Semantic Information Systems Engineering: A Query-based Approach for Semi-automatic Annotation of Web Services

A thesis submitted for the degree of Doctor of Philosophy

By

Mohammad Mourhaf AL Asswad

Department of Information Systems and Computing,

Brunel University

January 2011

ABSTRACT

There has been an increasing interest in Semantic Web services (SWS) as a proposed solution to facilitate automatic discovery, composition and deployment of existing syntactic Web services. Successful implementation and wider adoption of SWS by research and industry are, however, profoundly based on the existence of effective and easy to use methods for service semantic description. Unfortunately, Web service semantic annotation is currently performed by manual means. Manual annotation is a difficult, error-prone and time-consuming task and few approaches exist aiming to semi-automate that task. Existing approaches are difficult to use since they require ontology building. Moreover, these approaches employ ineffective matching methods and suffer from the Low Percentage Problem. The latter problem happens when a small number of service elements - in comparison to the total number of elements – are annotated in a given service.

This research addresses the Web services annotation problem by developing a semi-automatic annotation approach that allows SWS developers to effectively and easily annotate their syntactic services. The proposed approach does not require application ontologies to model service semantics. Instead, a standard query template is used: This template is filled with data and semantics extracted from WSDL files in order to produce query instances. The input of the annotation approach is the WSDL file of a candidate service and a set of ontologies. The output is an annotated WSDL file. The proposed approach is composed of five phases: (1) Concept extraction; (2) concept filtering and query filling; (3) query execution; (4) results assessment; and (5) SAWSDL annotation. The query execution engine makes use of name-based and structural matching techniques. The name-based matching is carried out by CN-Match which is a novel matching method and tool that is developed and evaluated in this research.

The proposed annotation approach is evaluated using a set of existing Web services and ontologies. Precision (P), Recall (R), F-Measure (F) and Percentage of annotated elements are used as evaluation metrics. The evaluation reveals that

the proposed approach is effective since - in relation to manual results - accurate and almost complete annotation results are obtained. In addition, high percentage of annotated elements is achieved using the proposed approach because it makes use of effective ontology extension mechanisms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AE: After Extension

ANNIE: A Nearly New Information Extraction System

API: Application Programming Interface

ASSAM: Automated Semantic Service Annotation with Machine learning

BE: Before Extension

BPEL4WS: Business Process Execution Language for Web Services

C: Concept

Cls: Class

CN: Compound Noun

Cor: Correspondence

CORBA: Common Object Request Broker Architecture

CorM: Correspondence of the Main Service Concept

DAML: DARPA Agent Mark-up Language

DCOM: Distributed Component Object Model

DSR: Design Science Research

ebXML: Electronic Business using eXtensible Mark-up Language

E-Commerce: Electronic Commerce

F: F-Measure

FOAM: Framework for Ontology Alignment and Matching

GATE: General Architecture for Text Engineering

HTML: Hyper Text Mark-up Language

IR: Information Retrieval

IRS: Internet Reasoning Service

IS: Information Systems

IT: Information Technology

JAPE: Java Annotation Pattern Engine

LCS: Least Common Subsumer

ML: Machine Learning

MSC: Main Service Concept

MWSAF: Meteor-s Web Service Annotation Framework

NLP: Natural Language Processing

OAEI: Ontology Alignment Evaluation Initiative

OASIS: Organisation of Advancements of Structured Information Standards

OIL: Ontology Interchange Language

OP: Object Property

OWL: Ontology Web Language

OWL-S: Ontology Web Language for Services

P: Precision

POS: Part Of Speech Tagger

QOM: Quick Ontology Matching

R: Recall

SAWSDL: Semantic Annotation for WSDL

Sc: Service Concept

SCM: Set of Candidate Matches

SM: Set of Matches

SOA: Service Oriented Architecture

SPARQL: SPARQL Protocol and RDF Query Language

Src: Service Related Concept

SS: Set of Candidates

SWRL: Semantic Web Rule Language

SWS: Semantic Web Services

UDDI: Universal Description, Discovery and Integration

URI: Universal Resource Identifier

URL: Universal Resource Locator

W3C: World Wide Web Consortium

WSDL: Web Service Description Language

WSFL: Web Service Flow Language

WSML: Web Service Modelling Language

WSMO: Web Service Modelling Ontology

XLANG: XML-based extension of WSDL

XML: eXtensible Mark-up Language

XSD: XML Schema Definition

DEDICATIONS

I would like to dedicate this thesis to my parents. They have taught me how to be goal-oriented and patient to achieve what I am aiming for. Their encouragement and inspiration had a great impact on the successful completion of this thesis. Thank you very much.

I am very grateful to my wife and son Abdulmalek who were very patient and supportive. Without their love and help, I would never be able complete this work.

I also express my gratitude to my brother, sisters and brothers in law. I am very appreciative for all their great support during all the stages of my PhD.

ACKNOWLEDGEMENTS

I would like to take this opportunity to sincerely thank my supervisor Professor Mark Lycett for his great guidance during all stages of my PhD course. His inspiration, encouragement, critical thinking and, above all, his constructive criticism and feedback, were invaluable to the research, writing and completion of this thesis. Thank you Professor Mark.

I am very grateful to my Second Supervisor Dr. Sergio de Cesare for all the constructive discussion and guidance. Sergio offered me unlimited support whenever and wherever I needed. Without his continuous help, this thesis would not be completed.

I would also like to express my sincere appreciation and gratitude to Dr. Simon Kent and Dr. David Bell for their very useful insights and support in solving numerous research and technical issues.

Finally, I am very thankful to all the DISC staff for the support and facilities they have provided to my during my PhD course.

PUBLICATIONS

AL Asswad, M.M., de Cesare, S. and Lycett, M. 2009, Toward a Research Agenda for Semi-Automatic Annotation of Web Services, International Conference on Informatics and Semiotics in Organisations (ICISO) - IFIP WG8.1 Working Conference, pp. 138 - 146.

AL Asswad, M. M., Al-Debei, M. M., de Cesare, S., Lycett, M. 2010, Conceptual Modelling and the Quality of Ontologies: A Comparison between Object-Role Modelling and the Object Paradigm, 18th European Conference on Information Systems, pp. 1 - 18.

AL Asswad, M. M., de Cesare, S., Lycett, M. 2011, A Query-based Approach for Semi-Automatic Annotation of Web Services, the International Journal of Information Systems and Social Change, vol. 2, no. 2, pp. 37-54.

Chapter 1: Research Introduction and Motivation

1.1 Background and Motivation

The last decade has witnessed an increasing interest in engineering information systems that communicate and interoperate more easily (Hasselbring, 2000). This is because existing information systems are no longer isolated but they need to exchange data and knowledge. In addition, the emergence of the Web as a platform for people and machine communication has added new requirements and facilities for systems interoperability (Isakowitz et al., 1998). In response to the increasing need for effective methods and approaches to system integration and communication, Web services have appeared as a systematic and extensible approach for system to system interactions (Curbera et al., 2002).

1.1.1 Web Services

Web services are software components that can be published and discovered by other services and applications. The Web service framework is based on three fundamental XML-based standards. These standards are SOAP, WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery and Integration) (Curbera et al., 2002). SOAP allows communication among services and between services and other software systems. WSDL is used to describe Web services as sets of communication endpoints that enable message exchange and

UDDI is a directory that stores information about services and allows developers to search for services.

Many applications require a business logic that cannot be achieved using a single service. Therefore, a number of services need to be composed together to perform the desired task. Web service composition however, is not an easy task. This is because composition involves: (1) Discovering the right services that can do the specified tasks; and (2) solving potential structural and semantic mismatches that may occur between parameters of candidate services. Mismatches occur because different service developers normally have different views of data structure and semantics of same or similar service elements.

For Web services to meet the requirements of modern applications, automatic discovery and composition of services is needed since the manual discovery and composition is difficult, time-consuming and error-prone (Agarwal et al., 2003). Moreover, in the future Web, intelligent agents may be responsible for service discovery and composition that should be performed automatically at run time (Narayanan and McIIraith, 2002). Unfortunately, existing Web service standards do not enable dynamic discovery and composition of services because they are missing important semantic constructs (Sivashanmugam et al., 2003a; Sycara et al., 2003). The use of semantics modelled in the form of ontologies can facilitate automatic service discovery and composition (McIIraith et al., 2001): This is because ontologies provide machine-understandable and precise definitions to service elements. This can facilitate service discovery and composition based on functional, non-functional and capability descriptions (Ringelstein et al., 2007). Moreover, semantic matching techniques can be used to resolve the semantic mismatching issues between parameters of composed services. Therefore it is important to describe Web services in a semantic manner using ontologies (McIIraith et al., 2001). Web service semantic description can be achieved using the ‘Semantic Web Services’ (SWS) initiative.

1.1.2 The Semantic Web and Ontology

The Semantic Web is defined by Berners-Lee et al. (2001 pp. 3-4) as "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". The Semantic Web therefore aims to provide computer-understandable and precise descriptions of static and dynamic Web resources (McIIraith et al., 2001). Consequently, an important contribution of the Semantic Web is the addition of semantics to syntactic Web services, where services offered and user requirements in relation to desired services are described semantically (Martin and Domingue, 2007).

Ontologies are significant components of the Semantic Web: They are used to model and provide semantics to different data elements on the Web. Ontologies have been employed in a wide range of applications such as; Artificial Intelligence and Knowledge-based systems (Janev and Vranes, 2009). In computational terms, an ontology can be defined as "a formal explicit specification of a shared conceptualisation" (Gruber, 1993 pp. 3). That is, an ontology is a definition of vocabulary, axioms and relations in a formal, shared and machine-understandable form (Jasper and Uschold, 1999).

Since ontologies are important elements to many applications and systems, they have to be designed and engineered using suitable design methods that produce good quality. Building a good ontology, however, is a hard task: It requires a very good level of technical and domain knowledge to provide semantically and syntactically sound ontologies (Devedzic, 2002). Technical knowledge is needed because a tangible ontology has to be encoded using an ontology representation language such as OWL (Ontology Web Language) (De Nicola et al., 2009). Using such a representation language requires knowledge of the language and its constructs such as classes, properties, cardinality restrictions and domain and range axioms. In addition, domain knowledge is needed to precisely collect, define and model domain concepts, their relations and axioms (De Nicola et al.,

2009). Using imprecise and incomplete domain knowledge may result in ontologies that are inaccurate representations of domain knowledge (Staab, 2004).

Currently, there are few approaches that seek to automate the ontology building task. These approaches are called ontology learning techniques (Grobelnik et al., 2009; Wei et al., 2010), which utilise methods such as Machine Learning (ML), Natural Language Processing (NLP) and Statistics (Gomez-Perez and Manzano-Macho, 2004). The learning process is generally composed of steps such as knowledge acquisition, concept filtering and relation learning and ontology organisation which improves the knowledge content of the new ontology (Missikoff et al., 2002; Zhou, 2007). Available ontology learning methods, however, provide ontologies that are of unsatisfactory quality (Zouaq and Nkambou, 2008) since they miss many important constructs such as axioms. In addition, the resulting ontologies are representations of the source documents rather than being precise models of described domains. Consequently, these ontologies may not be useful when they are shared between different applications since they overlook many ontological entities that are important for these applications.

Summarising the previous literature on manual and automatic ontology engineering, one can conclude that manual ontology building is difficult and labour-intensive task (Jiang and Tan, 2010) since it requires much domain and technical knowledge (Devedzic, 2002). In addition, existing automatic ontology building methods have many understandable limitations and thus the resulting ontologies may not be of satisfactory quality.

1.1.3 Semantic Web Services (SWS)

SWS has emerged as a promising solution to solve the discovery and composition problems of current syntactic Web services (Vitvar et al., 2007). The SWS idea is based on using ontologies to provide semantic descriptions to service elements. Generally speaking, semantically describing a service entails two significant

processes (Verma and Sheth, 2007): (1) Identifying the service elements that should be semantically described; and (2) annotating the identified elements to appropriate ontological entities. In the context of this research, annotation means referencing Web service elements to suitable ontological entities. Successful implementation of SWS in industry and research requires effective and easy to use SWS annotation approaches (Lara et al., 2004).

1.1.4 The Dilemma of Semantic Web Service Annotation

Web services are currently annotated by manual means. Manual annotation is a difficult, error-prone and time-consuming task for the following three reasons (Hepp, 2006):

1. The large number of potential domain ontologies that can be used for annotation. Over time it is expected that more domain ontologies will be available to SWS developers requiring a developer to search manually for most appropriate ontologies (Patil et al., 2004).
2. The big size of potential ontologies. Ontologies can contain hundreds or maybe thousands of entities. Using such heavy weight ontologies for annotation requires a developer to browse through their descriptions to find entities that suit the different service elements.
3. The big size of candidate Web services. Many services have a high number of elements that should be annotated.

Given these three problems, there is a pressing need in the SWS arena for effective and automatic Web service annotation mechanisms. A few approaches and tools have been developed aiming to automate the annotation task. What exists can be classified into learning-based, matching-based and workflow definition-based approaches. These categories and their limitations are briefly described as follows:

- Learning-based approaches employ ontology learning techniques to automatically build ontologies for annotation (Chifu et al., 2007). Retrospectively, existing learning techniques provide poor quality ontologies

as they miss many important constructs. In addition, resulting ontologies are representations of individual services rather than being precise and shared domain models.

- Matching-based approaches require manual building of application ontologies to capture the semantics of candidate services. The application ontologies are then matched against existing domain ontologies to find corresponding ontological entities for given service elements. Manual building of application ontologies is difficult as it requires much domain and technical knowledge. Moreover, employed matching techniques cannot perform accurate matching when labels of candidate services and ontological entities are composed of multiple words.
- The workflow definition-based approach uses "tried and tested" workflows and annotated services to derive annotation for new services. Existing annotated services and "tried and tested" workflows are hard to find in practical settings.

1.2 Research Aim and Objectives

Given the difficulty of manual annotation of Web services and the limitations of existing semi-automatic annotation approaches, the SWS area needs a new semi-automatic annotation approach that can help SWS developers to effectively and easily annotate their services. Subsequently, the aim of this research is:

> *To develop an effective and easy to use Web service semi-automatic annotation approach that utilises ontology matching techniques.*

In fulfilling this aim, a number of objectives are considered important to be achieved as follows:

O1. Analyse the previous Web service semi-automatic annotation approaches and study their limitations in order to derive a set of design requirements and strategies for the new approach.

O2. Design an initial annotation framework based on the derived requirements and strategies and the analysis of WSDL general structure.

O3. Develop and test the automated components of the annotation approach.

O4. Evaluate the final annotation approach using suitable evaluation methods, metrics and data.

O5. Draw conclusions from the building and evaluation phases and identify future research directions that are important to continue refining and developing this significant area of research.

1.3 Research Approach

To achieve the research aim and objectives, this research follows the Design Science Research (DSR) approach. DSR is a problem solving paradigm that aims to design innovative and effective artefacts that can solve significant research problems (Hevner et al., 2004). DSR is deemed appropriate for this project since the aim of this research is to design an effective and easy to use solution for the important problem of Web service annotation.

The DSR process comprises three significant activities of 'build', 'deploy' and 'evaluate' (March and Smith, 1995). Across these activities, the desired design artefact is developed, deployed and tested using suitable evaluation methods and metrics. The DSR process can be of iterative and/or incremental nature which implies that the 'build-deploy-evaluate' process can be repeated or incremented until satisfactory artefacts are obtained (Markus et al., 2002). In iterative DSR, the build-deploy-evaluate process is repeated a number of times to improve the artefact. On the other hand, incremental DSR means that the design of the required artefact is decomposed into more granular artefacts where each one is developed and evaluated during an increment (Simon, 1996 pp. 120). The DSR process of this project is an incremental one. This is because the proposed annotation approach contains different components where each component or set of components is developed and tested in a specific increment.

In presenting this research, the DSR cycle provided by Kuechler and Vaishnavi (2008) and presented in Figure 3.2 is utilised. This cycle is composed of five phases called awareness of problem, suggestion, development, evaluation and conclusion. Knowledge feedback is very significant in the DSR cycle. Knowledge is acquired during all phases of the design process and transferred to previous and subsequent phases. This knowledge is very important because it can help to improve the design process and the resulting artefacts (Kuechler and Vaishnavi, 2008). In designing the required annotation approach, six design increments are defined. These increments are briefly explained as follows:

(1) Increment 1 (Design of the Initial Framework): In this increment, the initial annotation approach is developed and its manual and automatic phases and components are identified. Three automatic phases are defined which are: (a) Concept extraction, (b) query execution and (c) SAWSDL annotation. The two manual phases are: (a) Concept filtering and query filling and (b) results assessment.

(2) Increment 2 (Design of the Concept Extraction Technique): The extraction technique is designed to automatically extract candidate service elements from given WSDL files. This technique is implemented using text analysis techniques.

(3) Increment 3 (CN-Match Design): CN-Match is the name-based matching mechanism that is employed by the query execution engine. String and linguistic matching mechanisms are used to implement CN-Match.

(4) Increment 4 (Structural Matching Design): The structural matching mechanism is the second matching technique that is utilised by the query execution engine. Structural matching is developed and used to improve query execution by measuring similarities between related elements of candidate service concepts and related classes of candidate ontological classes.

(5) Increment 5 (SAWSDL Annotator Design): The SAWSDL annotator is designed using text parsing and string similarity techniques: It takes correct matches and uses them to automatically annotate the given service based on the SAWSDL notation.

(6) Increment 6 (Design of the Ontology Extension Mechanisms): The extension mechanisms are developed and added to the annotation approach to allow the addition of appropriate ontological entities for service elements that do not have suitable correspondences. Two extension methods are developed, one for simple queries and the other for complex queries.

The role of design artefacts is central for any DSR project. Artefacts represent solutions to defined research problems (Orlikowski and Iacono, 2001). March and Smith (1995) classified DSR artefacts into constructs, methods, models and instantiations. March and Smith (1995) classification is used to classify the artefacts of this project. Table 1.1 presents the classification of artefacts.

Category	Artefact
Construct	Standard query template
Model	None
Method	Initial annotation framework CN-Match Structural matching mechanism Ontology extension mechanism
Instantiation	Concept extraction mechanism CN-Match Structural matching mechanism SAWSDL annotator Ontology extension mechanism Semi-automatic annotation framework

Table 1.1: The Classification of the DSR Artefacts of this Research

1.4 Structure of the Thesis

In presenting the research, this thesis is structured as follows:

Chapter 2 provides a review of related research literature. The issue of semi-automatic semantic description (i.e., annotation) of Web services is the main topic to explore. To understand the semantic annotation issue, it is important to investigate the Web services, ontologies and Semantic Web services (SWS) areas. Web service is a promising technology for supporting seamless connectivity between distributed application systems. Automatic discovery and composition of services is, however, very difficult when using current Web service standards. SWS is a proposed solution for solving the discovery and composition problems of Web services. There is no consensus in the SWS arena on the service elements that should be semantically annotated. Therefore, this chapter provides a set of synthesised elements that should be semantically described. Ontologies are very important components of Semantic Web applications such as SWS. Consequently, ontologies and their manual and automatic building methods are discussed. Few semi-automatic annotation approaches exist: They are classified as learning-based, workflow definition-based and matching-based approaches. These approaches are discussed and their limitations are illustrated.

Chapter 3 describes the research approach used in designing and evaluating the proposed semi-automatic Web service annotation framework. The Design Science Research (DSR) paradigm is chosen as the right approach for tackling the defined research problem. DSR is a problem solving paradigm that aims to provide novel and purposeful artefacts to solve significant research problems. The research carried out in this project is then described in light of the DSR research cycle. Five DSR phases are identified which are: (1) Awareness of problem; (2) suggestion; (3) development; (4) evaluation; and (5) conclusions. Later, the research increments performed during the design process of the annotation framework are illustrated along with the learning that happens during each increment. Since the evaluation is a very crucial activity in any DSR project, the

methods, metrics and data used to evaluate the developed annotation approach and all its automatic components are described. The design artefacts produced in this project are also presented and classified according to a widely used DSR artefact classification approach.

Chapter 4 presents the new annotation approach and identifies all its phases and components. The proposed approach is of a semi-automated nature and utilises a query template rather than application ontologies. The design of the new approach is based on a set of design requirements and strategies that are derived from limitations of previous annotation approaches and the analysis of the WSDL general structure. Analysing the WSDL structure allows the identification of WSDL elements that should be annotated. The proposed approach comprises five interrelated phases. Three phases are fully automatic which are concept extraction, query execution and SAWSDL annotation. Queries are executed by means of a novel query execution engine that employs name-based and structural matching mechanisms. The two manual phases are 'concept filtering and query filling' and results assessment. The design of all phases is described and the techniques used to implement the automatic parts are illustrated.

Chapter 5 presents the design and evaluation of CN-Match which is the name-based matching mechanism employed by the query execution engine. The chapter starts by highlighting the significance of matching Compound Nouns (CNs) in the area of ontology matching. A discussion about the structure and types of CNs from a linguistic point of view is followed. Previous approaches of CN matching are discussed and their limitations are provided. The given limitations provide a motivation for designing a novel CN-Matching mechanism that can automatically and effectively measure similarities between single terms, binary and triple CNs. Considerations and rules for the design of the new matching approach are derived from the deficiencies of previous approaches and the linguistic structure of CNs. Six design cases are identified for CN-Match design: These cases are distinguished based on the number of constituents in any two candidates. Later, the design and implementation of CN-Match are illustrated. To assess the

performance of CN-Match, it is evaluated using Precision (P), Recall (R), F-Measure (F) and Percentage metrics. Three sets of experiments are conducted using three different sets of exiting ontologies. The evaluation results are then discussed and important implications are derived.

Chapter 6 focuses on evaluating the proposed semi-automated annotation approach. The chapter starts by presenting the ontology extension mechanisms supporting the proposed annotation approach. Then, three illustrative cases are provided to explain the annotation steps and show how the annotation approach works in practice. Next, the framework evaluation method and metrics are presented. P, R and F metrics are again used in this evaluation. Five sets of experiments are performed and the evaluation results discussed. Finally, implications of the conducted evaluation and limitations of the proposed annotation approach are given.

Chapter 7 summarises the research findings and conclusions: It categorises the research contributions into contributions to theory and contributions to practice. Last, this chapter discusses how this research meets its defined objectives and directions for further research are explored.

To simplify the reading of this thesis, a mapping between the thesis chapters and objectives is provided in Figure 1.1.

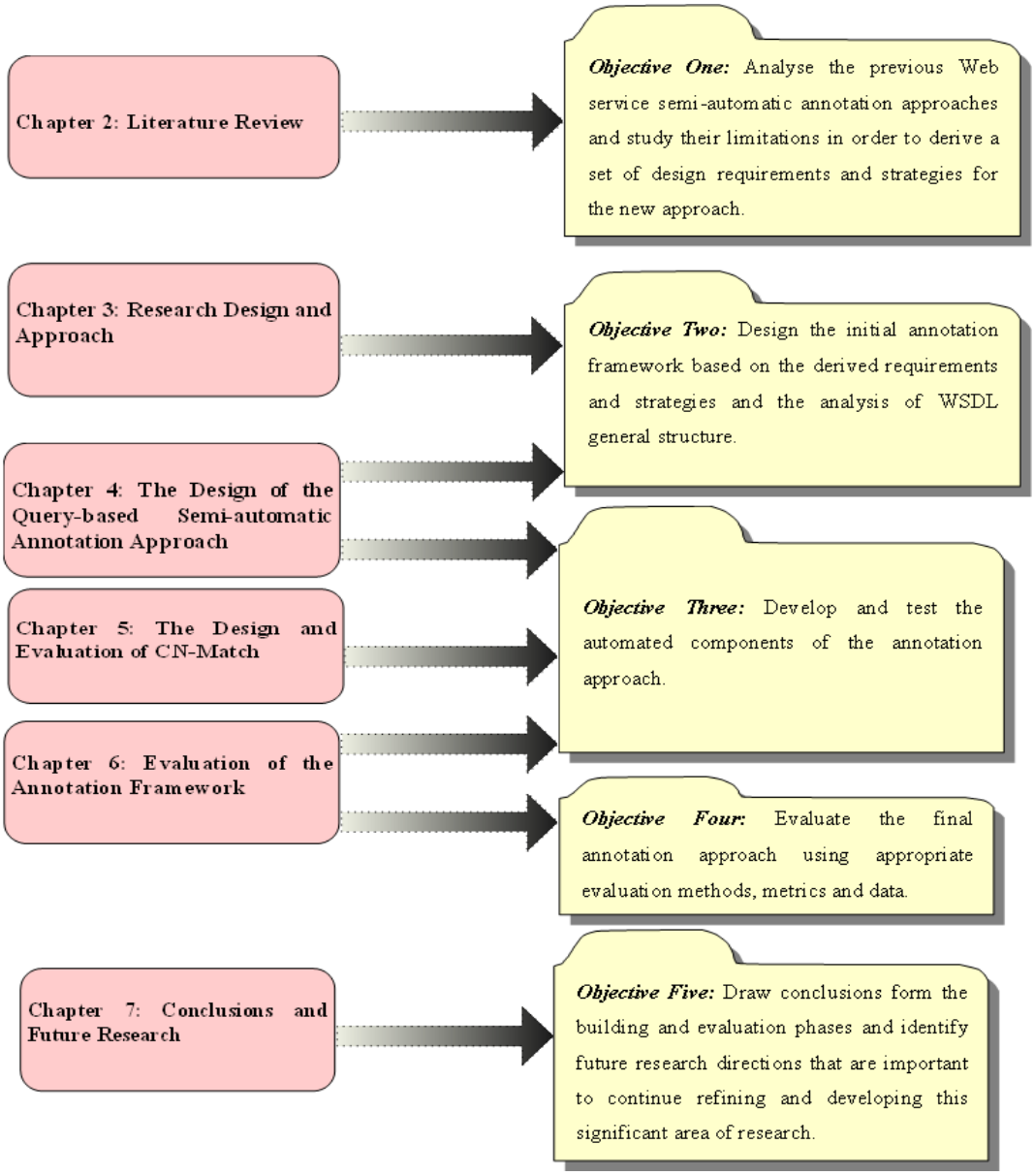

Figure 1.1: Mapping Between Chapters and Objectives

Chapter 2: Literature Review

2.1 Introduction

This chapter investigates the state-of-the-art in Web service semantic description and exposes the limitations of existing semi-automatic annotation methods. In order to examine the semantic annotation issue, it is necessary to present Web services, ontologies and Semantic Web services (SWS). Ontology matching is also a significant area for this research because matching techniques are important means for achieving the desired semi-automation of annotation.

The chapter is organised as follows: Section 2.2 presents Web services and their supporting technologies. Section 2.3 discusses ontologies, their engineering, learning and extension. Section 2.4 focuses on SWS and their major description frameworks. Section 2.5 discusses the importance of semi-automatic annotation of Web services and categorises existing semi-automatic annotation approaches into three categories. Section 2.6 illustrates the first and second categories which are learning and workflow definition-based approaches. Section 2.7 discusses matching-based annotation approaches which constitute the third category and illustrates the fundamental ontology matching mechanisms. Finally, Section 2.8 presents deficiencies of previous research and Section 2.9 summarises the chapter.

2.2 Web Services

Modern applications of distributed-systems such as; electronic commerce (e-commerce) and supply chain management require the development of platform-independent and distributed software components (Paolucci et al., 2002). This is because such components are necessary to facilitate flexible communication and integration across heterogeneous systems. To achieve this aim, distributed software entities should be made discoverable, composable and reusable by different applications and organisations (Cheng et al., 2006). Nevertheless, developing such network accessible components has been a complex and challenging undertaking (Stal, 2002). In response to this challenge, SOA (Service Oriented Architecture) has been developed as a distributed computing paradigm that offers software components described by publishable and discoverable interfaces to other applications and services existing on a network (Papazoglou et al., 2007). Services in SOA can be defined as open, platform-independent and self-describing software components (software-as-a-service) that enable fast and low cost composition of distributed application systems (Papazoglou and Georgakapoulous, 2003).

One can argue that SOA has roots in previous component-based software models such as CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model) (http://www.service-architecture.com/) since SOA services are themselves software components. Yet, services in SOA are different because they are capable of providing more granular business functionalities that can be discovered and composed in a flexible manner. In addition, these services are decoupled from implementation. These significant characteristics of services make them different from previous component-based software systems and enable them to be reused as effective solutions to business needs (Papazoglou, 2003; Stal, 2002).

The basic SOA architecture is composed of a service provider, service consumer (client) and service registry (Huhns and Signh, 2005). A service provider offers a service description and support for service use. A service client utilises the service in their application. Service descriptions are stored in a directory that makes these descriptions searchable by clients. Once an appropriate service is found a client can bind with the provider, invoke the service and implement its functionality (Papazoglou, 2003).

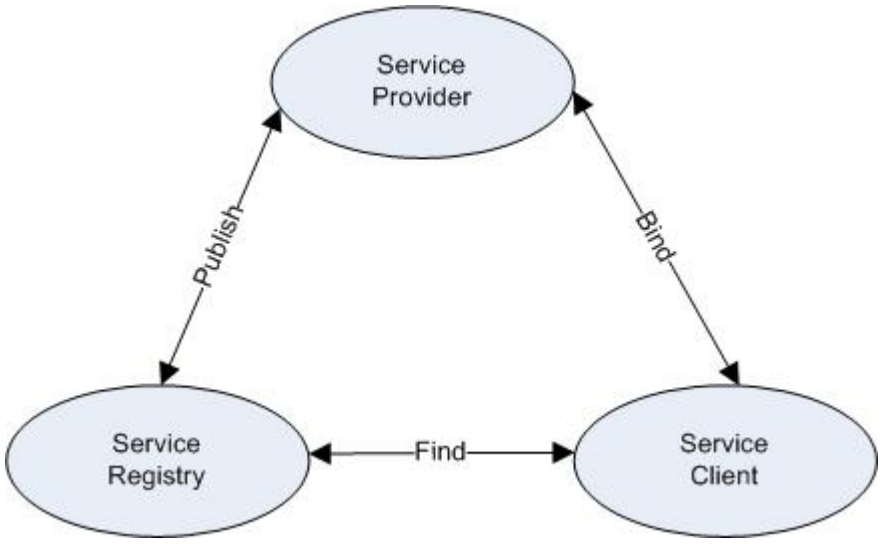

Figure 2.1: The Basic SOA Architecture (Source: Papazoglou, 2003)

The SOA literature provides a theoretical foundation but lacks fundamental pragmatic issues pertaining to service description, publishing, management, orchestration, coordination and security. The practical implementation of the SOA framework, however, can be realised by the Web service technology (Ferris and Farrell, 2003). Web services can be seen as the application of SOA on the Web. The innovation of Web services is a collaborative effort supported by different parties including the W3C (the World Wide Web Consortium) standards group, OASIS (Organisation of the Advancement of Structured Information Standards) and the open source community.

A Web service is defined as "a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can

be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition using XML-based messages conveyed by Internet protocols" (Austin et al., 2004 pp. 1). The Web service framework is based on three fundamental standards that facilitate the operations performed by the three basic elements of the SOA i.e., service provider, client and registry. These open and XML-based standards are WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration) and SOAP. The three standards are briefly illustrated as follows (Curbera et al., 2002).

- SOAP is a messaging protocol that facilitates message exchange among services and between a service provider and consumer. A SOAP message encompasses two components: (1) An envelope and (2) a model describing how the message should be processed by recipients and who should process the message.
- WSDL describes the service interface as a set of communication endpoints that enable message exchange. A WSDL file contains two kinds of description: (a) Service application description including XSD (XML Schema) definitions that provide specifications for data types of various service elements and (b) concrete binding information that allows the end user to access the service at its endpoints.
- UDDI is a centralised directory of service description. UDDI allows the description of services using a set of features called tModel. tModel contains information that describes service interface and category. Using tModel, a service client has to browse through different categories and may use keyword-based search in order to find their desired services (Brittenham et al., 2001). Subsequently, this discovery process must be performed with full human involvement and thus can be time-consuming, error-prone and unsuitable for applications that require on-the-fly discovery.

A more flexible, accurate and dynamic service selection mechanism than the one offered by UDDI is required for the following reasons:

1. Services in an open and dynamic environment such as the Web should be discovered at run-time by other services and software agents (Paolucci et al., 2002). Run-time discovery is fundamental to facilitate dynamic interoperability between different systems.
2. Exact matching between service advertisements and requests is very unlikely as service clients and providers may have very different knowledge about the same service (Benatallah et al., 2005). Therefore, more tolerant discovery mechanisms should be used in order to find services that offer more or less than what is required by clients (Cardoso and Sheth, 2003).
3. Services should be discovered based not only on their functional properties (e.g. inputs and outputs) but also their capabilities (what they offer) and behaviour (how they perform their tasks). The reason is that services could have the same inputs and outputs but perform very different functionalities and have different behaviour.

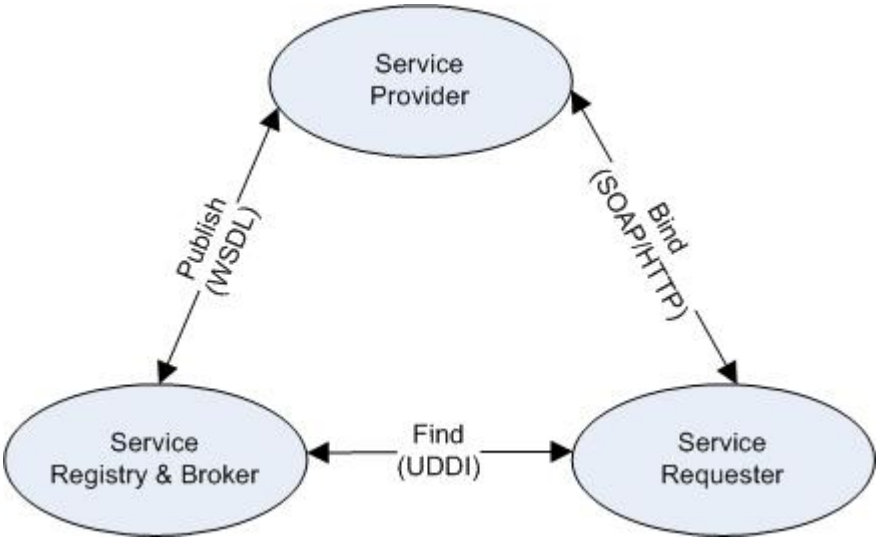

Figure 2.2: Web Services Architecture Model (Source: Huhns and Singh, 2005)

WSDL describes Web services as a set of ports that offer operations (Austin et al., 2004). Each operation performs a specific functionality by sending and receiving one or no message. Many applications and clients, however, require an

implementation of a complex business logic that cannot be achieved by using a single service operation or available composed services. Therefore, it is necessary to integrate services in a specific manner in order to achieve the desired complex functionality (Dustdar and Schreiner, 2005). Organising single operations of services or existing composed services in a specific sequence to achieve the desired business process is called Web services composition (Khalaf and Leymann, 2003). In response to the high importance of Web services composition, different standards have been proposed such as; WSFL (Web Services Flow Language) from IBM, XLANG (XML-based extension of WSDL) from Microsoft, ebXML (Electronic Business using eXtensible Mark-up Language) and BPEL4WS (Business Process Execution Language for Web Services).

The earlier XML-based composition standards differ on the necessary composition constructs and their semantics (Staab, 2003). Moreover, the Web services composition process is not an easy task: It is much more complicated than the design of conventional workflow-based systems for two reasons (Cardoso and Sheth, 2003):

- Web services discovery cannot be performed manually as the number of Web services potentially appropriate for the given composition can be massive. Therefore, an efficient automatic discovery method based on functional and operational characteristics is needed.

- The structural and semantic heterogeneities of the selected Web services. The structural heterogeneity happens because Web services use different data structures to describe their elements, while the semantic heterogeneity is due to the different interpretations of information used by the connected services in a process. These heterogeneity issues require schema and semantic mediation in order to enable interoperability between elements of connected services.

These two challenges highlight the importance of automatic discovery and composition because the manual process can be very difficult, time-consuming

and error-prone. For Web services to meet the needs of future Web applications, it is especially important to enable on-the-fly discovery and composition of Web services (Agarwal et al., 2003). Software agents may play important roles in the future Web where they may perform tasks such as service discovery, selection and composition on behalf of a human user (Narayanan and McIIraith, 2002). Unfortunately, using existing Web service standards alone does not enable the desired automation and agility because these standards lack the necessary semantic constructs (Sycara et al., 2003; Sivashanmugam et al., 2003a). The use of semantics represented in the form of ontologies can provide precise, machine-understandable and shared meanings of service elements and thus may enable automatic discovery and composition of Web services (McIIraith et al., 2001; Sycara et al., 2003). Moreover, semantic matching techniques can resolve the above mentioned heterogeneity issues. Therefore utilising semantics in the area of Web services seems to be a natural choice (McIIraith et al., 2001). This utilisation launched a new and active research area called 'Semantic Web Services' (SWS) which combines the Semantic Web initiative proposed by Berners-Lee et al. (2001) with the Web service technology.

The Semantic Web is defined in the widely cited Scientific American article written by Berners-Lee et al. (2001 pp. 3-4) as "an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". The Semantic Web idea was initially proposed for static Web resources but then extended to cover dynamic resources i.e., Web services (McIIraith et al., 2001). Therefore, an important contribution of the Semantic Web is the provision of semantic mark-ups of web services where services offered and capabilities required by potential consumers are described semantically (Martin and Domingue, 2007). In summary, Table 2.1 presents the most significant issue discussed in this Section.

Issue Number	Issue Description
Issue 1	Existing Web service standards do not facilitate automatic discovery and composition because they lack the necessary semantics.

Table 2.1: Issue Identified in Section 2.2

2.3 Ontology

Ontologies are fundamental components of SWS: They are used to provide precise, explicit and shared meanings of Web service elements. Therefore ontologies are discussed in detail in this section. Ontologies have been applied to a wide range of computer applications such as knowledge engineering and sharing, database design, Artificial Intelligence and Web services (Janev and Vranes, 2009).

Ontology can be defined as "a formal explicit specification of a shared conceptualisation" (Gruber, 1993, pp. 3); that is, a definition of concepts, axioms and relations between concepts in a formal, shared and machine-understandable format (Jasper and Uschold, 1999). In the context of the Semantic Web, things that exist in the domain under consideration should be represented in an ontological model (Gruber, 1995). The last sentence has two important implications on ontology modelling. The first is the notion of existence which refers to ontological commitment. This notion enables an ontology to precisely reflect real life phenomena. The second is a representation which is achieved using a formal language (Zuniga, 2001).

In response to the importance of ontologies in Semantic Web applications, formal ontology representation languages have been proposed. Examples are DAML (http://www.daml.org/), OIL (http://www.ontoknowledge.org/oil/) and OWL

(http://www.w3.org/TR/owl-guide/) which is based on DAML and OIL. These languages are based on logic in order to be formal and enable machine-readability.

2.3.1 Ontology Engineering and Learning

According to Guarino (1998 pp. 4) ontology is defined as "engineering artefact" which comprises a set of vocabulary to describe a certain reality and some axioms to restrict the interpretation of this vocabulary. Due to the important role of ontologies in information systems applications, special attention should be taken when engineering an ontology. The ontology must be of good quality, in relation to its content, in order to serve its intended purposes and be shared and reusable by different applications (Staab, 2004).

Building a good quality ontology, however, is not an easy task: It requires extensive technical and domain knowledge to ensure correctness of syntax and semantics (Devedzic, 2002). Technical knowledge is required because building tangible and usable ontologies entails representing these ontologies in one of the ontology representation languages such as OWL (De Nicola et al., 2009). Representing ontologies in an ontology representation language requires knowledge of this language and its constructs such as classes, properties, cardinality restrictions and domain and range axioms. In addition, knowledge of a development tool that can help developers during the representation process may be necessary to speed up the development process.

In addition, domain knowledge is needed in order to precisely capture and represent domain concepts, their relations and axioms (De Nicola et al., 2009). Lack of domain knowledge may result in an ontology that incorrectly models the domain or misses many significant concepts and axioms (Staab, 2004). To help ontology developers in building good quality ontologies, ontology building methodologies are proposed (De Nicola et al., 2009; Gruninger and Fox, 1995). These methodologies aim to provide guidelines that can make the development a

systematic process. For example, Pinto and Martins (2004) propose an ontology engineering process that is composed of the following five main phases;

- Specification: Identifies goals and scope.
- Conceptualisation: Constructs the conceptual model.
- Formalisation: Represents the conceptual model in a formal manner in order to define axioms.
- Implementation: Implements the ontology in an ontology representation language.
- Maintenance: Maintains the correctness of the resulting ontology.

Given the previous literature on ontology engineering, it can be inferred that manual ontology building is difficult and labour-intensive task (Jiang and Tan, 2010) since it requires domain and technical knowledge and can go through different steps such as conceptualisation and formalisation in order to produce good ontologies (Devedzic, 2002). There are few approaches that aim to automate the ontology building process, however. They utilise ontology learning techniques (Grobelnik et al., 2009; Wei et al., 2010) which, in turn, employ methods borrowed from other disciplines such as Machine Learning (ML), Natural Language Processing (NLP) and statistical mechanisms (Gomez-Perez and Manzano-Macho, 2004). The learning process requires resources for knowledge acquisition such as unstructured, semi-structured and structured documents or databases (Sanchez, 2010). Generally speaking, the learning process consists of three fundamental steps (Missikoff et al., 2002; Zhou, 2007): (1) Knowledge extraction which involves mining a resource to obtain the required ontological constructs; (2) ontology discovery which entails domain concepts filtering and relations learning; and (3) ontology organisation which involves harmonising the discovered knowledge and improving the knowledge content of the new ontology.

Given the current state of the ontology learning research, existing automatic ontology building methods have many understandable limitations. Consequently, resulting ontologies may not be of a satisfactory quality. These limitations are (Zhou, 2007; Zouaq and Nkambou, 2008):

- Existing learning methods focus on detecting relations of type generalisation-specialisation and miss other important relations such as whole-part.
- Current techniques can only provide binary relations which link two concepts together. Higher-degree relations are, however, very significant components and must be included in resulting ontologies.
- Produced ontologies contain many irrelevant concepts to given domains because effective automatic filtering techniques are still under development
- The quality of the produced ontologies is profoundly based on the quality and richness of the used resources.
- Ontologies that are learnt from specific documents are representations of these documents rather than being comprehensive and accurate representation of given domains. Consequently, these ontologies may not be useful when they are shared between different applications since they cannot capture and represent many ontological entities that are important for these applications.

2.3.2 Ontology Extension

Ontology extension is defined as the process of adding new ontological constructs to an existing ontology (Beneventano et al., 2003). These constructs can belong to any type of ontological entities such as classes and properties (Ovchinnikova and Kühnberger, 2006). Retrospectively, building an ontology from scratch is a very difficult and costly process; therefore, expanding an existing ontology is considered as an effective solution to many applications that use ontologies dynamically (Liu et al., 2005). So, once an application is changed or new requirements are added, the ontology can be updated to accommodate new semantics for the changes (Ovchinnikova and Kühnberger, 2006).

There are few approaches that aim to extend ontologies in an automatic or semi-automatic manner. For example, Jung et al. (2009) provide an ontology extension method to add concepts and relations extracted from textual documents using NLP techniques. A different ontology extension approach is proposed by Liu et al.

(2005): It expands ontologies semi-automatically by mining textual data of websites. In the later approach, Spreading Activation, which is a semantic network search method, is used to find the most relevant terms to the given domain: These terms are then incorporated into the original ontology (Liu et al., 2005). Developing an extension method is profoundly based on the nature of given knowledge resources. A knowledge resource such as a database or a text document provides knowledge that supports the addition of a subset of the required ontological constructs. For example, a text document could provide concepts and relations extracted from verbs. Extracting other important constructs such as cardinality and domain and range restrictions can be difficult using such a resource. Table 2.2 shows the significant issues identified in Section 2.3.

Issue number	Issue description
Issue 2	Manual ontology building is a hard and time-consuming task since it needs extensive technical and domain knowledge.
Issue 3	Automatic ontology building cannot provide good quality ontologies due to the immaturity of existing learning methods.
Issue 4	Ontology extension is proposed as an effective solution which allows expanding and reusing an existing ontology by adding new ontological constructs.

Table 2.2: Issues Identified in Section 2.3

2.4 Semantic Web Services (SWS)

The SWS proposal emerges as a solution to the problems (see section 2.2) of current syntactic Web services (Martin and Domingue, 2007; Vitvar et al., 2007). SWS can enable more agile and efficient discovery, composition and execution monitoring of services. The key principle of SWS is the use of ontologies to describe different service elements in a precise, shared and semantically rich manner. The SWS idea has attracted much attention and many approaches for

description (Feier et al., 2005; Jacek et al., 2007; Martin et al., 2007), discovery (Sycara et al., 2003; Pathak et al., 2005; Sbodio et al., 2010) and composition (Cardoso and Sheth, 2003; Wu et al., 2007; Yeganeh et al., 2010) have been proposed. However, successful implementation of automatic discovery, composition and interoperability of SWS is based on the availability of appropriate methods for SWS description (Lara et al., 2004). The SWS description is composed of service elements such as inputs and outputs annotated using suitable semantic metadata (Verma and Sheth, 2007). Web service annotation means explicitly describing the service's data and functional elements using concepts of shared ontologies in order to give these elements precise and machine understandable definitions (Martin et al., 2007). Subsequently, semantically describing a Web service entails two significant activities: (1) Electing the service elements that need to be semantically described; and (2) annotating these elements along with their data to appropriate ontological concepts.

The existing SWS literature does not agree on the service's elements that constitute a comprehensive SWS description. Table 2.3 provides a comparison between four neutral Semantic Web services description approaches. Drawing on (Cardoso, 2006; Nagarajan, 2006; Ringelstein et al., 2007; Sivashanmugam et al., 2003a), these papers are chosen on the basis that they consider SWS description regardless of any particular SWS approach such as; OWL-S, SAWSDL or WSMO. The reason behind this selection is that the synthesised elements are used as a benchmark for comparing the major SWS approaches in order to judge their completeness.

