning.owl">

  <owl:Ontology rdf:about="">

    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>

    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Grounding.owl"/>

    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Service.owl"/>

    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Profile.owl"/>

  </owl:Ontology>

  <process:AtomicProcess rdf:ID="getMeaningProcess">

    <process:hasOutput>

      <process:Output rdf:ID="getMeaningReturn">

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

        >getMeaningReturn</rdfs:label>

        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>

      </process:Output>

    </process:hasOutput>

    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

    >getMeaningProcess</rdfs:label>

    <process:hasInput>

      <process:Input rdf:ID="in0">

        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

        >in0</rdfs:label>

      </process:Input>

    </process:hasInput>
    <service:describes>

     <service:Service rdf:ID="getMeaningService">

      <service:supports>

       <grounding:WsdlGrounding rdf:ID="getMeaningGrounding">

        <grounding:hasAtomicProcessGrounding>

         <grounding:WsdlAtomicProcessGrounding rdf:ID="getMeaningAtomicProcessGrounding">

          <grounding:wsdlOutputMessage
  rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"


  >http://www.mindswap.org/axis/services/DictionaryService#getMeaningResponse</grounding:wsdl
  OutputMessage>

          <grounding:wsdlOutput>

           <grounding:WsdlOutputMessageMap>

            <grounding:owlsParameter rdf:resource="#getMeaningReturn"/>

            <grounding:wsdlMessagePart
  rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"


  >http://www.mindswap.org/axis/services/DictionaryService?wsdl#getMeaningReturn</grounding:ws
  dlMessagePart>

           </grounding:WsdlOutputMessageMap>

          </grounding:wsdlOutput>

          <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"


  >http://www.mindswap.org/axis/services/DictionaryService?wsdl</grounding:wsdlDocument>

          <grounding:owlsProcess rdf:resource="#getMeaningProcess"/>
```

```
            <grounding:wsdlInput>
              <grounding:WsdlInputMessageMap>
                <grounding:owlsParameter rdf:resource="#in0"/>
                <grounding:wsdlMessagePart
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

>http://www.mindswap.org/axis/services/DictionaryService?wsdl#in0</grounding:wsdlMessagePart>
              </grounding:WsdlInputMessageMap>
            </grounding:wsdlInput>
            <grounding:wsdlOperation>
              <grounding:WsdlOperationRef>
                <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

>http://www.mindswap.org/axis/services/DictionaryService?wsdl#DictionaryService</grounding:por
tType>
                <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

>http://www.mindswap.org/axis/services/DictionaryService?wsdl#getMeaning</grounding:operation
>
              </grounding:WsdlOperationRef>
            </grounding:wsdlOperation>
            <grounding:wsdlInputMessage
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"

>http://www.mindswap.org/axis/services/DictionaryService#getMeaningRequest</grounding:wsdlIn
putMessage>
```

```
          </grounding:WsdlAtomicProcessGrounding>
         </grounding:hasAtomicProcessGrounding>
         <service:supportedBy rdf:resource="#getMeaningService"/>
        </grounding:WsdlGrounding>
       </service:supports>
       <service:describedBy rdf:resource="#getMeaningProcess"/>
       <service:presents>
        <profile:Profile rdf:ID="getMeaningProfile">
         <profile:hasInput rdf:resource="#in0"/>
         <profile:textDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >Auto generated from
http://www.mindswap.org/axis/services/DictionaryService?wsdl</profile:textDescription>
         <profile:serviceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >getMeaning</profile:serviceName>
         <service:presentedBy rdf:resource="#getMeaningService"/>
         <profile:hasOutput rdf:resource="#getMeaningReturn"/>
        </profile:Profile>
       </service:presents>
      </service:Service>
     </service:describes>
  </process:AtomicProcess>
</rdf:RDF>
```

```
<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)  http://protege.stanford.edu -->
```