

ONTOLOGY LEARNING FOR SEMANTIC WEB SERVICES

**A thesis submitted towards the degree of
Doctor of Philosophy**

By

Auhood Alfaries

**School of Information Systems, Computing and Mathematics
Brunel University**

September 2010

ABSTRACT

The expansion of Semantic Web Services is restricted by traditional ontology engineering methods. Manual ontology development is time consuming, expensive and a resource exhaustive task. Consequently, it is important to support ontology engineers by automating the ontology acquisition process to help deliver the Semantic Web vision. Existing Web Services offer an affluent source of domain knowledge for ontology engineers. Ontology learning can be seen as a plug-in in the Web Service ontology development process, which can be used by ontology engineers to develop and maintain an ontology that evolves with current Web Services. Supporting the domain engineer with an automated tool whilst building an ontological domain model, serves the purpose of reducing time and effort in acquiring the domain concepts and relations from Web Service artefacts, whilst effectively speeding up the adoption of Semantic Web Services, thereby allowing current Web Services to accomplish their full potential

With that in mind, a Service Ontology Learning Framework (SOLF) is developed and applied to a real set of Web Services. The research contributes a rigorous method that effectively extracts domain concepts, and relations between these concepts, from Web Services and automatically builds the domain ontology. The method applies pattern-based information extraction techniques to automatically learn domain concepts and relations between those concepts. The framework is automated via building a tool that implements the techniques. Applying the SOLF and the tool on different sets of services results in an automatically built domain ontology model that represents semantic knowledge in the underlying domain.

The framework effectiveness, in extracting domain concepts and relations, is evaluated by its appliance on varying sets of commercial Web Services including the financial domain. The standard evaluation metrics, precision and recall, are employed to determine both the accuracy and coverage of the learned ontology models. Both the lexical and structural dimensions of the models are evaluated thoroughly. The evaluation results are encouraging, providing concrete outcomes in an area that is little researched.

ACKNOWLEDGEMENTS

I would like to acknowledge my deepest gratitude to those who have helped along the way and influenced the formation of my understanding.

- First, I would like to express my appreciation to my first supervisor Professor Mark Lycett. It is my great pleasure to acknowledge his invaluable suggestions, guidance and constant support during my research. I am very grateful to Prof. Lycett for providing a stimulating environment via the Fluidity research group. It is my good fortune to have been supervised by him and to have worked and learned from him.
- I am deeply grateful to my second supervisor Dr. David Bell for his valuable time, advice and support in all possible ways during my research.
- I am thankful to all my colleagues in SJ128 for the fruitful discussions we had many times at our desks. Thanks to my dearest colleague Laden Aldin, for her thoughtful comments and for the good times we had in Brunel University.
- I would like to express my gratitude to my cherished husband, Mosaad. I am so appreciative for his constant love, understanding and encouragement, for his taking up the extra responsibilities to our family and bearing the pressure both from work and home during my PhD. My thanks also go to my children; Fahad, Omar, Aljoharah and Abdulaziz for their incredible patience and understanding at times where I had to miss special moments with them.
- Finally, but not least, I would like to thank all of my extended family and friends for their belief in me. Very special thanks to my beloved mother, Aljoharah. For her prayers, continuous encouragement and support; and to whom I dedicate this thesis.

PUBLICATIONS

The work in this thesis has led to the following publications:

Alfaries, A., Bell, D. & Lycett, M. 2009, "Ontology Learning for Semantic Web Services", Proceedings of the 14th Annual UK Association of Information Systems Conference (UKAIS), Oxford University, Oxford, U.K, 31st March - 01st April, pp. 27-36.

Alfaries, A., Bell, D. & Lycett, M., "Service Ontology Learning Framework", work under review with IEEE Transactions on Services Computing (TSC).