Paper	Input	Output	Pre-condition	Effect	Category	Non-functional	Functionality	Cultural	Execution
Ringelstein et al. (2007)	X	X	X	X	X	X	-	-	X
Cardoso (2006)	X	X	X	X	X	X	X	X	-
Sivashanmugam et al. (2003a)	X	X	X	X	X	X	X	-	X
Nagrajan (2006)	X	X	X	X	-	X	X	-	X

Table 2.3: A Comparison between Four Neutral Semantic Web Service Description Approaches

From Table 2.3, it can be inferred that inputs, outputs, pre-conditions, effects, category, non-functional semantics, functionality and execution semantics are the common service elements that are described in the SWS arena. Table 2.4 provides a synthesis of the common service elements and offers a definition and an example of each element. The examples are based on an imaginary online pizza ordering Web service. This service has two operations; the first for 'reserving a pizza' and the second for 'paying the total price'.

Element	Definition	Example
Input	The formal definition of data in an input message of a service (Nagarajan, 2006).	For 'reserving a pizza' operation, the inputs are: PizzaName and Quantity
Output	The formal definition of data in an output message of a service (Nagarajan, 2006).	For 'reserving a pizza' operation, the output is the total price.

Pre-condition	Conditions that must be fulfilled before the execution of a service's operation. Pre –conditions describe constraints on the inputs' values as well as the state of the world before successful execution of a Web service (Ringelstein et al., 2007).	For 'paying the total price' operation, the pre-condition is that the used credit card must be valid.
Effect	Describes constraints on the returned values and the impact of a Web service execution. Effects are more comprehensive than post-conditions as they cover impacts of service execution on the external world (Sivashanmugam et al., 2003a).	'Paying the total price' operation has two effects; (1) the total price amount is transferred from the customer account to the restaurant account. The customer's balance is decreased by the total price amount and the destination account balance is increased by the same amount (2) the delivery process is commenced.
Category	Identifies the Web service category such as travel, and finance (Ringelstein et al., 2007).	Food selling service.
Non-functional Semantics	Formally describe quantitative or non-quantitative constraints that support the Web service discovery and selection such as cost and security (Cardoso, 2006).	The description of the security issues of the online pizza ordering service.
Functionality	Annotates operations' names of services. Functionality can be specified using a functionality ontology which has concepts of functionalities (Nagarajan, 2006).	For 'reserving a pizza' operation, the functionality is 'online pizza reservation'. For 'paying the total price' operation, the functionality is 'paying the total price'.
Cultural Semantics	Describe culture - specific semantics of a Web service such as currencies, time and date formats and measurement units	The total price must be paid in Great Britin Pound (GBP).
Execution Semantics	Formally define the operational behaviour of a service (Zaremba and Bussler, 2005). Execution semantics describe the flow of data and actions within a service or the flow of services in a process. Protocol semantics defined in (Ringelstein et al., 2007) are similar to execution semantics.	The customer must reserve the pizzas before paying the total price

Table 2.4: Descriptions and Examples for the Web Service Elements

2.4.1 SWS Description Frameworks

SWS can be achieved through the use of either formal Web service ontologies like WSMO and OWL-S, or by means for adding semantics to current Web service standards like SAWSDL. In this section, the SWS frameworks are reviewed briefly and then compared against the synthesised elements of SWS as a means of assessing the completeness of those SWS frameworks.

SAWSDL

SAWSDL (Semantic Annotation for WSDL) is a lightweight framework for Semantic Web Services (Jacek et al., 2007). SAWSDL defines a means for ontologically annotating elements of WSDL documents and XML schema (examples of such elements include input and output message structures, operations and interfaces). SAWSDL semantic annotations are independent of any particular ontology definition and mapping language. The only requirement for SAWSDL is that all concepts are identified with URIs. SAWSDL defines two extension attributes called Model Reference and Schema Mapping. Model Reference links (annotates) a concept in an ontology with a unit of structure in an XML schema or WSDL document. Model Reference can provide annotation for simple and complex XSD types of a WSDL document (Akkiraju and Sapkota, 2007). The Model Reference for a complex type can provide partial or full annotation. Full annotation happens when both of the complex type and its child elements are annotated. On the other hand, partial annotation takes place when either the complex type or its child elements are annotated but not both of them.

The Schema Mapping extension attribute is used during invocation to translate a semantically described concept to a syntactically defined one and vice versa. SAWSDL specification does not define how to represent pre-conditions and effects however; Akkiraju and Sapkota (2007) suggest the use of SWRL (Semantic Web Rule Language) rules to represent these elements. Moreover, SAWSDL does not deal with execution semantics.

SAWSDL is, arguably, promising and easier to use and understand by Web service developers in comparison to other SWS frameworks since it does not introduce any new notations or languages (Sivashanmugam et al., 2003b; Verma and Sheth, 2007). SAWSDL utilises WSDL which is a well known language amongst the Web service community.

OWL-S

OWL-S is an OWL upper ontology for services and comprises three complementary models: (a) A profile model which describes what the service does; (b) a process model which defines how the service works; and (c) a grounding model which describes how to access the service. A profile model can describe the functionality and non-functional properties of a Web service. Moreover, functional descriptions of the service including data transformation (input and output) and state transformation (preconditions and results) are described in the profile and process sub-ontologies. OWL-S does not define the rules for pre-conditions and effects but recommends the use of rule definition languages such as SWRL (Martin et al., 2007).

WSMO

Web Service Modelling Ontology (WSMO) is another project that provides an abstract foundation and a formal language called WSML (Web Service Modelling Language) to describe features of Semantic Web services (Vitvar et al., 2007). WSMO comprises four main components which are: (a) Ontologies; (b) Web services; (c) goals; and (d) mediators. Ontologies define the terminology used by other WSMO components. Web services provide access to service functionalities that have value for users. A goal describes the desired functionality from the user point of view. Finally, mediators solve heterogeneity issues that arise at different levels including data, process and protocol levels (Roman et al., 2005). WSMO defines Web service capabilities in terms of pre-conditions, assumptions, post-conditions and effects. The pre-condition and post-conditions define the state of information (constraints on the values of inputs and outputs) before and after the execution of a Web service, respectively. Assumptions and effects describe the

state of the world before and after the execution of a Web service, respectively. In WSMO Web services, the interface describes the Web service behaviour and has two components - choreography and orchestration. The choreography of a Web service is a specification of how to invoke and interact with the Web service. The orchestration describes how the Web service achieves its functionality by means of utilising other Web services (Feier et al., 2005). WSMO does not explicitly define the concepts of inputs and outputs of Web service operations. Instead, the capability part of the Web service describes constraints on the inputs and outputs of a service through the use of pre-conditions and post-conditions (Lara et al., 2004).

IRS III (Internet Reasoning Service)

IRS-III is a platform for developing and executing Semantic Web services. IRS-III service development is based on the WSMO framework. The IRS-III service ontology has similarities and differences with the WSMO specification. WSMO and IRS-III are similar in their description of non-functional properties, Web service capability, choreography, grounding and orchestration. A fundamental difference is the declaration of input and output parameters which is explicit in IRS-III and implicit in WSMO (Domingue et al., 2008).

Table 2.5 compares the four Semantic Web service description frameworks in order to show the frameworks' completeness against the synthesised SWS elements.

Framework	Input	Output	Pre-condition	Effect	Category	Non-functional	Functionality	Execution
OWL-S	**E**	**E**	**S**	**S**	**S**	**S**	**S**	**S**
WSMO	**I**	**I**	**S**	**S**	**S**	**S**	**S**	**S**
SAWSDL	**E**	**E**	**S**	**S**	**S**	**NS**	**S**	**NS**
IRS-III	**E**	**E**	**S**	**S**	**S**	**S**	**S**	**S**

Table 2.5: A Comparison between Semantic Web Service Frameworks against Synthesised Semantic Web Service Elements

Table Keys: E: Explicit, I: Implicit, S: Supported and NS: Not supported.

Table 2.6 Summarises the most significant issue discussed in Section 2.4.

Issue Number	Issue Description
Issue 5	Major SWS description frameworks differ on the SWS elements that constitute a comprehensive semantic description of a service.

Table 2.6: Issue Identified in Section 2.4

2.5 Importance and Categories of Web Services Semi-automatic Annotation Approaches

Annotation is a significant process in the area of Semantic Web. It enables different Web resources to have precise, machine-understandable and shared meaning by referencing these resources to appropriate concepts in shared ontologies. Manual annotation of Web services is a difficult task and requires

comprehensive human involvement. Thus, automating the annotation task is highly desired (Hepp, 2006). Few approaches and tools have been developed to semi-automatically annotate Web services. What exists can be categorised according to the method used in performing the annotation. These categories are machine learning-based, semantic matching-based and workflow definition-based. The following two sections present a review of different approaches within those categories. Section 2.6 illustrates the machine learning-based and the workflow definition-based approaches. Section 2.7 discusses the Matching-based approaches and the underlying ontology matching techniques. Table 2.7 presents the significant issue of Section 2.5.

Issue Number	Issue Description
Issue 6	Manual annotation of Web services is a difficult, error-prone and time-consuming task. Therefore, automating the annotation process is a pressing need in the SWS arena.

Table 2.7: Issues Identified in Section 2.5

2.6 Machine Learning-based and Workflow Definition-based Approaches

2.6.1 Machine Learning-based Approaches

A framework for learning domain specific taxonomies from textual descriptions of Web pages of Web services is proposed in (Chifu et al., 2007). The taxonomy learning process is composed of two steps; concept extraction followed by taxonomy building and pruning. Taxonomy concepts are extracted based on recognising linguistic patterns in a text corpus. The taxonomy learning process is based on hierarchical self-organising maps. The generated taxonomies represent

the domain ontologies necessary for annotating Web services. Taxonomy concepts are used to semantically annotate inputs and outputs of Web service operations (Chifu et al., 2007). Though useful, the work only explains the process of domain ontology building and says nothing about the process of Web service annotation. Moreover, the taxonomies built could be used to annotate the input and output parameters of a Web service but without considering the other important Web service elements that should also be annotated.

An approach to automatically create metadata from training data to semantically describing a Web service is developed by Heß and Kushmerick (2003). The training data comes from HTML pages documenting the service and the WSDL file of the service. Three different interrelated types of metadata are created. The first type is the category taxonomy that categorises Web services. The second type is the domain taxonomy, which describes the functionality of a specific service operation such as `'searching for a book'` or `'querying an airline timetable'`. The third taxonomy describes the semantic categories of input and output data such as `'book title'` or `'destination airport'` (Heß and Kushmerick, 2003). This approach considers the annotations of inputs, outputs, category and functionality only.

ASSAM is a tool developed by Heß et al. (2004) to semi-automatically annotate elements of a WSDL file. ASSAM suggests which ontology class should be used to annotate a WSDL element. The tool exploits an iterative relational classifier to semantically categorise Web services, their operations and parameters. The tool learns from previously annotated Web services, which provide training data (annotated Web services) from which ASSAM learns in order to predict annotations for new Web services. Furthermore, ASSAM uses a schema matching technique to aggregate data returned by a number of Web services that are semantically related. Having completed the annotation process, semantically annotated WSDL files can be exported into OWL-S and a concept file that contains complex data types. Though relatively comprehensive, ASSAM has limitations:

- Previously annotated services are prerequisite for ASSAM. Existing annotated services are not always available to be used as training data for other services: This deficiency limits the utility of ASSAM to annotating services belonging to domains that have many annotated services.
- The generated OWL-S process model allows only one atomic process per operation because ASSAM does not handle workflow definitions.

An approach which uses knowledge that exists in domain models to train the semantics of Web service data represented in WSDL documents is proposed in (Lerman et al., 2006). The system starts by querying a domain model in order to populate it with instances of all the semantic types. Two classifiers are then used. The first classifier, which is metadata-based, predicts the data types of inputs using concepts taken from WSDL documents. While the second classifier, which is content-based, predicts the semantics of output data after successfully invoking the Web service with correct input data. This approach differs from (Heß and Kushmerick, 2003) by adding a verification stage to guarantee a correct prediction of input data and generation of output data, but has some drawbacks. First, the system does not find any semantic metadata suitable for an input if an appropriate semantic type does not exist in the selected domain model. In this case, the input under consideration is left without annotation. Consequently, a service annotated using this approach is likely to have many elements that are not annotated. This latter problem is called the 'Low Percentage Problem'. Second, the search for appropriate semantic metadata becomes expensive when the Web service has more than two inputs (Chifu et al, 2007).

2.6.2 The Workflow Definition-based Approach

A framework for automatically annotating Web services based on so called "tried-and-tested" workflows is proposed in (Belhajjame et al, 2008). Constraints on the annotation of Web service operation parameters are inferred based on their links to other annotated operation parameters in the workflow.

The authors claim that their approach is effective in detecting errors in existing annotations. If a workflow produces correct results then the parameters of the linked operations are semantically compatible. Pre-existing annotated Web services and "tried-and-tested" workflows, which are not always available, are prerequisites for this approach to work.

2.7 Using Ontology Matching for Semi-automatic Annotation of Web Services

This class of annotation techniques utilises existing shared domain ontologies for annotation rather than developing new ones. Performing the desired semi-automation of annotation requires the following two processes: (1) WSDL files of services along with their XSDs have to be represented using application ontologies; and (2) shared domain ontologies and application ontologies are matched using ontology matching techniques. Due to the central role of matching techniques in automating the annotation process, they are discussed in detail in Subsection 2.7.1 while previous matching-based semi-automatic annotation approaches are presented in Subsection 2.7.2.

2.7.1 Ontology Matching

IS ontologies can be heterogeneous at the syntactic as well as the semantic levels. The syntactic differences result from using dissimilar representation languages. This issue can be solved by using the same language or translating an ontology from one language into another. While semantic mismatches occur due to differences in terminology, meaning, interpretation or conceptualisation between ontologies developed for the same universe of discourse (Sheth and Larson, 1990; Euzenat and Shviko, 2007 pp. 40).

Ontology heterogeneity prevents systems that use different ontologies from communicating and interoperating effectively (Shvaiko and Euzenat, 2008; Mascardi et al., 2009). The solution to this problem is ontology matching (Rodriguez and Egenhofer, 2003) which has synonyms in the literature such as; ontology mapping (Kalfoglou and Schorlemmer, 2003), semantic matching (Giunchiglia and Shvaiko, 2004) and semantic coordination (Bouquet et al., 2003).

Kalfoglou and Schorlemmer (2003 pp. 4) define ontology matching as "The task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected". In this Subsection, the focus is on the major similarity calculation methods because most of the developed annotation and matching tools and frameworks modify one or more of the methods and combine them in a way to achieve the desired goal. Figure 2.3 depicts the matching process between two ontologies.

In general, the input of the matching process is two ontological entities belonging to two different ontologies. The process utilises different similarity measurement techniques in order to perform a more accurate matching (Liping et al., 2007). The results of the individual techniques must be combined in an appropriate way to give an overall matching score for the entities under consideration (Euzenat and Valtchev, 2004). The matching score is then assessed against a threshold to decide whether the two given entities match or not.

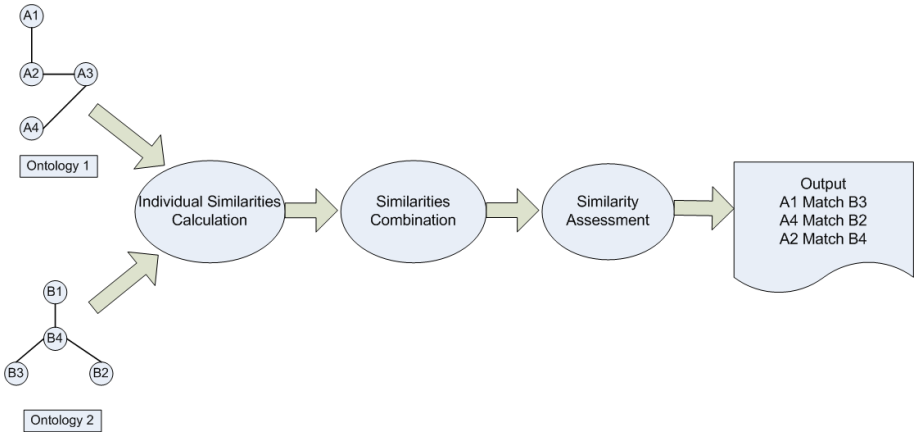

Figure 2.3: The General Matching Process Explained

Several authors have classified matching techniques (Shaviko and Euzenat, 2005; Rahm and Bernstein, 2001; Noy, 2004; Giunchglia and Shvaiko, 2004; Choi et al., 2006; Euzenat and Shaviko, 2007 pp. 61). The classification of Euzenat and Shaviko (2007 pp. 61) is adopted here as it is the latest and most comprehensive. They classify the basic techniques into four major non-mutually exclusive categories which are explained in some details as follows:

Name-based Techniques

These techniques utilise labels of ontological entities and can be further categorised into string-based and linguistic-based techniques. String-based techniques such as Edit Distance (Levenshtein, 1965) are syntactic-based techniques which consider a concept as a string of characters (see Cohen et al., (2003) for a comparison between string-based techniques). Linguistic-based methods exploit external thesauri such as WordNet (Miller, 1995) which accommodates concepts with synonym and hyponym relations. These methods can discover that `'Car'` and `'Automobile'` are synonyms. Thesauri usually organise linguistic resources in a graph or network like structure, which is used to find paths between nodes and calculate similarities. Effective use of linguistic methods is, however, subject to availability of linguistic relations between concepts under consideration. General purpose thesauri might miss domain specific concepts and thus could be unable to find some matches. Therefore, it is

possible to design domain specific linguistic databases and utilise them in applications (Budanitsky and Hirst, 2006). Labels of ontological entities are normally composed of single or multiple words (Castano et al., 2006; Nagy et al., 2009). Consequently, name-based matching methods must be able to effectively measure similarities between labels that contain multiple words. These labels are called Compound Nouns (CNs) (Sorrentino et al., 2009). A few name-based matching approaches can match labels that contain CNs, however, they cannot perform accurate automatic matching (Su and Gulla, 2004; Castano et al., 2006). Due to the high significance of CN matching in the context of this research, existing CN matching approaches and their limitations will be discussed in detail in Chapter 5. In addition, a novel and effective CN matching mechanism will be proposed and evaluated in Chapter 5.

Structure-based Techniques

These can be broken down into internal and relational structural techniques. Internal techniques exploit the internal structure of ontological entities such as; primitive datatypes, and cardinality restrictions. These techniques can be used for clustering purposes prior to the actual matching process (Rahm and Bernstein, 2001). Though, relational techniques make use of the structural relations such as super/subclass and properties of ontologies. For example, if a subclass in one ontology matches a subclass in another ontology, then their superclasses can potentially match. An example of using properties of ontologies for matching is when an ontology has two classes (B1 and B2) linked by a property X and another ontology has other two classes (D1 and D2) linked by a property Y. If B1 matches D1 and X matches Y then, B2 probably matches D2. This class of techniques is useful however; they have some deficiencies such as: (1) A matcher based on super/subclass relation might consider subclasses of a super-class as being the same; and (2) a matcher based on the relational structure might give inaccurate matching. This deficiency can be explained by the following example. Let us assume that an ontology has that two classes `'individual'` and `'Product'` linked by a property `'is bought by'` and another ontology which has two classes `'Organisation'` and `'Product'` linked by a property `'is`

`bought by'`. A relational-based matcher matching the previous two ontologies may find that `'Individual'` and `'Organisation'` are correspondences because `'Product'` match with `'Product'` and `'is bought by'` matches with `'is bought by'`. Therefore, relational techniques are usually used with other techniques. Structure-based techniques can be syntactic when the similarity calculation is based on syntactic means like string-based matching whilst they are semantic when the similarity computation is performed by semantic means. In the later case, meaning of ontological entities along with domain and structural knowledge are represented using logical formulae. Therefore, the problem of finding a match turns into a logical deduction (Bouquet et al., 2003) which can be carried out using a logical reasoner such as the SAT (propositional SATisfiability technique) decider.

Extensional techniques

These techniques are only feasible when instances of classes and properties are available. The main idea behind these techniques is: If two concepts have the same set of individuals then they can be the same. A more tolerant approach can define two concepts as overlapping when they share some of their individuals. An example of this category is formal concept analysis which can analyse the given data and organise it in a concept lattice (Zhao et al, 2006). Extensional techniques are not always applicable since individuals are not always available in ontologies.

Semantic-based techniques

These techniques are based on logical deduction and thus cannot work alone, requiring an initial processing to produce prior matching entities in order to deduce new matches. The prior matching can be produced by matching the two given ontologies to an external common ontology using name or structure-based techniques. Based on the prior matching results, logical formulas can be constructed. Having had the formulas, a reasoner such as; SAT decider or description logic reasoner can be used to deduce new matches.

The above techniques cannot perform a good matching when used in isolation; therefore, composing a number of techniques is usually needed in order to produce a better matching. Grouping individual techniques is generally based on producing a weighting system to combine the separate similarity scores. Weighting is an important mechanism as it enables maximising scores of more important techniques and minimise scores of less important ones. Weights can be assigned manually or automatically. Automatic weighting can be performed using methods such as machine learning (see Ehrig and Sure (2005) for an example). Manual weighting can present some problems when dynamic weighting is needed to optimise the overall similarity score.

The matching problem has gained monument during the last decade (Rahm and Berrnstein, 2001; Kalfoglou and Schorlemmer, 2003). Many fully and semi-automatic tools (Giunchiglia et al., 2004; Do and Rahm, 2002; Madhavan et al, 2001) have been proposed to solve the matching problem. Some of these attempts target the general matching issue. Examples are FOAM (Framework for Ontology Alignment and Matching) developed by Ehrig and Sure (2005), QOM (Quick Ontology Matching) proposed by Ehrig and Staab (2004) and Prompt (Noy and Musen, 2000). Other approaches are designed as solutions to specific problems such as; catalogue integration (Bouquet et al., 2003) and Web service composition (Wu et al., 2007; Pahi and Zhu, 2006).

Ontology matching is a very difficult and computationally expensive problem especially when considering the matching at a general level. It is more efficient and effective to design matching solutions for specific problems rather than designing general solutions, however. The reason is that the more internal and external ontological features and constraints are available to the matching process, the more accurate the matching is (Euzenat and Valtchev, 2004). To summarise the important issues discussed in Subsection 2.7.1, Table 2.8 presents the most significant ones.

Issue Number	Issue Description
Issue 7	Individual matching techniques do not normally provide satisfactory results thus combinations of these techniques are usually used to perform better matching.
Issue 8	Existing name-based matching techniques cannot provide accurate automatic CN matching.

Table 2.8: Issues Identified in Subsection 2.7.1

The following Subsection describes how matching techniques are utilised by the existing research to semi-automate the annotation of Web services.

2.7.2 Matching-based Semi-automatic Annotation Approaches

The METEOR-S Web Service Annotation Framework (MWSAF) which is part of the METEOR-S Project (http://lsdis.cs.uga.edu/projects/meteor-s/) is developed by (Patil et al., 2004) to add semantics to WSDL documents of Web services. MWSAF (METEOR-s Web Service Annotation Framework) uses ontology matching techniques to semi-automatically annotate WSDL documents with appropriate concepts from domain ontologies. The framework suggests a transformation of domain ontologies and the XML schema of WSDL documents into a common representation called SchemaGraph in order to enable structural matching. Once a common representation is achieved, every concept in the WSDL graph is matched against every concept in the domain ontology graph using two matching techniques. The two techniques are element level (name-based) and schema level (structural-based) matching techniques. The element level matching utilises N-Gram as a string-based mechanism and synonym-based similarity as a linguistic mechanism. Having completed the matching process, only the best matches are selected by a function called getBestMapping. Moreover, MWSAF classifies Web services into semantic categories taken from domain ontologies where the percentage of domain ontology concepts, used to annotate the Web

service, is considered (Patil et al., 2004). This framework suffers from some limitations:

- XSD and OWL transformation to schema graph is performed by manual means which makes this transformation hard to achieve. In addition, this transformation is no longer necessary as OWL is the dominant ontology representation language. This drawback limits the expressiveness of the produced ontology as OWL is more expressive than a graph.
- This approach can measure similarities between labels containing CNs using basic similarity mechanisms that ignore the linguistic structure of CNs (Compound Nouns). Other authors (Kim and Baldwin, 2005) have noted that this ignorance may result in imprecise similarity scores, however.
- This framework becomes computationally expensive when the number of candidate ontologies increases.

METEOR-S was modified to enable the generation of OWL-S descriptions from semantically annotated WSDL documents (Rajasekaran et al., 2005). The modified framework implements a Naïve Bayesian Classifier to classify a service into a specific domain. Then, an ontology describing the same domain is selected to annotate the service. This addition of the classifier makes the approach less computationally expensive but on the price of limiting the annotation scope to a single ontology. A service description may span more than one domain, however. Therefore, many service elements may not have appropriate ontological matches and thus may be left without annotation. Many none annotated service elements cause the Low Percentage Problem.

A framework for generating OWL-S descriptions from WSDL files was developed by Duo et al. (2005). The process of generating OWL-S starts with manually translating XML schema of a WSDL description into an intermediate OWL ontology. This transformation uses rules such as: (1) Complex, simple and global XSD elements are translated into Owl:Class; while (2) local element of type of simple type is translated into Owl:DatatypeProperty. Then, the intermediate ontology is mapped to existing domain ontologies using name-based

and structural similarity measures. The mapping result is used to generate the desired OWL-S description from the WSDL file. The mapping rules from WSDL to OWL-S are given as follows:

- A WSDL port type becomes a process model in OWL-S.
- An operation in WSDL is mapped to an atomic process in OWL-S.
- Inputs and outputs messages of an operation are mapped to inputs and outputs of an OWL-S atomic process.
- A WSDL message part type becomes an OWL-S parameter for that message part.

The approach has some drawbacks:

1. The implemented name-based matching uses Levenshtein Distance to measure similarities between labels containing single terms only. Though relatively effective, this approach cannot measure similarities between labels that have CNs.
2. The approach requires manual ontology building which is a hard and time-consuming task.
3. Many service elements could end up without annotation since this approach uses a limited set of ontologies and do not utilise an effective ontology extension mechanism.

Subsequently, services annotated using the latter approach may suffer from the Low Percentage Problem.

Lei et al. (2008) proposed an approach for annotating WSDL files. They claim that their approach can improve the efficiency of the annotation process by performing an initial name-based matching step to create a set of ontological concepts that are the best matches of the given XSD element. Having had the set, structural matching is implemented to find the best matches among elements of the set. Nevertheless, this approach is not a comprehensive one because it does not provide clear guidance for transforming an XSD to a temporary ontology.

Moreover, the approach uses very basic matching mechanisms that cannot provide accurate CN matching.

Another Web service annotation approach is developed by Zhang et al. (2008). This approach is similar to the one proposed by Duo et al. (2005) since it utilises the same XSD to ontology transformation rules. This approach produces services in the SAWSDL format. In this annotation mechanism, H-MATCH (Castano et al., 2006) is utilised as a matching tool to find correspondences between an intermediate ontology that represents a service and shared ontologies. H-Match can measure similarities between CNs but it requires the addition of CNs, that do not have entries in WordNet, to a newly constructed thesaurus. Once that is done, a similarity calculation can be carried out between CNs and other single terms or CNs that already exist in WordNet. The addition of new CNs to a newly developed thesaurus may delay and complicate the semi-automatic annotation process. This is because creating new entries needs some degree of human involvement to extend the constructed thesaurus with new entries. For more details about H-Match see Section 5.4.

2.8 Limitations of Previous Research

Unsurprisingly, given the immature nature of the state-of-the-art, current semi-automatic annotation frameworks need more development to be able to achieve the required Web service annotation task more efficiently and effectively. We therefore now use the outcomes of the previous section to summarise four important issues.

First, no approach can annotate all the required service elements provided in the synthesised set. Second, approaches based on ontology learning have some deficiencies:

- Ontology learning mechanisms are still under development and thus may produce semantically poor ontologies. The learned ontologies are usually in

the form of taxonomy where important ontological constructs such as; properties and axioms are not captured.

- The data required for learning purposes is not always available. For example, ASSAM requires annotated services as training data to annotate similar services. These existing annotated services cannot be always found due to the limited number of existing SWS.
- The resulting application ontologies are representations of sole services instead of being precise representations of shared domain knowledge. Therefore, matching the produced ontologies against shared ontologies is still required either at design time or at run time when service related activities such as discovery and composition are performed.

Third, the approach based on workflow definition is only effective for checking the correctness of an annotation rather than performing the annotation itself. The reason is that already annotated services that can be composed with new services and "tried and tested workflow" are very hard to find in practical cases. And fourth, using semantic matching to perform Web service semi-automatic annotation is promising due to the following reasons: (1) The existence of a family of matching techniques that can produce reasonable matching results; and (2) the ability to reuse and share existing ontologies for annotating many services. The later is a significant reason as manual ontology building is not an easy task and automatic ontology building, at best, produces semantically poor ontologies. Existing semantic matching-based techniques suffer from numerous limitations;

- They require manual development of application ontologies to model implicit semantics of WSDL files of candidate services. Manual ontology building is a tedious and difficult process that requires extensive domain and technical knowledge.
- Implemented matching approaches cannot provide effective and precise matching results when labels of service data and/or ontological entities contain CNs. The reason is that these matching approaches do not take the linguistic structure of CNs into consideration during the similarity calculation process (Kim and Baldwin, 2005).

- The Low Percentage Problem: Using existing annotation approaches may result in many unannotated service elements. This is because the shared ontologies used for annotation may not have suitable correspondences for all service elements that should be annotated. Used ontologies are relatively incomprehensive and miss some domain concepts. Subsequently, annotating Web services to these ontologies without a dynamic and effective ontology extension mechanism that can add necessary concepts to ontologies will lead to the *Low Percentage Problem*. This latter problem has a very negative impact on the performance and utility of annotation approaches.
- The matching process is computationally expensive when a framework uses many existing ontologies for annotation. This expensiveness limits the efficiency and usability of semi-automatic annotation. Therefore, a method is usually needed to reduce the number of potential ontologies and improve the efficiency of automatic annotation.

To summarise, Table 2.9 defines the significant issues discussed in Section 2.8.

Issue Number	Issue Description
Issue 9	All existing semi-automatic annotation approaches cannot annotate the whole set of synthesised elements.
Issue 10	Learning-based approaches suffer from the following limitations: (1) Poor quality ontologies; (2) training data is hard to find; and (3) ontologies are not representations of shared domain knowledge but models of individual services.
Issue 11	The workflow-based approach is only useful for checking annotated services but not to produce new annotations.
Issue 12	Matching based approaches are promising however they share the following deficiencies: (1) They require manual building of application ontology; (2) implemented name-based matching cannot perform accurate CN matching; (3) the low Percentage Problem; and (4) they can be computationally expensive when annotating to many ontologies.

Table 2.9: Issues Identified in Section 2.8

This research will address the issues that are relevant to its aim. Other issues can be addressed in future research. Table 2.10 presents the issues to be tackled in this research.

Issue Number	Issue Description	Reason for Addressing
Issue 1	Existing Web service standards lack the necessary semantics.	This research will add semantics to Web services.
Issue 2	Manual ontology building is hard.	To avoid manual ontology building in the provided approach.
Issue 4	Ontology extension is useful for expanding and reusing ontology.	To extend and reuse ontologies for annotation.
Issue 6	The difficulty and ineffectiveness of manual annotation of Web services.	The research semi-automate the annotation task.
Issue 7	Individual matching techniques do not provide satisfactory results and thus they should be combined.	To develop and use a combination of individual techniques in the annotation approach.
Issue 8	Existing name-based matching techniques cannot provide automatic and accurate CN matching.	Labels of Web service elements and ontological entities contain CNs.
Issue 12	Matching-based annotation approaches have limitations.	The research will overcome many limitations of existing approaches.

Table 2.10: An Overall Summary Table Highlighting the Issues to Be Addressed in the Research

2.9 Summary

This chapter presented previous research in the area of semi-automatic annotation of Web services. Web services and their industrial standards were discussed. The discussion showed that existing standards such as; WSDL, UDDI and SOAP do not support automatic discovery and composition of services because they miss the important semantic constructs modelled in ontologies. Due to their central role in the SWS area, ontologies, their engineering, learning and extension were presented. The SWS idea which is a proposed solution to automate the discovery and composition of Web services was then illustrated. After, the major SWS description frameworks were presented. Since those major frameworks do not agree on the service elements that should be annotated, a synthesis of service elements that should be semantically described was developed and used to assess the completeness of the major frameworks.

The SWS literature reports that SWS adoption by developers is low because of the difficulty of manual annotation of Web services. Consequently, automating the annotation process is a key issue for SWS success. Therefore, the existing semi-automatic annotation approaches were discussed and classified into three categories which are: Learning-based, workflow definition-based and matching-based approaches. Limitations of those annotation approaches were uncovered and issues like annotation difficulty and efficiency, matching effectiveness and the Low Percentage Problem were raised. This chapter concludes that matching-based annotation approaches are promising because exiting ontologies can be effectively shared between services and there are some matching approaches that can be combined and used to automate the annotation task. Those matching-based frameworks, however, suffer from numerous limitations which require attention.

In order to benefit from matching-based semi-automatic annotation, the difficulty of the annotation process which results from the manual ontology building of application ontologies should be minimised. Furthermore, the effectiveness and accuracy of matching especially when it comes to CNs should be improved. In

addition, the Low Percentage Problem has to be sorted in order to improve the adoption of semi-automatic annotation approaches. Finally, a method should be developed to automatically select ontologies when performing annotation. Such an automatic selection mechanism can reduce the computational expensiveness of matching approaches when multiple ontologies are used for annotation.

Chapter 3: Research Design and Approach

3.1 Overview

This chapter describes DSR (Design Science Research) as the research approach used in designing and testing the proposed semi-automatic Web service annotation framework. DSR is a problem solving paradigm: It offers solutions to research problems by providing useful design artefacts. These artefacts should be designed and evaluated using appropriate methods.

This chapter is structured as follows: Section 3.2 highlights the different research approaches employed in IS research showing the importance of selecting the right research method in answering the defined research question. Section 3.3 discusses in detail the Design Science Research (DSR) paradigm, its philosophies and processes. Section 3.4 describes the employment of DSR in the context of this research and illustrates the design increments and evaluation of the proposed approach and its artefacts. Section 3.5 defines the artefacts produced in this research and classifies them based on a widely used classification approach. Finally, Section 3.6 summarises the chapter.

3.2 Research Paradigms and Approaches in Information Systems

Research in Information Systems (IS) has attracted increasing attention in the last decade because IS can improve effectiveness and capabilities of organisations

(Nunamaker et al., 1991). The nature of IS research is complex because the IS field is multidisciplinary as IS has strong links with other domains such as medicine, engineering and social science. These richness and varieties in the IS field result in having different IS research methods (Land, 1992). Traditional IS research such as Chua (1986) and Orlikowski and Baroudi (1991) differentiate between two major research paradigms which are Positivist and Interpretive: These two types are briefly explained as follows:

- The IS research can be categorised as positivist when there are: (1) hypothesis generation and a set of quantifiable dependent and independent variables; (2) tests of proposed hypothesis; and (3) drawing of conclusions and inferences about the examined phenomenon from a representative sample set of the population (Orlikowski and Baroudi, 1991). Positivist researchers believe that hypothesises about reality can be tested independently of the researcher and the used tools.
- The IS research is described as interpretive when knowledge of reality is characterised by social context and factors such as language, shared understanding and meaning, tools and documents. Hence, interpretive research aims to investigate and understand the reality represented by IS context and the mutual influence between IS and its context (Walsham, 1993 pp. 62).

The other IS research approach that has emerged and characterised in the last decade is the DSR paradigm. DSR aims to produce useful and usable novel artefacts that can solve important research problems and change current social or organisational states into better ones (Hevner et al., 2004). Research iterations or increments are very significant in DSR processes since they result in generating knowledge and learning about the studied phenomenon and produce and improve the desired artefacts (Nunamaker et al., 1991; Vishnavi and Kuechler, 2004).

Hevner et al. (2004) provide a framework for IS research for both behavioural and design science. In their framework, they insist on the mutual relation between knowledge base and IS research. The existing knowledge base provides background knowledge that helps conducting the intended research work. In

addition, IS research should add significant contributions to the knowledge base. Contributions of IS research are, however, examined by applying them to business or social needs in appropriate environments. Research rigour is guaranteed by applying existing foundations and methodologies in a suitable manner.

It is worth mentioning that behavioural science and DSR are not isolated but mutually related to each other (Gregor and Jones, 2007). For example, DSR can utilise knowledge produced by behavioural science to improve existing IT artefacts while behavioural science can be effectively used to examine the impact of produced artefacts on organisations and individuals.

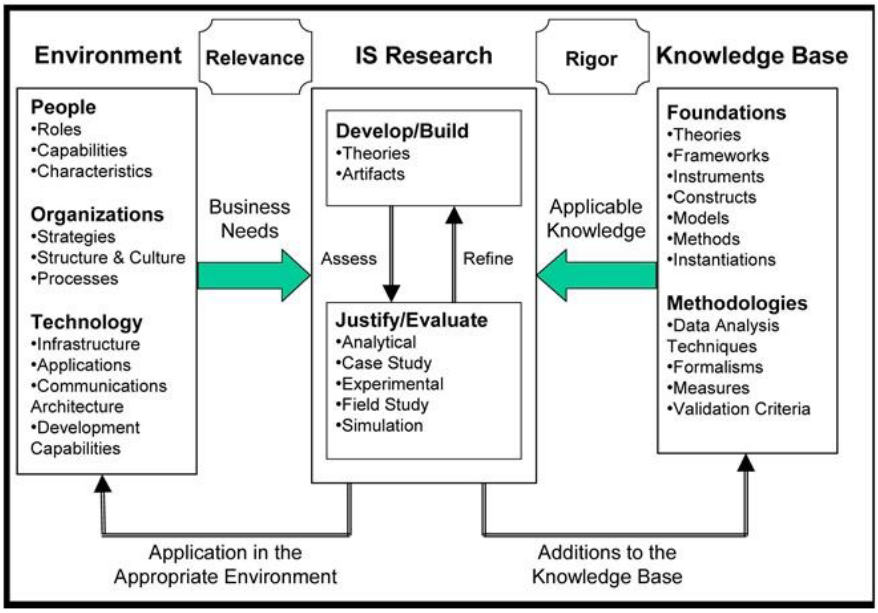

Figure 3.1: The Information Systems Framework (Source: Hevner et al., 2004)

This research tackles the problem of Web service annotation: This problem is significant since it prevents a wider adoption of SWS by Web service developers and researchers. Aiming to change the current state of SWS adoption, this research designs a new semi-automatic Web service annotation approach that overcomes limitations of existing approaches. Retrospectively, IS research is considered DSR when it aims to change a current state of an organisation into a

new state by developing novel IT artefacts (Hevner et al., 2004). Consequently, this research follows the DSR paradigm. The way in which DSR is employed in this research along with DSR processes and artefacts are discussed in detail in Sections 3.4 and 3.5.

3.3 The Design Science Research (DSR) Paradigm

DSR has recently attracted increasing attention in the IS and computing discipline: It is seen as another analytical perspective that can complement the behavioural science paradigm, traditionally the dominant paradigm in the IS area (Hevner et al., 2004; Vaishnavi and Kuechler, 2004; March and Storey, 2008). Research in DSR is impacted by Simon's view of the "Science of the Artificial" where the term artificial implies a hand-made product or artefact (Simon, 1996 pp. 123). The term design implies creating something novel that does not already exist in nature. Hevner et al. (2004 pp. 78) defines the notion of design as the "purposeful organisation of resources to accomplish a goal".

DSR is necessarily a problem solving paradigm that seeks to build IT artefacts addressing an existing problem (Nunamaker et al., 1991; Vaishnavi and Kuechler, 2004). Particularly, DSR focuses on developing and evaluating IT artefacts that are described as innovative, purposeful and novel (Hevner et al., 2004). Purposeful indicates that developed artefacts should potentially offer to organisations and individuals a 'utility' that addresses unresolved problems or provide better solutions that can enhance existing practices (Vaishnavi and Kuechler, 2004).

Unlike typical routine design activities which only focus on creating working artefacts, DSR signifies the systematic creation, capturing and communication of knowledge about and within the design process (Baskerville, 2008). Knowledge and understanding of design problems and solutions are obtained during the development, evaluation and application of designed artefacts (Simon, 1996 pp.

120). This is because the design process is seen as a learning process where understanding is enhanced as researchers are progressing in design activities. This understanding helps to improve the quality of the design process and the resulting design artefacts (See Figure 3.2).

3.3.1 DSR Processes

Researchers such as Hevner et al. (2004) and Kuechler and Vaishnavi (2008) argue that DSR projects are normally composed of certain activities or steps. The latter define five important project steps as the 'anatomy' of DSR (See Figure 3.2). The DSR process starts with awareness of problem then followed by suggestion, development and finally evaluation which in turn leads to a conclusion. All these phases are good opportunities for knowledge generation that can feed into earlier or subsequent steps.

A distinctive feature of DSR is its iterative or incremental nature which implies that the 'build-evaluate' process can be repeated or incremented until satisfactory artefacts are obtained (Markus et al., 2002). Unfortunately, most DSR scholars (Gregor and Jones, 2007; March and Story, 2008) focus on the iterative DSR and ignore incremental DSR. A description of incremental DSR, however, can be found in Simon (1996 pp. 120) stressing that the design process of a complex artefact can be broken down into more granular and semi-independent components. These components cumulatively make the desired artefact. In addition, Hevner et al. (2004) argue that DSR activities can be incremental. In incremental DSR, each artefact, part of artefact or a set of artefacts are designed during a DSR phase called an increment. It is worth mentioning that incremental design is necessarily associated with incremental learning since the understanding of the design process is improved as the design grows and more components of the final artefact are developed and evaluated.

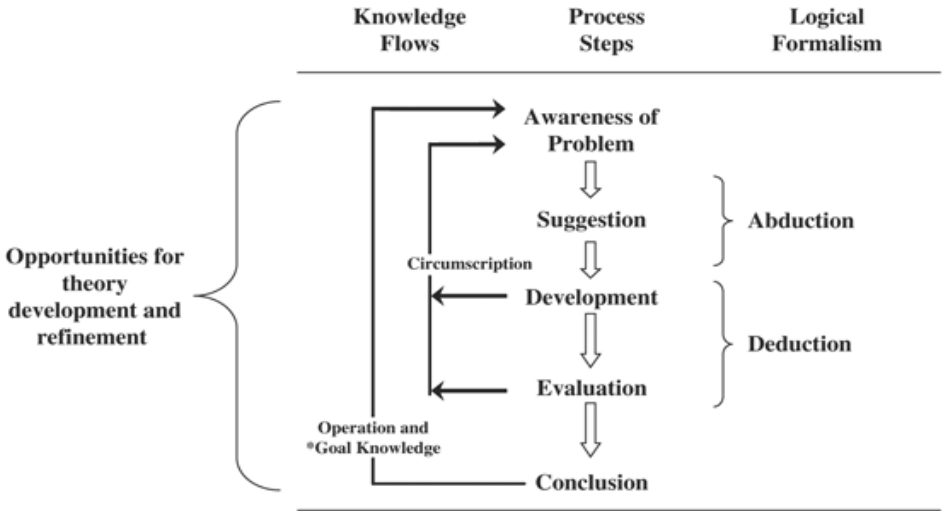

Figure 3.2: Reasoning in the Design-Science Research Cycle (Source: Kuechler and Vaishnavi, 2008)

3.3.2 DSR Evaluation

The evaluation process is significant in the context of DSR because it can generate feedback and knowledge that can lead to better understanding of the problem domain and improvements of the artefacts and design activities. To perform correct and effective evaluation, appropriate methods and metrics must be selected and used (Kuechler et al., 2005). Hevner et al. (2004) define a set of evaluation methods that can match different types of design artefacts. These methods are presented in Table 3.1.

Guideline	Description
Observational	**Case Study**: Study artefact in depth in business environment.
	Field Study: Monitor use of artefact in multiple projects**.**
Analytical	**Static analysis:** Examine structure of artefact for static qualities (e.g. complexity).
	Architecture Analysis: Study fit of artefact into technical IS architecture.
	Optimisation: Demonstrate inherent optimal properties of artefact or provide optimality bounds on artefact behaviour.
	Dynamic Analysis: Study artefact in use for dynamic qualities (e.g. performance).
Experimental	**Controlled Experiment:** Study artefact in controlled environment for qualities (e.g., usability).
	Simulation: Execute artefact with artificial data.
Testing	**Functional (Black Box) Testing:** Execute artefact interfaces to discover failures and identify defects.
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artefact implementation.
Descriptive	**Informed Argument:** Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artefact's utility.
	Scenarios: Construct detailed scenarios around the artefact to demonstrate its utility.

Table 3.1: Design Evaluation Methods (Source: Hevner et al., 2004)

3.3.3 DSR Artefacts

March and Smith (1995) and Baskerville (2008) describe DSR as a problem solving paradigm that aim to provide solutions to problems by designing useful artefacts. Orlikowski and Iacono (2001) call a DSR artefact "core subject matter". Consequently, the role of an artefact is central for any DSR project. Despite the significance of artefacts in design research, there is a lack of consensus about what constitutes a DSR artefact. Some researchers such as Orlikowski and Iacono (2001) and Benbasat and Zmud (2003) argue that IT artefacts are the only acceptable outputs of DSR. On the other hand, other researchers like (Winter, 2008) suggest that pure organisational artefacts such as those related to

organisational culture should be considered as valid DSR artefacts since the IS field is interested in not only technology but also organisations and individuals.