TABLE OF CONTENTS

ABSTRACT.....	II
ACKNOWLEDGEMENTS	III
PUBLICATIONS	IV
ACRONYMS.....	XI
CHAPTER 1 - INTRODUCTION.....	13
1.1 Background to the Problem	13
1.1.1 Service Orientation and the Role of Ontology.....	13
1.1.2 Ontology Engineering.....	14
1.2 Aims and Objectives:.....	16
1.3 Research Methodology	17
1.4 Thesis Overview	20
CHAPTER 2 - LITERATURE REVIEW	23
2.1 Introduction.....	23
2.2 Achieving Semantic Web Services/ Industry Perspective	24
2.2.1 Agents	27
2.2.2 Ontology	28
2.3 Tools used for Ontology Development.....	32
2.4 Ontology Development Challenge.....	34
2.5 Ontology Learning.....	35
2.5.1 Text-based Ontology Learning Approaches.....	37
2.5.2 Learning Approaches Based on Semi-structured Data	38
2.5.3 Learning Approaches Based on Structured Data	39
2.6 Overview of Ontology Learning Techniques	40
2.6.1 Machine Learning Techniques.....	40
2.6.2 Statistical Analysis.....	41
2.6.3 Linguistic Techniques.....	41
2.6.4 Rule-based Techniques	42
2.7 Related Work / Ontology Learning for Web Services.....	43
2.8 Summary	47
CHAPTER 3 – DESIGN RESEARCH METHODOLOGY	48
3.1 Introduction.....	48
3.2 Design Research Background.....	48
3.3 Design as an IS Research methodology.....	51
3.4 Design Research Evaluation	54
3.5 Applying Design Research	56
3.6 Research Evaluation	58
3.7 Research Design Iterations	62
3.8 Summary	69
CHAPTER 4 - ITERATION I	70
4.1 Introduction.....	70
4.2 Design Research and Output Artefacts	70
4.2.1 Design Research Artefacts	72

4.3	Artefact Building and Development	74
4.3.1	Tokenization	74
4.3.2	POS Tagging.....	75
4.3.3	Pattern Extraction	76
4.3.4	Ontology Building	77
4.4	Framework Prototype Implementation	77
4.5	Evaluation	84
4.5.1	Experimental Data	85
4.5.2	STE Performance	87
4.5.3	Pattern Evaluation.....	89
4.6	Specifying the Learning.....	91
4.7	Summary	92
CHAPTER 5 - ITERATION 2.....	94	
5.1	Introduction.....	94
5.2	Design Research and Output Artefacts	95
5.2.1	Design Research Artefacts	96
5.3	Artefact Building and Development	96
5.3.1	Document Pre-processing Phase.....	97
5.3.2	Relation Extraction	98
5.3.3	Ontology Building	99
5.3.4	Ontology Validation	100
5.4	Application and Implementation of SOLF	100
5.4.1	Pattern Extraction	101
5.4.2	Transformation Rule Development	108
5.4.3	Ontology Building	110
5.5	Evaluation	112
5.5.1	SIP Extraction Process Evaluation	112
5.5.2	Precision and Recall Evaluation Measures	113
5.5.3	Qualitative Evaluation	117
5.6	Specifying the Learning.....	118
5.7	Summary	119
CHAPTER 6 - ITERATION 3.....	120	
6.1	Introduction.....	120
6.2	Design Research and Output Artefacts	121
6.3	SOLF Refinement and Gold Standard Evaluation.....	122
6.3.1	Validate Ontology and Amend Patterns	124
6.3.2	Incorporating WSDL Structure in SOLF	124
6.3.3	Ontology Pruning.....	129
6.3.4	Experimental Data and Evaluation	129
6.3.5	Domain Coverage - Lexical Layer.....	131
6.3.6	Non Taxonomic Layer – Structural Evaluation.....	135
6.3.7	Taxonomic Layer – Structural Evaluation.....	139
6.4	Domain Expert Evaluation and SOLF Refinement	140
6.5	Specifying the learning	141
6.6	Summary	144
CHAPTER 7 - CONCLUSION	146	
7.1	Research Summary	146
7.2	Contributions and Conclusions	152

7.3 Limitations and Areas for Future Research	155
BIBLIOGRAPHY	158
APPENDICES	168
APPENDIX A - POS TAGGER	168
APPENDIX B - JAPE CODE	170
APPENDIX C - DATA SETS	172
APPENDIX D - EVALUATION SPREAD SHEETS	183