The classification of design artefacts provided by March and Smith (1995) is widely accepted in the DSR literature (Hevner et al., 2004), these being:

- Constructs: These are conceptual vocabulary and symbols that provides a language to define and share design problems and solutions.
- Models: They use design constructs to conceptualise the problem and its proposed solution in order to improve understanding.
- Methods: They define processes that aid the activities of searching the solution domain. These methods can be formal such as formal mathematical algorithms, informal such as natural language descriptions of approaches or a mixture of both.
- Instantiations: These are implementations of constructs, models and methods in a form of working systems. Instantiations allow IS researches to examine the applicability and appropriateness of design artefacts to their intended purposes in practical settings.

3.4 The Employment of DSR in the Context of this Project

Since the research presented in this work follows the design research paradigm, it is important to carefully and clearly present the research in the DSR format. We adopt the DSR cycle provided by (Kuechler and Vaishnavi, 2008) and presented in Figure 3.2 to illustrate the design activities carried out in this research. These activities are described as follows:

3.4.1 Awareness of problem

As presented earlier in Section 2.4, the interest in SWS has increased in the last few years because they promise to facilitate automatic discovery and composition of Web services. Semantically describing (annotating) a Web service is, however, a difficult and error-prone task when performed manually (See Issue 6). This is due to the size of Web services and ontologies as well as to the number of ontologies that can potentially annotate a given service (Hepp, 2006). Subsequently, many researches call for a solution to solve the problem of manual annotation of Web services. Having reviewed the literature, few approaches exist that aim to semi-automate the annotation task. These approaches, however, suffer from numerous deficiencies that significantly limit their usefulness (See Issues 10, 11 and 12). The main deficiencies are:

1. The approaches are difficult to use since they require ontology building which can be hard for many Web service developers.
2. The implemented matching mechanisms produce inaccurate results when the matching task involves compound nouns (CNs).
3. The Low Percentage Problem which indicates that many service elements may be left without annotation when appropriate ontological correspondences are missing.

This research is seen as a response for the calls of effective and easy to use semi-automatic Web service annotation approach which can improve the adoption of SWS by researchers and industrial practitioners. This research bridges the annotation gap by proposing a novel semi-automatic annotation approach that uses queries and employs improved name-based and structural matching mechanisms.

3.4.2 Suggestion

At this phase of the design process, the limitations of previous approaches are studied thoroughly and a subset of these deficiencies that require urgent solutions

is selected as a motivation for improvements. This subset is used to set up requirements which should be met by the new approach in order to be effective and useable. Then, the requirements are used to derive a set of design strategies for the new approach. Later, it becomes apparent that it is important to analyse the WSDL general structure in order to define what service elements can be annotated when having a WSDL file as the input for the annotation process. The processes of the suggestion activity are presented in Figure 3.3.

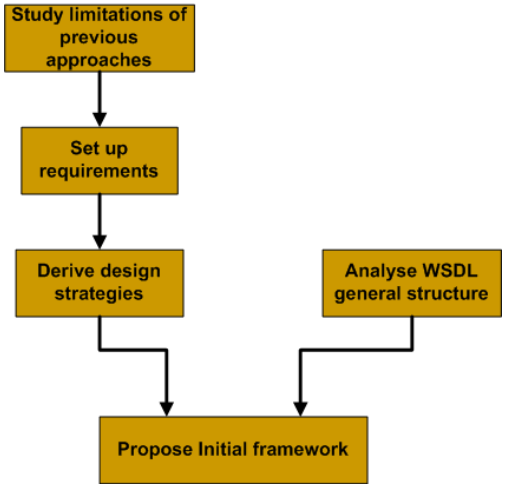

Figure 3.3: The Steps of the Suggestion Activity

3.4.3 Development

The development stage of any DSR project involves design increments or iterations carried out to provide the concrete artefacts proposed in the suggestion activity (Vaishnavi and Kuechler, 2004). The development stage of this project is composed of six design increments. In each increment an artefact or set of artefacts is developed. Additionally, every increment feeds issues and knowledge into the next increment. These issues and knowledge improve the understanding of the problem and solution domains and provide ideas and avenues to extend and improve the proposed solution. Figure 3.4 presents the architecture of design increments which are discussed in detail in the subsequent paragraphs.

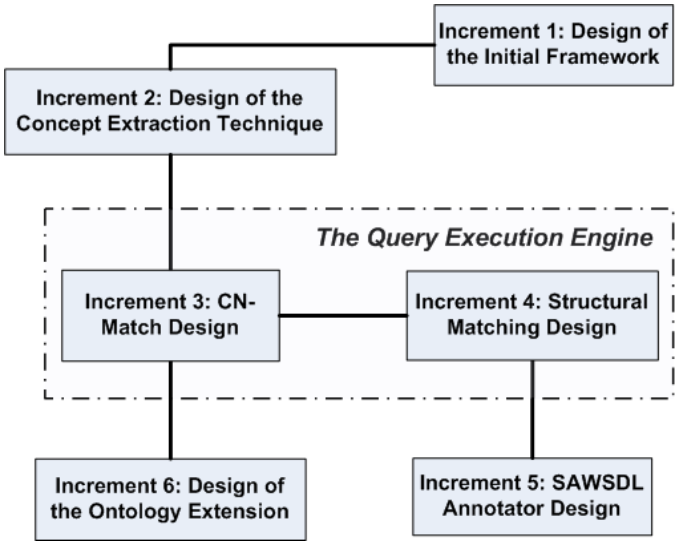

Figure 3.4: The Architecture of Design Increments

Increment 1: The Design of the Initial Annotation Framework

The design of the initial framework is driven by the design strategies and analysis results of WSDL structure. The initial framework identifies the components and phases of the approach. Five phases are defined: (1) The concept extraction phase; (2) the concept filtering and query filling phase; (3) The query execution phase; (4) the results assessment phase; and (5) the SAWSDL annotation phase. Three phases are fully automatic while the other two phases require the involvement of a human user. The three automatic phases are: (1) The concept extraction phase; (2) the query execution phase; and (3) the SAWSDL annotation phase. The role of concept extraction phase is to automatically extract the required concepts from a given WSDL files. The query execution phase is responsible for executing queries against existing ontologies using a query execution engine. The query execution engine involves two significant artefacts; the name-based matching mechanism which is called CN-Match and the structural matching mechanism. The output of a query execution is a set of recommended correspondences along with their matching degrees. The SAWSDL annotation component uses correct matches that result from manual assessment to automatically annotate given service elements based on the SAWSDL format using the Model Reference technology.

Increment 2: The Design of the Concept Extraction Technique

Since the input of the annotation process is always a WSDL file, it is necessary to extract candidate elements from this file before they can be used for the annotation task. Manual extraction of required WSDL elements is a tedious, difficult and time-consuming task; therefore the concept extraction mechanism has been developed to automate it. This mechanism employs a set of text analysis techniques of the GATE (General Architecture for Text Engineering) tool to automatically extract simple and complex XSD types. This mechanism is explained in detail in Section 4.7.

Increment 3: The Design of CN-Match

An important matching technique utilised by the query execution engine is the name-based matching: This technique measures similarities between labels of service elements and ontological entities. These labels can contain more than two constituents i.e., Compound Nouns (CNs). Unfortunately, existing name-based matching approaches cannot perform accurate and automatic similarity measurements when labels of candidates are CNs. The reason is that these approaches ignore the linguistic structure of CNs. Consequently, a new approach called CN-Match is developed. CN-Match performs automatic similarity measurements between single terms, binary and triple CNs. The design of CN-Match is based on a set of matching rules derived from the English linguistic literature on CN structure (See Section 5.3). CN-Match is implemented in Java 1.6.0 to provide a useful tool for the annotation approach and any other possible applications that require automatic CN matching.

Increment 4: Structural Matching Design

Individual matching approaches such as name-based matching cannot alone provide good results (See Issue 7). For instance, name-based matching may detect that `'Apple'` (which represents a fruit) and `'Apple'` (which represents the technology company) as matches since the labels are identical. Taking the structure as a similarity criterion may help in eliminating such wrong matches since the structures or context of candidates is considered. Consequently, a

combination of name-based and structural matching should be used for query execution. Therefore, a structural matching mechanism is developed and implemented. The implemented structural technique finds structural similarities based on matching labels of related elements of the given service element against related classes of a candidate ontological class. Related classes of an ontological class are those that have object property axioms with the given ontological class. Labels of object properties are, however, not taken into account because they do not have counterparts in queries of service elements.

Increment 5: SAWSDL Annotator Design

The annotation is performed based on the SAWSDL notation using the Model Reference technology: This technology adds a URI of the appropriate ontological correspondence to the given service element. Performing the SAWSDL annotation manually is a tedious and time-consuming task therefore, automation is needed and thus an annotator is designed. The new annotator parses a given WSDL document line by line and searches for the concept to be annotated. Once this concept is found, a model reference element with the URI of the appropriate ontological correspondence is added to the tag of the service element.

Increment 6: Design of the Ontology Extension Mechanism

The ontology extension mechanism is designed and added to support the annotation framework. This is because matching-based annotation approaches may suffer from the Low Percentage Problem if they do not use an appropriate and effective ontology extension method (See Issue 12). The provided mechanism extends the required ontology with appropriate ontological classes providing correspondences for the given service elements. Two extension methods are designed: One for simple elements and the other for complex types. The addition of these methods to the proposed annotation framework can alleviate the Low Percentage Problem since a higher percentage of service elements can be annotated. This learning about the latter problem and the added extension methods provide important knowledge that can be fed back to the knowledge base.

3.4.4 Evaluating the Semi-automatic Annotation Approach and its Components

This section describes in detail the evaluation methods and metrics of the annotation approach and its components. Evaluation methods such as testing (functional), experimental (controlled experiment) and descriptive (scenario) are employed to evaluate the different design artefacts provided by this project. In addition, measures such as Precision (P), Recall (R) and F-measure which are well known Information Retrieval (IR) metrics are utilised in the evaluation of some components.

A. Evaluating the Components of the Annotation Framework

This evaluation exercise concerns the evaluation of the automated components of the annotation framework.

A.1 Evaluating CN-Match:

CN-Match is evaluated using the experimental (controlled experiment) evaluation method. The evaluation is performed to ensure that CN-Match is capable of precisely measuring similarities between labels containing CNs. Precision (P), recall (R), and F-measure (F) are used as metrics for this evaluation. These three metrics are deemed appropriate since they are widely used to evaluate other name-based and ontology matching techniques (Euzenat et al., 2009; Giunchiglia et al., 2009). The evaluation is conducted by employing CN-Match to measure similarities between labels of classes taken from existing ontologies. Existing ontologies are utilised in the evaluation to avoid any potential bias that can result from using ontologies built for the specific purpose of this evaluation. The evaluation task is composed of two different sets of experiments. The first set comprises two experiments conducted to derive a suitable threshold for CN-Match and uses data belonging to two different domains which are the knowledge acquisition and the travel domains. The reason for having two experiments is to derive a general and domain independent threshold.

The second set of experiments is performed to evaluate the performance of CN-Match using P, R and F. Three different sets of existing ontologies (the Benchmark set, the Russian set and the Conference set) are used for this evaluation. These sets have ontologies describing different domains and having different CN coverage. Table 3.2 presents a brief description of these sets. For each set, a number of tests are performed to generate matching scores. Then, P, R and F values are measured and presented. Based on the provided results, important conclusions about CN-Match are drawn.

Set	Number of Tests	Domain Covered
Benchmark	4	Bibliographic
Russia	3	Country
Conference	8	Organisation of Conferences

Table 3.2: Description of Test Sets Used in Evaluating the Performance of CN-Match

A.2 Evaluating the Other Components of the Annotation Framework:

The other automatic components of the annotation framework which are the concept extraction mechanism, the structural matching mechanism and the SAWSDL annotation technique are evaluated according to the functional (black box) evaluation method. In other words, each component is employed individually to ensure that it is not faulty and can produce the expected results. Individual evaluations of these components are explained briefly as follows:

- Evaluation of the concept extraction technique: Three different WSDL files are used in this evaluation. The evaluation is devoted to ensure that the extraction method is able to extract all the required WSDL elements including simple types, complex types and complex relations.
- Evaluation of the structural matching mechanism: This mechanism is tested by calculating the structural matching scores of a number of pairs of classes belonging to two different ontologies.

- Evaluating the SAWSDL annotation mechanism: This mechanism is evaluated by providing the annotator with annotation results from the query execution engine and then employing this annotator to annotate simple and complex types of source WSDL files.

A.3 Evaluating the Ontology Extension Mechanism:

The evaluation of this technique is also conducted using the functional (black box) evaluation method. The evaluation is performed by employing the extension technique to automatically add appropriate ontological correspondences of some given service elements.

B. Evaluating the Whole Annotation Framework

The developed annotation framework is evaluated by applying it to a typical organisational scenario. The scenario itself is not directly identified from a live SWS project because no real industrial SWS applications are available. The scenario, however, is drawn from a proposed use case of the provided annotation approach in an organisational environment. According to Go and Carroll (2004) design that supports a specific scenario represents the development of a "proof of concept" research whose relevance is agreed and whose results are evaluated against the research objectives. Scenarios are accepted as an evaluation method and strategy in different disciplines, such as computer science, since they provide grounding for design and evaluation of research and support real-world use cases (Wack, 1985). The organisational scenario adopted in this research is presented graphically in Figure 3.5 and explained in detail in the following paragraph. In addition, the ontologies and Web services used in this evaluation and their search and selection processes are described in details in Subsection (B.1).

> *Scenario:*
> *Organisation X is specialised in SWS development: It receives WSDL files of Web services from other organisations and migrates these syntactic services into semantic ones by annotating them to*

existing ontologies based on the SAWSDL W3C recommendation. Organisation X owns a repository of OWL ontologies describing different domains. For every domain covered in the repository, there is one and only one ontology describing this domain. Consequently, Organisation X shares an ontology between elements belonging to different services but the same domain. The reason is that sharing ontologies between services allows this organisation to provide SWS that are pre-equipped with shared semantics that enable these services to interoperate easily and effectively. In addition, using a limited set of ontologies enable organisation X to maintain the quality of these ontologies and perform an ongoing maintenance of them. The annotation process at Organisation X is currently manual, difficult and time-consuming therefore, it is seeking to utilise an effective and easy to use semi-automatic annotation approach that can be used by any Web service developer.

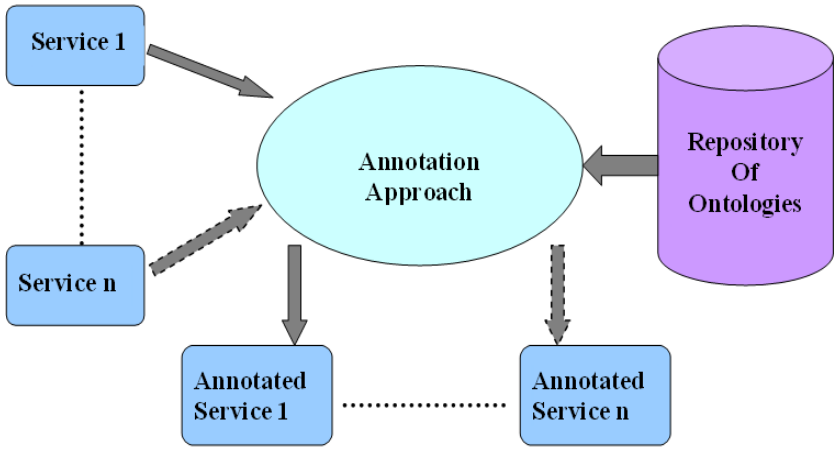

Figure 3.5: The Organisational Scenario of the Annotation Approach

B.1 Collecting Web Services and Ontologies for the Evaluation of the Annotation Approach

Existing Web services and ontologies are used to evaluate the proposed annotation approach. Using ontologies and Web services that are developed by different parties is deemed useful in this research because it allows us to mimic real life

cases when a set of Web services are annotated to a limited set of ontologies available in an ontology repository. Moreover, using available Web services and ontologies allows us to avoid any potential bias that may result from building ontologies pre-equipped with good matches of Web service elements. The selected ontologies and Web services are carefully gathered from available resources. In this subsection, the methods of searching for and selecting Web services and ontologies are illustrated. Moreover, these Web services and ontologies are briefly described.

Selecting Domains of Web Services

The selection process starts by finding domains that can have several Web services. This is an important selection strategy for the following reasons:

1. It limits the scope of the process of searching for ontologies. Finding appropriate ontologies describing many different domains is hard because a limited number of existing ontologies is available in open access repositories.
2. Having different Web services in the same domain enables us to experiment the extension exercise. The later reason is very important since annotating different Web services from the same domain using the same ontology allows us to extend this ontology when appropriate correspondences for service elements are unavailable.
3. Using the proposed approach to successfully annotate Web services belonging to different domains proves that it can be used to annotate a wide range of Web services and it is not limited to a single domain or a set of domains.

The domain selection is performed by conducting searches in available Web service repositories. As a result of the conducted search, we find that we can collect Web services from few domains to satisfy our selection strategy. An average of ten Web services is collected for each selected domain. The chosen domains are; Book, Stock Information, Weather, Communication and Payment. Many of the selected Web services contain data that belong to the User Information domain, however. Consequently, a decision to search for User Information ontology is taken.

Selecting Web Services for Each Domain

After selecting Web services' domains and sets, selected Web services are examined and those with relatively rich XSDs are taken. As a result, twenty five Web services are gathered to conduct the evaluation process. Every five of these twenty five services belong to one of the five selected domains. Table 3.3 provides some details about the selected Web services.

Domain	Web Service
Book	BookInfoPort, Service11.Accounts, Books, BookService, BookStore1
Weather	service38.Accounts, service47.Utility service43.Miscellaneous, service185 service51.Utility
StockInfo	Service3.Stock, Service7.Address, Service11.Stock, Service7.Stock, service17
Communication	Service9.Specialist, Service4.Specialist Service50.Miscellaneous, Service60.DeveloperTools Service80.Miscellaneous
Payment	Service72.Accounts, GeoCash Service24.Accounts, Service39.Accounts, Service68.Accounts

Table 3.3: Details of Selected Web Services

Selecting Ontologies for the Ontology Repository

Once the Web service selection is finalised, the process of searching for existing ontologies, that describe the general theme of the selected domains, is started. The selection criteria are:

1. Selected ontologies should be rich enough and have satisfactory level of details about the specified domains.
2. Ontologies should be developed by recognised bodies or research projects.

The search is performed in sixteen ontology repositories and search engines such as; the Ontology Yellow Pages, TONES Ontology Repository of Manchester University, Knowledgeforge, Protégé Ontology Library, Ontoselect, Sweet NASA repository and Swoogle Search Engine. As a result of the conducted search, two to four ontologies are selected for each domain. Then, for each domain, the richest ontology is selected and added to the repository. The final version of the repository contains five ontologies since the same ontology is selected for the Stock Information and Payment domains. Table 3.4 provides details about selected ontologies.

Ontology	Domain	Ontology developer
BookProperty	Book	ISLAB at the Hanyang University of South Korea
LSDIS-Finance	Stock Information and Payment	LSDIS research project at University of Georgia
WeatherConcepts	Weather	LSDIS research project at University of Georgia
MoguntiaDataTypes	Communication	Moguntia Semantic Web research project at Manchester University
Contact	User Information	The SWAP research project at the W3C

Table 3.4: Details of Selected Ontologies

B.2 Black Box Testing

After conducting the evaluation of the individual components of the annotation approach, these components are combined and evaluated together based on the black box testing method using the proposed organisational scenario. The purpose of this evaluation is to ensure that the components can work jointly to provide the desired annotation results. For this evaluation, three Web services from the Book,

Weather and Stock Information domains are selected from the set of the twenty five Web services. Then, these three services are annotated using the proposed semi-automatic annotation approach to ontologies residing in the repository. Last, the annotation steps and results are presented in Section 6.3 to prove that the annotation framework is capable of providing the expected outcomes.

B.3 Experimental (Controlled Experiment)

The experimental evaluation of the proposed annotation approach is undertaken through its application to the typical organisational scenario. P, R and F measures are used as metrics in this evaluation: These three measures are deemed appropriate for this evaluation because: (1) They are used in the evaluation of other similar annotation approaches such as METEOR-S (Patil et al., 2004); and (2) the annotation process is matching-based and these measure are normally used to evaluate matching approaches. Five sets of experiments are conducted. In each set, five Web services belonging to the same domain are annotated to the ontologies existing in the repository. For each experiment, values of P, R and F are recorded and then presented to show the strengths and weaknesses of the proposed approach.

3.5 Mapping the Artefacts of this Research to DSR Artefacts

In light of March and Smith (1995) classification of DSR artefacts (See Subsection 3.3.3), the artefacts produced in this project are presented and described as follows:

- **The initial annotation framework:** This framework represents a design method artefact because it provides steps to solve the problem tackled in this research. Steps of this method are executed by the other artefacts provided by this research.

- **The standard query template:** This template is categorised as a design construct since it is seen as a set of vocabularies that defines a part of the solution.
- **The concept extraction mechanism:** This extraction mechanism is a design instantiation artefact: It is a working prototype that can have input and provide outputs as results of processing.
- **CN-Match:** This name-based matching framework and tool is a key component of this research. CN-Match is classified as a design method and instantiation artefact: It is a method since it provides a set of steps that can be implemented to measure similarities between any two single terms, binary or triple CNs. In addition, CN-Match is considered as an instantiation because it is the Java-based implementation of the similarity measurement method. This working tool can be used not only for annotation purposes but also for the general ontology matching task and other matching activities. It is worth mentioning that the similarity measurement method can be implemented in other programming languages if it is going to be used in a none Java-based application.
- **The structural matching mechanism**: The structural matching is also a method and an instantiation. This matching mechanism is a method as it provides specific steps for measuring structural similarities. Furthermore, this mechanism is an instantiation because the structural similarity method is implemented in the Java programming language. This implementation provides a utility that can be employed by the query execution engine to measure similarities between related elements of service concepts and ontological classes.
- **The SAWSDL annotator:** This is an instantiation as it is an implementation that solves the problem of manual addition of model references to tags of service elements. The input of this annotator is the set of correct matches and the output is an annotated WSDL file.
- **The ontology extension mechanism:** This mechanism is a method and instantiation artefact: It is a method because it provides steps that can successfully perform ontology extension. These steps are implemented in the

Java language to provide a working system that can add new classes to ontologies based on a set of defined rules.

- **The semi-automatic annotation framework**: This framework is the main artefact of this research. It represents the solution to the defined research problem. This artefact is an instantiation that provides a purposeful utility which can help SWS developers in semi-automatically annotating Web services.

Table 3.5 provides classification of the DSR artefacts of this research based on March and Smith (1995) categories.

Category	Artefact
Construct	Standard query template
Model	None
Method	Initial annotation framework CN-Match Structural matching mechanism Ontology extension mechanism
Instantiation	Concept extraction mechanism CN-Match Structural matching mechanism SAWSDL annotator Ontology extension mechanism Semi-automatic annotation framework

Table 3.5: The Classification of the DSR Artefacts of this Research

3.6 Summary

This chapter presented the research method of designing and evaluating the semi-automatic annotation approach. In order to support the selection of DSR as the right method for undertaking this research, the different IS research methods were discussed and reasoning for choosing DSR was provided. After, the DSR paradigm, philosophy, processes, evaluation and artefacts were illustrated in detail. The research was then described in light of the DSR paradigm. In describing the research, increments that were performed to solve the defined problem were presented. In addition, the incremental learning that happened throughout the research activities was highlighted. Moreover, the methods, metrics and data used to evaluate the proposed annotation framework and its underlying artefacts were illustrated. Due to the central role of artefacts in any DSR project, the artefacts produced in this research were discussed and classified.

Chapter 4: The Design of the Semi-automatic Query-based Annotation Approach

4.1 Overview

This chapter proposes an annotation approach that can overcome a significant part of the limitations discussed in Chapter 2. The new approach is of a semi-automated nature and utilises query instances rather than application ontologies. In addition, this approach develops and uses a new matching mechanism for query execution employing name-based and structural matching techniques.

This chapter is organised as follows: Section 4.2 presents the design increments addressed in this chapter. Section 4.3 shows the importance of designing a new annotation approach and sets the design requirements for the new approach. Section 4.4 presents the design strategies derived from the provided requirements and the limitations of previous annotation approaches. Section 4.5 analyses the WSDL general structure to show what WSDL elements should be semantically described. Section 4.6 provides the overall design of the initial annotation approach and explains the five annotation phases. Section 4.7 illustrates the design of the concept extraction. Section 4.8 shows the design of the query execution. Section 4.9 discusses the design of the SAWSDL annotator and Section 4.10 summarises the chapter.

4.2 Design Increments Covered in this Chapter

The design of the annotation approach is composed of six increments as discussed in Subsection 3.4.3. This chapter illustrates the design of the initial annotation framework (Increment 1), the design of the concept extraction technique (Increment 2), structural matching design (Increment 4) and the SAWSDL annotator design (Increment 5). These three increments are shaded in Figure 4.1 to indicate that they are addressed in this chapter.

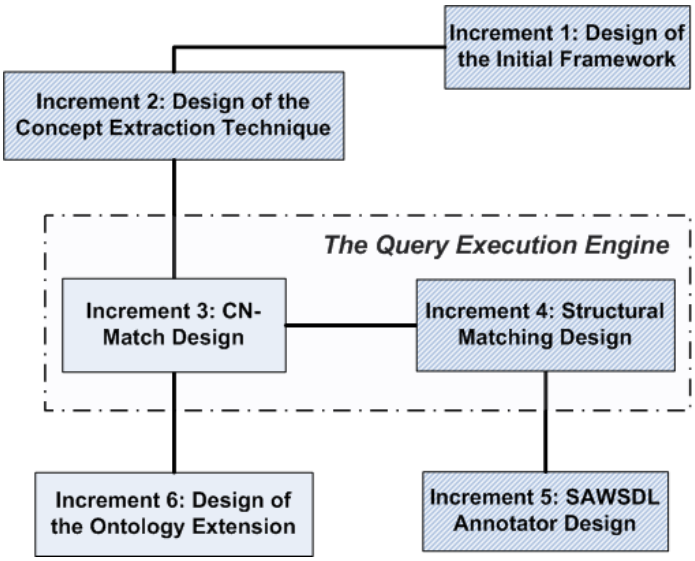

Figure 4.1: Design Increments Addressed in Chapter 4

4.3 The Need for a New Semi-automatic Annotation Framework

This research overcomes important limitations of existing annotation frameworks by proposing a novel approach. The proposed approach can be classified under the matching-based annotation category since a novel matching system is designed and implemented by the query execution engine. The approach presented is designed to overcome the following deficiencies:

1. The prerequisite of building an application ontology to model service semantics. Manual ontology building is a very hard process and automatic building is ineffective since low quality ontologies are produced (See Issues 2 and 3 in Table 2.2).
2. The inaccuracy of employed matching mechanisms: Earlier annotation approaches use matching techniques that cannot provide accurate matching results especially when labels of service elements as well as labels of ontological entities contain Compound Nouns (CNs) (See Issue 8 in Table 2.8).
3. The Low Percentage Problem: Many service elements are left without annotation for two reasons (See Issue 12 in Table 2.9): (1) The lack of an effective ontology extension mechanism that can expand the used ontologies when they miss corresponding classes of given service elements that belong to domains of used ontologies; and (2) the annotation of all service elements that belong to multiple domains to a single domain ontology.
4. Annotating service elements belonging to the same domain to multiple ontologies: Many earlier approaches especially learning-based ones do not allow the sharing of an ontology between service elements belonging to the same domain. These approaches build a domain ontology for each service and use it to annotate this service. This annotation process results in services that are annotated to non-shared ontologies and thus matching these ontologies is still required at run time when discovering or composing services. This extra automatic matching process which is performed by software agents may result in errors and delays in any future automatic discovery or composition task.

Resolving these four limitations is, arguably, seen more urgent than sorting out the other annotation problems such as; the annotation of all service elements and expensiveness of the annotation process (see Section 2.8 for a discussion about limitations). The reasons are:

1. The priority in the SWS area is to improve the SWS adoption by developing easy to use automatic annotation approaches (Patil et al., 2004). Approaches,

that can successfully annotate a subset of service elements and be extended to annotate other elements, are pressing needs.

2. The processing speed of computers is improving rapidly and thus having superfast computers can significantly decrease the computational cost of the annotation process.

To overcome the four previous deficiencies, the proposed approach should satisfy the following requirements:

R1. No application ontologies are needed to capture service semantics: The new approach should avoid the difficult process of application ontology building in order to make the approach usable by Web service developers who do not normally have knowledge and experience in ontology development.

R2. A name-based matching mechanism that can accurately measure similarities between labels containing CNs and single terms should be developed and used by the matching system.

R3. The proposed annotation approach should allow annotating a single service to multiple ontologies covering different domains.

R4. The proposed annotation approach should be able to annotate a high percentage of WSDL elements. In other words, the proposed approach should not suffer from the Low Percentage Problem.

R5. Elements that belong to the same domain but different services should share the same ontology. This will make the produced SWS ready for tasks such as discovery, composition and interoperability.

These five requirements lead to design strategies for the new semi-automatic annotation approach.

4.4 Design Strategies for the New Query-based Annotation Framework

To design an effective annotation approach, a set of design strategies that are derived from the earlier five requirements and knowledge acquired from reviewing previous annotation approaches must be adopted and presented before starting the implementation stage. These design decisions are:

1. The input of the proposed approach is a WSDL file and a set of domain ontologies describing different domains. A WSDL file is the only source of data that is always available with any service. Other service related files such as textual descriptions may not always be available (See Subsection 2.6.1).
2. The approach is query-based. A standard query template is designed and used in preference to the ontology building process used in previous matching-based approaches. This standard template can be filled with data extracted from WSDL files to produce query instances.
3. The approach is matching-based. Query instances will be executed using a novel query execution engine; which utilises name-based and structural matching mechanisms.
4. The approach is semi-automated. The output of query execution is a set of recommended correspondences along with their confidence degrees. The user of the annotation approach can select an appropriate correspondence from the provided set or rejects all matches if such a correspondence does not exist in the set.
5. The output of an annotation process is a service annotated based on the SAWSDL format.

4.5 WSDL Structure and Interpretation

As a precursor to the approach presented here and since WSDL files are the inputs of the proposed approach, it is necessary to analyse the WSDL general structure in

order to make clear what WSDL elements can be semantically described. In overall terms, a WSDL file is composed of an element declaration, type definition, interface, binding and service. The element declaration, type definition and interface provide an abstract definition of a service, while binding and service describe the implementation aspects of a service (Jacek et al., 2007).

Element declaration and type definitions are defined in the schema part of a WSDL file and provide data type definitions for input and output messages of operations and their parts. In an XSD, the elements that are direct children of a schema element are called global elements. Other XSD elements are called local elements. Furthermore, sub-elements of a complex type element are called direct child elements of that complex type. To give more insight onto WSDL structure, Figure 4.2 presents an example of a WSDL file of a Book Information service. The binding and service elements of this service are removed due to space limitation.

The data type definition (XSD) part of this WSDL document defines five global elements: `'Book'`, `'VendorPrice'`, `'ArrayOfBookInfo'`, `'Keyword'` and `'Source'`. These data types are used to define data of input and output message parts of WSDL operations. The `'Book'`, `'VendorPrice'` and `'ArrayOfBookInfo'` are defined as complex types while `'Keyword'` and `'Source'` are simple types. Every complex type has a set of child elements. For example, the `'Book'` complex type element has nine child elements: `'ISBN'`, `'Title'`, `'Author'`, `'PubDate'`, `'Publisher'`, `'Format'`, `'ImageUrl'`, `'TimeStamp'` and `'VendorPrice'`. On the other hand, elements that are of a simple type such as `'Keyword'` and `'Source'` do not have child elements.

Based on the previous brief analysis of WSDL elements, one can conclude that XSD elements including simple types and complex types along with their child elements should be annotated since they describe data of operations' messages. Other WSDL elements such as bindings and service define technical details and