List of Figures

Figure 1-1: Thesis Outline	22
Figure 2-1: Web Service Architecture	25
Figure 2-2: Ontology Learning Layer Cake (adopted from Cimiano, 2007).....	37
Figure 3-1: A Research Framework (March & Smith 1995).....	49
Figure 3-2: IS Research Framework (Hevner et al., 2004).....	53
Figure 3-3: Steps of Design Research (Vashnavi & Kuhler, 2004).....	57
Figure 3-4: Taxonomy of OL Evaluation Approaches	59
Figure 3-5: Research Iterations	63
Figure 4-1: Iteration 1 Overall Framework.....	72
Figure 4-2: WSDL sample file.....	75
Figure 4-3: Pattern Extraction Process	76
Figure 4-4: Service Term Extraction (STE)	77
Figure 4-5: SOLF Application Pipeline.....	79
Figure 4-6: WSDL POS Model	80
Figure 4-7: JAPE Sample Code	83
Figure 4-9: Snapshot of the Learned Domain Ontology Model	84
Figure 4-8: JAPE Rule for Concept Creation	83
Figure 4-10: WS2 Precision.....	89
Figure 5-1: Research Iterations.....	95
Figure 5-2: Service Ontology Learning Framework (SOLF)	97
Figure 5-3: WSDL to OWL SIP Mapping.....	99
Figure 5-4: ANNIC Pattern Extraction Query	103
Figure 5-5: Application Pipeline Processing Steps.....	111
Figure 5-6: JAPE Rule 1	111
Figure 5-7: JAPE Transformation Rule 1	111
Figure 5-8: A Sample of the Learned Domain Ontology Model.....	112
Figure 5-9: Pattern Recall Chart	115
Figure 5-10: Concept-Relation Precision Chart.....	116
Figure 6-1: Overall Design Research Iterations Framework	122
Figure 6-2: Service Ontology Learning Framework.....	123
Figure 6-3: Financial WSDL Code Sample	125
Figure 6-4: Sample Complex Relation JAPE Rule.....	126

Figure 6-5: Complex Relation Transformation Rule	127
Figure 6-6: Sample SOLF Ontology model (Group 2).....	128
Figure 6-7: Sample of the Financial Learned Ontology (SOLFO).....	130
Figure 6-8: Sample of Lexical Layer Evaluation Model	132
Figure 6-9: NonTP Evaluation Model	138
Figure 6-10: Sample Group 1 (Book) Ontology	144
Figure 0-1: Part-Of-Speech Tags (from GATE user Guide)	168
Figure 0-2: Part-Of-Speech Tags (from GATE User Guide)	169
Figure 0-3: JAPE code snippet illustrating code for Rules 1-4	170
Figure 0-4: JAPE Snippet, illustrating code for transformation rules TR3 and TR4	171
Figure 0-5: Matching WS1 WSDL and XSD Sample	172
Figure 0-6: Financial Ontology Model (Iteration 1).....	173
Figure 0-7: Financial Ontology Model (Iteration 2).....	174
Figure 0-8: Books Service Sample 1 Snippet	175
Figure 0-9: Books Service Sample 2 Snippet	176
Figure 0-10: Books GSO Snippet	177
Figure 0-11: Books SOLFO Snippet	178
Figure 0-12: Finance Sample 1 Snippet.....	179
Figure 0-13: Finance Sample 2 Snippet.....	180
Figure 0-14: Snippet Of Financial GSO	181
Figure 0-15: Snippet of Financial SOLFO	182
Figure 0-16: Method1-WS2 (XSD) Domain Expert Scoring	183
Figure 0-17: Method1& 2-WS2 (WSDL) Domain Expert Scoring.....	184
Figure 0-18: Method3-WS2 (XSD & WSDL) Domain Expert Scoring.....	185
Figure 0-19: Iteration 2 Financial Ontology Domain Expert Scoring	186
Figure 0-20: Iteration 3 Financial Gold Standard Ontology	187
Figure 0-21: Iteration 3 Financial SOLFO Gold Standard Evaluation	188
Figure 0-22: Iteration 3 Financial SOLFO Gold Standard Evaluation	189