thus do not require semantic annotation. XSD definition embeds implicit semantic information that requires disambiguation, however. For example, the relation between a complex type and each of its child elements is similar to an ontological property.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions>
  <wsdl:types>
    <s:schema>
      <s:complexType name="Book">
        <s:sequence>
          <s:element name="Isbn"/>
          <s:element name="Title"/>
          <s:element name="Author"/>
          <s:element name="PubDate"/>
          <s:element name="Publisher"/>
          <s:element name="Format"/>
          <s:element name="ImgUrl"/>
          <s:element name="TimeStamp"/>
          <s:element name="VendorPrice"/>
        </s:sequence>
      </s:complexType>
      <s:complexType name="VendorPrice">
        <s:sequence>
          <s:element name="Name" />
          <s:element name="SiteUrl" />
          <s:element name="PricePrefix" />
          <s:element name="Price" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="ArrayOfBookInfo">
        <s:sequence>
          <s:element name="Book" type="tns:Book" />
        </s:sequence>
      </s:complexType>
      <s:element name="Keyword" type="s:string" />
      <s:element name="Source" type="s:string" />
   </s:schema>
  </wsdl:types>
  <wsdl:message name="GetInfoHttpGetIn">
    <wsdl:part name="ISBN" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetInfoHttpGetOut">
    <wsdl:part name="Body" element="tns:Book" />
  </wsdl:message>
  <wsdl:message name="DoKeywordSearchHttpGetIn">
    <wsdl:part name="keyword" type="s:string" />
  </wsdl:message>
  <wsdl:message name="DoKeywordSearchHttpGetOut">
    <wsdl:part name="Body" element="tns:ArrayOfBookInfo" />
  </wsdl:message>
  <wsdl:message name="GetInfoHttpPostIn">
    <wsdl:part name="ISBN" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetInfoHttpPostOut">
    <wsdl:part name="Body" element="tns:bookInfo" />
  </wsdl:message>
```

Figure 4.2: WSDL File of the Book Information Service

Previous research (Duo et al., 2005; Zhang et al., 2008) defines sets of rules for interpreting the implicit semantic information embedded in an XSD definition of a WSDL file. A subset of these rules is adopted and implemented for the purpose of disambiguating the semantic information and extracting the set of concepts that will be considered during the annotation process. The rules are presented as follows.

> ***Rule One***: Each global complex or simple XSD element is considered as a concept that should be annotated.
> ***Rule Two***: Each local complex of simple XSD element is considered as a concept that should be annotated.
> ***Rule Three***: The set of child elements of a complex element formulates the set of related elements of a complex type concept.

Labels of complex and simple types do not necessarily carry significant meanings. Consequently, there are few types that should be filtered out and excluded from the annotation process. These types are:

A. Computing-specific terminologies: Computing-specific terminologies are those words or expressions that are reserved for programming languages such as Java and C++. Consequently, these terminologies do not carry a significant meaning outside of their programming languages and thus they are excluded from the annotation process. Examples of such computing-specific terminologies are `'ArrayOfBooks'` and the request response patterns such as `'BookSearchResponse'`.

B. Elements denoting processes rather than data. An example is `'GetWeatherByZipCode'`. These elements cannot be annotated using available ontologies because most existing ontologies are representations of data rather than being representations of methods or processes. An ontology can be defined as "a formal explicit specification of a shared conceptualisation" (Gruber, 1993 pp. 3); that is, a definition of concepts, axioms and relations between concepts in a formal, shared and machine-understandable format (Jasper and Uschold, 1999). Subsequently, we do not

expect existing ontologies used for annotation to contain correspondences for methods or processes. Nevertheless, some researchers argue that ontologies could be built to represent processes. Since this research merely utilises existing ontologies for annotation and most existing ontologies represent data, only service elements that denote data can be annotated. It is out of the scope of this project to build ontologies representing processes and use them for annotation.

C. Some service elements denote individuals (instances) rather than a class of individuals. An example is the service element `'Html'`. `'Html'` represents an individual of a class `'Languages'`. Since this annotation approach references service elements to classes only, the proposed approach cannot annotate service elements that denote individuals.

4.6 The Design and Phases of the Annotation Framework

The design strategies and analysis results presented earlier are used to design an annotation framework that meets the provided requirements. This section presents the design of the proposed framework. The framework is composed of phases where each phase performs a specific role and has an input and output. The phases are: (1) Concept extraction; (2) concept filtering and query filling; (3) query execution; (4) results assessment; and (5) SAWSDL annotation. These five phases are explained in detail as follows:

1. ***Concept extraction:*** The purpose of this phase is to automatically extract the service elements that will be annotated during the subsequent phases. The input of this phase is a WSDL file and the output is a set of extracted concepts. The set of extracted concepts contain simple types, complex types and relations between complex types and their child elements. Table 4.1 presents the output of the extraction phase of the Book Service shown in Figure 4.2. Section 4.7 shows how the concept extraction phase is automated.

Service Element	Child Elements
Book	ISBN, Title, Author, PubDate, Publisher, Format, ImgUrl, TimeStamp, VendorPrice
VendorPrice	Name, SiteUrl, PricePrefix, Price
ArrayOfBookInfo	Book
Keyword	None
Source	None

Table 4.1: Extracted Concepts and Relations from the Book Information Provider Service

2. ***Concept filtering and query filling:*** This is a manual process. The input is a set of extracted concepts and the output is a set of query instances. The set of extracted concepts may include some concepts that should be excluded from the annotation process since they do not carry significant meanings. These concepts can belong to one of the three categories defined earlier in Section 4.5. For example, the complex type `'ArrayOfBookInfo'` should be excluded since it denotes a syntactic definition of an array of things of type `'Book'`. Nevertheless, the `'Book'` concept is considered for annotation and therefore it can provide semantics for `'ArrayOfBookInfo'` when it is annotated. Figuring out the concepts that belong to the earlier categories cannot be performed automatically due to the lack of effective filtering techniques. Consequently, the filtering process is manual.

   The query filling part involves instantiating the standard query template to create query instances for simple and complex types. It is worth mentioning that the query filling phase is straightforward and requires neither domain knowledge nor technical knowledge since the filling process follows pre-defined steps. Figure 4.3 presents the Standard Query Template.

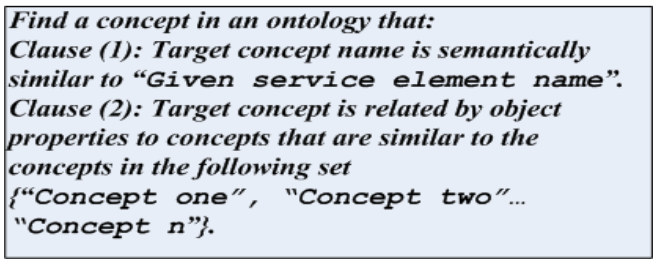

Figure 4.3: The Standard Query Template

The Query Template is designed to provide a standard format for all query instances. This Query Template has place holders for a service element and its related service elements. The template contains two clauses as shown in Figure 4.3. When filling a query for a complex type, the label of the complex type is used in Clause (1) and labels of related concepts are used in Clause (2). The resulting query instance for the complex type `'Book'` is given in Figure 4.4.

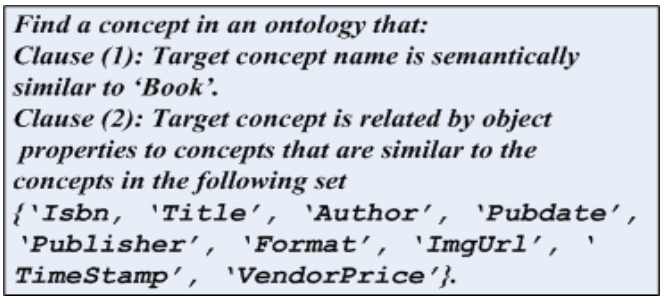

Figure 4.4: The Query Instance of the `'Book'` Complex Type

All query instances for simple types contain Clause (1) only because they do not have related concepts (child elements). Figure 4.5 shows the query instance for the simple type `'Keyword'`.

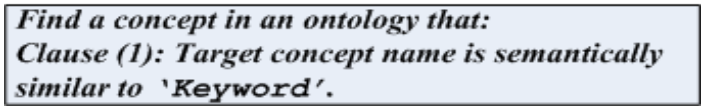

Figure 4.5: The Query Instance of the `'Keyword'` Simple Type

3. ***Query execution:*** This is a fully automatic phase whose inputs are a query instance and an ontology. The output of the query execution phase is a set of query results. The role of this phase is to execute query instances against an ontology using the query execution engine.
4. ***Results assessment:*** The input of this phase is a set of query results and the outputs are a set of appropriate correspondences and a set of inappropriate correspondences. In this phase, the user receives a set of matches as a result of query execution. The user then verifies the matches (recommendations) and chooses the correct ones as appropriate correspondences and the wrong ones as inappropriate correspondences. Nevertheless, this assessment process should be performed manually by a human user because fully automatic matching is still under development and thus human involvement can significantly increase the accuracy of query results.
5. ***SAWSDL annotation:*** This is a fully automatic process. The input is the set of appropriate ontological correspondences and the output is the annotated WSDL elements based on the SAWSDL format.

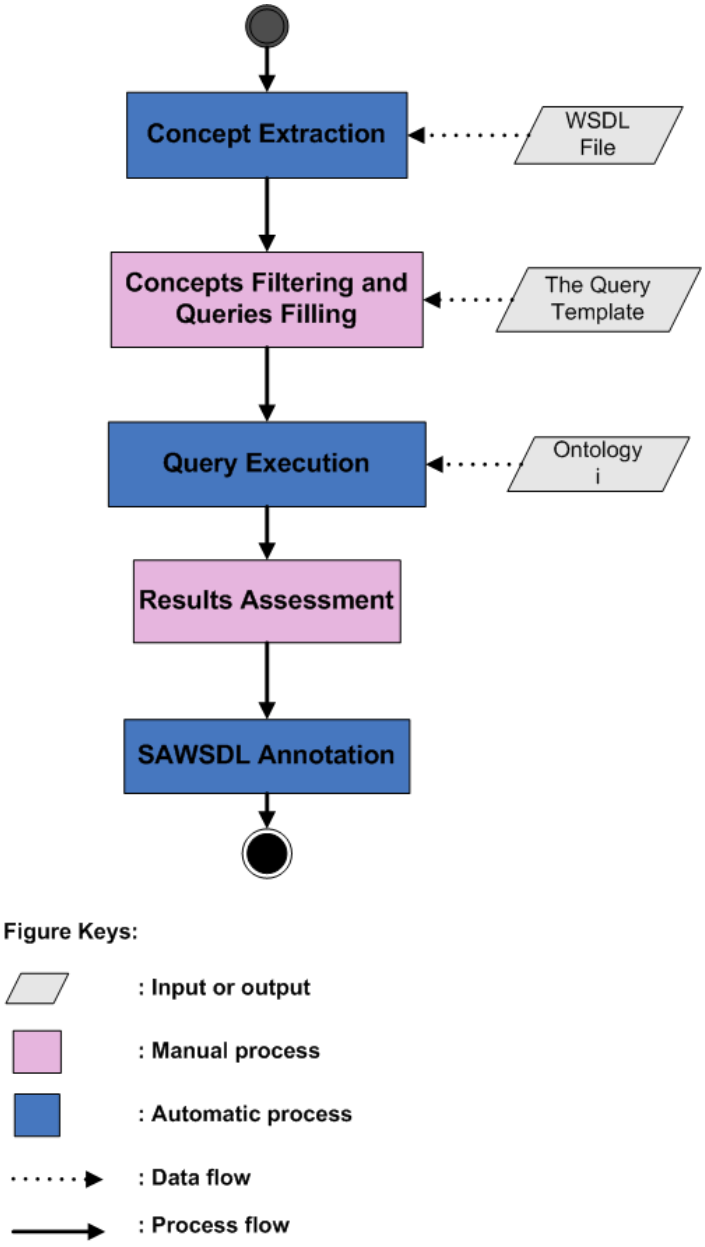

Figure 4.6: The Process Flow of the Annotation Framework

Table 4.2 provides a summary of the five phases and their important characteristics.

Phase	Automatic/Manual	Input	Output
Concept extraction	Automatic	WSDL file	A set of extracted concepts
Concepts filtering and queries filling	Manual	A set of extracted concepts	Query instances
Query execution	Automatic	Query instances and ontology	Queries' answers
Results assessment	Manual	Recommended correspondences	Appropriate and inappropriate correspondences
SAWSDL annotation	Automatic	Appropriate correspondences	Annotated elements

Table 4.2: A Summary of the Five Annotation Phases

The following three Sections; 4.7, 4.8 and 4.9 present the design of the three automatic phases which are concept extraction, query execution and SAWSDL annotation, respectively.

4.7 The Concept Extraction Phase

This phase is designed to allow automatic extraction of necessary concepts and relations between concepts from the given WSDL file based on the three rules noted earlier in Section 4.5. Retrospectively, WSDL files are very significant source of service knowledge that accompanies any service. Manual extraction of knowledge existing in a WSDL file is a difficult, time consuming and tedious task. Consequently, automating the extraction process is needed. In automating the extraction process, text analysis techniques that are packaged in the ANNIE system (A Nearly New Information Extraction System) of the GATE tool

(General Architecture for Text Engineering) are utilised (Cunningham et al., 2002). GATE is an open source tool created by the Natural Language Processing Research Group at the Sheffield University. Gate is a system that takes text as an input and provides a table of annotations applicable to this text as an output (Cunningham et al., 2002). Using ANNIE, a developer can bundle a set of components to create a sequence of processing resources called a pipeline. Examples of these ANNIE processing resources are: Sentence Splitter, Part Of Speech (POS) Tagger, Tokenizer and JAPE (Java Annotations Patterns Engine) Rules.

The ANNIE pipeline designed to automate the concept extraction phase is collaborative effort between the author and Alfaries (2010): This pipeline contains the following language processing components:

1. **Document Reset:** This resource returns the document to its original state by removing all annotations and their sets: It is always required as a preparation step before processing any text document by a pipeline.
2. **ANNIE Tokenizer:** The tokenizer splits the text into very simple tokens such as numbers, punctuation and words of different types. For example, the tokenizer differentiates between words in uppercase and lowercase, and between certain types of punctuation. The tokenizer is defined in terms of JAPE Rules and can be specified for a particular language. For instance, the English tokenizer is a processing resource that comprises a tokenizer and a JAPE transducer that is specific for the English language. The transducer has the role of adapting the generic output of the tokenizer to the requirements of the English part-of-speech tagger. The English Tokenizer should always be used on English text that needs to be processed later by the POS Tagger.
3. **ANNIE Sentence Splitter:** The sentence splitter is a cascade of finite-state transducers which splits the text into sentences. The splitter utilises a gazetteer list to distinguish sentence-marking full stops from other kinds. Each sentence is annotated with the type `'Sentence'`. Each sentence break (such as a full stop) is also annotated as `'Split'`.

4. **JAPE Rules:** JAPE is a pattern matching transducer consisting of regular expressions. A JAPE transduction contains a set of phases, each of which consists of a set of pattern/action rules. Patterns can be described in the following three ways:
   - Specifying a string of text. For example, one can write {Token.String = "of"}.
   - Specifying the absence or presence of an annotation previously provided by a tokenizer or another processing resource. For example, {!Lookup} describes the absence of a lookup annotation.
   - Specifying the attribute-value pairs of an annotation. For instance, the pattern {Token.length != 4} states that the length of the token must not be 4.

Figure 4.7 presents the developed pipeline for the automatic concept extraction task.

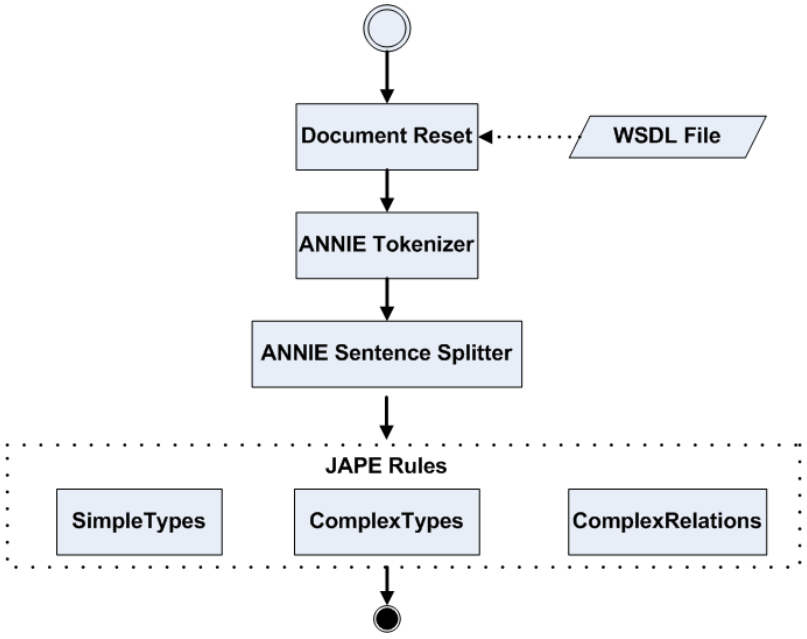

Figure 4.7: The Automatic Concept Extraction Pipeline

Three JAPE rules are developed to facilitate the desired automation of concept and relation extraction. These three rules are:

- SimpleTypes rule: To detect simple types.

- ComplexTypes rule: To capture complex types.
- ComplexRelations rule: To find the relations between complex types and their child elements.

4.8 The Query Execution Phase

A query instance is executed during the query execution phase requiring a query instance and an ontology as inputs. The Query Execution Engine is the means of performing the similarity calculation.

Each iteration of query execution takes a query instance and a candidate ontological class as inputs and produces a similarity score in the range [0-1] as an output. This score indicates how similar a query instance concept and an ontological class are. If this score is over a defined threshold, then the corresponding ontological class is added to the set of candidates (SS). Otherwise the matching is ignored. After executing the candidate query instance against all classes in ontology *i*, all classes over threshold are taken as candidates. If the SS set is empty, i.e., there are no recommended correspondences; the service element is added to the set of missing correspondences. A graphical representation of the query execution phase is given in Figure 4.8.

To allow an effective and accurate query execution, a new query execution engine is designed and implemented specifically for the purpose of semi-automatic annotation of Web services. This execution engine implements name-based and structural matching mechanisms. Name-based matching is achieved using CN-Match which is a novel name-based matching tool. Structural similarity is used to measure similarities between related concepts of a service element and those of the ontological class when executing the query. The following two subsections present CN-Match and the implemented structural matching mechanism.

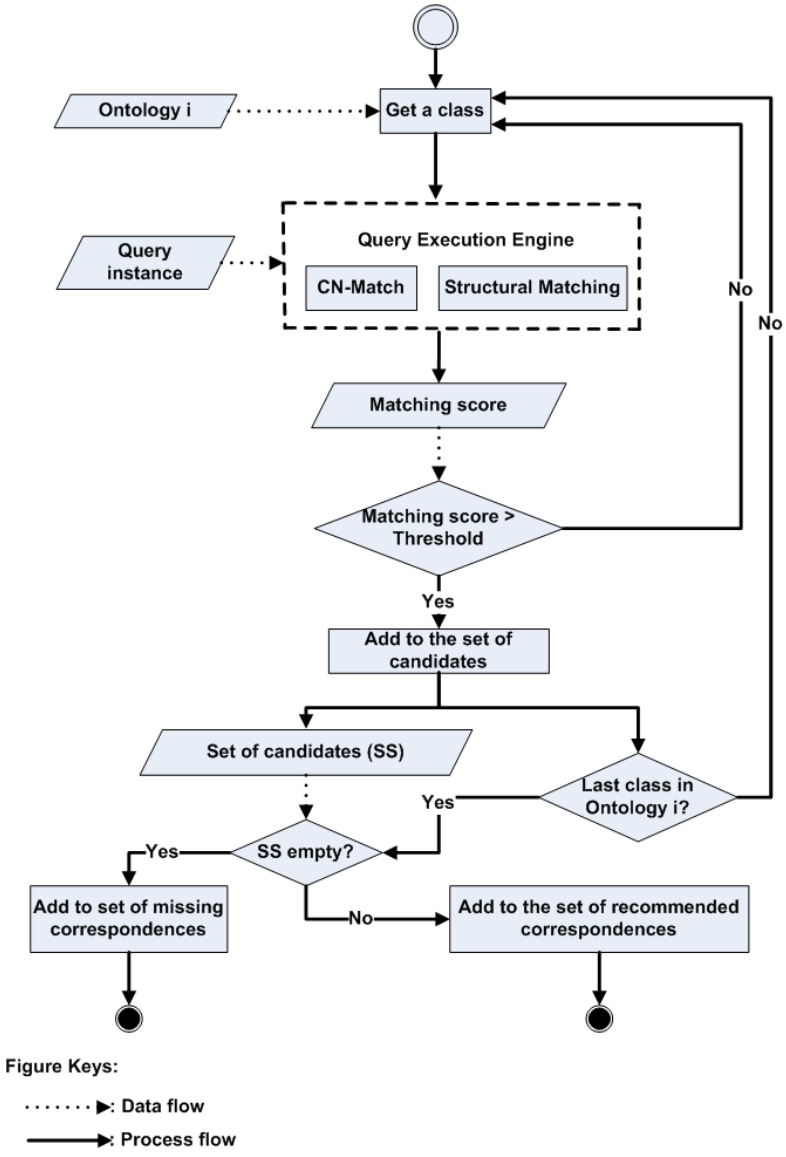

Figure 4.8: The Query Execution Phase

4.8.1 CN-Match

CN-Match is a novel and automatic name-based matching approach that can calculate similarities between labels containing single terms and compound nouns (CNs). The reason for developing and using CN-Match is that existing name-based matching techniques do not provide accurate matching results when labels of candidates are CNs - primarily because these techniques ignore the linguistic structure of CNs - (Kim & Baldwin, 2005). The CN-Match similarity calculation

mechanism is based on the fact that similarities between any two CNs can be derived successfully from similarities between their constituents (Kim & Baldwin, 2005). CN-Match employs string-based and linguistic-based matching techniques in its similarity calculation. CN-Match design, implementation and evaluation are discussed thoroughly in Chapter 5.

4.8.2 Structural Similarity

Structural matching is usually performed to enable more accurate similarity measurements between ontological entities (Euzenat & Shvaiko, 2007). Two ontological classes could have the same label but might denote different meanings. Let us assume that the main concept in a query instance is `'Book'` which denotes the `'written book'` and the candidate ontological concept has label `'Book'` which means `'reserve'`. Matching these two concepts using name-based matching only will provide a full matching score however, they have different meanings. Performing structural similarities can figure out that the previous two candidates are not similar since their related concepts are unlikely to be the same. Therefore, it is important to take the structural similarity into consideration.

Generally speaking, measuring structural similarities between two classes belonging to two different ontologies involves matching their super-classes, sub-classes and properties and their domains or ranges (Euzenat & Shvaiko, 2007). In the context of this research, structural similarity between a query instance concept and a candidate ontological class is performed based on calculating similarities between related concepts of the query instance concept (service element) and related concepts of the candidate ontological class. Related concepts of an ontological class are those classes that are linked to this class and its superclasses through object properties. Related concepts of superclasses are taken into account because an ontological class inherits the relations (properties) of its superclasses. Super and subclasses of a candidate ontological class are ignored in this structural

similarity approach because they do not have counterparts in query instances. In other words, the interpretation of an XSD of a WSDL file do not provide neither implicit nor explicit super or sub classing relations between defined data types (Jacek et al., 2007). Related concepts of ontological classes are extracted from candidate ontologies using the OWL API (Euzenat, 2004). The OWL API is a Java API that allows developers to manipulate ontologies represented in the Web Ontology Language (OWL) formalism. Figure 4.9 presents the process flow of the implemented structural matching approach.

The structural matching process starts by obtaining two concepts; one from the set of related concepts of the query instance concept (Set 1) and one from the set of related concepts of the candidate ontological class (Set 2). The similarity between labels of these two concepts is measured using CN-Match. If the resulting score is higher than the CN-Match threshold then the corresponding concept is added to the set of candidate matches. Otherwise, this matching is ignored. When a related concept from Set 1 is matched against all concepts of Set 2, the candidate with the highest score is selected as a match of the given Set 1 element. This match is added to the set of matches. If the set of candidate matches is empty, then there is no match for the given related concept.

The earlier steps are repeated for every concept from Set 1. Once all concepts of Set 1 are taken, the content of the 'Set of matches' SM is checked. If SM is not empty, the final structural similarity score is calculated according to Equation (4.1). Otherwise, the structural similarity score is 0.

$$S_{S=}\frac{\sum_{i=1}^{i=n} Si}{n} \qquad (4.1)$$

Where:

Ss is the final structural similarity score.

n is the number of elements in set 1.

Si is the highest similarity score between each element of set 1 and all elements of set 2.

The final score of a query execution which represents the similarity score between a query instance concept and a candidate ontological class is calculated as a weighted sum of the name-based and structural similarities scores as given in Equation 4.2. In the context of this research, W_n and W_s are given equal values as of 0.5 to provide equal weights to structural and name-based matching. Equal weights are given because both similarity measurements are equally important for the automatic annotation task.

$$S = W_n \times S_n + W_s \times S_s \qquad (4.2)$$

Where:

S is the final score.

Wn is the weight of the name-based matching.

Sn is the name-based matching score between the label of the candidate service element and the label of the selected ontological class.

Ws is the weight of the structural matching.

Ss is the structural similarity score.

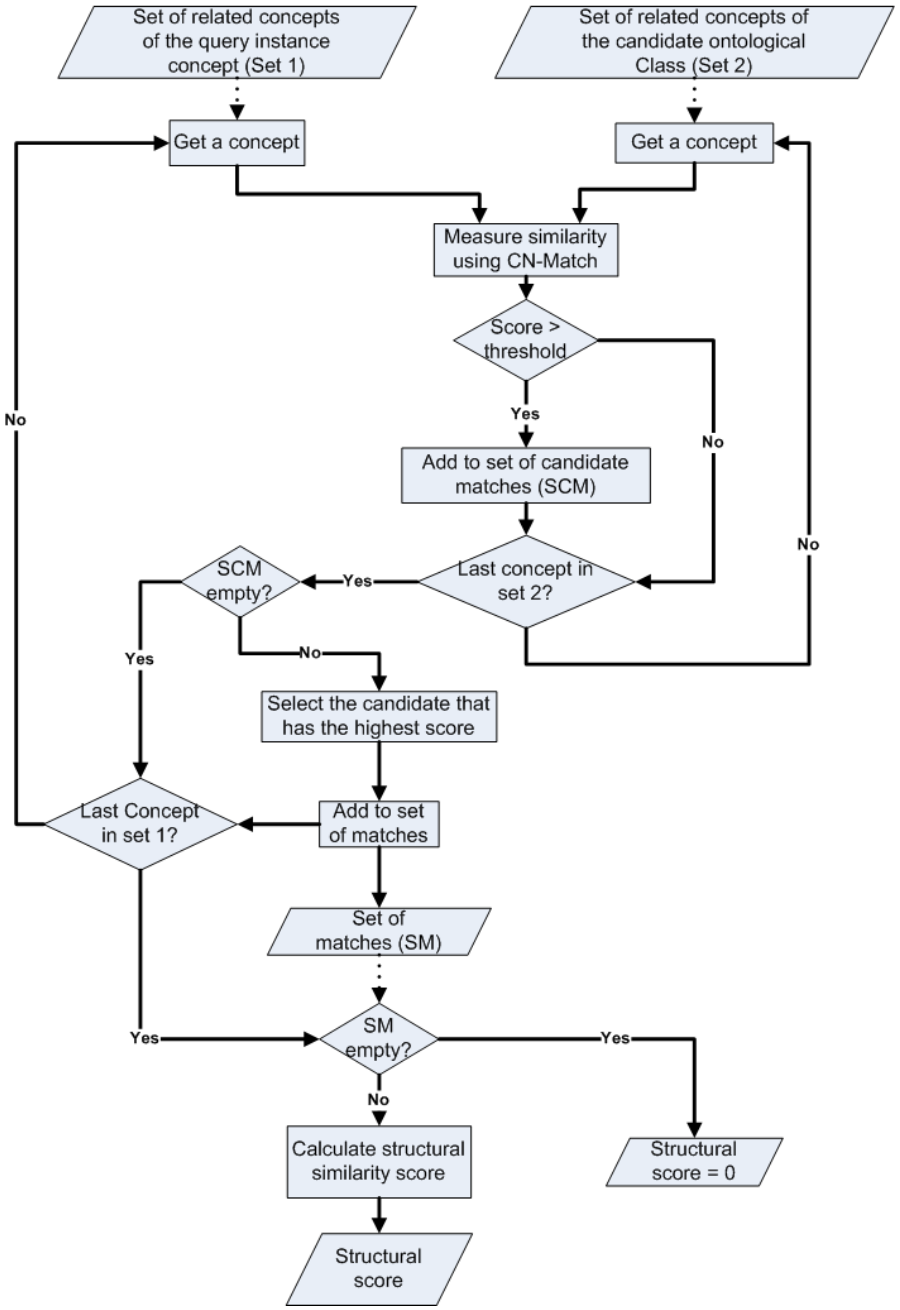

Figure 4.9: The Structural Matching Method

4.9 The SAWSDL Annotation Phase

During the SAWSDL Annotation Phase, the appropriate correspondences that are provided by the results assessment phase are used to annotate the candidate

service concepts. Service annotation is done based on the SAWSDL format (Jacek et al., 2007) which is a W3C recommendation for SWS description (See Subsection 2.4.1 for a discussion on SAWSDL).
The annotation process is performed automatically once an appropriate correspondence is provided for a given query instance. The automatic annotation is conducted by adding a model reference element (URI) to the tag of the given service element. This automatic addition is carried out by parsing the given WSDL file line by line and checking if the given service element exists in this line. If the element exists in the current line, a model reference for the appropriate ontological class is added to the current line; i.e., the tag of the current element. Otherwise, the parser moves to the next line and performs the same checking process.

For simple queries, a model reference is added to the simple element tag only. For complex queries, if the label of the given complex type carries a significant meaning, model references are added to the tags of the complex type and to all of its child elements. This latter annotation is called full annotation. If the complex type does not carry a significant meaning because it falls in one of the three categories provided in Section 4.5, then model references are added to child elements only. The name of the later annotation is partial annotation. It is worth mentioning that full and partial annotations are both considered as valid SAWSDL annotations (Jacek et al., 2007).

When the given service element does not have an appropriate correspondence, this service element is added to the set of non-annotated elements (See Figure 4.6). In this case, either the query instance should be executed against another ontology that exists in the repository if the query concept belongs to a different domain, or the current ontology should be extended with an appropriate correspondence of the given service element. The extension method is fully explained in Section 6.2.

4.10 Summary

This chapter presented the proposed query-based annotation approach: This approach overcomes very important limitations of existing annotation frameworks. These limitations are: (1) The need for building application ontologies to represent service semantics; (2) the inaccuracy of implemented similarity measurement techniques; (3) the Low Percentage Problem; and (4) the annotation of service elements belonging to same domain to different domain ontologies. In eliminating these deficiencies, a set of design requirements were set up. Based on these requirements and knowledge acquired from reviewing previous approaches, design strategies were considered to lead the design process.

The proposed approach takes a WSDL file and ontologies as inputs and produces an annotated WSDL file as an output. The approach is composed of five phases which are concept extraction, concepts filtering and query filling, query execution, results assessment and SAWSDL annotation. The concept extraction, query execution and SAWSDL annotation are fully automatic processes while the concept filtering and query filling and results assessment phases are manual ones. The design of the three automatic phases was discussed in detail in this chapter. The concept extraction is performed using text analysis techniques implemented using the GATE tool. The query execution engine utilises name-based and structural matching mechanisms. Name-based matching is performed using CN-Match which is a novel and effective CN matching mechanism. The SAWSDL annotator is designed using text parsing and string look up techniques.

Chapter 5: The Design and Evaluation of CN-Match

5.1 Overview

This chapter presents the design and evaluation of CN-Match. CN-Match measures similarities between labels composed of single terms, binary and triple compounds. The design of CN-Match is based on a set of considerations and rules derived from limitations of previous CN matching approaches and the linguistic structure of CNs. To perform accurate matching, six design cases are identified and adopted in CN-Match design. CN-Match design represents Increment 3 of the design approach which is shaded in Figure 5.1.

This chapter is organised as follows: Section 5.2 discusses the significance of matching Compound Nouns (CNs) in the area of ontology matching. Section 5.3 provides literature about the structure and types of CNs from a linguistic point of view. Section 5.4 discusses previous CN matching approaches and presents their limitations. Section 5.5 provides considerations and rules for the design of CN-Match. Section 5.6 illustrates the design and implementation of CN-Match. Section 5.7 presents the evaluation of CN-Match to ensure its applicability and assess its performance. Section 5.8 provides a discussion derived from the evaluation results and Section 5.9 summarises the chapter.

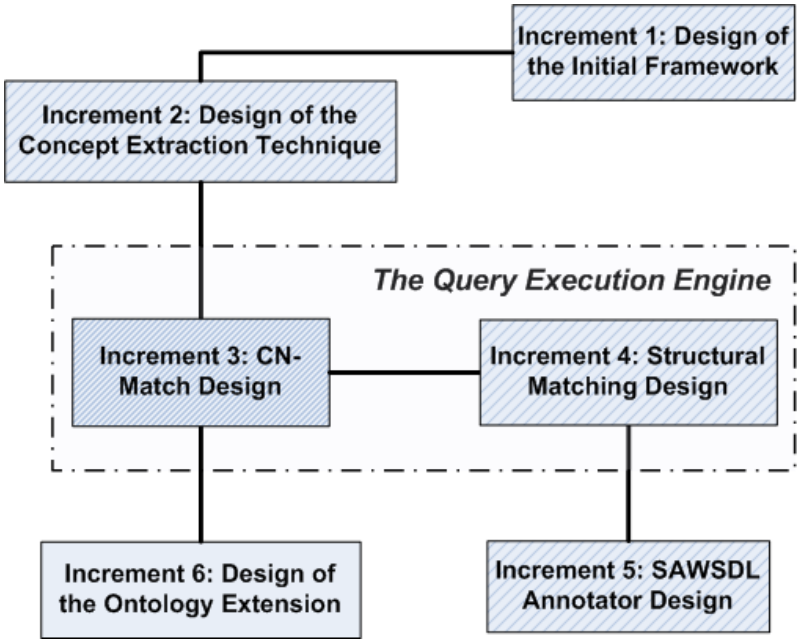

Figure 5.1: The Design Increment Addressed in Chapter 5

5.2 Motivation - The Importance of CN Matching

Ontology matching research has attracted increasing attention in the Semantic Web area. Ontology matching is considered as a very promising solution to the ontology heterogeneity dilemma (See section 2.7.1). Name-based matching is one of the key ontology matching mechanisms: It has been widely used by matching approaches and tools (Euzenat and Valtchev, 2004; Ehrig and Sure, 2005; Castano et al., 2006; Choi et al., 2006; Tagarelli et al., 2009). The weight of this matching technique comes from the fact that similar ontological constructs (classes, properties and individuals) are very likely to have similar names of labels.

Labels of ontological constructs can be composed of multiple words i.e., Compound Nouns (Castano et al., 2006; Nagy et al., 2009; Sorrentino et al., 2009). The reason is that CNs are very commonly used in the English language and constitute a considerable amount of words denoting ontological concepts (Girju et al., 2005). Examples of well known ontologies that contain CNs are the Ka ontology (Horrocks, 2003) and the Portal ontology (Akt Partners, 2010).

5.3 Compound Nouns Structure and Types

A CN is a noun that is made up of two or more nouns (Girju et al., 2005). Plag (2003 pp. 186) categorises CNs into three main categories; endocentric, exocentric and copulative. An endocentric compound is one that has a (modifier, head) structure and its meaning is inherited from the meaning of its head. In linguistic terms, a head refers to the most important unit in complex linguistic structures such as CNs (Plag, 2003 pp. 189). In an endocentric compound, the constituent at the right side of a compound is called a "head" while other constituents are called modifiers or descriptors (Kim and Baldwin, 2005). An example of an endocentric CN can be `'Tennis Player'` where `'Player'` is the head and `'Tennis'` is the modifier. It is well known in linguistic research that the set of things denoted by an Endocentric compound can be seen as a subset of things denoted by its head (Kim and Baldwin, 2005). Exocentric is a type of compound that does not have a (modifier, head) structure and it denotes a characteristic of a person. The head of an exocentric compound is implicit, is located outside a compound and refers to a human being. Examples of exocentric compounds are `'loud mouth'` and `'grey head'` which refer to `'loud mouthed person'` and `'grey headed person'` respectively. Copulative compounds, on the other hand, have two constituents that contribute equally to the overall meaning of CNs. Examples of copulative compounds are, `'singer-songwriter'` and `'doctor-patient'` in `'doctor-patient gap'`.

In this matching approach, we only consider endocentric compounds for the following reasons (Sorrentino et al., 2009): (1) Endocentric compounds are the most common type of compounds in the English language; and (2) exocentric and copulative compounds usually exist in dictionaries such as WordNet, therefore, their meanings can be looked up by a direct use of a thesaurus. Consequently, from this point forward, we will refer to endocentric compounds as compound nouns (CNs).

CNs can also be classified based on the number of their constituents. For example, a binary CN is one that is composed of two constituents and a triple compound is composed of three constituents. An example of a triple CN can be `'Player First Name'`. Nevertheless, the interpretation of triple CNs is not an easy task because triple CNs can be syntactically ambiguous (Lauer, 1995 pp. 34). The reason for ambiguity is that single terms and binary CNs are considered the building blocks of triple compounds and these blocks can be organised in two different ways to form triple CNs (Plag, 2003 pp. 170). To further explain the two possibilities of a triple CN structure; let us assume that we have the triple CN [$C_1C_2C_3$] that is composed of three constituents [C_1], [C_2] and [C_3]. The first possibility happens when the first and second constituents ([C_1] and [C_2] respectively) form a binary compound that is a descriptor of the head [C_3]. The second possibility occurs when the second constituent [C_2] and the head [C_3] together form a compound [C_2C_3] that is described by the first constituent [C_1] which represents the modifier of the CN. An example of the first case is `'Tennis Player Name'` where `'Tennis Player'` is a binary CN that represents the modifier and `'Name'` is the head of the triple CN. An example of the second case is `'Player First Name'` where `'First Name'` is the head and `'Player'` is the modifier of the triple CN. This confusion can cause difficulties for Natural Language Processing (NLP) techniques and CN matching tools that automatically interpret and match CNs (Girju et al., 2005; Kim and Baldwin, 2005).

5.4 Previous Research on CN Matching

Previous research has looked at the problem of similarity measurement between CNs in the context of ontology matching using different methods. These methods normally combine name-based matching with other mechanisms such as structural and extensional when addressing an ontology matching problem. In the following paragraphs, existing CN matching approaches are reviewed and analysed and their limitations in relation to CN similarity measurement are provided.

Sorrentino et al. (2009) propose a method for semi-automatically normalising labels of schema elements that contain CNs and abbreviations. The purpose of this normalisation is to maximise the number of labels that can be compared during a further matching process. The normalisation process of CNs is based on disambiguating their meanings by creating new WordNet entries for those CNs that do not already exist in WordNet. However, this approach requires an initial manual process to associate the right relationship to the pairs of super-concepts (beginners) of CN constituents. This type of relation is used to create a WordNet gloss for new entries. Furthermore, a human user has to select the right relationships among a set of given relations between new entries and existing ones in order to fit new entries into the WordNet hierarchy. This approach could be useful for applications that do not require full automation of the CN matching process because it is based on the idea of adding new CNs to WordNet. In addition, CNs that are composed of more than two constituents are not considered in this approach because the WordNet lexicon can accommodate single terms and binary CNs only.

DSSim (Nagy et al., 2009) is another ontology matching algorithm that takes into account similarities between labels denoted by CNs. Similarity computation between any two compounds is based on similarities between the semantic relations that hold between constituents of each CN candidate. However, the process of detecting semantic relations is based on manually created classification rules. These rules classify a relation between constituents of any given binary CN into one of a set of pre-defined semantic relations that best describes the meaning of the given compound. The creation of the classification rules is based on the use of comments associated with labels of given compounds (definitions of compounds) (Nagy et al., 2009). This approach, however, deals with binary compounds only and requires human intervention to define the set of relations based on given comments that are not always available.

Su and Gulla (2004) propose an approach for improving semi-automatic matching of ontologies. The matching process is composed of two phases; semantic

enrichment and similarity calculation. Semantic enrichment of underling ontologies is performed by adding instance information to ontologies. Instance information is taken from documents accompanying concepts of ontologies. The similarity calculation phase uses linguistic and structural similarity and takes into account binary CNs of labels. A similarity score between a CN and another word (whether a CN or a single term) is the average of similarity scores of each constituent and the other word. This method of similarity calculation can yield imprecise matching scores when matching a single term against a binary CN because head contribution to a CN meaning is more than that of the modifier as it is well known among the linguistic community (Kim and Baldwin, 2005).

H-Match (Castano et al., 2006) is an ontology matching tool designed to match ontologies in open networked systems. This approach is based on creating a thesaurus that exploits WordNet linguistic structures. CNs that do not exist in WordNet are added to the constructed thesaurus using a set of "offline extension steps". These steps are: (1) Entries are defined for each single term, CN and constituents of a CN; (2) terminological relations that hold between entries are defined using WordNet linguistic relations and a set of rules that are exploited from compounds structure. These rules are "broader term" that is defined between a head of a compound and a compound, and "related terms" that is defined between a modifier and the compound itself. H-Match, however, necessitates the addition of CNs that do not have entry in WordNet to the constructed thesaurus prior to similarity calculation between these CNs and other single terms or CNs that already exist at WordNet. This necessity could affect applications that require full automation of CN matching because creating new entries requires some degree of human involvement to extend the constructed thesaurus with new entries.

Based on the preceding analysis of previous research on measuring similarities between CNs for ontology matching, one can conclude that earlier approaches have some or all of the following limitations.

1. They can match binary CNs against other binary CNs and single terms only.

2. The similarity calculation between a pair of binary CNs in (Sorrentino et al., 2009; Castano et al., 2006) is based on adding CNs that do not have entries in WordNet to the thesaurus prior to matching. This could be useful for future processing as linguistic similarity scores are saved and can be retrieved easily. This additional process, however, can have some limitations in that: (a) It may require offline processing and human intervention and thus cannot be applied for settings that require high automation; and (b) CN production is an active process and thus new CNs need always to be added.

3. Similarity calculation between a pair of binary CNs in DSSim is based on the similarity between the semantic relation that holds between constituents of the first CN and this of the second CN. In line with (Downing, 1977; Finin, 1980; Lapata, 2002), we argue that the number of possible types of semantic relations between constituents of binary CNs is infinite. Therefore it is very hard to obtain a comprehensive set of predefined relations that could hold between constituents of any CN. Subsequently, it is very hard to achieve automatic matching that takes into account similarities between relations holding between constituents of candidate CNs.

4. The approach of Su and Gulla (2004) computes similarities between two binary CNs or a binary CN and a single term based on similarities between constituents (heads and modifiers). However, their approach does not take the linguistic structure of CNs into consideration. The reason is that their similarity calculation mechanism between a binary CN and a single term is based on finding the average of similarity scores between each constituent of the CN and the single term. As discussed earlier, the head contribution to the whole meaning of a CN is more than that of the modifier.

5.5 Considerations and Rules for the Design of CN-Match

The analysis of previous research on CN similarity measurement shows some limitations that require attention. To overcome these limitations, we adopt the

following set of considerations (requirements) when designing CN-Match.

R1. CN-Match should be able to measure similarities between single terms, binary CNs and triple CNs.

R2. CN-Match should perform automatic name-based matching in order to facilitate full automation of query execution. Retrospectively, the query execution phase is fully automatic and thus it must be performed with no human intervention (See Sections 4.6 and 4.8). Consequently, unlike other approaches that require the addition of CNs that do not exist in WordNet to a thesaurus prior to similarity calculation, CN-Match should perform automatic and dynamic similarity calculation for cases that involve CNs which do not have entry in WordNet.

R3. Similarity calculation in CN-Match is based on measuring similarities between constituents of CNs with respect to their linguistic structures. Subsequently, the overall matching score is a weighted sum of individual similarities between pairs of constituents. Similarities of internal relations between constituents of CNs are not taken into account for two reasons. First, the set of possible relations between constituents of binary CNs is infinite and thus similarities between these relations cannot always be detected (Finin, 1980). Measuring these similarities becomes even harder when considering triple CNs because each triple CN contains two relations, one inside the binary CN and one between the binary CN and the single term. Second, similarities between any two CNs can be calculated based on similarities between their constituents (Finin, 1980; Lauer, 1995; Plag, 2003 pp. 189).

In order to perform effective similarity calculation between CNs, the linguistic structure of CNs must be taken into consideration during the calculation process (Plag, 2003 pp. 189). Therefore, we set up the following rules that are derived from literature on CN linguistic structure (see Section 5.3) to restrict and lead the similarity calculation process performed by CN-Match.

> ***Rule 1****:* The meaning of a CN is mostly inherited from the meaning of its head (Kim and Baldwin, 2005). This is because the set of things

denoted by a CN is considered as a subset of things denoted by its head. An example is the `'Book Price'` denoting a `'Price'` concept that is specific for books only.

Rule 2: Similarity measurement between any two CNs can be successfully derived from similarities between their constituents (modifiers and heads) (Lauer, 1995; Finin, 1980; Plag, 2003 pp. 189). Simply put, matching a CN against another CN involves matching the modifier of the first CN against the modifier of the second CN and the head of the first CN against the head of the second CN.

Rule 3: A triple CN can always be decomposed into a Binary CN and a single term (Plag, 2003 pp. 170). Either the head or the modifier of a triple CN can be a binary CN while the other will be the single term.

Since the CN-Match similarity measurement process involves measuring similarities between candidates that can differ in relation to the number of constituents, it is necessary to analyse all the possible matching cases in terms of the number of constituents that any pair of candidates may have (See Table 5.1 for a brief explanation of cases). The reason is that taking into account all the cases of matching a single term, a binary CN or a triple CN against other single terms, binary CNs, or triple CNs will result in different cases where each case requires a special processing and different weights.

First Candidate		**Second Candidate**		
		Single Term	**Binary CN**	**Triple CN**
	Single Term	Case One	Case Two	Case Four
	Binary CN	Case Two	Case Three	Case Five
	Triple CN	Case Four	Case Five	Case Six

Table 5.1: CN Matching Cases

In order to show the need for having different cases within CN-Match processing, let us discuss how the process of matching a triple CN against another triple CN is different from the process of matching a triple CN against a single term. For this example, the triple CN `'Tennis Player Name'` is matched against the triple CN `'Player First Name'` and the single term `'Name'`.

When matching `'Tennis Player Name'` against `'Player First Name'`, `'Tennis Player Name'` is broken down into the binary CN `'Tennis Player'` which represents the modifier and the single term `'Name'` which represents the head (Rule 3). Similarly, `'Player First Name'` is decomposed into the single term `'Player'` which represents the modifier and the binary CN `'First Name'` which represents the head. Based on Rule 2, the binary CN `'Tennis Player'` is matched against the single term `'Player'` because they both represent the modifiers of the first and second triple CNs, respectively. Moreover, the single term `'Name'` is matched against the binary CN `'First Name'` because they both represent the heads of the first and second triple CNs, respectively. The overall score is a weighted sum of similarities of modifiers and heads. However, the method used to derive weights is explained later in Section 5.6.2.

Unlike the former case, matching the CN `'Tennis Player Name'` against the single term `'Name'` involves matching the head of the triple CN which is `'Name'` against the single term `'Name'` only (see Rule 1). Therefore, no similarity between modifiers is involved in this calculation.

5.6 The Design and Implementation of CN-Match

In order to prove the applicability of the proposed similarity measurement approach and make it usable by applications, CN-Match has been implemented using the Java Programming Language version 1.6.0. A set of techniques has been

utilised to enable the desired similarity measurement for the six cases of similarity calculation. The adopted techniques, the six defined cases and the process flow of CN-Match are discussed in the following subsections.

5.6.1 Similarity Measurement Techniques Used to Implement CN-Match

CN-Match involves matching of single terms and CNs using string-based and linguistic-based similarities. For linguistic-based similarities, synonym similarity and path length-based linguistic similarity of WordNet 2.1 are implemented. These techniques are deemed useful for matching labels in the context of ontology matching and have been utilised by different matching tools and frameworks (Choi et al., 2006; Ehrig and Sure, 2005; Euzenat and Shvaiko, 2007 pp. 78). These techniques are described briefly as follows.

1. ***String-based similarity*:** In this similarity measurement, a word is considered as a sequence of letters. Therefore, similarity is calculated based on existence of the same characters at specific positions of the two candidates (Euzenat and Shvaiko, 2007 pp. 76). For this similarity, we use Levenshtein Distance which is a method proposed by (Levenshtein, 1965) to compute the distance between two strings. The distance is calculated based on the number of insertions, deletions and substitutions of letters required to transform one string into another. The higher the distance is, the more different the two strings are. In implementing Levenshtein distance, Gilleland (2009) produced a method to calculate string-based similarity which can provide similarity scores between two given strings in the range [0 1] where 0 means no similarity and 1 means identical. However, before calculating similarity between any two candidates using string matching, it is normally necessary to stem the two candidates using a stemmer. A stemmer such as Porter Stemmer (Porter, 2006) is used to remove suffixes from words and thus transfer these words into their origin. For example, a stemmer will convert ‘Computers’ into ‘Computer’.
2. ***Synonym Similarity*:** This similarity measurement utilises WordNet synsets. It is performed based on the following consideration (Euzenat and Shvaiko,

2007 pp. 89). Any two candidates are synonyms if one candidate exists in the synset of the other. However, the output of synonym similarity is either 1 when the two candidates are synonyms or 0 otherwise. Synonym similarity is implemented in CN-Match using WordNet 2.1 Thesaurus of the MorphAdorner API (Burns, 2006).

3. ***Path length-based linguistic similarity*:** This measurement exploits the lexical relations of the WordNet hierarchal structure by using a path length-based method (Budanitsky and Hirst, 2006; Lin and Sandkuhl, 2008). In path length measures, the shorter the path between any two nodes, the more similar the two concepts represented by these two nodes are. An example of a path-based length method is the Wu and Palmer method (Wu and Palmer, 1994). The Wu-Palmer method calculates similarity between two concepts in a graph by finding the path length between the least common subsumer (LCS) of the nodes of these two concepts and the root node. The value of the resulting path length is then divided by the sum of the path length from the node of each individual concept to the root element. Wu-Palmer similarity for WordNet is implemented in CN-Match using the "Wu-Palmer Similarity" method of the 'JWNLDistance' class of the Alignment API 3.6 (Euzenat, 2004). This later similarity measurement provides similarity scores in the range [0 1].

In the context of this research, the two linguistic-based similarity techniques are merged in CN-Match into a single algorithm called 'Linguistic Similarity'. The reason for this merge is to improve the design of CN-Match by creating one linguistic similarity class that can easily be used by other classes. The new linguistic similarity process is demonstrated in Figure 5.2.

It is worth mentioning that the 'Linguistic Similarity' algorithm can measure similarities between two candidates that both have entry in WordNet. These candidates can be single terms and binary CNs only. Similarity measurements involving CNs that do not have entry in WordNet are performed by a set of heuristics that are implemented by the six similarity measurement cases of CN-Match.

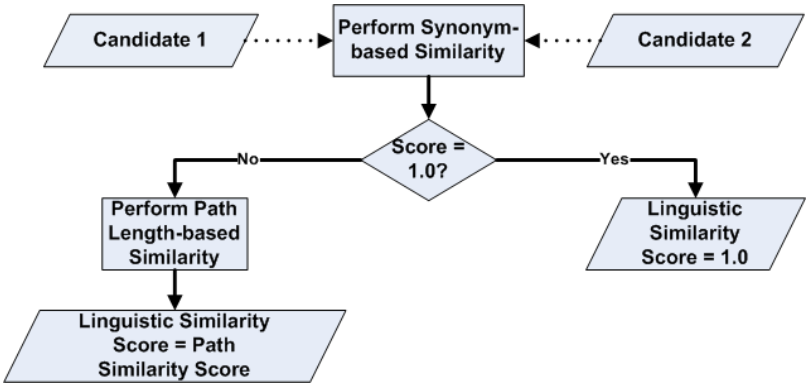

Figure 5.2: Linguistic Similarity Process Flow

5.6.2 The Six Cases of Matching Single Terms, Binary and Triple CNs

CN-Match performs similarity calculation between candidates that can be composed of numbers of constituents ranging from one to three. As explained earlier in Section 5.5, it is necessary to distinguish between the different cases and explain each one individually. This subsection provides a detailed explanation of these cases. In explaining the cases, we use symbols such as [$C_{11}C_{12}$] and [$C_{21}C_{22}C_{23}$]. These symbols are used to generalise and simplify the explanation of cases. The first index in a constituent of a CN refers to the order of this CN. The second index refers to the order of the constituent in a CN. For example, the leftmost 1 in [C_{11}] indicates that the binary CN [$C_{11}C_{12}$] is the first (source) candidate. While the rightmost 1 in [C_{11}] indicates that [C_{11}] is the first constituent of the binary CN [$C_{11}C_{12}$]. Similarly, 2 in [C_{12}] indicates that [C_{12}] is the second constituent of the binary CN [$C_{11}C_{12}$]. For example, if `'TennisPlayer'` is the first candidate in a matching process then `'Tennis'` would be [C_{11}] and 'Player' would be [C_{12}].

Similarity scores of Cases 2 to 6 are calculated as a weighted sum of individual similarity scores. The similarity scores are generally given by the following equation:

$$S = W_1 \times S_1 + W_2 \times S_2 \quad (5.1)$$

Where:

S is the overall score.

W_1 is the weight of the modifiers' similarity.

W_2 is the weight of the heads' similarity.

S_1 is the score of the modifiers' similarity.

S_2 is the score of the heads' similarity.

When measuring similarity between a single term and a CN as in Cases 2 and 4, similarity between the head of the candidate CN and the single term is taken into account only and no similarity between modifiers is involved. Subsequently, the overall score is given by the following equation:

$$S = W_3 \times S_3 \quad (5.2)$$

Where:

S is the overall score.

W_3 is the weight of similarity between the CN head and the single term.

S_3 is the score of similarity between the CN head and the single term.

The values of the preceding weights have been assigned based on linguistic structure of CNs and verified by conducting two experiments. First, initial values of W_1 and W_2 were assigned based on Rule 2 which declares that similarity between any two CNs can be derived from the similarities between their modifiers and between their heads. Similarly, an initial value of W_3 which is the weight of similarity between a single term and a CN is set up by taking into consideration Rule 1 which indicates that the meaning of a CN is mostly inherited from the meaning of its head.

These initial values were then evaluated by conducting two experiments that involve matching ontologies describing the book selling and the academic domains. The used ontologies contain single terms and CNs. During these

experiments, the initial values were changed and the similarity scores were evaluated in order to find the most appropriate values of weights. The two experiments lead to 0.5, 0.5 and 0.96 as the most appropriate values for W_1, W_2 and W_3 respectively.

Case 1

This case measures similarities between any two single terms [C_1] and [C_2] and is called 'Single Terms Matching'. Similarity between [C_1] and [C_2] is first measured using string-based similarity. If the resulting score is 1.0, then the final score is 1.0 which means that the terms are identical. Otherwise, linguistic similarity is used to measure similarity between [C_1] and [C_2] and the final score is the linguistic similarity score. An example of this case can be the matching of `'Price'` against `'Cost'`.

However, 'Single Terms Matching' and 'Linguistic Similarity' are the basic techniques that are utilised by the five subsequent cases to perform the required similarity calculations. Therefore, the following two assumptions hold for the Cases 2 to 6.

- 'Single Terms Matching' is the method that is always used to measure similarities between any two single terms. Consequently, whenever calculating a similarity between a pair of single terms the 'Single Terms Matching' method is implemented.
- 'Linguistic Similarity', on the other hand, is the method used for similarity measurements that involve binary CNs that exist in WordNet. Therefore, from this point forward, any similarity measurement that involves two binary CNs that both have entry in WordNet or a binary CN and a single term that also both have WordNet entry is performed using 'Linguistic Similarity'.

Case 2

The second case performs matching of a single term [C_{11}] against a binary CN [$C_{21}C_{22}$] and has two possibilities which are highlighted in Table 5.2 and explained

as follow.

P1. If both [C_{11}] and [$C_{21}C_{22}$] have WordNet entries, then similarity between them is measured using 'Linguistic Similarity'. An example of this case is comparing `'Person'` as being [C_{11}] against `'Sales Representative'` as being [$C_{21}C_{22}$] which both are available in the WordNet.

P2. But if [$C_{21}C_{22}$] or [C_{11}] does not have entry, similarity between [C_{11}] and the head of [$C_{21}C_{22}$] which is [C_{22}] is measured using 'Single Terms Matching'. An example of this occasion is measuring similarity of `'Book Price'` against `'Cost'`. The `'price'` is the head and it is matched against the single term `'Cost'`.

Possibility	Explanation
P1	C_{11} against C_{21} C_{22}
P2	C_{11} against C_{22}

Table 5.2: Possibilities of Case 2

Case 3

The third case examines the matching between two binary CNs [$C_{11}C_{12}$] and [$C_{21}C_{22}$]. An example of these binary CNs could be `'Book Price'` and `'Book Cost'`. The following possibilities can be distinguished.

P1. If both CNs have WordNet entries, then similarity between them is measured.

P2. If only one CN has entry and the head of the other CN has entry as well, similarity between the CN that has entry and the head of the other CN is measured.

P3. If both CNs do not have WordNet entry, then similarities between the heads of the first and second CNs and the modifiers of the first and second CNs are measured and the overall score is a weighted sum of the two resulting scores.

Table 5.3 summarises the different possibilities of Case 3.

Possibility	Explanation
P1	$C_{11}C_{12}$ against $C_{21}C_{22}$
P2	C_{12} against $C_{21}C_{22}$
P3	$C_{11}C_{12}$ against C_{22}
P4	C_{11} against C_{21} & C_{12} against C_{22}

Table 5.3: Possibilities of Case 3

Case 4

The current case looks at measuring similarities between a single term [C_{11}] and a triple CN [$C_{21}C_{22}C_{23}$]. An example is matching `'Name'` against `'Person First Name'`. Two possibilities are identified and shown in Table 5.4.

P1. If the binary CN [$C_{22}C_{23}$] and the single term [C_{11}] both have WordNet entries, then [$C_{22}C_{23}$] is considered as the head of the triple CN and thus similarity between [C_{11}] and [$C_{22}C_{23}$] is measured.

P2. Otherwise, similarity between [C_{11}] and [C_{23}] is measured because [C_{23}] is considered as the head of the triple CN. In this case [$C_{21}C_{22}$] is the modifier of the triple CN.

Possibility	Explanation
P1	C_{11} against $C_{22}C_{23}$
P2	C_{11} against C_{23}

Table 5.4: Possibilities of Case 4

Case 5

The fifth case examines matching a binary CN [$C_{11}C_{12}$] against a triple CN [$C_{21}C_{22}C_{23}$]. According to Rule 2 (see Section 5.5) the Triple CN [$C_{21}C_{22}C_{23}$] can be broken down into either ([C_{21}] and [$C_{22}C_{23}$]) or ([$C_{21}C_{22}$] and [C_{23}]). An example of this case is when measuring similarity between the triple CN `'Sales Representative Name'` and the binary CN `'Person Name'`. For this example, the triple CN can be decomposed into a binary CN `'Sales Representative'` which represents the modifier and a single term `'Name'`

which is the head.

In general, this case involves the following four possibilities:

P1. If [$C_{22}C_{23}$] and [$C_{11}C_{12}$] both have entries, then [$C_{22}C_{23}$] represents the head of the triple CN. Subsequently, similarity between [$C_{22}C_{23}$] and [$C_{11}C_{12}$] is measured and score S_1 is obtained.

P2. If [$C_{22}C_{23}$] and [C_{12}] both have entries, then similarities between ([$C_{22}C_{23}$] and [C_{12}] which are heads) and between ([C_{21}] and [C_{11}] which are modifiers) are measured. Score S_2 is obtained as a weighted sum of the two previous similarity measurements.

P3. If [$C_{21}C_{22}$] and [C_{11}] both have entries, then similarities between ([$C_{21}C_{22}$] and [C_{11}] which are modifiers) and between ([C_{23}] and [C_{12}] which are heads) are measured. The overall score S_3 is a weighted sum of the two previous scores.

P4. If none of [$C_{22}C_{23}$], [$C_{21}C_{22}$] and [$C_{11}C_{12}$] have entry, similarities between ([C_{11}] and [C_{22}]) and ([C_{12}] and [C_{23}]) are measured and score S_4 is obtained.

After taking into account all possibilities, S_1, S_2, S_3 and S_4 are compared and the highest one will be the final score. Table 5.5 highlights the different possibilities of Case 5.

Possibility	Explanation
P1	$C_{11}C_{12}$ against $C_{22}C_{23}$
P2	C_{11} against C_{21} & C_{12} against $C_{22}C_{23}$
P3	C_{11} against $C_{21}C_{22}$ & C_{12} against C_{23}
P4	C_{11} against C_{22} & C_{12} against C_{23} Or C_{11} against C_{21} & C_{12} against C_{23}

Table 5.5: Possibilities of Case 5

Case 6

Case 6 looks at measuring similarity between a pair of triple CNs [$C_{11}C_{12}C_{13}$] and [$C_{21}C_{22}C_{23}$]. However, the first triple CN can be broken down into ([C_{11}] and [$C_{12}C_{13}$]) or ([$C_{11}C_{12}$] and [C_{13}]). Similarly, [$C_{21}C_{22}C_{23}$] can be broken down into ([C_{21}] and [$C_{22}C_{23}$]) or ([$C_{21}C_{22}$] and [C_{23}]). For instance, when matching the triple CN `'Sales Representative Name'` against the triple CN `'Person First`

`Name'` the first one is broken down into the binary CN `'Sales Representative'` and the single term `'Name'` as shown in the example of case five. The second triple CN, however, is decomposed into a single term `'Person'`, which is the modifier, and a binary CN `'First Name'` which forms the head of this triple CN.

The following four general possibilities are considered when matching a triple CN against another triple CN.

P1. P1. If both of [C11C12] and [C21C22] have entries, similarities between ([C11C12] and [C21C22] which are modifiers) and between ([C13] and [C23] which are heads) are measured. The overall score S1 is a weighted sum of the results of the two previous measurements. A very similar processing happens when [C12C13] and [C22C23] are heads and [C11] and [C21] are their modifiers. The overall score is S2.

P2. If $[C_{11}C_{12}]$, $[C_{13}]$, $[C_{21}]$ and $[C_{22}C_{23}]$ all have entries, similarities between ($[C_{11}C_{12}]$ and $[C_{21}]$ which are modifiers) and between ($[C_{13}]$ and $[C_{22}C_{23}]$ which are heads) are measured. The overall score S_3 is a weighted sum of the two previous measurements. A very similar processing happens when $[C_{11}]$, $[C_{12}C_{13}]$, $[C_{21}C_{22}]$ and $[C_{23}]$ have entries in WordNet. In this case, heads are $[C_{12}C_{13}]$ and $[C_{23}]$ and modifiers are $[C_{11}]$ and $[C_{21}C_{22}]$. The overall score of the later case is S_4.

P3. If only $[C_{11}C_{12}]$ and $[C_{22}]$ have entries, similarity between $[C_{11}C_{12}]$ and $[C_{22}]$ is measured. In this case, $[C_{21}]$ is considered as a modifier of the binary CN $[C_{21}C_{22}]$ where $[C_{22}]$ is its head. $[C_{21}C_{22}]$ as a whole forms the modifier of the triple CN $[C_{21}C_{22}C_{23}]$. Moreover, $[C_{13}]$ and $[C_{23}]$ are considered as heads of the two triple CNs and similarity between them is measured. The overall score S_5 is a weighted sum of the results of the two previous similarity measurements. A very similar processing happens for cases when only one of $[C_{21}C_{22}]$, $[C_{12}C_{13}]$ or $[C_{22}C_{23}]$ has entry in WordNet where scores S_6, S_7, and S_8 can be obtained.

P4. If none of $[C_{11}C_{12}]$, $[C_{12}C_{13}]$, $[C_{21}C_{22}]$ and $[C_{22}C_{23}]$ have entry, similarities between ($[C_{11}]$ and $[C_{21}]$) and ($[C_{12}]$ and $[C_{22}]$) and ($[C_{13}]$ and $[C_{23}]$) are measured. In this case, no indication can be found to figure out how the two triple CNs can be decomposed. Therefore, the only possible way to calculate similarity is by

measuring similarities between each constituent of the first triple CN against its counterpart in the other triple CN. The overall score S_9 is a weighted sum of the three individual scores.

After taking into account all possibilities, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8 and S_9 are compared and the highest score will be taken as the final score. Table 5.6 summarises all the possibilities of Case 6.

Possibility	Explanation
P1	$C_{11}C_{12}$ against $C_{21}C_{22}$ & C_{13} against C_{23} Or C_{11} against C_{21} & $C_{12}C_{13}$ against $C_{22}C_{23}$
P2	$C_{11}C_{12}$ against C_{21} & C_{13} against $C_{22}C_{23}$ Or C_{11} against $C_{21}C_{22}$ & $C_{12}C_{13}$ against C_{23}
P3	$C_{11}C_{12}$ against C_{22} & C_{13} against C_{23} Or C_{11} against $C_{21}C_{22}$ & C_{13} against C_{23} Or C_{11} against C_{22} & $C_{12}C_{13}$ against C_{23} Or C_{11} against C_{21} & C_{13} against $C_{22}C_{23}$
P4	C_{11} against C_{21} & C_{12} against C_{22} & C_{13} against C_{23}

Table 5.6: Possibilities of Case 6

Table 5.7 presents mappings between each case and the method implementing it.

Case	Method
Case 1	Perform Single Terms Matching
Case 2	Perform Single Term And Binary CN Matching
Case 3	Perform Binary CN Matching
Case 4	Perform Single Term and Triple CN Matching
Case 5	Perform Binary And Triple CN Matching
Case 6	Perform Triple CN Matching

Table 5.7: Mapping Cases to Methods

Figure 5.3 presents the class diagram of CN-Match design in order to show interactions between different classes used to implement the logic of CN-Match.

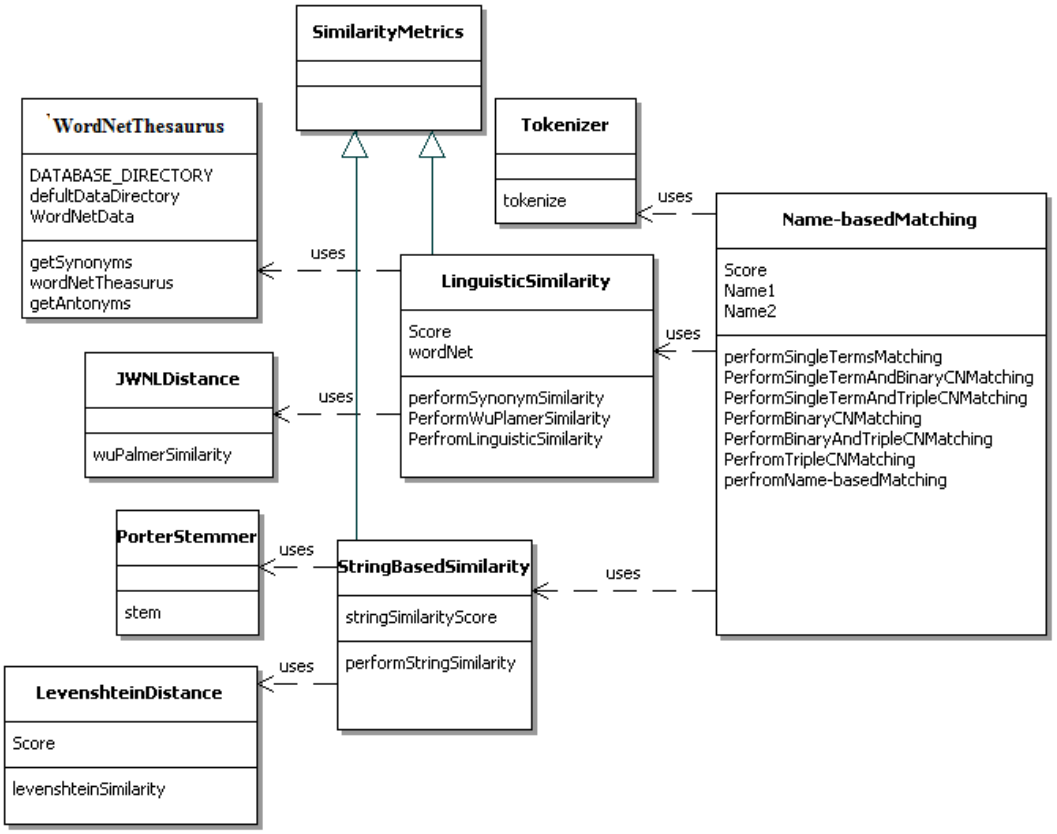

Figure 5.3: CN-Match Class Diagram

5.6.3 Process Flow of CN-Match

To fully explain the design of CN-Match, the process flow of similarity measurement between any two given candidates is explained in this subsection. The process flow starts by tokenising the two candidates and then automatically selecting the appropriate matching case out of the six cases based on the number of constituents in each candidate. After selecting the appropriate case, similarity measurement is calculated based on the logic flow and assigned weights of the selected case. Finally, the process flow of CN-Match is presented in Figure 5.4.

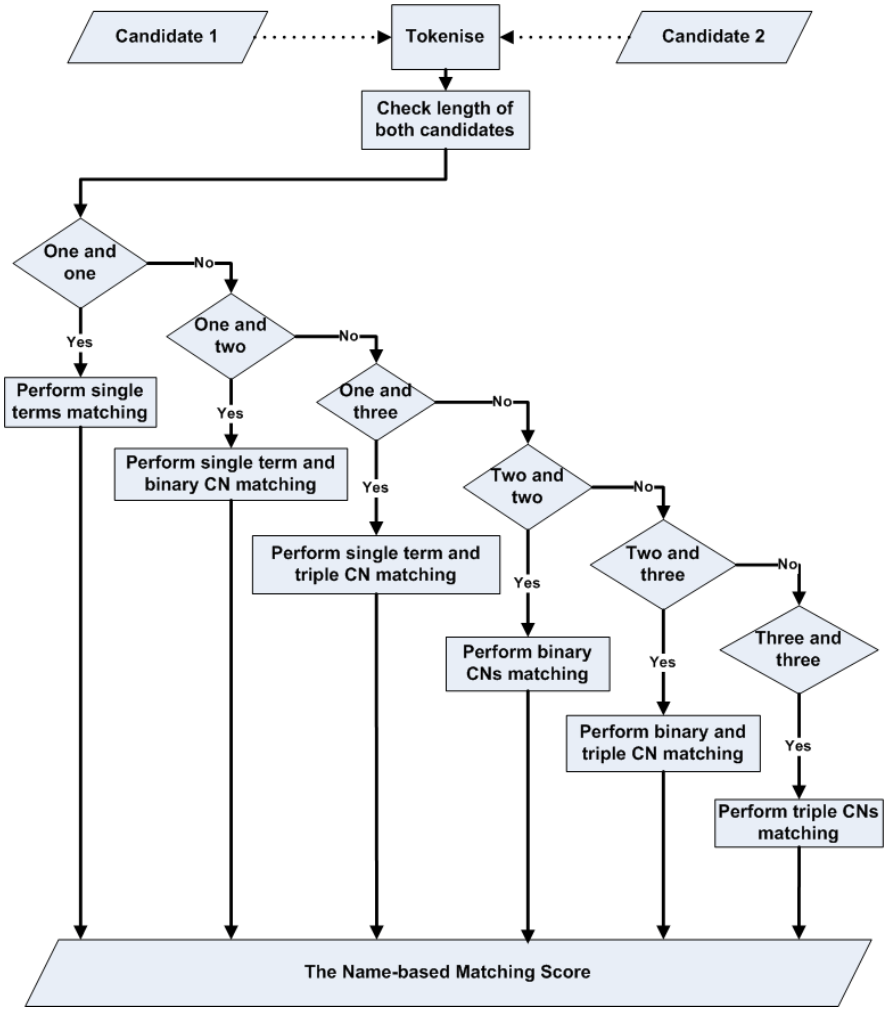

Figure 5.4: The Process Flow of CN-Match

5.7 Evaluation of CN-Match

Having designed and implemented CN-Match, it is very important to evaluate the quality of similarity results it produces. The evaluation is designed to ensure and demonstrate the effectiveness of CN-Match as a matching approach for finding correspondences between labels of ontological constructs.

A comparative evaluation between CN-Match results and other results obtained by ontology matching systems that cater for CN matching is inappropriate: This is because the CN-Match target is different from the target of the approaches reviewed in Section 5.4. These approaches tackle the general ontology matching

problem and combine additional matching techniques with the name-based matching. CN-Match, however, presents an automatic and systematic approach for measuring similarities between ontological labels containing CNs based on CN linguistic structure and using string and linguistic-based similarities. The quality of matching results of CN-Match is evaluated using three ontological test sets. To avoid bias and ensure the applicability of CN-Match, published and recognised test sets are utilised to evaluate CN-Match performance.

The metrics used in this evaluation are precision (P), recall (R) and F-Measure (F). These three measures are borrowed from the Information Retrieval research and have proved to be effective in evaluating performance of retrieval algorithms (Buckland and Gey, 1994). In Information Retrieval research, P indicates the purity (cleanliness) of retrieval and R denotes completeness of retrieval results (Buckland and Gey, 1994). These two measures are then adopted by the ontology matching research to measure the effectiveness of similarly measurement approaches (Euzenat et al., 2009; Giunchiglia et al., 2009). In the ontology matching context, P indicates cleanliness of similarity results and R means completeness of these results (Giunchiglia et al., 2009). However, using these two measures individually could result in misleading evaluation. The reason is that there is a trade-off between P and R. For example, P can be maximised but at the expense of R and vice versa. Hence, a third measure called the F-measure (F) is normally used to combine P and R (Castano et al., 2006). The values of P, R and F fall in the range [0 1]. These three metrics are defined in the following equations.

$$p = \frac{|A \cap M|}{|A|} \qquad (5.3)$$

$$R = \frac{|A \cap M|}{|M|} \qquad (5.4)$$

$$F = \frac{2 \times P \times R}{(P + R)} \qquad (5.5)$$

A is the set of total number of matches automatically found.

M is the set of total number of matches manually found.

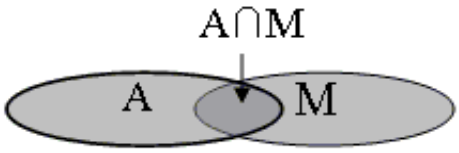

Figure 5.5: The Intersection between Automatic and Manual Matches

Based on the previous definitions of P and R, one can understand that these two metrics require reference results (M) to compare against in order to find out the correct automatic results (Ehrig and Euzenat, 2005). This reference (gold standard) is usually obtained by performing manual matching.

5.7.1 Experiments Design

We have conducted three experiments to evaluate the matching results of CN-Match. For each experiment, the input of the matching process is two sets of labels extracted from two ontologies. In these experiments, labels are extracted from ontologies using SPARQL queries. SPARQL is a query language for RDF graphs (Prud'Hommeaux and Seaborne, 2009). ARQ 2.8.0 API (Hewlett-Packard, 2009) is used as a SPARQL execution engine. An example of a query to extract class labels from an ontology is given in Figure 5.6.