List of Tables

Table 2-1: Summarized Ontology Types	29
Table 2-2: Summarized Approaches to SWS	32
Table 3-1: Summarized Evaluation Criteria with Artefact Types (Hevner et al., 2004) .	55
Table 3-2: Design Evaluation Methods (Hevner et al., 2004)	56
Table 3-3: Comparison of OL Evaluation Methods	61
Table 3-4: Research Products Versus Research Processes	67
Table 3-5: Summary of Research Iterations	68
Table 4-1: Iteration Steps – Input Output Model.....	73
Table 4-2 : WSDL Tokenized Model	80
Table 4-3: Pattern Extraction Model	81
Table 4-4: Summarized Generic Patterns	82
Table 4-5: Summary Information Representing Used Web Services.....	86
Table 4-6: WSTM Extracted from WS3	87
Table 4-7: Concept Evaluation Model.....	88
Table 4-8: Default Tokenizer WSDL Model.....	91
Table 5-1: Iteration Steps Input Output model	96
Table 5-2: Output of WSDL (WS1) Tokenizer Step	100
Table 5-3: Output of the WSDL (WS1) POS Tagger.....	101
Table 5-4: Web Service 1 Pattern Extraction Model.....	104
Table 5-5: Web Service 2 Pattern Extraction Model.....	105
Table 5-6: Web Service 3 Pattern Extraction Model.....	106
Table 5-7: Relative Frequency of SIP Across Three Web Services	107
Table 5-8: Pattern Relation-Identification Model.....	108
Table 5-9: Sample Pattern-Relation Identification Model.....	109
Table 5-10: Summarized Transformation Rules.....	110
Table 5-11: Pattern Recall Summary	114
Table 5-12: Summarized Results for Precision	116
Table 6-1: Formal Definition of SOLF Output Phases.....	124
Table 6-2: Summarised Precision and Recall for Group 1 and Group 2	135
Table 6-3: Summarized NonTP and NonTR Results.....	139
Table 6-4: Summarized Domain Expert Precision	141
Table 7-1: Design Research Products X Activities	149

ACRONYMS

- **ANNIC:** ANNotations In Context
- **API:** Application Programme Interface
- **ASIUM:** Acquisition of Semantic knowLedge Using Machine learning methods
- **DAML:** DARPA Agent Markup Language
- **DOLCE:** Descriptive Ontology for Linguistic and Cognitive Engineering
- **Design Research:** Design Research
- **GATE:** General Architecture for Text Engineering
- **GSO:** Gold Standard Ontology
- **GUI:** Graphic User Interface
- **HTML:** Hyper Text Markup Language
- **HTTP:** Hyper Text Transfer Protocol
- **IE:** Information Extraction
- **IRS:** Internet Reasoning Service
- **JAPE:** Java Annotation Pattern Engine
- **LATINO:** Link Analysis and Text-Mining Toolbox
- **LP:** Lexical Processing
- **LR:** Lexical Recall
- **ML:** Machine Learning
- **NLP:** Natural Language Processing
- **NN:** Noun
- **NNP:** Proper Noun
- **NonT:** Non-Taxonomic
- **NonTR:** Non-Taxonomic Recall
- **OI:** Ontological Improvements
- **OL:** Ontology Learning
- **OLT:** Ontology Learning Techniques
- **OWL:** OWL Web Ontology Language
- **OWL-DL:** OWL Description Logics
- **OWL-full:** Version of OWL
- **OWLIM:** A Semantic Repository
- **OWL-Lite:** Version of OWL
- **OWL-S:** Web Ontology Language for Web Services
- **POS:** Part of Speech
- **PSL:** Process Specification Language
- **RDF:** Resource Description Framework
- **RPC:** Remote Procedure Call
- **SAWSDL:** Semantic Annotation for Web Service Description Language
- **SIP:** Structured Interpretation Patterns
- **SOA:** Service Oriented Architecture
- **SOAP:** Simple Object Access Protocol
- **SOLF:** Service Ontology Learning Framework
- **SOLFO:** SOLF Ontology
- **STE:** Service Term Extraction
- **SUMO:** Suggested Upper Merged Ontology
- **SWS:** Semantic Web Services