```
SELECT ?x
WHERE
{
?x rdf:type owl:Class.
}
```

Figure 5.6: A SPARQL Query to Extract Labels of Classes

After extracting labels of the two candidate ontologies, each label of the first ontology is compared against all labels of the second ontology in order to find its matches. A match is defined as the one that has a matching score over a defined constant threshold.

A threshold is defined as a cut-off-value that separates correct results from incorrect ones (Castano et al., 2006). Having obtained matching pairs that have scores higher than the threshold, these pairs are compared against the gold standard to check if they exist in this standard. If they exist, then they are added to the list of automatically detected correct matches. Otherwise, these matches are ignored. Next, P, R and F values are calculated using Equations 5.3, 5.4 and 5.5. Figure 5.7 presents the process flow of conducted experiments.

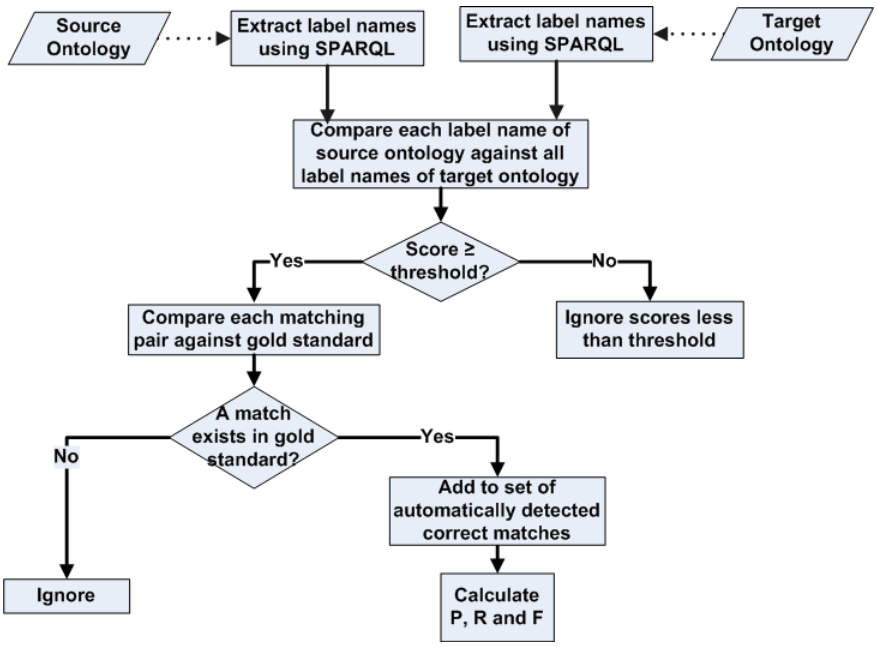

Figure 5.7: Process Flow of Experiments

Nevertheless, the threshold value plays a significant role in the performance of any matching approach. Subsequently, the threshold value should be derived carefully using a set of experiments.

5.7.2 Threshold Derivation

In order to set up an accurate threshold, two experiments were conducted. These two experiments involve matching concepts of: (1) The Yahoo! (Yahoo!, 2010) and the Open Directory Project (The Open Directory Project, 2010) travel categories; and (2) the Portal and Ka ontologies. The travel categories of the Yahoo! and the

Open Directory Project contain concepts describing the travel domain. For the purpose of the first experiment, 40 concepts were considered. The Portal and Ka ontologies contain concepts that describe the knowledge acquisition domain. Each one of these two ontologies contains more than one hundred classes. Performing the preceding two matching tests is deemed appropriate for our experiment because: (1) They contain good ratio of CNs; (2) the two tests have been used before to test other matching techniques (Castano et al., 2006; Su and Gulla, 2004); and (3) these two test cases are different in terms of domain and number of concepts and thus can provide an average threshold that is more generally applicable than a threshold derived from a single experiment.

In each experiment, the threshold value is changed and P, R and F values were measured for each threshold value. In order to calculate P and R, a gold standard is needed. Unfortunately, these two matching cases do not have published gold standards and thus it was necessary to produce them manually. To this aim, six computer science PhD students (three for each experiment) were asked to perform a manual matching between concepts of the first and second sets of each experiment based on their intuitive understanding of the involved domains. They were allowed to use a dictionary when required. A matching pair is counted as a correct match when it is given by at least two out of three students. Otherwise, the matching pair is ignored.

Figure 5.8 presents F values for both of the travel category experiment and the Portal and Ka experiment. The reason for considering the F measure as an indication of good results is because it combines both of the P and R values. The best F value of the travel category experiment is obtained when the threshold is 0.86. The best F value of the Portal and Ka experiment is achieved when the threshold is 0.87. Therefore, the final threshold is taken as 0.865 which is the average of the two previous thresholds.

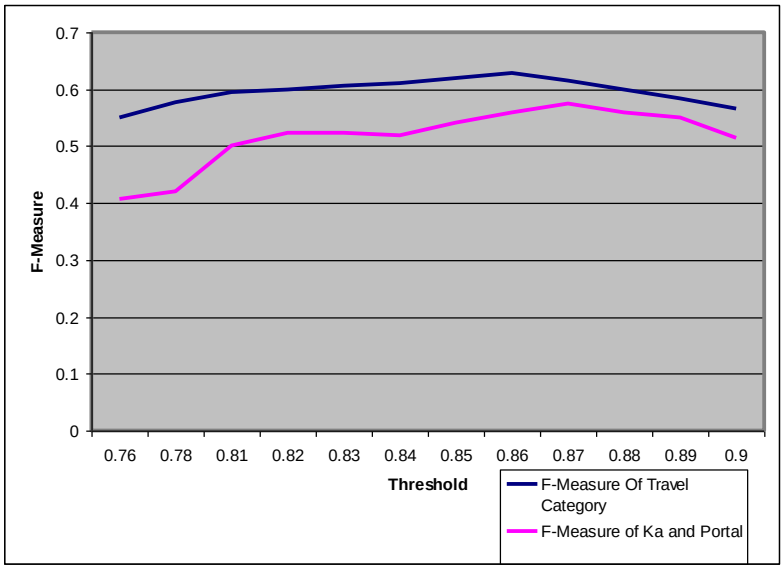

Figure 5.8: Changes of F-Measure against Changes in Threshold

5.7.3 Evaluation Test Sets and Results

Having acquired the threshold, the performance of CN-Match is now evaluated using three published test sets which are the Benchmark of OAEI 2009 (Euzenat et al., 2009), the Russia test set of the FOAM (Ehrig, 2009) and the conference test set of the OAEI 2009 (Euzenat et al., 2009). These test sets cover different domains and contain different levels of CN coverage. These three test sets are chosen for the following reasons. First, they all have published gold standards which make the evaluation of CN-Match more valid and less biased. Second, they all contain a good number of CNs. And third, they describe domains (Bibliography, Country and organisation of Conferences) that are understandable by the general audience.

A. The Benchmark Test Set

The benchmark test set of the OAEI (Euzenat et al., 2009) contains tests that compare ontologies describing the bibliographic domain. This test set is composed of fifty four different tests that are designed to evaluate strengths and weaknesses of different matching systems that implement different techniques. Since the focus of this evaluation is on testing the performance of CN-Match which is a name-based

matching system using recognised and published ontologies, a subset (301 to 304) of the Benchmark test set is used. The reason is that the first fifty tests (101 to 104 and 201 to 266) match the reference ontology (101) against modified versions of this ontology. On the contrary, 301, 302, 303 and 304 tests match the reference ontology against other four different ontologies that are created and used by different organisations. These four ontologies are BibTex/MIT, BibTex/UMBC, Karlsruhe and INRIA. These four ontologies contain 30 classes in average where about 15% of their labels are CNs. Figure 5.9 presents the P, R and F values of these four tests.

Figure 5.9 shows that the worst F value is achieved by the 302 test. The reason is that the 302 gold standard provides more domain specific pairs than the other three tests. Examples of these domain specific correspondences are (Unpublished, Resource), (Collection, Book) and (Part, Publication) just to mention a few. These domain-specific matches cannot be discovered using a general purpose thesaurus such as WordNet.

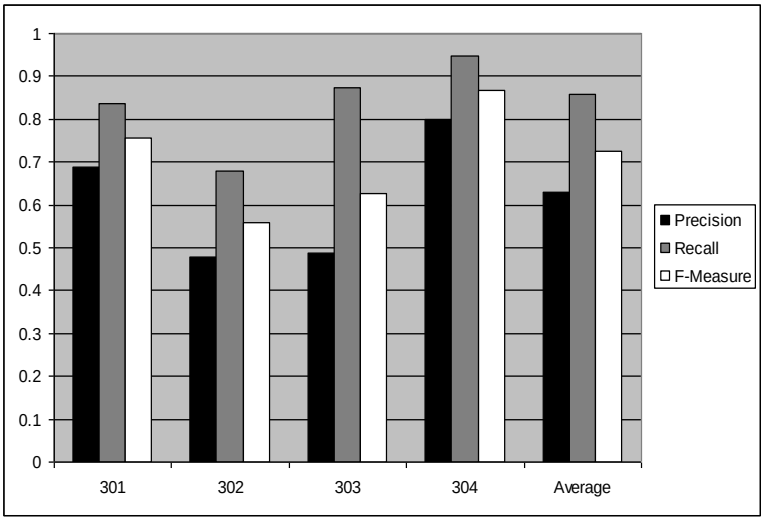

Figure 5.9: P, R and F of the Benchmark Test Set

B. The Russia Test Set

This test set includes three individual tests that involve matching three pairs of ontologies describing the location and cultural aspects of the Russia country (Ehrig,

2009). These six ontologies individually contain an average of 130 classes of which about 30% are CNs. This test set contains much more classes and a higher percentage of CNs in comparison to the Benchmark test set. Figure 5.10 presents the P, R and F values of the three Russia tests. Using Figure 5.10, one can conclude that the worst F value is obtained by the 'Russia1, 2' test. This is because there are domain-specific matches provided by the gold standard that cannot be captured by CN-Match. Examples of these matching pairs are (TravelEvent, RecreationSport), (CultureOffer, PublicSpace) and (PoliticalFact, Overview).

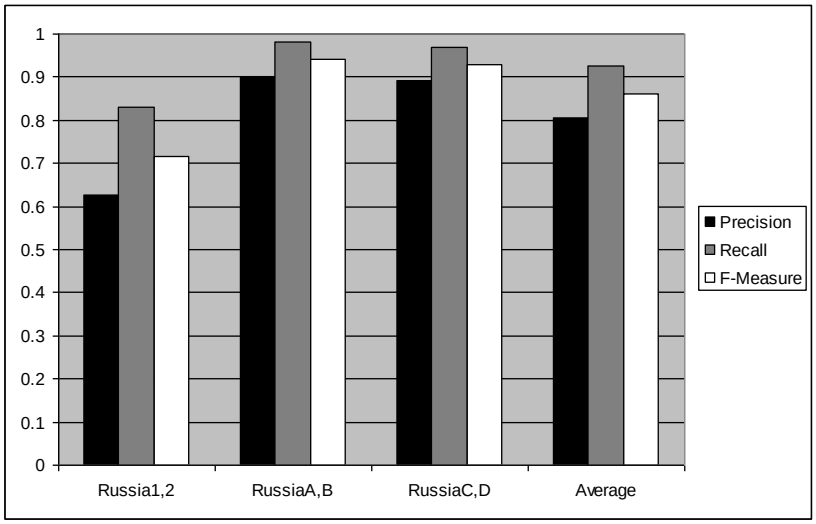

Figure 5.10: P, R and F of the Russia Test Set

C. The Conference Test Set

This test set contains 15 ontologies describing the organisation of conferences (Euzenat et al., 2009). 21 tests have gold standards. Out of these 21 tests, 8 tests that involve ontologies containing the highest number of classes and percentages of CNs have been selected for the purpose of this evaluation. The ontologies of selected tests have a number of classes ranging from 49 (Sigkdd ontology) to 140 (Iasted ontology). Moreover, the average percentage of class labels that are CNs is 65%. This percentage is significantly higher than the previous two test sets. This makes this test set very suitable for the purpose of CN-Match Evaluation. Figure 5.11 shows the P, R and F values of the seven conducted tests.

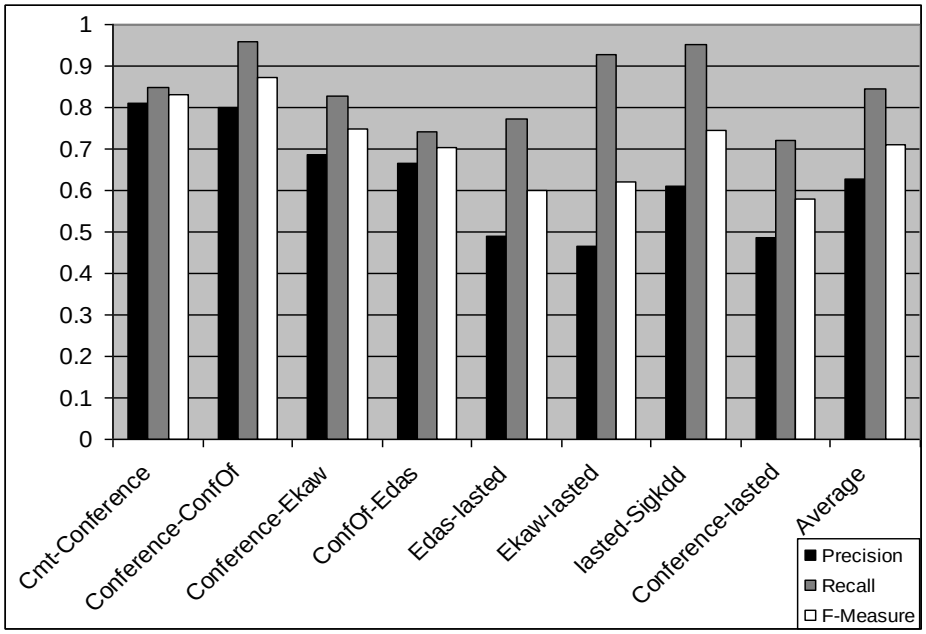

Figure 5.11: P, R and F of the Conference Test Set

Having looked at Figure 5.11, one can tell that Edas-Iasted, Ekaw-Iasted and Conference-Iasted have achieved the lowest F value. The reason is that the gold standards of these tests provide domain-specific concepts relevant to the Iasted ontology that match with concepts from other ontologies. Examples of these matching pairs are (SlideSet, Transparency), (Paper, Submission) and (WelcomeTalk, WelcomeAddress) from the Ekaw-Iasted test and (PassiveConferenceParticipant, Listener), (ActiveConferenceParticipant, Speaker) and (CameraReadyContribution, FinalManuscript) from the Conference-Iasted test.

5.8 Discussion

CN-Match measures similarities between labels containing all three types (endocentric, exocentric and copulative) of CNs. Exocentric and copulative CNs can be matched using linguistic and/or string matching. Using WordNet, meanings of exocentric and copulative CNs can be looked up. Endocentric CNs, which are the most common in the English Language, can be matched using the six pre-defined cases and the matching techniques utilised by CN-Match.

Three test sets have been used to evaluate the performance of CN-Match in terms of P (cleanliness of matching results), R (completeness of results) and F which is the harmonic measure of the previous two metrics. The resulting values of these measures (See Figures 5.9, 5.10 and 5.11) demonstrate the effectiveness of CN-Match. Importantly, high R values (The lowest is 0.68 obtained by Test 302) were obtained indicating that a high percentage of correct matches were retrieved by CN-Match.

Unsurprisingly, different values of the three measures were obtained by the different test sets. These differences are due to the following reasons: (1) Issues of the used datasets and their gold standards; and (2) issues of the techniques implemented by CN-Match. Calculating the values of P and R necessitates the existence of a gold standard, which is used as a baseline against which resulting automatic matches are compared (see Equations 5.3 and 5.4). Therefore, the accuracy and completeness of a gold standard significantly affects the resulting P and F values which are the metrics against which the performance of the underlining matching system is evaluated. Unfortunately, existing gold standards are currently generated by manual means. This leads to incompleteness and sometimes inaccuracy of the produced matches of these standards. For example, the gold standard of the 304 test of the Benchmark dataset misses some important matches such as the pairs (Organisation, University) and (Book, Reference). Moreover, the 303 test provides some imprecise matches such as the pairs (Book, BookTitle) and (Collection, BookTitle).

Since CN-Match is a domain-independent name-based matching tool, it makes use of WordNet which is a general purpose thesaurus. Consequently, the performance of CN-Match is affected by WordNet performance. As WordNet is a general purpose linguistic database, domain-specific matches that have same meanings in a specific domain but different meanings in general or in different domains might not be discovered by WordNet and subsequently by CN-Match. Thus, if a test contains a very high number of domain specific correspondences then the resulting F value

could be lower than that of other tests which have moderate numbers of domain-specific matches.

Based on the conducted evaluation, the lowest F value of 0.57 is obtained by the 302 test of the Benchmark test set. While, the highest F value of 0.94 is obtained by the RussiaA,B test of the Russia test set. The reason for having such a relatively low value for the 302 test is mainly because of the existence of many domain-specific and imprecisely structured CNs. On the contrary, the reason for having the highest F value for the RussiaA,B test is that most of the matching pairs are domain agnostic matches that have same meaning in general as well as in the underling domain. We understand that having domain specific correspondences is sometimes unaffordable, but accounting for these matches requires the existence of domain-specific thesauri that cover many different domains and can be used in automatic settings. Unfortunately, very few domain-specific thesauri can be found. This is because building a domain-specific thesaurus is a very difficult task that requires extensive effort and domain knowledge (Chen et al., 2003). An example of a domain specific thesaurus is the UMLS (Unified Medical Language System) which describes biomedical concepts (Lindberg et al., 1993). Moreover, using domain specific thesauri within an automatic and domain-agnostic matching task may result in additional difficulties since a domain of a concept should be discovered dynamically and a concept could belong to several domains and might have different meanings in different domains (Giunchiglia et al., 2006).

Another important WordNet-related factor that can affect the performance of CN-Match is the quality of matching results produced by WordNet and similarity measurement approaches that make use of WordNet such as the Wu-Palmer similarity measure. WordNet-based linguistic similarity measures have some understandable limitations. Incomplete as well as imprecise linguistic similarity results could sometimes be obtained by WordNet-Based measures. For example, some matching candidates - that are expected to be synonyms or semantically similar - might sometimes not be discovered by a WordNet-based similarity measure. Just to give an example, (Citizen, Inhabitant) are not synonyms in

WordNet 2.1. Moreover, the similarity score of (Woman, Human) is relatively low when using Wu-Palmer similarity of WordNet. These later issues can affect any matching system that uses WordNet and its similarity approaches.

5.9 Summary

This chapter described CN-Match which is the name-based matching approach implemented by the query execution engine of the proposed semi-automatic annotation approach. The chapter began by showing the importance of matching CNs in ontology matching research. Next, a review of the structure and types of CNs from a linguistic perspective was provided. Also, previous approaches that match CNs and limitations of these approaches were given. To setup an appropriate design for the automatic name-based matching approach that can perform precise and effective CN matching, considerations and design rules were then adopted and presented. The design cases and implementation of CN-Match were then followed.

The performance of CN-Match, which is a matching approach capable of measuring similarities between all types of CNs, was evaluated using three published and well recognised test sets. Precision (P), which indicates cleanliness of results, recall (R), which denotes completeness of results, and F-Measure (F), which is a harmonic measure that combines P and R, were the metrics used in the evaluation of CN-Match. The evaluation revealed that CN-Match achieved good results in terms of P, R and F. Importantly; high R values were obtained indicating that almost all correct matches in relation to gold standards were retrieved. The matching results, however, differ from a test to another. These differences were mainly because of two reasons: (1) The nature of the test data, the amount of domain specific matches contained in a specific test and the gold standard of this test; and (2) the basic matching techniques that are implemented by CN-Match. Although the evaluation results were affected by few CN-Match dependent and independent factors, these results were generally very promising and proved the

effectiveness of CN-Match since very good results in terms of P, R and F values were obtained.

.

Chapter 6: The Evaluation of the Annotation Framework

6.1 Overview

This chapter discusses the ontology extension mechanisms and the evaluation of the semi-automatic annotation approach. The ontology extension mechanisms are developed and utilised by the annotation approach to add ontological entities to ontologies used for annotation. The design of the extension mechanisms represents Iteration 6 of the research design as demonstrated in Figure 6.1. The evaluation of the annotation approach is important to assess its performance and assure its appropriateness.

The chapter is organised as follows: Section 6.2 illustrates the design of the extension mechanisms. Section 6.3 presents three illustrative cases that explain the annotation steps and show how the annotation approach works in practice. Section 6.4 illustrates the experimental evaluation methods and metrics and presents the evaluation results. Section 6.5 discusses the evaluation results and highlights the strengths and limitations of the approach. Last, Section 6.6 summarises this chapter.

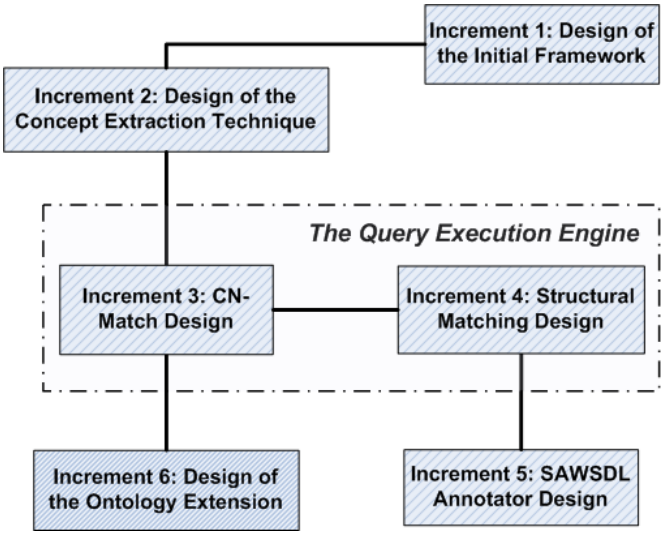

Figure 6.1: The Design Increment Covered in Chapter 6

6.2 Ontology Extension

In the context of this research, ontologies used for annotation should be extended when they do not have correspondences for query concepts (service concepts). Since a limited set containing unique ontologies is used for annotation (See the Scenario in Subsection 3.4.4), ontology extension is a very important activity as it allows us to increase the number of annotated service elements by extending the used ontologies with appropriate correspondences for service elements. In other words, Ontology extension is proposed as a solution to alleviate the Low Percentage Problem (See Section 2.8).

6.2.1 The Method of Extension

Ontology extension is defined as the process of adding new entities to existing ontologies (Liu et al., 2005). In the context of this research, an ontology is extended by adding a class and/or object property. When extending an ontology, a class should not be added to the ontology as an isolated entity. This is because ontological classes normally participate in relations (properties) that form axioms.

Object properties can be seen as links that relate two or more classes together. Subsequently, a newly added class should be appropriately linked to other classes in an ontology.

The proposed query-based annotation approach defines two types of queries called simple and complex queries. Subsequently, we differentiate between two types of extension methods based on the query type. This is because simple and complex queries require different kinds of extensions. Complex queries have a main concept and a set of related concepts that are conceptually linked to the main concept with implicit relations that are derived from the XSD structure of complex types. Therefore, the user already has an idea about the classes that the added class can be linked to. On the other hand, simple queries contain a single concept and thus no clue about related concepts is given by a simple query.

A. The extension method for simple queries:

In order to extend the ontology with a concept that can be a correspondence for a simple query concept, the following steps can be followed:

1. The user finds a concept (or concepts) that can possibly have a conceptual relation with the given query concept.
2. Name-based matching is used to check if such a concept exists in the ontology.
3. If the name-based matching detects a matching class for the provided concept, this class can be used as a basis for extension. In other words, a class that denotes the given query concept and an object property can be added to the ontology. The class that denotes the given query concept and the class found by the matcher are domain and range for the new object property.
4. If the name-based matching does not detect a match, the user can think of another concept or a corresponding class to the given query concept can be added directly to the ontology.

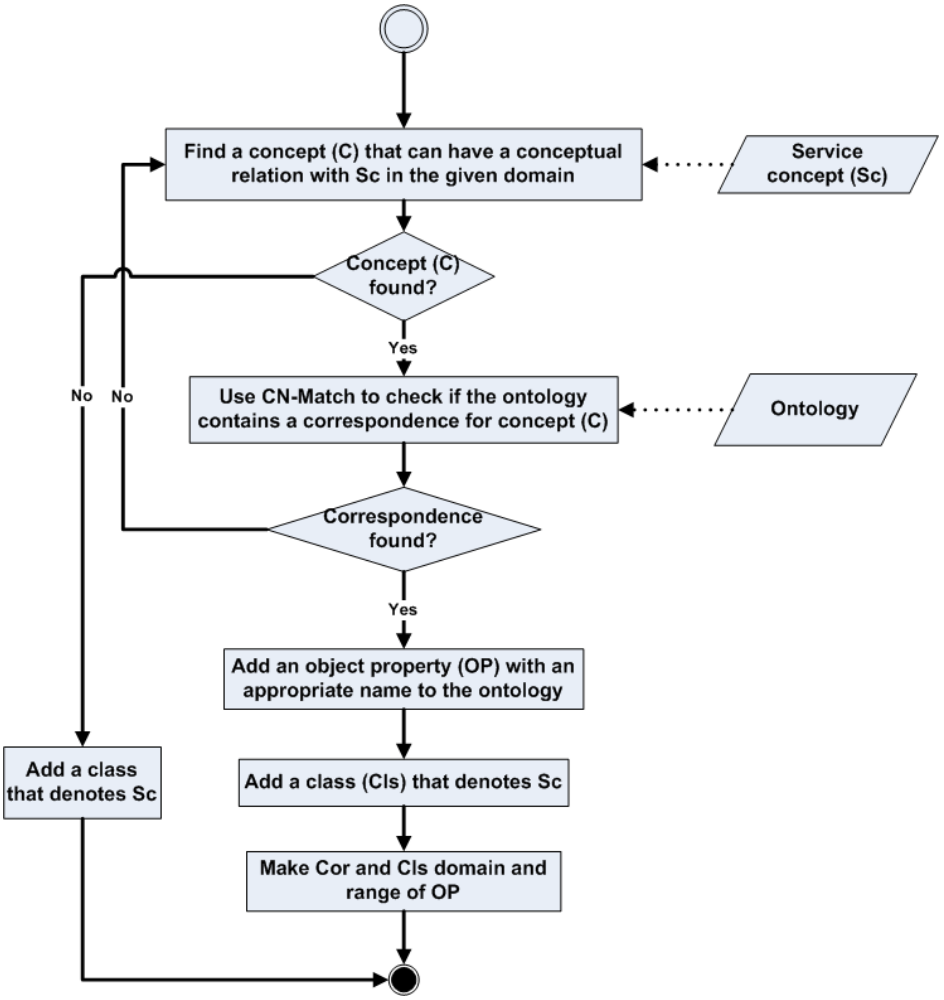

Figure 6.2: Extension for Simple Query Concept

B. The extension method for complex queries:

When a correspondence for the main service concept is missing:

1. Create a class that denotes the given main service concept.
2. Find a correspondence (Cor) of a service related concept.
3. Create an object property (the object property name is given by the user).
4. The new class and the class that is the correspondence of the service related concept are the domain and range of the new object property. Either the new class or the correspondence of the service related concept is the domain of the object property and the other will be the range of this property.

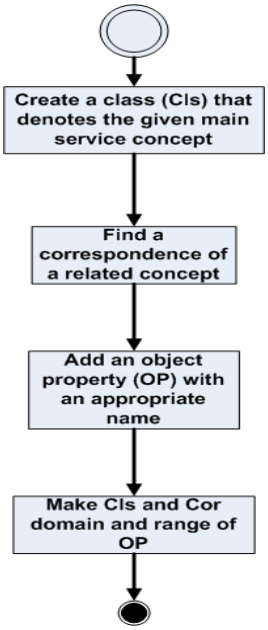

Figure 6.3: Extension for Main Service Concept of Complex Query

When a correspondence for a related service concept is missing:

1. Use name-based matching to check if the given ontology has a correspondence for the given service related concept. This checking process is important since the structural matching calculates similarities between related service concepts and related classes of candidate ontological classes only. The required correspondence of the given related concept could exist in the ontology but might not be linked through an object property to the correspondence of the main service concept. Consequently, it is very important that all classes of the given ontology are matched against missing service related concepts to prevent any potential redundancy that may happen when adding a duplicated correspondence of a service related concept.
2. If the ontology contains such a correspondence, create an object property and give it an appropriate name. The found correspondence and the correspondence of the main service concept are the domain and range of the new object property.
3. But, if the ontology does not contain such a correspondence, create a class that denotes the given service related concept. Create an object property (the object property name is given by the user). The created class and the correspondence

of the main service concept are the domain and range of the new object property.

Figure 6.4 shows the extension process of a related concept.

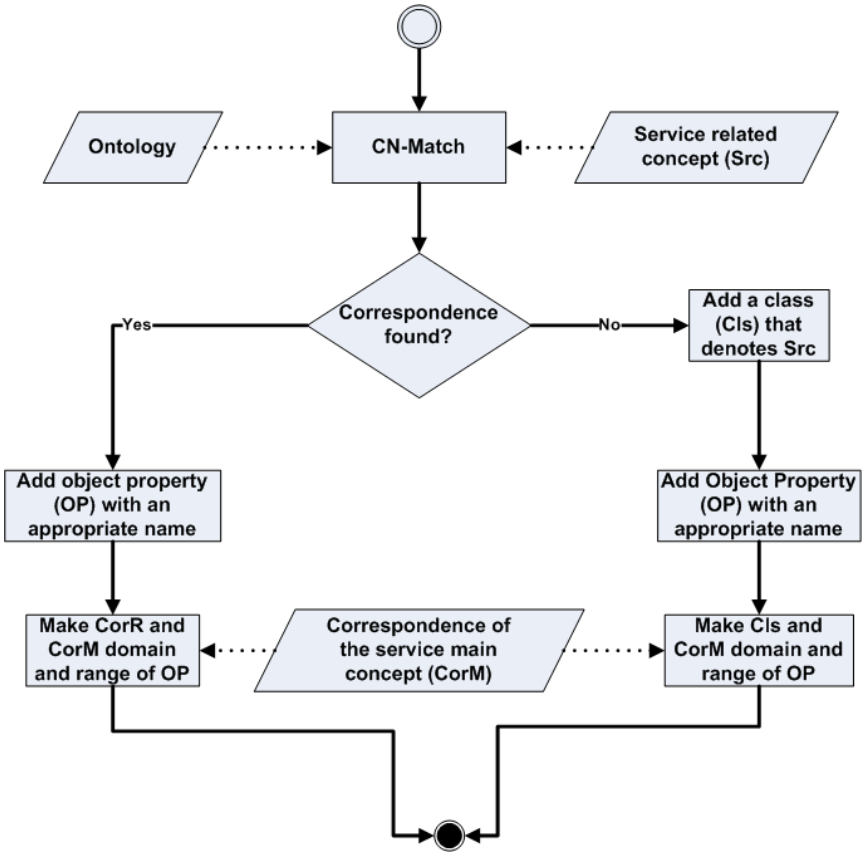

Figure 6.4: Extension for Service Related Concept of Complex Query

6.3 Illustrative Cases

In this section, the annotation framework is utilised to annotate three different Web services that belong to the selected set of twenty five Web services. The reasons for providing these three working examples are: (1) To show the reader how the framework performs in practice; and (2) to perform a black box testing to ensure that the proposed approach is free from faults and it can lead to the desired

aim of this research. All the annotation steps including ontology extension are applied and explained.

6.3.1 Illustrative Case (1): The BookInfoPort Service

The BookInfoPort service allows users to search for book information such as `'authors'`, `'title'`, `'Isbn'` and `'publisher'` using one of these parameters as a search criterion. The steps followed to annotate this service are described in the subsequent paragraphs.

A. Concept Extraction:

The novel concept extraction mechanism (See Section 4.7) is used to automatically extract simple and complex XSD data types of the BookInfoPort service. The output of the concept extraction step is given in Table 6.1.

Service Element	Child elements
Book	Title, Edition, PublicationPlace, Isbn, Availability, Author, Publisher, PublicationDate, ListPrice, DiscountPrice, DiscountPercent
GetBookInfoByISBN	CustomerAccount, CustomerSubAccount, LoginName, LoginPassword, ISBN
BookInfoResponseType	Status, Book, Marc
GetBookInfoByISBNResponse	GetBookInfoByISBNResult

Table 6.1: The Output of the Concept Extraction Process

Please note that the output does not contain simple types because the XSD of the given service does not have simple types.

B. Concepts Filtering and Queries Filling:

Each complex type has a set of related concepts. The complex type represents the Main Service Concept (MSC) in the jargon of this project. Related concepts of a

main service concept are the child elements that are placed to the right of an MSC in the Table (See Table 6.1). For instance, the 'Book' complex type has nine related concepts which are: 'Title', 'Author', 'ISBN', 'Publisher', 'PublicationDate', 'PublicationPlace', 'Edition', 'DiscountPrice' and 'Availability'.

When a label of a complex type is one of the categories that cannot be annotated (see Section 4.5), only child elements of complex types can be semantically described. Annotating child elements of complex types provides partial semantic descriptions for these types (Akkiraju and Sapkota, 2007).

Query for 'Book':

The 'Book' MSC will have a complex query since this MSC is defined as a complex type in the XSD of the service. The constructed query is:

Find a concept in an ontology that:
Clause (1): Target concept name is semantically similar to 'Book'.
Clause (2): Target concept is related by object properties to concepts that are similar to the concepts in the following set: {'Title', 'Author', 'Isbn', 'Publisher', 'PublicationDate', 'PublicationPlace', 'Edition', 'DiscountPrice', 'Availability'}.

Queries for child elements of 'GetBookInfoByISBN':

Since the label of the 'GetBookInfoByISBN' complex type denotes a method, we ignore this complex type and only annotate its related concepts. Therefore, we create simple queries for the following concepts: 'CustomerAccount', 'CustomerSubAccount', 'LoginName', '*LoginPassword*'. No query for 'Isbn' is created since the concept 'Isbn' already exists in the 'Book' query. The created queries are as follows:

Find a concept in an ontology that:
Clause (1): Target concept name is semantically similar to 'CustomerAccount'.

Similar queries to the earlier one are created for 'CustomerSubAccount', 'LoginName' and 'LoginPassword'.

Queries for child elements of 'BookInfoResponseType':

'BookInfoResponseType' is a complex type that does not carry a significant meaning: It belongs to the first category of complex types that can only have partial annotation (See Section 4.5). Therefore, we only annotate child elements of this complex type. The 'BookInfoResponseType' has three child elements which are: 'Status', 'Book' and 'Marc' (MAchine Readable Cataloging). A query has already been created for the 'Book' concept so there is no need to construct a query for it. Simple queries of 'Status' and 'Marc' are shown below.

Find a concept in an ontology that:
Clause (1): Target concept name is semantically similar to 'Status'.

Find a concept in an ontology that:
Clause (1): Target concept name is semantically similar to 'Marc'.

C. Query Execution, Results Assessment, Ontology Extension and SAWSDL Annotation:

The query execution, results assessment, ontology extension and annotation steps are explained together because they are interrelated.

Executing the 'Book' Complex Query:

This query is executed against the 'BookProperty' ontology. The execution output is provided in Figure 6.5.

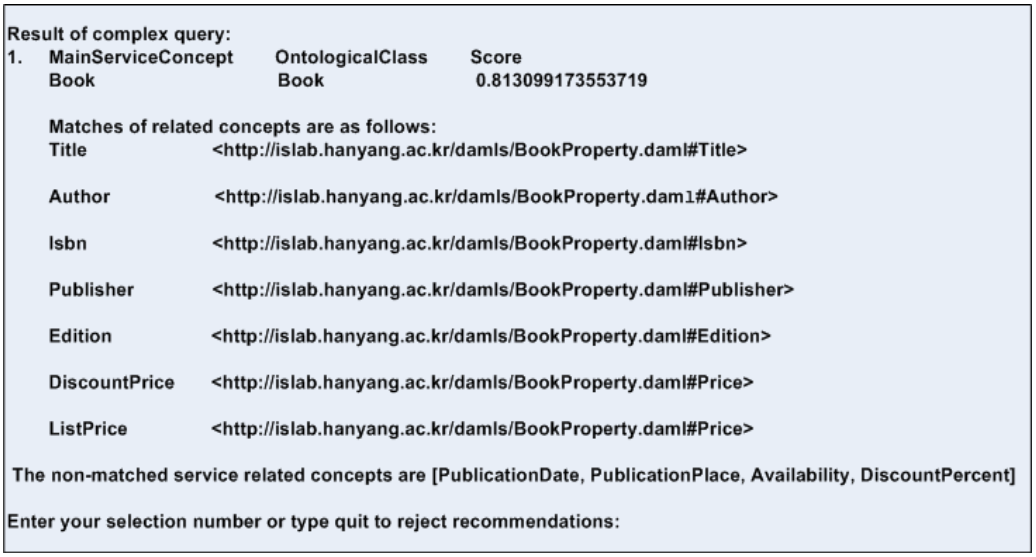

Figure 6.5: 'Book' Query Result

The output presented in Figure 6.5 tells that the 'Book' ontological concept is a correspondence for the 'Book' service concept. In addition, the output provides: (1) The set of service related concepts that have potential correspondences as well as their corresponding ontological concepts; and (2) the set of non-matched service related concepts. The latter set contains the service related concepts that do not have correspondences in the 'BookProperty' ontology: This set provides a foundation for the extension process.

The user can accept the recommendation and type in 1 to indicate their selection. Next, extending the 'BookProperty' ontology is required in order to add correspondences for the elements of the set of non-matched service related concepts. The extension is performed using the developed extension mechanisms (see Section 6.2). To use the extension mechanism, the user has to provide appropriate labels for new classes and object properties. When conducting extension for the 'PublicationDate' related concept, the user provides 'PublicationDate' and 'hasPublicationDate' as inputs for the extension mechanism. As a result of extension, the 'PublicationDate' class and the 'hasPublicationDate' property are added to the ontology. The domain of the property is 'Book' while the range is 'PublicationDate'.

Similarly, three more extension processes are performed to add corresponding classes for 'PublicationPlace', 'Availability' and 'DiscountPercent'. The added classes are shown in Figure 6.6.

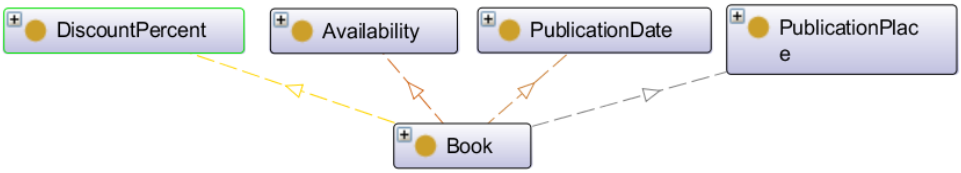

Figure 6.6: Extending the Book Ontology with 'PublicationDate', 'Availability', 'PublicationPlace' and 'DiscountPercent'

After extending the 'BookProperty' ontology with the required entities, the query is executed again and the non-matched service related concepts are annotated to the new classes using the annotation mechanism. The annotated 'Book' type is presented in Figure 6.7.

```
<s:complexType name="Book" sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/
BookProperty.daml#Book"  >
        <s:sequence>
          <s:element name="Title" sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/
BookProperty.daml#Title"/>

          <s:element name="Author" sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/
BookProperty.daml#Author"/>

          <s:element name="Isbn" sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/
BookProperty.daml#Isbn"/>

          <s:element name="Publisher" sawsdl:modelReference = "http://islab.hanyang.ac.kr/
damls/BookProperty.daml#Publisher"/>

          <s:element name="PublicationDate" sawsdl:modelReference = "http://
islab.hanyang.ac.kr/damls/BookProperty.daml#PublicationDate"/>

          <s:element name="PublicationPlace" sawsdl:modelReference = "http://
islab.hanyang.ac.kr/damls/BookProperty.daml#PublicationPlace"/>

          <s:element name="Edition" sawsdl:modelReference = "http://islab.hanyang.ac.kr/
damls/BookProperty.daml#Edition"/>

          <s:element name="DiscountPrice" sawsdl:modelReference = "http://
islab.hanyang.ac.kr/damls/BookProperty.daml#Price"/>

          <s:element name="Availability" sawsdl:modelReference = "http://islab.hanyang.ac.kr/
damls/BookProperty.daml#Availability"/>

          <s:element name="ListPrice" sawsdl:modelReference = "http://islab.hanyang.ac.kr/
damls/BookProperty.daml#Price"/>

          <s:element name="DiscountPercent" sawsdl:modelReference = "http://
islab.hanyang.ac.kr/damls/BookProperty.daml#DiscountPercent"/>
        </s:sequence>
      </s:complexType>
```

Figure 6.7: The Annotated 'Book' Complex Type

Executing the 'CustomerAccount', 'CustomerSubAccount, 'LoginName' and 'LoginPassword' Queries:

First the 'CustomerAccount' query is executed against the 'BookProperty' ontology. The query execution engine fails to find any appropriate correspondence for 'CustomerAccount' in the 'BookProperty' ontology. This is a good result because the queried ontology does not have an appropriate correspondence for the given query concept. Since we could not find a correspondence for the concept under consideration, we will try to execute the query against another ontology existing in the repository. The other ontology is the 'Contact' ontology which describes user information. Again the query execution engine does not find any correspondence for 'CustomerAccount' in the 'Contact' ontology. Subsequently, the decision here is to extend the 'Contact' Ontology with an appropriate concept that can be used to annotate 'CustomerAccount'.

The reason for extending the 'Contact' ontology rather than the 'BookProperty' ontology is that the 'BookProperty' domain ontology covers concepts related to the Book domain and therefore, the 'BookProperty' ontology may not include a description of 'CustomerAccount'. On the other hand, the 'Contact' Ontology provides semantics related to the 'Person' concept and can include a description of 'CustomerAccount'. The 'Contact' ontology is extended by adding a class called 'CustomerAccount' and an object property named 'hasCustomerAccount' that links the 'CustomerAccount' class to the 'Person' class. The proposed extension mechanism for simple queries is used to perform the required extension. Figure 6.8 shows the extended part of the ontology.

Figure 6.8: Extension for 'CustomerAccount'

After performing the extension, we re-execute the 'CustomerAccount' query and get the results shown in Figure 6.9.

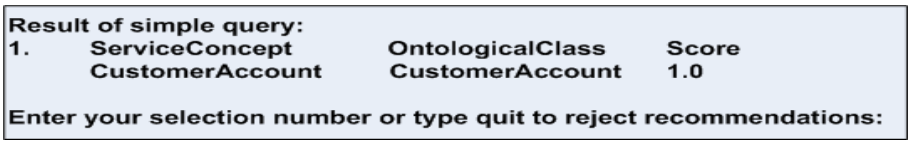

Figure 6.9: 'CustomerAccount' Query Result after Annotation

Once the query is executed, the user can select the appropriate option (in this case, the user can enter 1) and the service element will be annotated automatically using the SAWSDL annotator.

```
<s:element name="CustomerAccount" sawsdl:modelReference = "http://www.w3.org/2000/10/swap/pim/contact#CustomerAccount"/>
```

Figure 6.10: Annotated 'CustomerAccount'

In a similar manner, the 'CustomerSubAccount' query is executed against the 'Contact' Ontology. The query execution engine provides 'CustomerAccount' as a recommended correspondence. This recommendation is rejected since 'CustomerSubAccount' is different from 'CustomerAccount' in the context of the given service. Consequently, the ontology is extended by adding a class called 'CustomerSubAccount' and an object property called 'hasCustomerSubAccount': This property links the 'CustomerSubAccount' class to the 'Person' class.

Also, the 'LoginName' query is executed against the 'Contact' ontology. No appropriate correspondence is provided by the execution engine. Subsequently, this ontology is extended by adding a class called 'LoginName' and a new object property called 'hasLoginName'. Finally, the 'LoginPassword' query is executed against the 'Contact' ontology but no match is found for this service concept. Therefore, the 'Contact' ontology is

extended by adding the 'LoginPassword' class and the 'hasPassword' object property.

Executing the 'Status' and 'Marc' queries:

The 'Status' query is executed against the 'BookProperty' ontology but no matches are found. Therefore, ontology extension is required to add a correspondence for 'Status'. The extension is performed by adding a class labelled 'Status' and an object property called 'hasStatus'. The domain of the 'hasStatus' property is 'Book' and the range is 'Status' as the concept 'Status' denotes a status of a book. After extending the 'BookProperty' ontology with 'Status', the query is executed against this ontology to annotate the 'Status' service concept. Figure 6.11 presents the query result.

```
Result of "Status" simple query:
        OntologicalClass        Score
1.      Status                  1.0
Enter your selection number or type quit to reject recommendations:
```

Figure 6.11: Query Result of 'Status' after Extension

When the user accepts the recommendation by entering 1, the 'Status' service concept is automatically annotated to the 'Status' ontological class as shown in Figure 6.12.

```
<s:element name="Status" sawsdl:modelReference = "http://islab.hanyang.ac.kr/
damls/BookProperty.daml#Status"/>
```

Figure 6.12: Annotation of the 'Status' Concept

In a similar way, the 'Marc' query is executed against the 'BookProperty' ontology because 'Marc' is a characteristic of a book. The query execution, however, does not return any match. Therefore, extension is needed. The extension is conducted by adding a class called 'Marc' and an object property labelled 'hasMarc'.

To summarise, Table 6.2 presents an overview of the `'BookInfoPort'` service annotation activity.

Query elements	Type	Corresponding classes Before extension	Corresponding classes after extension	Ontology
Book (Isbn, Title, Author, PublicationDate, Publisher, Edition, DiscountPrice, ListPrice, PublicationPlace, Availability, DiscountPercent)	C	Book Isbn Title Author Publisher Edition Price Price	Book Isbn Title Author PublicationDate Publisher Edition Price Price PublicationPlace Availability DiscountPercent	BookProperty
CustomerAccount	S		CustomerAccount	Contact
CustomerSubAccount	S		CustomerSubAccount	Contact
LoginName	S		LoginName	Contact
LoginPassword	S		LoginPassword	Contact
Status	S		Status	BookProperty
Marc	S		Marc	BookProperty

Table 6.2: Summary of BookInfoPort Annotation Exercise

Table keys:

S: Simple query.

C: Complex query.

6.3.2 Illustrative case (2): The Service43.Miscellaneous Web service

The Service43.Miscellaneous Web service has operations that provide information about the weather phenomena. The Web service data is described using a rich

XSD which is composed of nineteen complex types and five simple types. Six complex types are computing-specific terminologies which are: `'ArrayOfPrecipitation'`, `'ArrayOfPhenomenon'`, `'ArrayOfstring'`, `'ArrayOfExtreme'`, `'ArrayOfLayer'` and `'ArrayOfStation'`. The remaining thirteen complex types and the five simple types are used to construct complex and simple queries. All the queries are executed against the Weather ontology. The queries' concepts along with queries' results for BE and AE cases are given in Table 6.3.

Query elements	**Type**	**Corresponding classes Before extension**	**Corresponding classes after extension**	**Ontology**
Direction(Compass, Degrees)	C	Direction DirectionCompass	Direction DirectionCompass Degree	Weather
Temperature(Ambient, Dewpoint, RelativeHumidity)	C	Temperature	Temperature Ambient Dewpoint RelativeHumidity	Weather
Extreme(Temperature, Type, Hours)	C	Extreme Temperature ExtremeType	Extreme Temperature ExtremeType Hours	Weather
Visibility (Distance, Qualifier)	C	Visibility Qualifier	Visibility Distance Qualifier	Weather
Layer (Altitude, Extent, Type)	C	Layer ExtremeType	Layer Altitude Range ExtremeType	Weather
WeatherReport (TimeStamp, Station, Phenomena, Precipitation,	C	WeatherReport Station	WeatherReport TimeStamp Station Phenomenon Precipitation	Weather

Extreme, Pressure, Sky, Temperature, Visibility, Wind)		 Pressure Sky Temperature Visibility Wind	Extreme Pressure Sky Temperature Visibility Wind	
Phenomenon(Type, Intensity)	C	Phenomenon PhenomenonType PhenomenonIntensity	Phenomenon PhenomenonType PhenomenonIntensity	Weather
Wind (PrevailingSpeed, GustSpeed, PrevailingDirection, VaryingFromDirection VaryingToDirection)	C	Wind Direction	Wind PrevailingSpeed GustSpeed Direction VaryingFromDirectio nVaryingToDirection	Weather
Pressure (Altimeter, Slp, Delta, DeltaHours)	C	Pressure	Pressure Altimeter Hours	Weather
Station (Elevation, Latitude, Longitude, Name, Region, Country)	C	Station	Station Altitude Latitude Longitude Name Region Country	Weather
Sky (CeilingAltitude, Layers)	C	Sky	Sky Altitude Layer	Weather
Precipitation (Amount, Hours)	C	Precipitation	Precipitation Amount Hours	Weather
Range (From, To)	C	Range	Range	Weather
DirectionCompass	S	DirectionCompass	DirectionCompass	Weather

ExtremeType	S	ExtremeType	ExtremeType	Weather
PhenomenonType	S	PhenomenonType	PhenomenonType	Weather
VisibilityQualifier	S	VisibilityQualifier	VisibilityQualifier	Weather
PhenomenonIntensity	S	PhenomenonIntensity	PhenomenonIntensity	Weather

Table 6.3: Summary of Service43.Miscellaneous Annotation Exercise

Thirty service elements are annotated without any extension while fifty nine concepts are annotated after completing the extension process. The six complex types that denote computing-specific terminologies are partially annotated by linking their child elements to appropriate ontological classes: This annotation process is performed by constructing simple queries for child elements. For example, The 'ArrayOfStation' complex type has 'Station' as a child element. Consequently, annotating the 'Station' child element would result in partial annotation of 'ArrayOfStation'. Figure 6.13 presents the partial annotation of 'ArrayOfStation'.

```
<xsd:complexType name="ArrayOfStation">
       <xsd:sequence>
              <xsd:element name="item" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Station"/>
       </xsd:sequence>
</xsd:complexType>
```

Figure 6.13: Partial Annotation of the 'ArrayOfStation' Service Concept

Only four elements are not annotated by the annotation approach. These elements are 'Slp', 'Delta', 'From' and 'To'. Although extension is performed to provide correspondences for these four elements, the query execution engine is unable to detect these correspondences as matches for these four elements. The reasons for missing these matches are:

1. The 'Slp' concept is an abbreviation for the compound 'Sea Level Pressure'. When performing the extension, 'Sea Level Pressure' is added to the ontology rather than the abbreviation itself. Since our name-based matching mechanism uses WordNet which does not define 'Slp' as a

match for 'Sea Level Pressure', the name-based matcher failed to detect this match.

2. The 'Delta' service concept should be matched to the 'DeltaPressure' ontological class. Since the used name-based matching is based on the CN convention, it cannot match 'Delta' to 'DeltaPressure'. The reason is that the service concept 'Delta' is not a correspondence for the head of the ontological class 'DeltaPressure' which is 'Pressure'.
3. The service concepts 'To' and 'From' should be annotated to the ontological classes 'LowestValue' and 'HighestValue' respectively: These two service concepts are not matches for the heads of the ontological classes and thus the matcher cannot detect these matches. A snapshot of the annotated service is provided in Figure 6.14.

```
<xsd:complexType name="Sky" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Sky">
       <xsd:sequence>
              <xsd:element name="ceiling_altitude" sawsdl:modelReference = "http://
lsdis.cs.uga.edu/WeatherConcepts.owl#Altitude" type="xsd:double"/>
              <xsd:element name="layers" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Layers" type="xsd1:ArrayOfLayer"/>
       </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Precipitation" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Precipitation">
       <xsd:sequence>
              <xsd:element name="amount" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Amount" type="xsd:double"/>
              <xsd:element name="hours" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Hours" type="xsd:int"/>
       </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Range" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
WeatherConcepts.owl#Range">
       <xsd:sequence>
              <xsd:element name="from" type="xsd:double"/>
              <xsd:element name="to" type="xsd:double"/>
       </xsd:sequence>
</xsd:complexType>
```

Figure 6.14: A Snapshot of the Annotated Service43.Miscellaneous Service

6.3.3 Illustrative case (3): The Stock Information Service

The Service7.Stock offers operations that provide stock and market news, headlines and briefings. The parameters of this service are described using an

XSD definition that contains twenty four complex types. Only seven complex types have meaningful labels. The other seventeen labels are computing-specific terminologies. An example of a computing-specific terminology is the `'ArrayOfBriefing'` service element. The seventeen complex types that do not have meaningful labels are partially annotated by creating simple queries for their child elements.

Seven complex queries are constructed for the seven meaningful complex types. The main service concepts and the related concepts of the queries along with the correspondences of all service concepts are given in Table 6.4. All the queries are executed against the LSDIS-Finance ontology.

Query elements	**Type**	**Corresponding classes Before extension**	**Corresponding classes after extension**	**Ontology**
Briefing (Title, Text)	C		Briefing Title Text	LSDIS-Finance
StockHeadlines (Symbol, HeadlineCount)	C		StockHeadlines Symbol HeadlineCount	LSDIS-Finance
StockNews (Headline, Ticker, Date, Time, Source, Url, Error)	C		StockNews StockHeadlines Ticker Date TimeInterval Source Url Error	LSDIS-Finance
Ticker (Symbol)	C	Ticker	Ticker Symbol	LSDIS-Finance
MarketNews (Headline, Time, Source,	C		MarketNews StockHeadlines TimeInterval Source	LSDIS-Finance

Url, Summary)			Url Summary	
MarketNewsItem (Headline, Date, Source, Url, Content, Error)	C		MarketNewsItem StockHeadlines Date Source Url Content Error	LSDIS- Finance
EarningAnnouncement (AnnouncementDate, Symbol, Company, EpsEstimate, AnnouncementTime, Message)	C		EarningAnnouncement Date Symbol Company EpsEstimate TimeInterval Message	LSDIS- Finance

Table 6.4: Summary of Service7.Stock Annotation Exercise

Table 6.4 shows that only one service element which is `'Ticker'` is annotated before extension and thirty six elements are annotated after completing the extension. Figure 6.15 presents a snapshot of the annotated service.

```
<s:complexType name="Briefing" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Briefing">
       <s:sequence>
              <s:element name="Title" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Title" type="s:string" />
          <s:element name="Text" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Text" type="s:string" />
              <s:element name="Html" type="s:string" />
       </s:sequence>
</s:complexType>
<s:complexType name="StockNews" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#StockNews">
       <s:sequence>
              <s:element name="Headline" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#StockHeadline" type="s:string" />
              <s:element name="Ticker" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Ticker" type="s:string" />
              <s:element name="Date" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Date" type="s:string" />
              <s:element name="Time" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#TimeInterval" type="s:string" />
              <s:element name="Source" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Source" type="s:string" />
              <s:element name="Url" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Url" type="s:string" />
              <s:element name="Error" sawsdl:modelReference = "http://lsdis.cs.uga.edu/
LSDIS_Finance.owl#Error" type="s:string" />
       </s:sequence>
</s:complexType>
```

Figure 6.15: A Snapshot of the Annotated Service7.Stock Service

6.4 Experimental Evaluation

Evaluation is a very significant step in any research project (Hevner et al., 2004): It reveals the strengths and weaknesses of research approaches and can lead to improvements of proposed solutions. Appropriate and successful evaluation requires the selection of suitable evaluation methods and metrics. The following subsections illustrate in detail the method and metrics used to evaluate the proposed semi-automatic annotation approach.

6.4.1 The Experiment Design and Metrics

The evaluation examines the performance of the proposed annotation approach by testing it in practical settings. The evaluation is performed by annotating a set of

twenty five Web services belonging to five different domains. The annotation results represent the required evaluation data. The annotation results are then measured using four different evaluation metrics. These metrics are Precision (P), Recall (R), F-measure (F) and Percentage of annotated elements.

As noted in Chapter 5, P, R and F are metrics used to measure the performance of information retrieval approaches (Bukland and Gey, 1994) and ontology matching mechanisms (Euzenat et al., 2009). In this evaluation, these three metrics are also deemed appropriate because the annotation process involves matching executed by the query execution engine. Furthermore, previous Web service semi-automatic annotation approaches (Patil et al., 2004) used P, R and F in their evaluation. The formulas of P, R, and F are given in Equations 5.3, 5.4 and 5.5 in Chapter 5.

In the context of this evaluation, recall is important because it measures the completeness of the annotation results in relation to the gold standard. Precision, is also a significant measure for this evaluation because high precision means that clean results are provided to the user after query execution. Although the proposed annotation approach is semi-automated in the sense that all results over a specified threshold are presented to the user who has to take a decision, providing many wrong recommendations may have a negative impact on the annotation process. This is because many incorrect options could make the selection of the appropriate correspondence a time-consuming and an error-prone task.

The percentage metric is used in this evaluation to provide a measure of how many elements of a given service are annotated in relation to the total number of candidate service elements. Unlike P, R and F which only measure the performance and effectiveness of the matching mechanism, percentage gives a clearer idea about the performance of the whole annotation approach. For example, a full mark of P, R and F could be obtained for a given case however; many service elements could be left without annotation. That's it; percentage can be used to show how good the annotation framework is in alleviating the 'Low

Percentage Problem'. Equation 6.1 shows the method of calculating the Percentage metric.

$$Percentage = \frac{NumberOfAnnotatedElements}{TotalNumberOfServiceElements} \qquad (6.1)$$

The 'NumberOfAnnotatedElements' represents the number of all elements that are automatically annotated either fully or partially. The 'TotalNumberOfServiceElements' stands for the number of all XSD elements in a given service. Although, there are some sorts of elements (See Section 4.5) that are excluded from the annotation process, these elements are taken into account when calculating Percentage. Some of these concepts can, sometimes, be partially annotated. Figure 6.16 presents an example of partial annotation.