- **SWSF:** Semantic Web Services Framework
- **SWSO:** Semantic Web Services Ontology
- **TAO:** Transitioning Applications to Ontologies
- **TP:** Taxonomic Precision
- **TR:** Transformation Rule
- **UDDI:** Universal Description, Discovery, and Integration
- **UPML:** United Problem Solving Method Development Language
- **URI:** Uniform Resource Identifier
- **VB:** Verb
- **W3C:** World Wide Web Consortium
- **WebODE:** An Ontology Editing Tool
- **WS:** Web Services
- **WSDL:** Web Service Description Language
- **WSDL-S:** Web Service Description Language - Semantic
- **WSMF:** Web Service Modelling Framework
- **WSMO:** Web Service Modelling Ontology
- **WSMX:** Web Service Modelling eXecution Environment
- **WSTM:** Web Service Term Model
- **XML:** Extensible Markup Language
- **XSD:** XML Schema Definition

CHAPTER 1 - INTRODUCTION

1.1 Background to the Problem

1.1.1 Service Orientation and the Role of Ontology

Service Oriented Architecture (SOA) is an emerging architectural approach with the potential to better accommodate the changing enterprise. SOA unifies business processes by encapsulating modules as well-defined interoperable services delivering large applications as a collection of services (Papazoglou & van den Heuvel, 2007). Currently, Web Services are the predominant technological means of delivering on the SOA ideal and there is a clear increase in organizational interest in both the architecture and delivery mechanism (Azoff, 2007; Heffner & Peters, 2008; Martin, 2007a; Tsai et al., 2006; Yu et al., 2008). Recent surveys (Meyer, 2006) indicate that Web Service creation and application development via Web Services is under way within 50% and 33% of the US and western European organizations surveyed respectively. Larger organisations are the primary adopters of SOA, primarily due to a greater need for integrating applications and services to adapt to dynamically changing processes.

Though increasing in popularity, several barriers to adoption exist including organizational complexity, the need for manual intervention and a lack of application support (such as easy to adopt tools) (Gedda, 2007). In particular, the need for manual intervention in discovery and adoption stands out as a challenge - Web Services cannot be automatically discovered and composed as the description of those services lack the necessary semantics (Martin, 2007b). This point is explicitly recognized by the Semantic Web community (Berners-Lee, Hendler & Lassila, 2001; Shadbolt, Hall & Berners-Lee, 2006), who argue that full automation of service discovery and composition is indispensable and is necessary for dynamic, flexible and machine understandable services and, as a consequence, an infrastructure that meets the business ideal (Maedche & Staab, 2003).

Semantic Web Services are introduced to enable automatic service discovery and composition (Sheth, 2006) by providing the infrastructure that meets the ultimate

business needs. The infrastructure is based on the use of ontologies as the core component that facilitates the semantic layer. Ontologies, in computer science, are defined by Studer et al. (1998, p.184) as: “ a formal, explicit specification of a shared conceptualization.”. Each term in this definition represents an important aspect of ontologies in providing and catering for the Semantic Web vision. The first part - formal, explicit specification – of the definition implies that the explicit specification is described using formal machine readable language, like description logic (Bruijen, 2009). The conceptualisation part provides the abstract view model of the underlying domain described by the ontology. Finally, the shared aspect provides the stakeholders with an ability to share an ontological conceptualization commitment (Bruijen, 2009). Importantly, ontologies are categorized in different types according to their use. For example top-level ontologies are used to give an abstract view of the world whereas lower level ontologies are domain specific.

The literature clearly indicates that Web Service domain ontologies are the general means by which semantics are added to Web Services, therefore, providing a solution for automating their service tasks. Semantic Web Services benefit from ontologies in two ways: (1) reasoning facility to automate the Web Service usage tasks, (2) providing a shared conceptualization of a domain to corporate stakeholders (Bruijn, 2009). The demand therefore is to develop ontologies from existing services and to enable those ontologies to adapt and evolve in line with the domain and any demands made on it (Cuel et al., 2008).

1.1.2 Ontology Engineering

The importance of achieving Semantic Web Services emphasises the need for a faster and less expensive ontology development process. Manual ontology acquisition is a tedious, expensive and error prone task that can slow down the ontology development process (Ding & Foo, 2002; Staab & Maedche, 2001; Maedche & Staab, 2001). Ontology engineers are generally required to develop a domain knowledge base using ontologies, and they are also required to ensure that these ontologies are updated and maintained by extending the knowledge base with new domain concepts. ‘Ontology learning’ is the term used to refer to automatic or semi-automatic acquisition of knowledge from different sources of data (Buitelaar, Cimiano & Magnini, 2007; Zhou, 2007; Buitelaar & Cimiano, 2008). Enormous

power could be added to the Semantic Web by automating the manual knowledge acquisition process; this process normally involves domain experts mining legacy systems and underlying documentation in order to harvest domain concepts and identify taxonomic and non-taxonomic relations between those concepts. Applying artificial intelligence automated techniques to extract domain knowledge from legacy systems can certainly assist domain engineers, consequently contributing towards faster ontology development (Maedche & Staab, 2001).