```
<s:complexType name="GetAreaCodesForCity">
      <s:sequence>
                  <s:element name="City" sawsdl:modelReference = "http://
moguntia.ucd.ie/owl/Datatypes.owl#City"/>
                  <s:element name="State" sawsdl:modelReference =
"http://moguntia.ucd.ie/owl/Datatypes.owl#State"/>
      </s:sequence>
</s:complexType>
```

Figure 6.16: Partial Annotation of a Complex Type

In retrospect, calculating the values of P, R and F requires a gold standard against which the results of automatic annotation are compared in order to find these values. Gold standards should, ideally, be provided with the test sets like what we had in the evaluation of CN-Match (See Section 5.7.3). Unfortunately, no gold standards that describe the annotation of Web services to ontologies can be found. Therefore, five sets of Web services are selected and used in this evaluation exercise (see Subsection 3.4.4 for full description of services and their selection method). Every set contains five Web services belonging to a specific domain. Five ontologies are also selected and placed in an ontology repository. The selected ontologies have never been used to annotate the twenty five Web

services. Consequently, generating the gold standards of the annotation is required.

Since the annotation process involves ontology extension, the values of the four preceding metrics are calculated before and after extension. The purpose of performing the measurements in both cases is to show the impact of the extension process on the annotation results. Consequently, two gold standards are required for each Web service annotation activity (one for the annotation before extension and for after extension). Hence, fifty gold standards were generated manually prior to conducting the semi-automatic annotation. For validity purposes, theses gold standards are checked by three people who are native English speakers and have very good knowledge about the five domains and the Semantic Web technology. Few changes are made to the gold standards based on recommendations from the three people.

6.4.2 Evaluation Method

To perform the required evaluation the five sets of Web services are annotated using the ontology repository. To annotate a given service, queries are constructed for simple as well as complex XSD types of the given service. These queries are then executed using the query execution engine to provide the annotation results. The results of these queries cumulatively make the result of a given service. Generally speaking, a query answer provided by the query execution engine contains zero or more results. Zero or one of the provided results is correct. Consequently, to measure P, R, F and Percentage, results of each query are classified into correct-retrieved (true positive), incorrect-retrieved (true negative) and missing or correct-not retrieved (false negative). This classification of results is obtained by comparing these results against the gold standard of a specific service. Table 6.5 categorises the matching results.

	Correct	Incorrect
Retrieved	True Positive	True Negative
Not-Retrieved	False Negative (Missing)	False Positive

Table 6.5: Classification of Matching Results

Table 6.6 presents partial results of a service called 'GeoCash' which belongs to the Payment domain. This service allows users to search for locations of ATM machines.

Query elements	Correct-Retrieved	Incorrect-Retrieved	Missing
AtmLocations (AtmMachine, Error)	Location AtmMachine Error	Address	
Error (Desc, Number, Location)	Error Number Location	Confirmation WorkingKey	Description

Table 6.6: Partial Results of the GeoCash Service

Table 6.6 shows elements of two queries. The first query has 'AtmLocations' as a main concept and 'AtmMachine' and 'Error' as related concepts. The 'AtmLocation' main concept has 'Location' as a correct-retrieved match and 'Address' as an incorrect-retrieved match. No missing matches are found for this query. The second query has 'Error' as a main concept and three related concepts which are 'Desc', 'Number' and 'Location'. The main concept 'Error' and the related concepts 'Desc' and 'Location' have correct-retrieved correspondences which are 'Error', 'Number' and 'Location' respectively. In addition, the main concept 'Error' and the related concept 'Number' have 'Confirmation' and 'WorkingKey' as

incorrect-retrieved matches, respectively. The related concept `'Desc'` has a missing match which is `'Description'`.

To summarise, the steps of calculating P, R, F and Percentage are given as follows:

1. Generate the required gold standard using the service under consideration and the ontologies available in the repository.
2. Perform the annotation activity for each query.
3. Record the results of the executed query.
4. Group the results of all queries together.
5. Compare the obtained grouped results against the relevant gold standard.
6. Calculate P, R, F and Percentage for the annotated service.

To present the evaluation steps in a graphical format, Figure 6.17 shows the applied method.

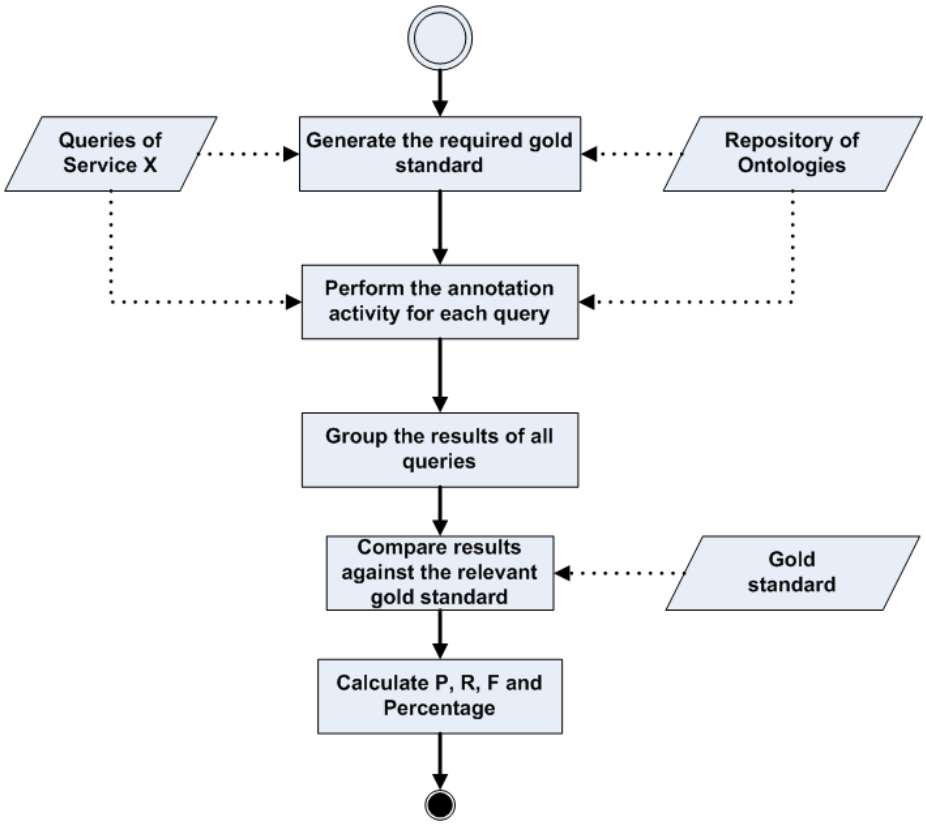

Figure 6.17: The Evaluation Method of the Annotation Framework

6.4.3 Evaluation Results

In the following paragraphs and Figures, the evaluation results in terms of the four selected metrics are presented. The results of each domain are discussed separately. For every domain, two types of results are presented. The first type is for the annotation results before extension (BE) while the second type is for results after extension (AE). The charts on the left hand side show the BE results while the ones on the right hand side shows the AE results.

A. Results of the Book Domain

The Book domain contains five services that provide operations for searching book information. Good results in terms of R, P, F and Percentage values are obtained. These results differ between BE and AE cases. For example, R values of the `'BookInfoPort'` service increase from 0.9 to 0.94 and those values drop from 1.0 to 0.63 in the `'Books'` service. The reason for the latter decrease in the R value is that the `'Books'` service contains elements that have acronyms or badly-formed CNs as their labels: These elements are not annotated in the BE case as no ontological correspondences exist for them. Therefore extension is performed. For example, two correspondences are added for two concepts that are called `'NumPages'` and `'TOC'`. The added correspondence of the first concept is `'PagesNumber'` while the added correspondence of the second concept is `'ContentsTable'`. These two extended classes are not detected by the execution engine because `'NumPages'` is not well-formed and `'TOC'` is an acronym. Consequently, these missing matches are classified as correct-not retrieved and hence a lower R value is obtained. Retrospectively, the matching techniques employed by the execution engine cannot match a full expression against its acronym. The reason is that matching acronyms against their original expressions is very hard in domain-independent settings since a specific acronym could denote different meanings in different domains.

The reason behind adding `'PagesNumber'` and `'ContentsTable'` as correspondences rather than adding classes that carry identical labels to the service concepts is that the quality of labels of ontological classes should be kept as good as possible. In other words, adding classes that have vague or not well-formed labels will affect the quality of ontology as an ontology should always be a shared and precise conceptualisation (Gruber, 1993). In addition, low quality and unclear labels of classes are unlikely to be detected as correct correspondences in any future annotation activities performed by a matcher that is based on the name-based matching technology. Consequently, those badly-formed labels of classes would be of very limited use.

After extension, the P values increase for all services: This is because the number of correct-retrieved matches increases while the number of incorrect-retrieved matches almost stays the same. The values of F measure increase in all cases except for the Books service. This increase is because the decrease in R values is much lower than the increase in P values.

The Percentage values of AE are always higher than those values of BE. This is because extension provides correspondences for service elements that do not have matches before extension. The later result is good since it means that the proposed approach can beat the 'Low Percentage Problem' that previous approaches suffer from. Figure 6.18 presents the results of the Book domain.

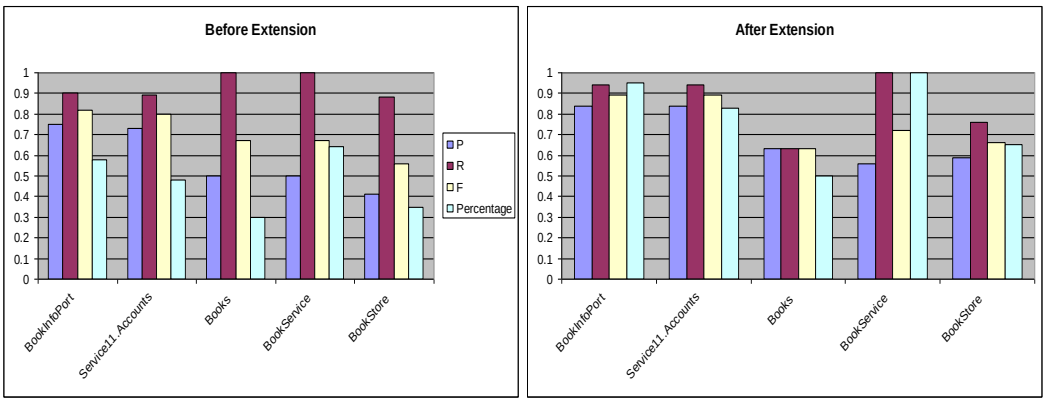

Figure 6.18: Results of the Book Domain

B. Results of the Weather Domain

The second domain in this evaluation exercise is the Weather domain. This domain has five services that provide operations offering weather and forecast information. The P and Percentage values of the Service38.Accounts, Service3.Miscellaneous, Service47.Utility and Service185 increase after extension in comparison to these values before extension. On the contrary, the values of R for these four services decrease after extension. The reason for having less R values of AE cases in comparison to these values in the BE cases is that the execution engine fails to detect some of the extended concepts as correct matches of their corresponding service concepts. As mentioned earlier in the results of the Book domain, only meaningful and well-formed labels of classes are added during the extension activity. These newly added classes might not match with their corresponding service concepts when labels of these service elements are not well-formed or composed of parts of words or abbreviations.

For the Service51.Utility, the values of the four metrics stay the same for BE and AE cases: This is because no extension is performed since all service concepts that can be annotated have corresponding ontological concepts. Although all concepts that can be possibly annotated have correspondences, the value of Percentage is not one. This is because there are service elements that cannot be annotated either partially or fully. These later elements belong to the categories defined in Section 4.5. Figure 6.19 shows the evaluation results of the Weather domain.

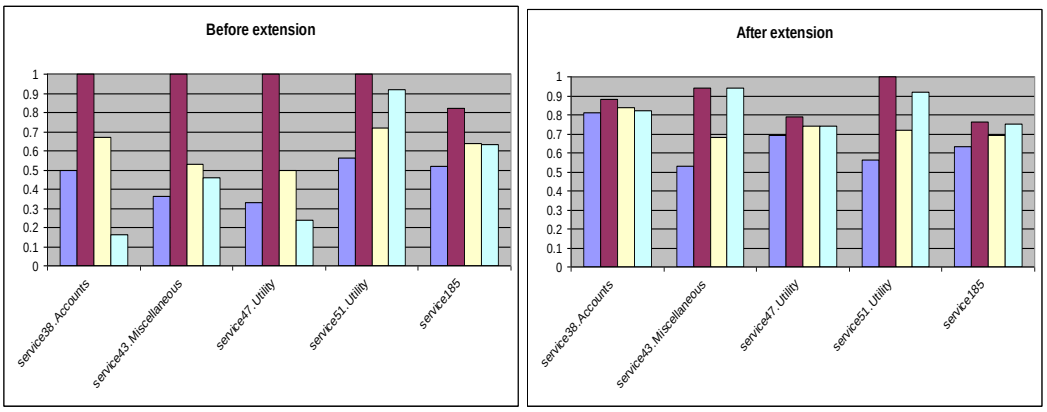

Figure 6.19: Results of the Weather Domain

C. Results of the Stock Information domain

The third domain that is used in this evaluation is the Stock Information domain which also has five services. These services provide stock quotes and market news. Before extension, one service which is Service11.Stock has no annotated elements. This later result is proved by having no values for the four metrics. Moreover, low percentages of annotated elements in BE cases are provided for the other four services. Full R values, however, are obtained for these services which indicate that all possible correct matches are retrieved by the query execution engine.

After extension, all the five Web services have most of their elements annotated since high percentages of annotated elements are given. The service Service11.Stock which has no annotated elements before extension has all of its elements annotated after extension since the value of the Percentage metric is 1. The R values of the Service3.Stock, Service7.Address and Service17.Stock decrease after extension. The reason for this decrease is the same as the one mentioned earlier in the previous two domains. The P and F values significantly increase after extension. Figure 6.20 presents the evaluation results of the Stock Information domain.

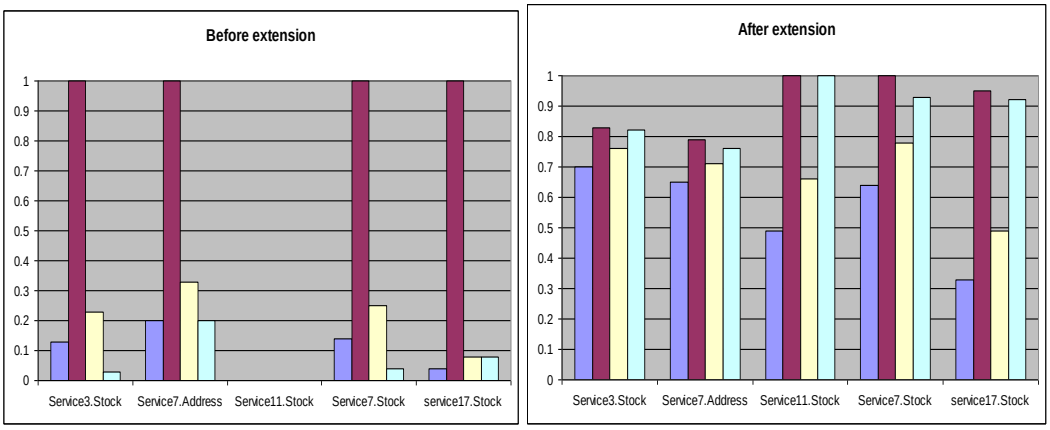

Figure 6.20: Results of the Stock Information Domain

D. Results of the Communication Domain

The fourth domain used in this evaluation is the Communication domain. Services of this domain provide operations for sending emails, SMS, and faxes and for making calls. Before extension, the Service50.Miscellaneous has no annotated elements. The other four services have some of their elements annotated. This later result is shown by the low values of the Percentage metric of these four services. Full R values are obtained by the Service9.Specialist, Service4.Specialist and Service60.DeveloperTools while 0.5 R value is given to the Service80.Miscellaneous. In addition, low P, F and Percentage values are obtained for the four services.

After extension, the values of P, F and Percentage significantly increase for all the services. The R values stay the same for Service9.Specialist and Service4.Specialist, slightly decrease for Service60.DeveloperTools and increase for Service50.Miscellaneous and Service80.Miscellaneous. Figure 6.21 presents the results of the Communication domain.

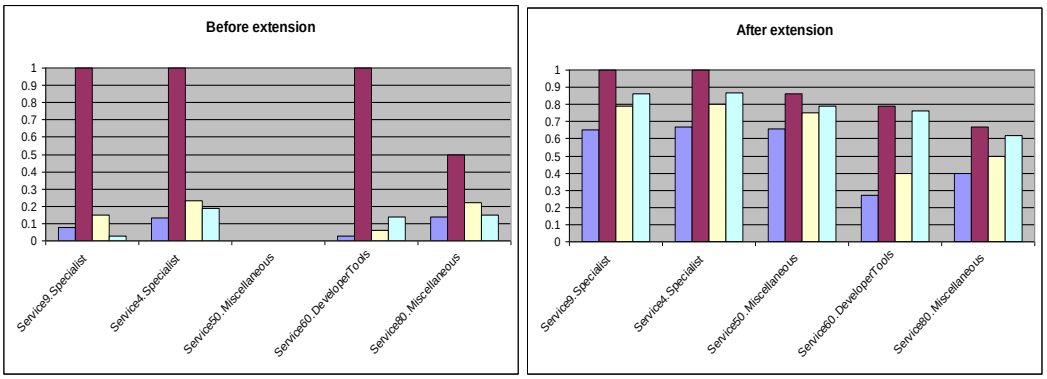

Figure 6.21: Results of the Communication Domain

E. Results of the Payment Domain

The last domain in this evaluation exercise is the Payment domain. Before extension, Service72.Accounts has no annotated elements but the other four services have high R values and low P, F and Percentage values. After extension, the P, F and Percentage values significantly increase. The R values increase for Service72.Accounts and Service68.Accounts, slightly decrease for Service24.Accounts and GeoCash and does not change for Service39.Accounts. Figure 6.22 provides the evaluation results of the Payment Domain.

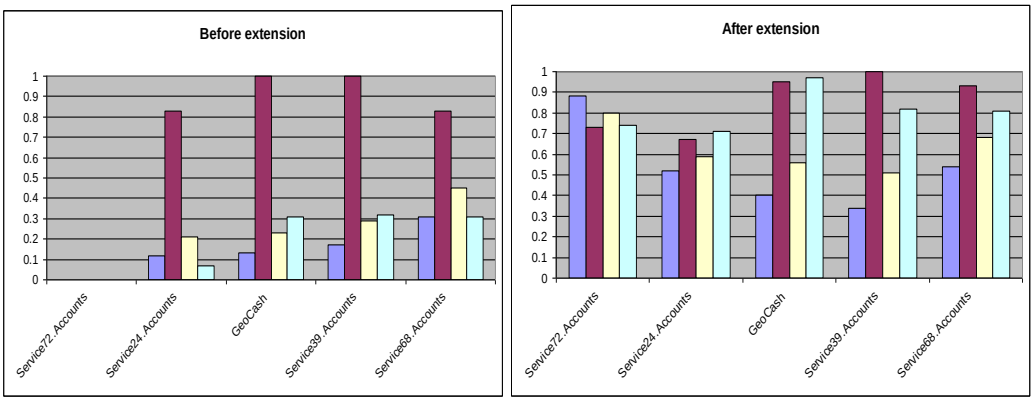

Figure 6.22: Results of the Payment Domain

6.5 Discussion and Limitations

After describing the results of individual domains, averages of P, R, F and Percentage per domain are calculated. These averages allow us to generalise evaluation results and draw important conclusions.

6.5.1 Discussion of Averages across Domains

Presenting averages is very important because they can be used to compare and discuss the evaluation results across the five domains. The comparison leads to implications and conclusions and capture the limitations of the proposed annotation approach.

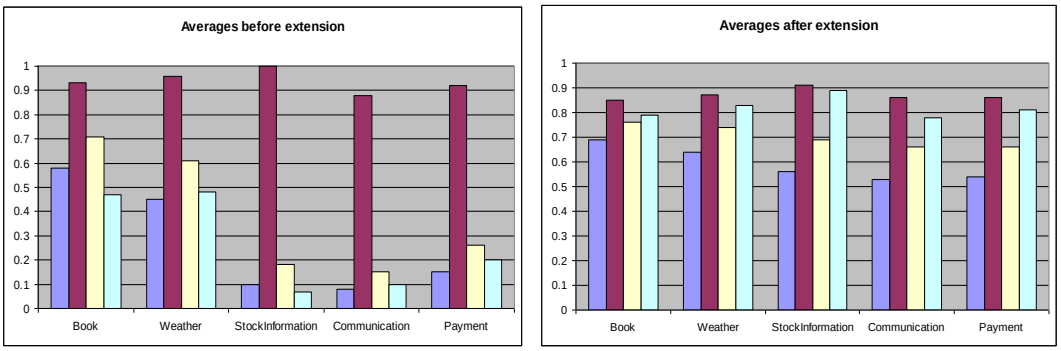

Figure 6.23: Averages of Metrics for the Five Domains

By comparing the averages before extension against the averages after extension and comparing the averages across domains, the following observations and interpretations are presented:

A. R values are the highest amongst all metrics. R values range from 0.83 for the Book domain after extension to 1.0 for the Stock Information domain before extension. Having significantly high R values is a very important merit which proves that the proposed approach is effective since nearly all possible correct correspondences are retrieved by the proposed semi-automatic annotation approach.

B. R values slightly decrease after extension. Some service elements may not have appropriate correspondences before extension because such correspondences are missing from the ontologies. Consequently, relevant annotation gold standards do not contain such correspondences. After performing extensions and adding appropriate correspondences, the gold standards are updated to accommodate the newly added correspondences.

Sometimes, service elements can have labels that are acronyms or badly-formed CNs. Classes added by the extension process as correspondences of those elements can, however, have labels different from labels of those elements. This is because names of ontological classes must be meaningful and well-formed to precisely represent semantics. Since the matching techniques employed by the execution engine cannot always match an acronym against its original expression or a well-formed CN against a badly-formed CN, some correspondences may not be retrieved by this execution engine. Subsequently, a decrease in R values may happen.

The decrease in R values is, however, very small since most extended classes are retrieved by the query execution engine as correspondences of service concepts. The latter result is proved by the relatively low changes in the averages of R values shown in Figure 6.24 where the maximum decrease is 9%.

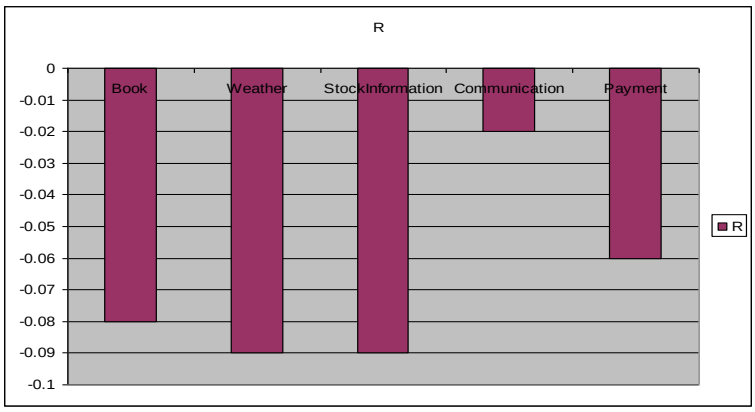

Figure 6.24: Changes in Averages of R Values

C. P values significantly increase after extension: This is because the extension process adds correspondences that are very likely to be detected by the execution engine. Given that the P Equation is represented by the ratio of correct-retrieved to total number of automatic matches and the extension process increases the number of correct-retrieved, the P value will increase. Figure 6.25 shows the changes in P values from BE to AE.

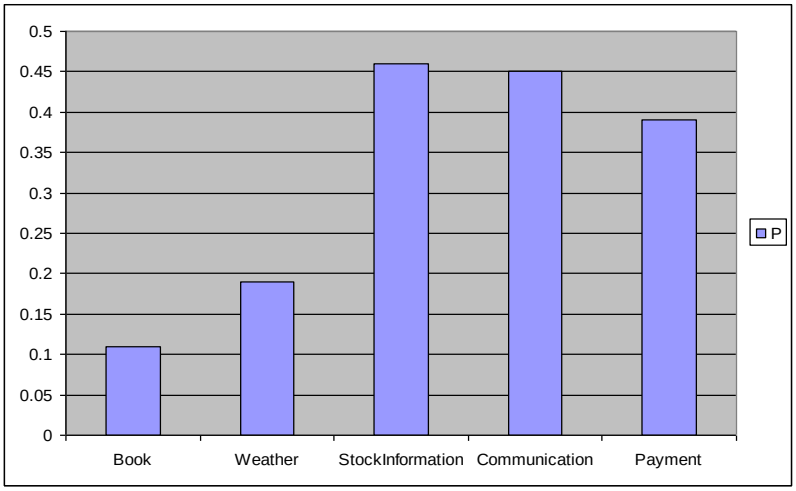

Figure 6.25: Changes in Averages of P Values

D. F values increase after extension. The reason is that F combines P and R and the increase in P is much higher than the decrease in R. Figure 6.26 presents the changes in F values.

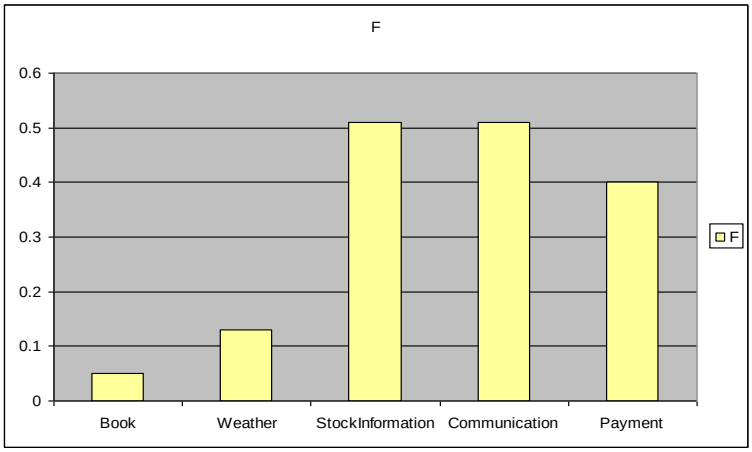

Figure 6.26: Changes in Averages of F Values

E. Percentage values significantly increase after extension. Figure 6.27 shows that the changes in Percentage for the Book (32%) and Weather (38%) domains are moderate while the changes in StockInfo (82%), Communication (68%) and Payment (61%) are high. The reasons behind these percentage results are: (1) The Book and Weather domains are relatively small domains and have concepts that are shared by almost all their Web services; and (2) The StockInformation, Communication and Payment domains are relatively big domains and many different concepts exist in each of these domains. Subsequently, extensions seem to be regularly required to add ontological correspondences of new service concepts in the last three domains.

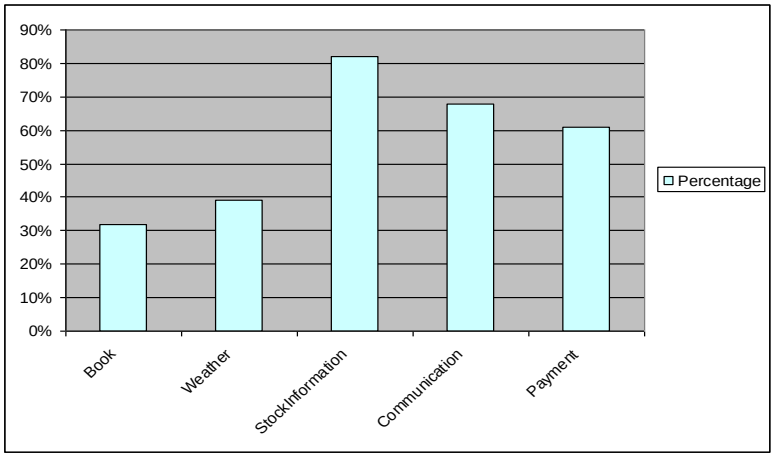

Figure 6.27: Changes in Averages of Percentage Values

6.5.2 Implications of Presented Results

A. The Effectiveness of CN-Match as a Name-based Matching Mechanism

Many elements of the services and labels of ontologies used in this evaluation are CNs. Consequently, having an effective CN matching mechanism is very crucial to the success of the proposed annotation approach. CN-Match allows the query-execution engine to detect appropriate matches that are neither synonymous nor identical to the given service concepts. Sometimes, more general concepts are appropriate correspondences to the given query concepts. Just to give a single

example, the concept `'AtmLocation'` which belongs to the Payment domain is correctly matched to the ontological class `'Location'`. In addition, less general concepts can also be suitable matches of service concepts. For instance, the service concept `'Price'` which belongs to the Book domain is correctly matched to the ontological class `'BookPrice'`.

These more and less general matches that are composed of different number of constituents would not be retrieved without having CN-Match as an effective and accurate CN matching mechanism. Moreover, many correct correspondences that are detected by the query execution engine are synonyms or have constituents that are synonyms. Not only synonyms but also names that carry similar meanings are captured by the engine. An example of these similar names that are detected as matches is the pair `'BookName'` and `'Title'` which belongs to the Book domain. Synonyms and semantically similar matches are retrieved because CN-Match effectively and correctly employs WordNet and its path-based matching mechanism. Given all these good matching results, CN-Match is appropriately designed and used by the query execution engine.

B. Completeness of Results in Relation to Manual Results

The preceding experimental results show that R values are significantly high. Almost complete annotation results in relation to manual results (gold standard) are retrieved by the employed query execution engine. These complete results are achieved because of the effective use of the name-based and structural matching techniques. In retrospect, the core name-based matching mechanism allows the query execution engine to detect almost all candidate correspondences that have similar labels. In addition, the structural matching mechanism positively contributes to the completeness of results because it uses a balanced structural matching framework.

The employed framework is balanced because it utilises neither too little nor too many matching restrictions. The employed structural matching approach searches for ontological classes that have related classes which are semantically similar to related concepts of the given query concept. This structural matching criterion is

appropriate for the purpose of the proposed annotation approach since related concepts of a service concept constitute the context of the given service concept. This context is very significant to the meaning of the given concept as it allows the execution engine to differentiate between homonyms (same spelling but different meaning). Moreover, the context similarity allows the engine to detect correspondences that have similar related concepts but different names.

On the other hand, other annotation approaches like (Patil et al., 2004) use other structural matching criteria such as the similarity between data types of service elements and data types of related concepts. We argue that imposing unnecessary restrictions on the employed structural matching approach will have negative impacts on the completeness of results because appropriate correspondences should not always satisfy all the imposed restrictions. For example, the identification of a primitive data type of a related concept can be a subjective matter. In other words, it is unlikely that developers always agree on the same primitive data type for a given concept. For example, a related concept called `'Price'` could have `'Integer'`, `'Double'` or `'String'` as its primitive data type.

C. Significance of Semi-automation of Annotation

Given that the query execution engine makes use of a set of matching techniques that do not always give perfect results, minimum human intervention can make big improvements. A fully automated approach that would select the correspondence that has the highest score as the correct match would result in many missing and incorrect correspondences and hence a lower P and R values would be obtained. The reason is that it is not always necessary that the candidate with the highest confidence degree is the most appropriate one since current matching techniques such as WordNet and string matching can have their understandable limitations. In the context of the developed annotation approach, the user has to select the appropriate correspondence from the given set of recommendations. The selection decision is based on the user basic knowledge and thus it is easy to make. The choice is based on two simple criteria: (1) The

similarity of the main service concept to the candidate ontological class; and (2) similarities of related concepts of the main service concept to related concepts of the candidate ontological class.

D. Usefulness of Queries

Queries are used in the context of this research as alternatives to ontologies that model the semantics of candidate services. Retrospectively, ontology building is a very difficult task requiring extensive domain and technical knowledge (Gruber, 1993). Consequently, making ontology building a prerequisite to annotation activities would make these activities very difficult and time consuming. Therefore, the proposed query-based annotation approach can speed up and simplify the annotation process since it does not require any extensive technical or domain knowledge. The reason behind this simplicity is that all annotation steps are standard, straightforward and easy to use. For example, filling the query template from the concept extraction outputs requires no technical or domain knowledge and can be performed by any computer literal person.

One might argue that building an ontology to represent semantics of a candidate service would result in better and more accurate annotation results. We argue that building an ontology to capture the service semantics and then matching it against existing ontologies to generate correspondences would not only make the annotation task very difficult but also introduce many unnecessary restrictions on the matching process. Ontology to ontology matching requires much more matching techniques such as super and subclasses matching which when applied would decrease the opportunity of retrieving the required matches (Euzenat and Valtchev, 2004). The proposed approach proves that fewer matching restrictions would generate more candidate results and thus increase the opportunity of retrieving correct matches. This later result is proved by having high R scores which indicate that nearly all possible correspondences in relation to gold standards are retrieved by the query execution engine.

E. Importance of Extension

The extension process is a very significant and successful technique that supports the proposed annotation approach. This conclusion is derived from the annotation results that show significant improvements in Percentage values as well as good increase in P and F values after performing extensions. The extension approach can be seen as an effective solution to the 'Low Percentage Problem' introduced by previous annotation approaches. Having a high percentage of correctly annotated elements would significantly increase the usefulness and efficiency of the whole annotation approach and consequently improve the adoption of the SWS technology.

F. The Importance of Annotation to Shared Ontologies

According to the scenario adopted in this research (See Subsection 3.4.4), services are annotated to ontologies that are shared with other services. Annotation to shared ontologies is important because resulting services are ready to interoperate and be composed by software agents. When services are annotated to different ontologies, automatically matching these ontologies at run time becomes a prerequisite for any future composition or interaction activities (Patel et al., 2004). Given the imperfect nature of current automatic ontology matching approaches, this matching process could delay the required interaction and composition activities and be error-prone. Annotating services to shared ontologies will significantly minimise the need for automatic ontology matching processes as most of the required services are annotated to the same set of ontologies (Patel et al., 2004).

6.5.3 Limitations of the Proposed Approach

Although promising results are achieved by the proposed annotation mechanism, limitations exist that provide future research. These limitations can be classified as matching-dependent, annotation-dependent and extension-dependent.

A. Matching-dependent limitations

The preceding evaluation results reveal that full P, R, F and Percentage values are not always obtained by the proposed annotation approach: This is because there are a few types of service elements and ontological labels that are either unlikely to be correctly matched or cannot be matched using the current matching mechanism. These types are:

- Elements that do not carry significant meanings: These elements belong to the three categories already defined in Section 4.5. These categories are: (1) Elements representing method names; (2) elements that denoting computing-specific terminologies; and (3) elements representing individuals rather than classes.
- Elements and ontological labels composed of more than three constituents: CN-Match can measure similarities between CNs that have a maximum of three constituents. For some annotation cases, matching labels composed of more than three constituents is needed. Designing a matching mechanism that can cater for those more complicated CNs is a hard task since syntactic ambiguity of CNs increases as the number of their constituents increases (See section 5.3). Significantly more modifier-head structural cases than those currently covered by CN-Match must be taken into account: This is because a CN that has four constituents, for example, has more modifier-head structural possibilities than a CN that is composed of three constituents only. The six design cases of CN-Match can, however, provide a foundation for developing new cases that can match more complicated CNs.
- Elements and ontological classes denoting acronyms or parts of words: CN-Match employs WordNet and Sting-based matching as its basic matching techniques. Subsequently, CN-Match inherits the limitations of these basic techniques. WordNet, which is an English Thesaurus, contains a very limited set of acronyms. Consequently, many service elements and ontological labels that have acronyms as their names cannot be correctly matched to other elements or labels that are denoted by the full expressions of those acronyms. In addition, matching acronyms against their original expressions is very difficult in a domain-independent context because an acronym can denote

different meanings in different domains. Further, matching a part of word such as 'App' to its original word cannot be performed by CN-Match. Catering for the later type of matches is very hard since the same part of word can belong to many full words. For example, 'App' can be a part of 'Approach', 'Apple', 'Application' and many others. Therefore, using parts of words to denote service elements should be avoided because it can cause much confusion and make the matching process very hard.

- Domain-specific elements and ontological classes: CN-Match is a domain agnostic name-based matching mechanism because it makes use of WordNet which is a domain-independent thesaurus. Subsequently, some names that are very domain specific are unlikely to be correctly matched and annotated using the current annotation approach. For example, the Service60.Developers which belongs to the Communication Domain has two service elements which are 'From' and 'To'. These two elements should be annotated to 'Sender' and 'Receiver', respectively. These two matches are not retrieved by the execution engine as they do not match using WordNet and its path-based mechanism. Matching such correspondences, however, requires domain specific thesauri that define all matches in particular domains.
- Elements and ontological classes that have many identical or very close matches in WordNet: Since WordNet is a general purpose thesaurus; it might give same or very close confidence degrees when matching a given element against different ontological classes. These confidence degrees result in having many potential correspondences to the given element and thus increase the number of incorrect-retrieved candidates. This increase in incorrect-retrieved results raises the number of recommendations presented to users and thus decreases P values.

B. Annotation-dependent Limitation

The proposed annotation approach can provide semantics for a subset of service elements identified in the set of synthesised elements (See Section 2.4). The approach can offer semantic descriptions for elements that use XSD for their descriptions because the input of the annotation approach is XSD data of WSDL files. These elements are; 'Input', 'Output', 'Precondition' and 'Effect'. Other

elements such as; 'Category', 'Non-functional Semantics', and 'Execution Semantics' are not supported in the proposed approach because the semantics required for their descriptions cannot be extracted from WSDL and XSD files.

C. Limitations of Extended Ontologies

The extended ontologies miss some ontological structures. Although these missing structures do not directly affect the annotation process, they might have some impact on the software agents that will make use of these ontologies. These missing structures result from the nature of the extension mechanism adopted in this annotation approach. The extension mechanism is designed to add classes and object properties in order to create correspondences for elements of simple and complex queries. Generally speaking, an ontology can contain more axioms and structure than what is added by the proposed extension mechanism. Examples of these axioms and structures are super and subclasses, cardinality restrictions and data type properties. Additions of these extra structures, however, requires more complicated rules to detect the type of relations or axioms between a given service concept and existing or added ontological entities. In addition, adding these structures by the extension process might complicate and delay the extension and annotation process. Future research can study the feasibility of including additional structures and their impact on the annotation and SWS processes. Further, the quality of extended ontologies is not fully evaluated using appropriate evaluation metrics. This is because evaluation of ontologies is hard and requires much work and time.

6.6 Summary

This chapter concentrated on evaluating the effectiveness of the proposed annotation approach. The chapter started by providing a rational for the ontology extension process and presenting the extension mechanism for simple and complex queries. Next, three illustrative cases explaining the annotation steps were presented to show the annotation framework in practice and provide extra clarifications to the annotation process. Thereafter, the evaluation method and

results were presented. P, R, F and Percentage were utilised as evaluation metrics. For every Web service, these four metrics were measured before and after extension to show the impact of the extension activity on the annotation process. Later, averages of P, R, F and Percentage were calculated for every domain of the five domains. These averages were then compared to show how the annotation results differe across domains. These differences led to significant implications about the proposed semi-automatic approach. The conducted evaluation demonstrated that the proposed approach is very effective and promising since nearly complete annotation results, in relation to manual results, were obtained. Moreover, the evaluation proved that the extension process is very useful because significant improvements in percentages of annotated elements were obtained after performing the extension. Finally, some limitations of the proposed approach were given which can be seen as motivations for improvements in future research.

Chapter 7: Conclusions

7.1 Overview

This chapter discusses the research conclusions and presents contributions and future research avenues. Section 7.2 gives a summary of this research by providing the theme and findings of all thesis chapters. Section 7.3 demonstrates the research conclusions and contributions to theory and practice. Section 7.4 discusses how this research meets its defined objectives. Section 7.5 illustrates the research limitations and Section 7.6 presents future research directions.

7.2 Summary of the Research

The research presented in this thesis aimed at designing a semi-automatic Web service annotation framework that can help Web service developers in transforming their syntactic Web services into semantic ones in an effective and easy to use manner.

Chapter 2 reviewed previous research in the area of semi-automatic annotation of Web services. Web services and their industrial standards were discussed. The discussion highlighted that existing standards such as; WSDL, UDDI and SOAP are unable to support automatic discovery and composition of services because they ignore important semantic constructs. The proposed solution for automation is Semantic Web services (SWS). Due to the central role of ontologies in the SWS area, ontologies, their engineering, learning and extension were illustrated. It was

concluded that manual ontology building is difficult and knowledge-intensive and automatic ontology building is still under development and cannot provide good ontologies. Further, the major SWS description frameworks were presented and compared against a set of synthesised elements to assess their completeness.

The SWS literature reports that SWS adoption by developers is low because Web services are currently annotated manually. Manual annotation is hard, error-prone and time-consuming and thus automation is needed. Next, the existing semi-automatic annotation approaches were discussed and classified into: Learning-based, workflow definition-based and matching-based approaches. Limitations of these approaches were uncovered and issues like annotation difficulty and efficiency, matching effectiveness and the Low Percentage Problem were raised.

Chapter 3 discussed the research method of designing and evaluating the proposed semi-automatic annotation approach. In order to provide a rationale for selecting Design Science Research (DSR) as the right research approach for conducting this research, the fundamental IS research methods were presented. DSR is a problem solving paradigm that aims to develop purposeful and effective artefacts that can solve significant research problems. Later, the research was described in light of the DSR research cycle. This cycle is composed of five fundamental phases called awareness of problem, suggestion, development, evaluation and conclusion.

The DSR approach adopted in this project is of an incremental nature. This means that the design of the approach comprises granular components that were developed during different increments of the design process. Six increments were identified and presented in this design process which are: (1) The design of the initial framework; (2) the design of the concept extraction technique; (3) CN-Match design; (4) structural matching design; (5) SAWSDL annotator design; and (6) design of the ontology extension mechanisms. The incremental learning happened throughout the research increments was also highlighted. Later, the methods, metrics and data used to evaluate the proposed annotation framework and its underlying artefacts were presented. Functional (black-box) and

experimental evaluation methods were used to evaluate the annotation approach and its components. To avoid any potential bias when conducting the evaluation, existing ontologies and Web services were used. Artefacts are very important in any DSR project: They are the fundamental outputs of DSR research activities. DSR artefacts are generally categorised into constructs, models, methods and instantiations. In light of this classification, the artefacts of this research were discussed and presented.

Chapter 4 presented the proposed Query-based semi-automatic annotation framework: This approach eliminates four important deficiencies of existing matching-based annotation frameworks. These limitations are: (1) The prerequisite of building application ontologies to represent service semantics; (2) the inaccuracy and ineffectiveness of implemented matching approaches; (3) the Low Percentage Problem; and (4) the annotation of service elements belonging to same domain to different domain ontologies. In overcoming these limitations, a set of design requirements were defined. Based on these requirements and knowledge acquired from previous annotation approaches, design strategies were identified to direct the design process.

The inputs of the proposed approach are the WSDL file of the candidate Web service and ontologies from the repository and the output is an annotated WSDL file based on the SAWSDL notation. The annotation process is composed of five phases which are concept extraction, concept filtering and query filling, query execution, results assessment and SAWSDL annotation. The concept extraction, query execution and SAWSDL annotation are fully automatic processes while the concept filtering and query filling and result assessment phases are manual ones. The design of the three automatic phases was illustrated in detail in this chapter. The concept extraction mechanism utilises text analysis techniques of the GATE tool. The query execution engine employs name-based and structural matching mechanisms. Name-based matching is performed by a novel approach called CN-Match. CN-Match can perform effective and accurate name-based matching between labels containing multiple words. The SAWSDL annotator is designed

using text parsing and string look up techniques. The query filling phase makes use of a standard query template that is filled with data extracted from WSDL files.

Chapter 5 presented the design and evaluation of CN-Match which is the name-based matching approach implemented by the query execution engine. The chapter began by highlighting the significance of matching CNs in the ontology matching arena. Thereafter, a review of the structure and categories of CNs from a linguistic point of view was provided. Later, previous approaches that match CNs were discussed and their limitations were uncovered. To provide an appropriate design for CN-Match, design considerations and rules were identified and presented. Six design cases were defined and used by CN-Match: These cases were differentiated based on the number of possible constituents in any two candidate labels. String and linguistic based matching techniques were implemented by CN-Match.

To test CN-Match, its performance was evaluated using three published and well recognised test sets. Precision (P), Recall (R) and F-Measure (F) were the metrics used to evaluate CN-Match. The evaluation results highlighted significant conclusions that were presented later in the Discussion Section. Different P, R and F values were obtained by the different tests. These differences were due to two reasons: (1) The nature of the test data, the amount of domain specific matches contained in a specific test and the gold standard of this test; and (2) the underlying matching techniques, especially the linguistic one, that are implemented by CN-Match. Although the evaluation results were impacted by few CN-Match dependent and independent factors, these results were very promising and proved the effectiveness of CN-Match since high values of P, R and F were achieved.

Chapter 6 was dedicated to the evaluation of the proposed annotation approach. The chapter began by explaining the ontology extension approach and its implementation which was added to the functionality of the annotation

framework. Two extension methods were provided - one for simple queries and the other for complex queries -. Thereafter, three illustrative cases that show the annotation process in practice and explain the annotation steps in detail were presented. Later, the evaluation method, measures and results were presented. P, R, F and Percentage were utilised as evaluation metrics. For every Web service, these four metrics were measured before and after extension to show the effect of extension on the annotation process. Next, averages of P, R, F and Percentage were calculated for every domain of the five domains. These averages were then compared to find how the annotation results differe across domains. These differences provide significant implications about the proposed semi-automatic approach.

The conducted evaluation showed that the proposed approach is effective since nearly complete annotation results, in relation to manual results were obtained. Moreover, the evaluation highlighted the significance of the extension method for the annotation process. The extension mechanism allows the proposed annotation approach to overcome the Low Percentage Problem. Finally, few deficiencies of the annotation approach were provided which can be considered as motivations for future improvements to the current approach.

7.3 Research Conclusions and Contributions

The contributions made throughout this project are diverse and cover theoretical and practical facets. This research adds value to research and practice communities concerned with Web services, Semantic Web, SWS and ontology matching and extension. The novel integration of these research areas also enhances the value added by this research. The annotation approach is composed of phases that employ different components to perform the required annotation task. Each component in its own right provides granular contribution when considering its application to the SWS arena.

7.3.1 Contributions to the Knowledge Base

A. The Semi-automatic Annotation Method

This research adds to the knowledge base an annotation method that provides an effective solution to the problems associated with the manual annotation of Web services. This method is unique when compared with existing annotation approaches such as Patil et al. (2004) and Hepp (2006) since it combines and integrates innovative components together to facilitate the desired automation of annotation. These components are: (1) The concept extraction technique; (2) the query execution mechanism; (3) the SAWSDL annotation technique; and (4) the ontology extension mechanism. The new annotation method has the following advantages:

- **The automation of concept extraction:** In previous annotation approaches, the process of extracting required concepts for annotation from given WSDL files was manual. This manual extraction activity can be difficult and tedious especially when WSDL files of candidate services are relatively large. The annotation approach developed in this research automates the extraction process to facilitate easy and fast extraction of required WSDL concepts.
- **The usefulness and ease of use of queries:** The new annotation approach makes use of queries to eliminate the difficult and problematic process of manual or automatic ontology building. Retrospectively, manual ontology building is hard and error-prone and automatic ontology building by means of learning is ineffective. Using queries to capture and represent semantics of service elements is a pragmatic but useful approach. This is because instantiating queries from the Standard Query Template is easy to perform since it does not require much technical or domain knowledge.
- **The effectiveness of the query execution engine:** The annotation results that are outputs of query execution are encouraging since they are almost complete in relation to gold standards (See Figure 6.23). Queries are executed using an effective and novel query execution mechanism that combines name-based and structural matching. The name-based matching is performed by means of CN-Match which can automatically and precisely measure similarities

between labels containing different numbers of constituents. Results of CN-Match are precise because the linguistic structure of CNs is taken into account when measuring similarities. The evaluation results of CN-Match prove that it can provide clean and almost complete matching results (See Section 5.7.3). Structural matching searches for ontological classes that have related classes semantically similar to related concepts of a given query concept. The structural matching contributes to the completeness of results because it implements a balanced structural matching method: It is balanced because it utilises neither too few nor too many structural matching restrictions. This balance allows the execution engine to retrieve almost all correct matches. The latter result is proved by the high R values obtained in the evaluation of the annotation approach (See Figure 6.23). A more restricted structural matcher would likely impose more constraints on the matching process and thus may result in retrieving a lower number of correct results. Consequently, lower values of R may be obtained. Another advantage of structural matching is that it allows the execution engine to differentiate between homonyms (same spelling but different meaning) since it is unlikely that they have similar related elements. In addition, structural similarity enables the engine to find correspondences that have dissimilar labels but similar related concepts.

- **Significance of semi-automation of annotation:** The proposed approach is semi-automated because the 'concept filtering and query filling' and 'results assessments' are necessarily manual. The concept filtering should be performed by manual means since effective automatic filtering methods cannot be found. Such automatic methods would have to automatically recognise the concepts that should be excluded. We understand that the concept filtering process is performed to exclude the types of concepts presented earlier in Section 4.5 however; automatically detecting the concepts that belong to these categories is currently not possible. This is because concepts of each category do not necessarily share specific features that can be recognised by a computer algorithm. For example, the category of 'elements denoting processes' can include instances that have many different formats and use many different words. Additionally, the results assessment has to be

done by a human user because the employed matching techniques such as linguistic and string mechanisms are not perfect and have their understandable limitations. A fully automated approach that would select the correspondence with the highest score as the correct match would result in many missing and incorrect correspondences and hence a potentially lower P and R values. This is because it is not always necessary that the candidate with the highest confidence degree is the most appropriate one.

- **Importance of annotation to shared ontologies:** Annotation to shared ontologies is important because resulting services are pre-equipped with shared semantics that make the interoperability of these services with other services annotated to same or related ontologies much easier and effective. This is because, when services are annotated to diverse ontologies, automatic matching of these ontologies at run time becomes a prerequisite for future composition or interaction activities (Patel et al., 2004). Given the imperfect nature of existing automatic ontology matching techniques, the automatic matching may delay the required interaction and composition activities and be error-prone.

B. The CN-Match Method

CN-Match provides a novel CN similarity calculation method that takes the CN linguistic structure into consideration when performing matching. CN matching is very important for Web service annotation because many service elements and ontological entities have labels that are CNs. CN-Match design was motivated by the limitations of existing name-based matching methods. The existing methods cannot perform accurate and automatic matching between labels containing multiple words: This is because they ignore the linguistic structure of CNs. Therefore, it was necessary to review the types and structure of CNs using the linguistic knowledge base. The intersection of the CN linguistic literature and the ontology matching literature provides significant knowledge that enables the definition of a set of rules for CN and single terms matching. The rules were then employed to derive six design cases for CN-Match. CN-Match measures similarities between labels containing all three types (endocentric, exocentric and

copulative) of CNs. Exocentric and copulative CNs can be matched using linguistic and/or string matching. Using WordNet, meanings of exocentric and copulative CNs can be looked up. Endocentric CNs, which are the most common in the English Language, can be matched using the six pre-defined cases and the matching techniques utilised by CN-Match.

CN-Match is a useful matching approach: It provides appropriate matches that are not only identical or synonymous but also having semantically similar constituents. Semantic similarity is measured using the path-based similarity of WordNet. Path-based matching measures similarities between concepts by exploiting the lexical relations of the WordNet hierarchy.

C. The Ontology Extension Method

The provision of a method for semi-automatic ontology extension is a contribution to the existing body of knowledge. The provided method can be used not only for Web service annotation but also for other applications such as annotation of HTML pages, semantic query answering and knowledge-based systems. The ontology extension idea is very new and thus few extension approaches exist in the ontology literature (Ovchinnikova and Kühnberger, 2006). Most existing approaches extend ontologies used for semantic description of textual and static Web resources (Jung et al., 2009; Liu et al., 2005). Consequently, there is a need for ontology extension methods that can serve dynamic Web resources. This is because dynamic resources are different from static ones in terms of the nature of provided data and the way in which this data can be used for extension.

The proposed extension approach is different from other approaches such as Jung et al. (2009) and Liu et al. (2005) in that: (1) It adds not only classes but also object properties to extended ontologies; and (2) it performs an important check using CN-Match to find out if a similar concept to the candidate one exists in the ontology. If such a concept exists, then it will be used instead of adding a new

class. This later check process prevents any potential redundancies in extended ontologies.

7.3.2 Contributions to Practice

Interesting and important contributions of this research are the software prototypes that can be used by different practitioners in the SWS domain. These prototypes can improve the current practice in the Web service annotation and ontology matching domains. The most significant contributions to practice provided by this research are explained as follows:

A. The Semi-automatic annotation prototype

The novel annotation method is implemented in the Java 1.6.0 programming language to provide a useful utility to SWS developers and enable effective evaluation of the proposed method. The new annotation prototype facilitates a semi-automatic, accurate and easy to use annotation process. The evaluation results prove that the annotation prototype can effectively annotate Web service belonging to a wide range of domains: This makes the annotation approach very useful for any business or organisation that is interested in transforming their syntactic Web services to semantic ones.

Moreover, the new prototype utilises an effective semi-automatic ontology extension method. The extension method allows the proposed approach to overcome the Low Percentage Problem: This problem makes previous approaches ineffective since many service elements cannot be annotated. The new approach can provide annotation to those service elements that do not have appropriate correspondences in candidate ontologies by adding new and suitable ontological entities. Subsequently, a much higher percentage of service elements can be annotated. Overcoming the Low Percentage Problem is believed to improve the adoption of SWS by developers and industry.

B. CN-Match: The Name-based Matching Prototype

Name-based matching is one of the most significant matching criteria in the ontology matching domain. Therefore, there is a pressing need for effective and accurate matching tools especially those that can match labels composed of multiple words. CN-Match provides to the practice community a name-based matching prototype that can perform automatic and accurate similarity measurements between labels containing single terms and CNs. CN-Match can match endocentric, exocentric and copulative CNs. CN-Match prototype can be used by many applications that require effective name-based matching.

7.4 Meeting the Research Objectives

This section shows how this research successfully achieves the objectives formulated at the start of this project and presented in Section 1.2.

Objective 1: *"Analyse the previous Web service semi-automatic annotation approaches and study their limitations in order to derive a set of design requirements and strategies for the new approach".* Objective 1 was met in Chapter 2. SWS literature related to semantic annotation of Web services was reviewed in order to identify the approaches that perform semi-automatic annotation. Few approaches were found. These approaches were then categorised into; learning-based, workflow definition-based and matching-based approaches. Each category has a number of deficiencies that were identified. The limitations were studied carefully to discover avenues for improvements. The matching-based category was found promising because: (1) It allows sharing ontologies between different services; and (2) the existence of a family of matching approaches that can be improved, customised and integrated to achieve better and more accurate semi-automatic annotation. Later, the set of limitations that would be addressed were defined. Based on these limitations, a set of requirements for improvements were derived. Thereafter, a set of design strategies for the new approach were

identified based on the defined requirements. The limitations, requirements and strategies were then presented in Chapter 4.

Objective 2: *"Design the initial annotation framework based on the derived requirements and strategies and the analysis of WSDL general structure"*. Objective 2 was achieved in Chapters 3 and 4. Chapter 3 defined the research approach followed in this research, presented the research phases and detailed the research increments performed to design the desired annotation approach and its components. Later, WSDL general structure was analysed to specify the service elements that should be semantically described. The design strategies and the analysis results were then used to design the initial annotation framework and identify its phases and manual and automatic components. As significant parts of the annotation approach, the standard query template and the method of instantiating simple and complex queries were then defined.

Objective 3: *"Develop and test the automated components of the annotation approach".* This objective was met in Chapters 4, 5 and 6. After identifying the manual and automated components of the new annotation approach, the automatic components were developed and evaluated. The concept extraction mechanism was based on a set of text analysis techniques. The query execution engine was then designed: This engine employs name-based matching and structural matching techniques. Name-based matching is achieved by CN-Match which was designed and evaluated using three sets of existing ontologies to test its performance. Structural matching was then developed and implemented. Later, the two mechanisms were integrated in an appropriate manner where weights were assigned to each individual matching technique. To improve the automation of the annotation approach, a SAWSDL annotator was then developed to automatically annotate candidate services based on the SAWSDL format by adding a Model Reference element to tags of given elements. The annotator utilises text parsing and string look up in performing the automatic annotation. Finally, the components were integrated and tested together to make the required approach. The test showed that the approach required an effective and automatic ontology

extension method that can add appropriate correspondences for service elements that do not have suitable matches in the ontology repository. Subsequently, two novel ontology extension methods –one for simple queries and one for complex queries- were developed and used.

Objective 4: *"Evaluate the final annotation approach using appropriate evaluation methods, metrics and data".* This objective was accomplished in Chapter 6. Twenty five existing Web services that belong to five different domains were selected for this experimental evaluation. Five ontologies were also chosen to annotate the twenty five services. Precision (P), Recall (R), F-Measure and Percentage were used as evaluation metrics. Measuring P, R and F requires a gold standard. Since the twenty five services had never been annotated using the given set of ontologies, generating gold standard was required. Two gold standards –one for before extension and one for after extension- were generated for each service. Later, the values of P, R, F and Percentage were measured for each experiment.

Objective 5: *"Draw conclusions from the building and evaluation phases and identify future research directions that are important to continue refining and developing this significant area of research"*. This objective was met in Chapter 7. The design and evaluation phases allowed us to derive very important conclusions about the provided annotation approach. The proposed approach has many advantages in that; it is effective and easy to use. In addition, the evaluation provided few limitations of the new approach. The limitations were illustrated in Section 7.5. The provided limitations are good opportunities for identifying future research directions that can further improve the current annotation approach.

7.5 Limitations

Although the annotation approach presented in this thesis is argued to be effective and easy to use, the approach and its evaluation have some limitations which are explained in the following subsections.

7.5.1 Matching-dependent Limitations

The annotation approach makes use of a query execution engine which utilises name-based and structural matching mechanisms. The implemented name-based matching approach cannot always provide very accurate matching results. This is because there are some types of labels that cannot be matched using the provided matching technique. These categories are:

- **Ontological labels and service elements that have more than three constituents**: Unlike previous name-based matching approaches that can match single terms and binary CNs only, CN-Match can measure similarities between single terms, binary and triple CNs. Although, it makes a clear contribution, CN-Match cannot perform matching between labels composed of more than three constituents.
- **Domain-specific labels:** These labels have specific meanings in specific domains. Since CN-Match makes use of WordNet which is a domain-independent thesaurus, CN-Match cannot match some domain-specific labels. Consequently, some labels that are domain-dependent are unlikely to be correctly matched and annotated using the provided annotation method.
- **Labels denoting acronyms or parts of words:** CN-Match cannot always match service elements and ontological classes that are denoted by acronyms. This is because string and linguistic techniques implemented by CN-Match are unable to match many acronyms. String matching can match identical acronyms only. In addition, WordNet contains a very limited set of acronyms and thus similarities between many acronyms cannot be measured using WordNet. Moreover, CN-Match is sometimes unable to match a full word or

and expression against a part of a word or an expression that is composed of few letters of original words. Successfully matching a part of a word against its original word is very hard because a specific part could belong to many different words. Therefore, we argue that using parts of words in labels of classes and service elements should be avoided since they can result in much confusion and complications in the annotation process.

7.5.2 Annotation-dependent Limitations

The proposed approach can offer semantics for a subset of the synthesised elements defined in Section 2.4. These elements use XSD for their descriptions. These elements are: 'Input', 'Output', 'Precondition' and 'Effect'. The reason is that the input of the annotation approach is XSD data of WSDL files. Other elements such as 'Category', 'Non-functional Semantics', and 'Execution Semantics' are not annotated using the proposed approach because semantics required for their descriptions cannot be extracted from XSD or WSDL files. Providing an approach that can offer semi-automatic or automatic annotations for all elements of the synthesised set is a very hard and challenging task that requires multiple input data. Knowledge should then be extracted from given data in order to provide useful inputs for the annotation approach. Such knowledge requires advanced automatic or semi-automatic semantic extraction and modelling techniques which are currently hard to achieve due to the time constraints of this PhD project.

7.5.3 Extension-dependent Limitations

Ontologies extended by the proposed extension mechanism can miss some ontological structures. Although these missing structures may not have a direct impact on the annotation process, they could affect the software agents that would make use of extended ontologies for service discovery and composition purposes. These structures are missed because the extension mechanism is designed to add

classes and object properties in order to create correspondences for elements of simple and complex queries. An ontology can contain more axioms and structure than what is added by the developed extension mechanism. For instance, it could be argued that more axioms and structures such as super and subclasses, cardinality restrictions and data type properties should also be added by the extension method. Adding these extra ontological restrictions, however, requires more complicated rules to detect types of relations or axioms between candidate service concepts and existing or added ontological entities. Further, the addition of these structures could cause difficulties and delays to the extension and annotation processes.

7.5.4 Limitations of the Evaluation Metrics

The evaluations of the developed annotation approach and CN-Match use Precision (P), Recall (R) and F-Measure as metrics. These measures require the existence of gold standards against which automatic results are compared to identify correct-retrieved, incorrect-retrieved and correct-missing matches. Subsequently, the accuracy and completeness of "gold standards" is a key issue for the evaluation process. Unfortunately, the "gold standards" are currently created by manual means which makes them incomplete, subjective and error-prone. In addition, manual generation of these standards is very hard. Having such standards as baselines for measuring P, R and F values can affect the evaluation results and conclusions.

7.6 Future Work

The provided limitations offer significant opportunities for future research that can continue the development of the proposed annotation approach.

- **Improvements to the query execution engine:** The query execution engine presented in this thesis makes use of WordNet which is a domain-independent

English thesaurus. Future research can develop domain-dependent thesauri and use them for matching. Developing and employing such thesauri is, however, a hard task as creating domain-specific thesauri requires extensive domain knowledge. Further, employing such thesauri within an automatic and domain-independent matcher can introduce numerous difficulties since the domain of a given concept has to be discovered dynamically. Additionally, a single concept can belong to different domains and denote different meanings in different domains. Another improvement to the query execution engine could result from extending the design of CN-Match to be able to measure similarities between labels containing more than three constituents. Matching more complicated CNs is, however, not an easy task because the syntactic ambiguity of a CN increases as the number of its constituents grows.

- **Improvements to the ontology extension method:** The ontology extension method developed in this research can extend ontologies by adding classes and object properties only. Other important ontological components such as datatype properties and sub-super class relationships can also be added during extension. Adding these extra structures could, however, complicate and delay the annotation and extension activities and thus the feasibility of this addition should be studied carefully in real, practical and industrial settings.
- **The need for more effective gold standard generation methods:** The quality of gold standards is a key issue for calculating the values of P, R and F. Therefore, high quality of gold standards is desired. Creating such good standards requires the development of generation methods that are more effective, objective and accurate than the existing manual generation methods.
- **The integration of the annotation approach with Web service development tools:** The provided annotation approach can annotate existing Web services. Integrating the annotation approach with available Web service development tools can facilitate annotation that is simultaneous with the Web service development process. This simultaneous process would allow dynamic annotation of each new service element to an appropriate ontological class. Developing such an integrated annotation approach can improve the adoption

of SWS since resulting services would be pre-equipped with suitable semantic constructs that make them semantic services rather than syntactic ones.

- **The use of the Pragmatic Web to support service description:** SWS does not support the creation of adaptive and personalised Web service applications. This is because the semantic Web is unable to capture context-aware aspects of meaning (Liu, 2008) since ontologies can represent concepts, relations and axioms only. The pragmatic Web is a proposed solution to represent context-based meaning by using semiotics (symbols) (Singh, 2002). Therefore, future research will focus on combining the Semantic Web and the Pragmatic Web to improve service description.
- **The development of an effective ontology selection method:** The developed annotation approach uses a set of ontologies residing in a repository for annotation. Selecting the appropriate ontology from the repository for a given annotation task is currently performed by manual means. Developing and using an effective ontology selection mechanism could further simplify the annotation process.

References

Agarwal, S., Handschuh, S. & Staab, S. 2003, "Surfing the Service Web", *In Proceedings of the 2nd International Semantic Web Conference*, Springer, pp. 211-226.

Akkiraju, R. & Sapkota, B. 2007, 26 January 2007-last update, *Semantic Annotations for WSDL and XML Schema — Usage Guide* [Homepage of World Wide Web Consortium], [Online]. Available: http://www.w3.org/TR/2007/WD-sawsdl-guide-20070126/ [2008, 15 May].

AKT partners, 2010, *The Portal Ontology*. Available: http://www.aktors.org/ontology/ [2010, January/18].

Alfaries, A. 2010, *Ontology Learning for Semantic Web Services*, PhD edn, Brunel University, BURA.

Austin, D., Barbir, A., Ferris, C. & Garg, S. 2004, 11 February 2004-last update, *Web Services Architecture Requirements, W3C Working Group Note 11 February 2004* [Homepage of W3C], [Online]. Available: http://www.w3.org/TR/wsa-reqs/ [2009, 01 Sep].

Baskerville, R. 2008, "What Design Science is not", *European Journal of Information Systems*, vol. 17, no. 5, pp. 441-443.

Belhajjame, K., Embury, S.M., Paton, N.W., Stevens, R. & Goble, C. 2008, "Automatic annotation of Web Services Based on Workflow Definitions", *ACM Transactions on the Web*, vol. 2, no. 2, pp. 1-34.

Benatallah, B., Hacid, M., Leger, A., Rey, C. & Toumani, F. 2005, "On automating Web services discovery", *The International Journal on Very Large Data Bases*, vol. 14, no. 1, pp. 84-96.

Benbasat, I. & Zmud, R. 2003, "The Identity Crisis within the IS Discipline: Defining and Communicating the Discipline's Core Properties", *MIS Quarterly*, vol. 27, no. 2, pp. 183-194.

Beneventano, D., Bergamaschi, S., Guerra, F. & Vincini, M. 2003, "Synthesizing an Integrated Ontology", *IEEE Internet Computing*, vol. 7, no. 5, pp. 42-51.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001, "The Semantic Web", *Scientific American*, vol. 284, no. 5, pp. 34-43.

Bouquet, P., Serafini, L. & Zanobini, S. 2003, "Semantic Coordination: A New Approach and an Application", *Proceeding of 2nd the International Semantic Web Conference*, Springer, Berlin Heidelberg, pp. 130-145.

Brittenham, P., Curbera, F., Ehnebuske, D. & Graham, S. 2001, *Understanding WSDL in a UDDI registry, Part 1*, [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-wsdl/ [2009, 06/25].

Buckland, M. & Gey, F. 1994, "The Relationship between Recall and Precision", *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12-19.

Budanitsky, A. & Hirst, G. 2006, "Evaluating WordNet-based Measures of Lexical Semantic Relatedness", *Computational Linguistics*, vol. 32, no. 1, pp. 13-47.

Burns, P. 2006, *MorphAdorner: Morphological Adorner for English Text*.

Cardoso, J. 2006, "Approaches to Developing Semantic Web Services", *International Journal of Computer Science*, vol. 1, no. 1, pp. 8-21.

Cardoso, J. & Sheth, A. 2003, "Semantic E-Workflow Composition", *Journal of Intelligent Information Systems*, vol. 21, no. 3, pp. 191-225.

Castano, S., Ferrara, A. & Montanelli, S. 2006, "Matching Ontologies in Open Networked Systems: Techniques and Applications", *Journal on Data Semantic*, vol. 5, pp. 25-63.

Chen, Z., Liu, S., Wenyin, L., Pu, G. & Ma, W. 2003, "Building a web thesaurus from web link structure", *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, , pp. 48-55.

Cheng, H.K., Tang, Q.C. & Zhao, J.L. 2006, "Web Services and Service-Oriented Application Provisioning: An Analytical Study of Application Service Strategies", *IEEE Transactions on Engineering Management*, vol. 53, no. 4, pp. 520-533.

Chifu, V.R., Salomie, I. & Chifu, E.S. 2007, "Taxonomy Learning for Semantic Annotation of Web Services", *Proceedings of the 11th WSEAS International Conference on Computers*, ACM, pp. 300-305.

Choi, N., Song, I.Y. & Han, H. 2006, "A Survey on Ontology Mapping", *SIGMOD Record*, vol. 35, no. 3, pp. 34-41.

Chua, W.F. 1986, "Radical Developments in Accounting Thought", *The Accounting Review*, vol. 61, no. 4, pp. 601-632.

Cohen, W., Ravikumar, P. & Fienberg, S. 2003, "A comparison of string metrics for matching names and records", *Proceedings of the workshop on Data Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining*, pp. 1-6.

Cunningham, h., Maynard, D., Bontcheva, K. & Tablan, V. 2002, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications", *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, pp. 1-8.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. & Weerawarana, S. 2002, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing*, vol. 6, no. 2, pp. 86-93.

De Nicola, A., Missikoff, M. & Navigli, R. 2009, "A Software engineering approach to ontology building", *Information Systems*, vol. 34, no. 2, pp. 258-275.

Devedzic, v. 2002, "Understanding Ontological Engineering", *Communications of the ACM*, vol. 45, no. 4, pp. 136-144.

Do, H.H. & Rahm, E. 2002, "COMA: A System for Flexible Combination of Schema Matching Approaches", *Proceedings of the 28th International Conference on Very Large Data Bases* ACM, pp. 610-612.

Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B. & Pedrinaci, C. 2008, "IRS-III: A broker-based approach to semantic Web services", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 2, pp. 109-132.

Downing, P. 1977, "On the Creation and Use of English Compound Nouns", *Language*, vol. 53, no. 4, pp. 810-842.

Duo, Z., Juan-Zi, L. & Bin, X. 2005, "Web Service Annotation Using Ontology Mapping", *Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*, IEEE, pp. 235-242.

Dustdar, S. & Schreiner, W. 2005, "A Survey on Web Services Composition", *International Journal on Web and Grid Services*, vol. 1, no. 1, pp. 1-30.

Ehrig, M. 2005, *Framework for Ontology Alignment and Mapping Test Ontologies and Alignments*. Available: http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies.htm [2009, 12/03].

Ehrig, M. & Euzenat, J. 2005, "Relaxed precision and recall for ontology matching", *In Proceedings of the K-CAP Workshop on Integrating Ontologies (IntOnt)*, pp. 25-33.

Ehrig, M. & Staab, S. 2004, "QOM - Quick Ontology Mapping", *Proceeding of the Third International Semantic Web Conference*, Springer Berlin, Heidelberg, pp. 683- 711.

Ehrig, M. & Sure, Y. 2005, "FOAM-framework for ontology alignment and mapping; results of the ontology alignment initiative", *Proceedings of the Workshop on Integrating Ontologies*, pp. 72-76.

Euzenat, J. 2004, "An API for Ontology Alignment", *Lecture Notes in Computer Science*, , no. 3298, pp. 698-712.

Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Trojahn, C., Vouros, G. & Wang, S. 2009, "Results of the Ontology Alignment Evaluation Initiative 2009", , pp. 1-54.

Euzenat, J. & Shvaiko, P. 2007, *Ontology Matching*, Springer Berlin Heidelberg, New York.

Euzenat, J. & Valtchev, P. 2004, "Similarity-based ontology alignment in OWL-lite", *Proceedings of the European Conference on Artificial Intelligence*, pp. 333-338.

Feier, C., Roman, D., Polleres, A., Domingue, J. & Fensel, D. 2005, "Towards Intelligent Web Services: The Web Service Modelling Ontology", *International Conference on Intelligent Computing*, pp. 1-10.

Ferris, C. & Farrell, J. 2003, "What Are Web Services?", *Communications of the ACM*, vol. 46, no. 6, pp. 31-31.

Finin, T. 1980, "The Semantic Interpretation of Nominal Compounds", AIII Press, pp. 310-312.

Fonseca, F. 2007, "The double role of ontologies in information science research: Research Articles", *Journal of the American Society for Information Science and Technology*, vol. 58, no. 6, pp. 786-793.

Gilleland, M. 2009, *Levenshtein Distance, in Three Flavours*. Available: http://www.merriampark.com/ld.htm [2009, November].

Girju, R., Moldovanb, D., Tatub, M. & Antoheb, D. 2005, "On the Semantics of Noun Compounds", *Computer Speech & Language*, vol. 19, no. 4, pp. 479-496.

Giunchiglia, F. & Shvaiko, P. 2004, "Semantic matching", *The Knowledge Engineering Review*, vol. 18, no. 3, pp. 265-280.

Giunchiglia, F., Shvaiko, P. & Yatskevich, M. 2004, "S-Match: an algorithm and an implementation of semantic matching", *Proceedings of the European Web Symposium*, Springer, pp. 61-75.

Giunchiglia, F., Shvaiko, P. & Yatskevich, M. 2006, "Discovering Missing Background Knowledge in Ontology Matching", ACM, pp. 382-389.

Giunchiglia, F., Yataskevich, M., Avesani, P. & Shvaiko, P. 2009, "A Large Dataset for the Evaluation of Ontology Matching", *The Knowledge Engineering Review*, vol. 24, no. 2, pp. 137-157.

Go, K. & Carroll, J.M. 2004, "The blind men and the elephant: views of scenario-based system design", *ACM Interactions*, vol. 11, no. 6, pp. 44-53.

Gómez-Pérez, A. & Manzano-Macho, D. 2004, "An overview of methods and tools for ontology learning from texts", *The Knowledge Engineering Review*, vol. 19, no. 3, pp. 187-212.

Gregor, S. & Jones, D. 2007, "The Anatomy of a Design Theory", *Journal of the Association for Information Systems*, vol. 8, no. 5, pp. 312-335.

Grobelnik, M., Mladenic, D. & Fortuna, B. 2009, "Semantic Technology for Capturing Communication Inside an Organization", *IEEE Internet Computing*, vol. 13, no. 4, pp. 59-67.

Gruber, T.R. 1993, "A Translation Approach to Portable Ontology Specification", *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220.

Gruber, T.R. 1995, "Toward principles for the design of ontologies used for knowledge sharing", *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907-928.

Gruninger, M. & Fox, M.S. 1995, "Methodology for the design and evaluation of ontologies", *In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 1-10.

Guarino, N. 1998, "Formal Ontology and Information Systems", *Formal Ontology in Information Systems*, pp. 3-15.

Hasselbring, W. 2000, "Information System Integration", *Communications of the ACM*, vol. 43, no. 6, pp. 32-38.

Hepp, M. 2006, "Semantic Web and semantic Web services: father and son or indivisible twins?", *IEEE Internet Computing*, vol. 10, no. 2, pp. 85-88.

Heß, A., Johnston, E. & Kushmerick, N. 2004, "ASSAM: A Tool for Semi-automatically Annotating Semantic Web Services", *The Semantic Web*, Springer Berlin, Heidelberg, pp. 320-324.

Heß, A. & Kushmerick, N. 2003, "Learning to Attach Semantic Metadata to Web Services", *The Semantic Web*, Springer Berlin, Heidelberg, pp. 258-273.

Hevner, A.R., March, S.T. & Park, J. 2004, "Design Science in Information Systems Research", *MIS Quarterly*, vol. 28, no. 1, pp. 75-105.

Hewlett-Packard Development Company 2009, *ARQ 2.8.0* [Homepage of Sourceforge.net], [Online]. Available: http://jena.sourceforge.net/ARQ/license.html [2009, 12/20].

Horrocks, I. 2003, *The Knowledge Acquisition Ontology*. Available: http://www.daml.org/ontologies/398 [2010, January/20].

Huhns, M.N. & Singh, M.P. 2005, "Service-oriented computing: key concepts and principles", *IEEE Internet Computing*, vol. 9, no. 1, pp. 75-81.

Isakowitz, T., Bieber, M. & Vitali, F. 1998, "Web information systems", *Communications of the ACM*, vol. 41, no. 7, pp. 78-80.

Jacek, K., Tomas, V., Carine, B. & Joel, F. 2007, "SAWSDL: Semantic Annotations for WSDL and XML Schema", *IEEE Internet Computing*, vol. 11, no. 6, pp. 60-67.

Janev, V. & Vranes, S. 2009, "Semantic Web Technologies: Ready for Adoption?", *IT Professional*, vol. 11, no. 5, pp. 8-16.

Jasper, R. & Uschold, M. 1999, "A Framework for Understanding and Classifying Ontology Applications", *Workshop on Ontologies and Problem-Solving Methods*, pp. 1-11.

Jiang, X. & Tan, A. 2010, "CRCTOL: A Semantic-based Domain Ontology Learning System", *Journal of the American Society for Information Science and Technology*, vol. 61, no. 1, pp. 150-168.

Jung, J., Oh, K. & Jo, G. 2009, "Extracting Relations towards Ontology Extension", *In Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, ACM, , pp. 242-251.

Kalfoglou, Y. & Schorlemmer, M. 2003, "Ontology mapping: the state of the art", *The Knowledge Engineering Review*, vol. 18, no. 1, pp. 1-31.

Khalaf, R. & Leymann, F. 2003, "On Web Services Aggregation", *Proceeding of the VLDB Technologies for e-Services Workshop*, Springer LNCS, pp. 1-13.

Kim, S.N. & Baldwin, T. 2005, "Automatic interpretation of noun compounds using WordNet similarity", pp. 945-957.

Kuechler, B. & Vaishnavi, V. 2008, "On Theory Development in Design Science Research: Anatomy of a Research Project", *European Journal of Information Systems*, vol. 17, no. 5, pp. 489-504.

Kuechler, B., Vaishnavi, V. & Petter, S. 2005, "The Aggregate General Design Science Cycle as a Perspective on the Evolution of Computing Communities of Interest", *Computing Letters*, vol. 1, no. 3, pp. 123-128.

Land, F. 1992, "The Information Systems Domain", *Information Systems Research: Issues, Methods, and Practical Guidelines*, ed. R. Galliers, Blackwell Scientific Publications, Oxford, pp. 6-13.

Lapata, M. 2002, "The Disambiguation of Nominalizations", *Computational Linguistics*, vol. 28, no. 3, pp. 357-388.

Lara, R., Roman, D., Polleres, A. & Fensel, D. 2004, "A Conceptual Comparison of WSMO and OWL-s", *In Proceedings of the European Conference on Web Services (ECOWS 2004)*Springer Berlin, Heidelberg, pp. 254-269.

Lauer, M. 1995, *Designing Statistical Language Learners: Experiments on Noun Compounds*, Macquarie University.

Lei, Z., Xiaoying, Y., Yanni, Y. & Bo, S. 2008, "An Improved Semantic Annotation Method of Web Services Based on Ontology", *In Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control and Management* ACM, pp. 580-584.

Lerman, K., Plangrasopchok, A. & Knoblock, C.A. 2006, "Automatically labelling the inputs and outputs of web services", *In Proceedings of the National Conference on Artificial Intelligence* AAAI, Charlotte, USA, pp. 1363-1369.

Levenshtein, V.I. 1965, "Binary Codes Capable of Correcting Spurious and Deletions of Ones", *Problems of Information Transmission*, vol. 1, no. 1, pp. 8-17.

Lin, F. & Sandkuhl, K. 2008, "A Survey of Exploiting WordNet in Ontology Matching", Springer Boston, pp. 341-350.

Lindberg, D., Humphreys, B. & McCray, A. 1993, "The Unified Medical Language System", *Methods of Information in Medicine*, vol. 32, no. 4, pp. 281-291.

Liping, Z., Guangyao, L., Yongquan, L. & Jing, S. 2007, "Design of ontology mapping framework and improvement of similarity computation", *Journal of Systems Engineering and Electronics*, vol. 18, no. 3, pp. 641-645.

Liu, K. (2008) "Pragmatic Computing - A Semiotic Perspective to Web Services", *Proceedings of the 2007 International joint Conference on e-Business and Telecommunications* Springer, pp. 3 - 15.

Liu, W., Weichselbraun, A., Scharl, A. & Chang, E. 2005, "Semi-Automatic Ontology Extension Using Spreading Activation", *Journal of Universal Knowledge Management*, vol. 0, no. 1, pp. 50-58.

Madhavan, J., Bernstein, P.A. & Rahm, E. 2001, "Generic Schema Matching with Cupid", *Proceedings of the 27th International Conference on Very Large Data Bases* ACM, pp. 49-64.

March, S.T. & Smith, G.F. 1995, "Design and Natural Science Research on Information Technology", *Decision Support Systems*, vol. 15, no. 4, pp. 251-266.

March, S.T. & Storey, V.C. 2008, "Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research", *MIS Quarterly*, vol. 32, no. 4, pp. 725-730.

Markus, M.L., Majchrzak, A. & Gasser, L. 2002, "A Design Theory for Systems that Support Emergent Knowledge Processes", *MIS Quarterly*, vol. 26, no. 3, pp. 179-212.

Martin, D., Burstein, M.H., McDermott, D., McIlraith, S.A., Paolucci, M., Sycara, K., McGuinness, D.L., Sirin, E. & Srinivasan, N. 2007, "Bringing Semantics to Web Services with OWL-S", *World Wide Web*, vol. 10, no. 3, pp. 243-277.

Martin, D. & Domingue, J. 2007, "Semantic Web Services, Part 1", *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 12-17.

Mascardi, V., Locoro, A. & Rosso, P. 2009, "Automatic Ontology Matching Via Upper Ontologies: A Systematic Evaluation", *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1-14.

McIlraith, S.A., Son, T.C. & Zeng, H. 2001, "Semantic Web Services", *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46-53.

Miller, G.A. 1995, "WordNet: a lexical database for English", *Communications of the ACM*, vol. 38, no. 11, pp. 39-41.

Missikoff, M., Navigli, R. & Velardi, P. 2002, "Integrated Approach to Web Ontology Learning and Engineering", *IEEE Computer*, vol. 35, no. 11, pp. 60-63.

Nagarajan, M. 2006, "Semantic Annotations in Web Services" in *Semantic Web Services, Processes and Applications* Springer US, pp. 35-61.

Nagy, M., Vargas-Vera, N. & Stolarski, P. 2009, "DSSim results for OAEI 2009", *In Proceedings of the 2009 Ontology Alignment Evaluation Initiative*, pp. 1-10.

Narayanan, S. & McIlraith, S.A. 2002, "Simulation, Verification and Automated Composition of Web Services", *Proceedings of the 11th International conference on World Wide Web*, pp. 77-88.

Noy, N. 2004, "Semantic integration: a survey of ontology-based approaches", *SIGMOND Record*, vol. 33, no. 4, pp. 65-70.

Noy, N. & Musen, M. 2000, "PROMPT: algorithm and tool for automated ontology merging and alignment", *Proceeding of the 17th National Conference on Artificial Intelligence*, pp. 1-6.

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

Orlikowski, W.J. & Baroudi, J. 1991, "Studying Information Technology in Organizations: Research Approaches and Assumptions", *Information Systems Research*, vol. 2, no. 1, pp. 1-28.

Orlikowski, W.J. & Iacono, C.S. 2001, "Research Commentary: Desperately Seeking the 'IT' in IT Research – A Call to Theorizing the IT Artefact", *Information Systems Research*, vol. 12, no. 2, pp. 121-134.

Ovchinnikova, E. & Kühnberger, K. 2006, "Aspects of Automatic Ontology Extension: Adapting and Re-generalising Dynamic Updates", *In Proceedings of the Second Australasian Workshop on Advances in Ontologies*, ACM, pp. 51-60.

Pahi, C. & Zhu, Y. 2006, "A Semantical Framework for the Orchestration and Choreography of Web Services", *Electronic Notes in Theoretical Computer Science*, vol. 151, no. 2, pp. 3-18.

Paolucci, M., Kawamura, T.R., Payne, T.R. & Sycara, K. 2002, "Importing the Semantic Web in UDDI" in *Web Services, E-Business and The Semantic Web* Springer Berlin / Heidelberg, pp. 815-821.

Papazoglou, M.P. 2003, "Service -Oriented Computing: Concepts, Characteristics and Directions", *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, IEEE Computer Society, pp. 3-12.

Papazoglou, M.P. & Georgakopoulos, D. 2003, "Service -Oriented Computing", *Communications of the ACM*, vol. 46, no. 10, pp. 25-28.

Papazoglou, M.P., Traverso, P., Dustdar, S. & Leymann, F. 2007, "Service -Oriented Computing: State of the Art and Research Challenges", *Computer*, vol. 40, no. 11, pp. 38-45.

Pathak, J., Koul, N., Caragea, D. & Honavar, V.G. 2005, "A framework for semantic Web service discovery", *Proceedings of the 7th annual international workshop on Web information and data management*, pp. 45-51.

Patil, A., Oundhakar, S., Sheth, A. & Verma, K. 2004, "Meteor-s web service annotation framework", *Proceedings of the 13th international conference on World Wide Web* ACM, New York, USA, pp. 553-562.

Pinto, H.S. & Martins, J.P. 2004, "Ontologies: How can They be Built?", *Knowledge and Information Systems*, vol. 6, no. 4, pp. 441-464.

Plag, I. 2003, *Word Formation in English*, 1st edn, Cambridge University Press.

Porter, M.F. 2006, "An Algorithm for Suffix Stripping", *Program*, vol. 40, no. 3, pp. 211-218.

Prud'Hommeaux, E. & Seaborne, A. 2008, , *SPARQL Query Language for RDF* [Homepage of W3C], [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/ [2009, 12/15].

Rahm, E. & Bernstein, P.A. 2001, "A survey of approaches to automatic schema matching", *The International Journal on Very Large Data Bases*, vol. 10, no. 4, pp. 334-350.

Rajasekaran, P., Miller, J., Verma, K. & Sheth, A. 2005, "Enhancing Web Services Description and Discovery to Facilitate Composition", *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition* Springer Berlin, Heidelberg, pp. 55-68.

Ringelstein, C., Franz, T. & Staab, S. 2007, "The Process of Semantic Annotation of Web Services" in *Semantic Web Service: Theory Tools and Application*, pp. 217-239.

Rodriguez, M.A. & Egenhofer, M.J. 2003, "Determining Semantic Similarity among Entity Classes from Different Ontologies", *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 2, pp. 442-456.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. & Fensel, D. 2005, "Web Service Modelling Ontology", *Applied Ontology*, vol. 1, no. 1, pp. 77-106.

Sánchez, D. 2010, "A methodology to learn ontological attributes from the Web", *Data & Knowledge Engineering*, vol. 69, no. 6, pp. 573-597.

Sbodio, M.L., Martin, D. & Moulin, M. 2010, "Discovering Semantic Web Services Using SPARQL and Intelligent Agents", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. Article In Press.

Sheth, A. & Larson, J. 1990, "Federated database systems for managing distributed, heterogeneous, and autonomous databases", *ACM Computing Surveys*, vol. 22, no. 3, pp. 183-236.

Shvaiko, P. & Euzenat, J. 2005, "A Survey of Schema-Based Matching Approaches", *Journal on Data Semantic*, pp. 146-171.

Shvaiko, P. & Euzenat, J. 2008, "Ten challenges for ontology matching", *Proceeding of the 7th International Conference on Ontologies, Databases and Applications of Semantics* Springer Berlin, pp. 1164-1182.

Simon, H.A. 1996, *The Sciences of the Artificial*, 3rd edn, MIT Press, Cambridge, MA.

Singh, M. (2002) "The Pragmatic Web", *IEEE Internet Computing*, vol. 6, no. 3, pp. 4-5.

Sivashanmugam, K., Sheth, A., Miller, J., Verma, K., Aggarwal, R. & Rajasekaran, P. 2003, "Metadata and Semantics for Web Services and Processes", *In Datenbanken und Information systeme (Databases and Information Systems)*, pp. 1-19.

Sivashanmugam, K., Verma, K., Sheth, A. & Miller, J. 2003, "Adding Semantics to Web services Standards", *Proceedings of the International Conference on Web Services*, pp. 1-7.

Sorrentino, S., Bergamaschi, S., Gawinecki, M. & Po, L. 2009, "Schema Normalization for Improving Schema Matching", Lecture Notes In Computer Science, pp. 280-293.

Spyns, P., Meersman, R. & Jarrar, M. 2002, "Data modelling versus Ontology engineering", *ACM SIGMOD Record*, vol. 31, no. 4, pp. 12-17.

Staab, S. 2003, "Web services: been there, done that?", *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 72-85.

Staab, S. 2004, "Why Evaluate Ontology Technologies? Because it Works!", *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 74-81.

Stal, M. 2002, "Web Services: Beyond Component -Based Computing", *Communications of the ACM*, vol. 45, no. 10, pp. 71-76.

Su, X. & Gulla, J.A. 2004, "Semantic Enrichment for Ontology Mapping", Springer, pp. 217-228.

Su, X. & Gulla, J.A. 2006, "An Information Retrieval Approach to Ontology Mapping", *Data & Knowledge Engineering*, vol. 58, no. 1, pp. 47-69.

Sycara, K., Paolucci, M., Ankolekar, A. & Srinivasan, N. 2003, "Automated discovery, interaction and composition of Semantic Web services", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 1, pp. 27-46.

Tagarelli, A., Longo, M. & Greco, S. 2009, "Word Sense Disambiguation for XML Structure Feature Generation", Springer, pp. 143-157.

The Open Directory Project 2010, *The Open Directory Project Travel Directory*. Available: http://www.dmoz.org/Recreation/Travel/ [2010, 01/15].

Vaishnavi, V. & Kuechler, W. 2004, January 20, 2004-last update, *Design Research in Information Systems* [Homepage of Association for Information Systems], [Online]. Available: http://desrist.org/design-research-in-information-systems [2009, August/ 01].

Verma, K. & Sheth, A. 2007, "Semantically Annotating a Web Service", *IEEE Internet Computing*, vol. 11, no. 2, pp. 83-85.

Vitvar, T., Mocan, A., Kerrigan, M., Zaremba, M., Zaremba, M., Moran, M., Cimpian, E., Haselwanter, T. & Fensel, D. 2007, "Semantically-enabled service oriented architecture: concepts, technology and application", *Service Oriented Computing and Applications*, vol. 1, no. 2, pp. 129-154.

Wack, P. 1985, "Scenarios: Uncharted Waters Ahead", *Harvard Business Review*, pp. 1-17.

Walsham, G. 1993, *Interpreting Information Systems in organisations*, 1st edn, Wiley series in Information Systems, Chichester.

Wei, W., Barnaghi, P. & Bargiela, A. 2010, "Probabilistic Topic Models for Learning Terminological Ontologies", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 1028-1040.

Winter, R. (2008) 'Design Science Research in Europe'. *European Journal of Information Systems*, vol. 17, no. 5, pp. 470-475.

Wu, Z., Gomadam, K., Ranabahu, A., Sheth, A. & Miller, J. 2007, "Automatic Composition of Semantic Web Services using Process and Data Mediation", *Proceedings of the 9th International Conference on Enterprise Information Systems*, pp. 453-462.

Wu, Z. & Palmer, M. 1994, "Verbs Semantics and Lexical Selection", *In Proceedings of the 32nd annual meeting on Association for Computational Linguistics* ACM, pp. 133-138.

Yahoo! 2010, *The Yahoo Travel Directory*. Available: http://dir.yahoo.com/Recreation/Travel/ [2010, 01/20].

Yeganeh, S.H., Habibi, J., Rostami, H. & Abolhassani, H. 2010, "Semantic Web Service Composition Tested", *Computers and Electrical Engineering*, vol. 36, no. 5, pp. 805-817.

Zaremba, M. & Bussler, C. 2005, "Towards Dynamic Execution Semantics in Semantic Web Services", *Workshop on Web Service Semantics: Towards Dynamic Business Integration*, pp. 1-9.

Zhang, M., Duan, Z. & Zhao, C. 2008, "Semi-automatically annotating data semantics to Web services using ontology Mapping", *Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design*, IEEE Computer Society, pp. 470-475.

Zhao, Y., Wang, X. & Halang, W. 2006, "Ontology Mapping based on Rough Formal Concept Analysis", *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, IEEE Computer Society, pp.1-7.

Zhou, L. 2007, "Ontology learning: state of the art and open issues", *Information Technology and Management*, vol. 8, no. 3, pp. 241-252.

Zouaq, A. & Nkambou, R. 2008, "Building Domain Ontologies from Text for Educational Purposes", *IEEE Transactions on Learning Technologies*, vol. 1, no. 1, pp. 49-62.

Zuniga, G.L. 2001, "Ontology: its transformation from philosophy to information systems", *Proceedings of the international conference on Formal Ontology in Information Systems* ACM, pp. 187-197.

Appendix A
Concept Extraction Code

Complex Relation

```
Phase: ComplexRelation
Input: Token Tokens COTag InnerElement SeqTag
Options: control = applet

Macro: EndElementTag

 (
 {Token.string == "type"}
 {Token.string == "="}
 {Token.kind == punctuation}
 {Token.kind == word}
 {Token.kind == punctuation}
 {Token.kind == word}
 {Token.kind == punctuation}
 )
Rule: fullElementTag
Priority: 50
(
	//This looks for the complexType of <ComplexType name =
	(
	(
		(
			{COTag}
			{Token.string == "name"}
			{Token.string == "="}
			{Token.kind == punctuation}
		)
	(
			{Token.kind == word}
		):className
		(
			{Token.kind == punctuation}
			{Token.string == ">"}
		)
	)
	|
	// This looks for the other complexType when the element tag comes before
	((
			{Token.string == "<"}
			{Token.kind == word}
			{Token.string == ":"}
			{Token.string == "element"}
			{Token.string == "name"}
			{Token.string == "="}
			{Token.kind == punctuation}
		)
		(
			{Token.kind == word}
		):className
		(
```

```
                        {Token.kind == punctuation}
                        {Token.string == ">"}
                )
                (
                        {COTag}
                        {Token.string == ">"}
                )
        )
        )
        ({SeqTag})

        // This looks for the first element.
        ((
                        {InnerElement}
                        (
                                {Token.kind == word}

                        ):elementAtt1
                        {Token.kind == punctuation}
                        (EndElementTag)?
                        ({Token.string == "/"})
                        ({Token.string == ">"})
                ) ?
        ):att1
        // This looks for another element.
        ((
                        {InnerElement}
                        (
                                {Token.kind == word}
                        ):elementAtt2
                        {Token.kind == punctuation}
                        (EndElementTag)?
                        ({Token.string == "/"})
                        ({Token.string == ">"})
                ) ? ):att2

        // This looks for another element.
        ((
                        {InnerElement}
                        (
                                {Token.kind == word}
                        ):elementAtt3
                        {Token.kind == punctuation}
                        (EndElementTag)?
                        ({Token.string == "/"})
                        ({Token.string == ">"})
                ) ? ):complexRelation
-->
:complexRelation.ComplexRelaion = {rule = "ComplexRelation", Class =
:className.Token.string , Attribute=:elementAtt1.Token.string },
:att1.Att1 ={ Class = :className.Token.string, Attribute=:elementAtt1.Token.string},
:att2.Att2 ={ Class = :className.Token.string, Attribute=:elementAtt2.Token.string}
```

Simple Element

Phase: simpleElement
Input: Token Tokens InnerElement COTag

```
Options: control = applet

Macro: EndingTag
 (
 {Token.string == "type"}
 {Token.string == "="}
 {Token.kind == punctuation}
 {Token.kind == word}
 {Token.kind == punctuation}
 {Token.kind == word}
 {Token.kind == punctuation}
 )

Rule: SimpleElement
Priority: 50
(
        {InnerElement}
        ({Token.kind == word}): SE
        {Token.kind == punctuation}
        (EndingTag)?
        {Token.string == "/"}
        {Token.string == ">"}
)
-->
:SE.SimpleElement = {rule = "SimpleElement"}
```

Complex Type

```
Phase: complexOpen
Input: Token Tokens
Options: control = applet
//defines opening tag till the name including "
Rule: complexOpenning
Priority: 50
(
({Token.string == "<"})
({Token.kind == word})
({Token.string == ":"})
({Token.string == "complexType"})
): complexOpenningTag
-->
:complexOpenningTag.COTag = {rule = "complexOpen"}
```

Appendix B

Examples of CN-Match Code

Single Term and Binary CN Matching

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import net.didion.jwnl.JWNLException;
import org.semanticweb.owl.align.AlignmentException;
public class SingleWordAndCompoundMatching {
   double score = 0.0;
   public SingleWordAndCompoundMatching()
   {}
   public double performSingleAndCompoundMatching(String label1, String label2) throws
IOException, JWNLException, FileNotFoundException, AlignmentException
   {
      ArrayList<String> list1 = new ArrayList<String>();
      ArrayList<String> list2 = new ArrayList<String>();
      ArrayList<String> listOfC21C22 = new ArrayList<String>();
      CheckingWordNetEntryOfSingleWords checkingWordNetEntryOfSingleWords = new
CheckingWordNetEntryOfSingleWords();
      CheckingWordNetEntryOfCompounds checkingWordNetEntryOfCompounds = new
CheckingWordNetEntryOfCompounds();
      SingleWordsMatching singleWordsMatching = new SingleWordsMatching();
      LinguisticSimilarity linguisticSimilarity = new LinguisticSimilarity();
      Tokenizer tokenizer = new Tokenizer();
      list1 = tokenizer.tokenize(label1);
      list2 = tokenizer.tokenize(label2);
      String C11, C21C22;
      double score1 = 0.0, score2 = 0.0;
      if(list1.size()==2)
      {
         C21C22 = list1.get(0) + list1.get(1);
         listOfC21C22 = list1;
         C11 = list2.get(0);
      }
      else //list2 size is 2
      {
         C21C22 = list2.get(0) + list2.get(1);
         listOfC21C22 = list2;
         C11 = list1.get(0);
      }
      boolean C11Entry = checkingWordNetEntryOfSingleWords.checkingEntry(C11);
      boolean C21C22Entry =
checkingWordNetEntryOfCompounds.checkingEntryOfCompounds(listOfC21C22);
      String C21 = listOfC21C22.get(0);
      String C22 = listOfC21C22.get(1);
      /* If C21C22 has entry,
      *       check if C11 has entry, if yes, perform linguistic matching between C11 and C21C22
and get score */
      if (C21C22Entry)
      {
         if (C11Entry)
         {
```

```
            score = linguisticSimilarity.performLinguisticSimilarity(C11, C21C22);
            System.out.println("Score= " + score);          }       }
            /*if C21C22 does not have entry,
         *     match C11 with C21 using singleWordsMatching and get score1. Score = 0.2*score1.
         *     match C11 with C22 using singleWordsMatching and get score2  Score = 0.8*score2.
         */
      else
      {
         score1 = singleWordsMatching.performSingleWordsMatching(C11, C21);
         score2 = singleWordsMatching.performSingleWordsMatching(C11, C22);
         if (score2 >= score1)
         {
            score = 0.8*score2;
            System.out.println("Score2= " + score2);
         }
         else
         {
            score = 0.2*score1;
            System.out.println("score1= " + score1);
         }  }
      return score;
   }}
```

Single Term and Triple CN Matching

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import net.didion.jwnl.JWNLException;
import org.semanticweb.owl.align.AlignmentException;

/*
 * Single Word Label:  C11
Triple Compound Label: C21C22C23
 */
public class SingleWordAndTripleCompoundMatching {

   double score = 0.0;

   public SingleWordAndTripleCompoundMatching()
   {}

   public double performSingleWordAndTripleCompound(String label1, String label2) throws
IOException, JWNLException, FileNotFoundException, AlignmentException
   {

      ArrayList<String> list1 = new ArrayList<String>();
      ArrayList<String> list2 = new ArrayList<String>();
      ArrayList<String> listOfC21C22C23 = new ArrayList<String>();
      Tokenizer tokenizer = new Tokenizer();
      String C11, C21C22C23;
      list1 = tokenizer.tokenize(label1);
      list2 = tokenizer.tokenize(label2);
      double score1 = 0.0, score2 = 0.0, score31 = 0.0, score32 = 0.0, score33 = 0.0;
      int indicator = -1;
      CheckingWordNetEntryOfSingleWords checkingWordNetEntryOfSingleWords = new
CheckingWordNetEntryOfSingleWords();
```

```
CheckingWordNetEntryOfCompounds checkingWordNetEntryOfCompounds = new
CheckingWordNetEntryOfCompounds();
SingleWordsMatching singleWordsMatching = new SingleWordsMatching();
LinguisticSimilarity linguisticSimilarity = new LinguisticSimilarity();
/* C11
 * C21C22C23 */
if (list1.size() ==3)
{
   listOfC21C22C23 = list1;
   C21C22C23 = label1;
   C11 = label2;
}
// Label2 is a Triple Compound
else
{
   listOfC21C22C23 = list2;
   C21C22C23 = label2;
   C11 = label1;
}
String C21C22 = listOfC21C22C23.get(0) + listOfC21C22C23.get(1);
String C22C23 = listOfC21C22C23.get(1) + listOfC21C22C23.get(2);
String C21 = listOfC21C22C23.get(0);
String C22 = listOfC21C22C23.get(1);
String C23 = listOfC21C22C23.get(2);
ArrayList<String> listOfC21C22 = new ArrayList<String>();
ArrayList<String> listOfC22C23 = new ArrayList<String>();
listOfC21C22.add(0, listOfC21C22C23.get(0));
listOfC21C22.add(1, listOfC21C22C23.get(1));
listOfC22C23.add(0, listOfC21C22C23.get(1));
listOfC22C23.add(1, listOfC21C22C23.get(2));

boolean C21C22Entry =
checkingWordNetEntryOfCompounds.checkingEntryOfCompounds(listOfC21C22);
boolean C22C23Entry =
checkingWordNetEntryOfCompounds.checkingEntryOfCompounds(listOfC22C23);
boolean C11Entry = checkingWordNetEntryOfSingleWords.checkingEntry(C11);

/* Case1: if C21C22 has entry,
 *      check if C11 has entry, if yes, perform linguistic matching between C11 and C21C22
and get score1. */
if (C21C22Entry)
{
   if (C11Entry)
   {
      score1 = linguisticSimilarity.performLinguisticSimilarity(C21C22, C11);
   }
}

/* Case2: if C22C23 has entry,
 *      check if C11 has entry, if yes, perform linguistic similarity between C22C23 and C11
and get score2.*/
if (C22C23Entry)
{
   if (C11Entry)
   {
      score2 = linguisticSimilarity.performLinguisticSimilarity(C22C23, C11);
   }
}
```

```
        /* Case3: if neither C21C22 nor C22C23 has entry,
         *      match C11 with C21 using SingleWordsMatching and get score31.
         *      match C11 with C22 using SingleWordsMatching and get score32.
         *      match C11 with C23 using SingleWordsMatching and get score33.*/
         if (!C21C22Entry&&!C22C23Entry)
         {
            score31 = singleWordsMatching.performSingleWordsMatching(C11, C21);
            score32 = singleWordsMatching.performSingleWordsMatching(C11, C22);
            score33 = singleWordsMatching.performSingleWordsMatching(C11, C23);
         }
        double scores[] = {score1, score2, score31, score32, score33};

        /* We select the highest score because it means that the single word is most probably similar
in meaning to the one which has the highest score with*/
        for (int i = 0; i<scores.length; i++)
        {
           if (scores[i] >= score)
           {
              score = scores[i];
              indicator = i;
           }
        }
        /* If score1 is the highest, score =  */

        if (indicator == 0)
        {
           score = 0.2*score1;
        }
        else if (indicator == 1)
        {
           score = score2;
        }
        else if (indicator == 2)
        {
           score = 0.1*score31;
        }
        else if (indicator == 3)
        {
           score = 0.1*score32;
        }
        else if (indicator == 4)
        {
           score = 0.8*score33;
        }

        return score;
    }
}
```

Appendix C

An Example of an Annotated Service: Bookinfoport

```
<xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="GetBookInfoByISBN"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="GetBookInfoByISBN"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="GetBookInfoByISBN">
      <s:element name="GetBookInfoByISBN">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Isbn" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Isbn" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="CustomerAccount"
sawsdl:modelReference = "http://www.w3.org/2000/10/swap/pim/contact#CustomerAccount"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="CustomerSubAccount"
sawsdl:modelReference = "http://www.w3.org/2000/10/swap/pim/contact#CustomerSubAccount"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="LoginName" sawsdl:modelReference
= "http://www.w3.org/2000/10/swap/pim/contact#LoginName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="LoginPassword"
sawsdl:modelReference = "http://www.w3.org/2000/10/swap/pim/contact#LoginPassword"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetBookInfoByISBNResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetBookInfoByISBNResult"
type="tns:BookInfoResponseType" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="BookInfoResponseType">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Status" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Status"  type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Book" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Book"  type="tns:Book" />
          <s:element minOccurs="0" maxOccurs="1" name="Marc" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Marc"  type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="Book" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Book"  >
        <s:sequence>
```

```
        <s:element minOccurs="0" maxOccurs="1" name="Title" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Title" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="Author" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Author" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="Isbn" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Isbn" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="Publisher" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Publisher" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="PublicationDate"
sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/BookProperty.daml#PublicationDate"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="PublicationPlace"
sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/BookProperty.daml#PublicationPlace"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="Edition" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Edition" type="s:string" />
        <s:element minOccurs="1" maxOccurs="1" name="DiscountPrice" sawsdl:modelReference
= "http://islab.hanyang.ac.kr/damls/BookProperty.daml#Price" type="s:double" />
        <s:element minOccurs="0" maxOccurs="1" name="Availability" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Availability" type="s:string" />
        <s:element minOccurs="1" maxOccurs="1" name="ListPrice" sawsdl:modelReference =
"http://islab.hanyang.ac.kr/damls/BookProperty.daml#Price" type="s:double" />
        <s:element minOccurs="1" maxOccurs="1" name="DiscountPercent"
sawsdl:modelReference = "http://islab.hanyang.ac.kr/damls/BookProperty.daml#DiscountPercent"
type="s:double" />
       </s:sequence>
     </s:complexType>
     <s:element name="BookInfoResponseType" nillable="true"
type="tns:BookInfoResponseType" />
   </s:schema>
  </wsdl:types>
  <wsdl:message name="GetBookInfoByISBNSoapIn">
    <wsdl:part name="parameters" element="tns:GetBookInfoByISBN" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoByISBNSoapOut">
    <wsdl:part name="parameters" element="tns:GetBookInfoByISBNResponse" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoByISBNHttpGetIn">
    <wsdl:part name="ISBN" type="s:string" />
    <wsdl:part name="CustomerAccount" sawsdl:modelReference =
"http://www.w3.org/2000/10/swap/pim/contact#CustomerAccount"  type="s:string" />
    <wsdl:part name="CustomerSubAccount" sawsdl:modelReference =
"http://www.w3.org/2000/10/swap/pim/contact#CustomerSubAccount"  type="s:string" />
    <wsdl:part name="LoginName" type="s:string" />
    <wsdl:part name="LoginPassword" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoByISBNHttpGetOut">
    <wsdl:part name="Body" element="tns:BookInfoResponseType" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoByISBNHttpPostIn">
    <wsdl:part name="ISBN" type="s:string" />
    <wsdl:part name="CustomerAccount" sawsdl:modelReference =
"http://www.w3.org/2000/10/swap/pim/contact#CustomerAccount"  type="s:string" />
    <wsdl:part name="CustomerSubAccount" sawsdl:modelReference =
"http://www.w3.org/2000/10/swap/pim/contact#CustomerSubAccount"  type="s:string" />
    <wsdl:part name="LoginName" type="s:string" />
    <wsdl:part name="LoginPassword" type="s:string" />
  </wsdl:message>
```

```
  <wsdl:message name="GetBookInfoByISBNHttpPostOut">
    <wsdl:part name="Body" element="tns:BookInfoResponseType" />
  </wsdl:message>
  <wsdl:portType name="BookInfoServiceSoap">
    <wsdl:operation name="GetBookInfoByISBN">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Stock Status Check by ISBN and
Customer ID</documentation>
      <wsdl:input message="tns:GetBookInfoByISBNSoapIn" />
      <wsdl:output message="tns:GetBookInfoByISBNSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="BookInfoServiceHttpGet">
    <wsdl:operation name="GetBookInfoByISBN">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Stock Status Check by ISBN and
Customer ID</documentation>
      <wsdl:input message="tns:GetBookInfoByISBNHttpGetIn" />
      <wsdl:output message="tns:GetBookInfoByISBNHttpGetOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="BookInfoServiceHttpPost">
    <wsdl:operation name="GetBookInfoByISBN">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Stock Status Check by ISBN and
Customer ID</documentation>
      <wsdl:input message="tns:GetBookInfoByISBNHttpPostIn" />
      <wsdl:output message="tns:GetBookInfoByISBNHttpPostOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="BookInfoServiceSoap" type="tns:BookInfoServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <wsdl:operation name="GetBookInfoByISBN">
      <soap:operation soapAction="GetBookInfoByISBN/GetBookInfoByISBN" style="document"
/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="BookInfoServiceHttpGet" type="tns:BookInfoServiceHttpGet">
    <http:binding verb="GET" />
    <wsdl:operation name="GetBookInfoByISBN">
      <http:operation location="/GetBookInfoByISBN" />
      <wsdl:input>
        <http:urlEncoded />
      </wsdl:input>
      <wsdl:output>
        <mime:mimeXml part="Body" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="BookInfoServiceHttpPost" type="tns:BookInfoServiceHttpPost">
    <http:binding verb="POST" />
    <wsdl:operation name="GetBookInfoByISBN">
      <http:operation location="/GetBookInfoByISBN" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
```

```
      </wsdl:input>
      <wsdl:output>
        <mime:mimeXml part="Body" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="BookInfoService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />
    <wsdl:port name="BookInfoServiceSoap" binding="tns:BookInfoServiceSoap">
      <soap:address location="http://edi.btol.com/bookinfoservice/bookinfoport.asmx" />
    </wsdl:port>
    <wsdl:port name="BookInfoServiceHttpGet" binding="tns:BookInfoServiceHttpGet">
      <http:address location="http://edi.btol.com/bookinfoservice/bookinfoport.asmx" />
    </wsdl:port>
    <wsdl:port name="BookInfoServiceHttpPost" binding="tns:BookInfoServiceHttpPost">
      <http:address location="http://edi.btol.com/bookinfoservice/bookinfoport.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```