The goal of ontology learning is to support and facilitate ontology construction. Ontology learning is a long way from being fully automatic, but it can be effectively integrated in a wider ontology engineering framework (Zhou, 2007; Buitelaar & Cimiano, 2008; Maedche & Staab, 2004; Maedche, 2002; Cimiano et al., 2009). Drawing upon that statement, it is clear that ontology learning can play a key role towards achieving Semantic Web Services.

A number of ontology learning methods have been introduced over the last few years (Zhou, 2007; Buitelaar & Cimiano, 2008; Cimiano et al., 2009). These methods are considered to be general ontology learning methods, and have not been tested or applied and evaluated on the Web Service domain. Semantic Web Services impose a special kind of ontology learning application area due to the fact that they contain both structured and unstructured data (Yu, 2007). Due to the role that ontology development plays in Semantic Web Services, and the fact that only limited research has been found in this area, further research on ontology learning techniques that cater for extracting domain ontologies from Web Services is required.

Several approaches have been proposed to facilitate the automatic extraction of ontological elements from different types of knowledge sources, ranging from structured, semi-structured and unstructured sources (Zhou, 2007). An Ontology Learning (OL) system can be considered as a reverse engineering process where input data sources are used by the system to learn relevant domain concepts and relations, and an ontology is produced as an output of the system. OL approaches are classified according to the data sources used as input to the system (Maedche & Staab, 2004). The emphases found in the proposed OL approaches, are mainly aimed at applying OL on unstructured data sources, commonly referred to as textual

sources. Progressing ontology development for Web Services can benefit greatly from applying current OL techniques on Web Service artefacts and evaluating their applicability on real data Web Service sources.

With the Semantic Web Services vision and the rapid increase in the number of available Web Services, here, the research focus is on applying ontology learning techniques on Web Services artefacts as an application domain of the Semantic Web. It is important to look intensely into and to investigate the effect of applying OL on the current Web Service XML-based standards such as SOAP and WSDL, as they provide a rich source of legacy domain knowledge (Sabou, 2005). Providing appropriate tools that assist in and automate ontology development - taken in the large part from ontology learning - is essential for a dynamic service vision to be realized.

The challenge, therefore, is to develop ontologies from existing services and to enable those ontologies to adapt and evolve in line with the domain and any demands made on it (Cuel et al., 2008). Adopting knowledge extraction techniques in the form of Ontology Learning provides an automated means of dealing with these issues, as it allows automatic knowledge acquisition from different sources of Web Services, for the purpose of reducing the cost, time and effort required by ontology engineers to build domain specific ontologies (Buitelaar, Cimiano & Magnini, 2007).

1.2 Aims and Objectives:

The aim of this research is to automate the ontology development process and to develop a methodological ontology learning framework tailored for Web Services.

The objectives of the work are to:

1. Review the available ontology learning approaches and tools in order to provide an understanding of the state-of-the-art of ontology learning and Web Services.
2. Develop ontology learning techniques for service concept and relation extraction and to automate these techniques by building a prototype application to test the applicability of the techniques using real Web Services.

3. Develop a methodological Service Ontology Learning Framework (SOLF) that incorporates the techniques for concept and relation extraction.
4. Implement a tool that facilitates the framework and evaluates the application of the framework, and assess the impact of the framework on the state-of-the-art of ontology learning.
5. Validate the research outcome by testing the generality of the extracted patterns and rules on services from other domains.

1.3 Research Methodology

Design research is chosen as the research method for executing this research. The objective of Design Research is to produce a relevant IT based solution to a significant business problem (Hevner et al., 2004) with a focus on the utility of the artefact. the approach applies a set of analytical techniques from the problem space to understand, explain and improve the designed artefact. Design research is considered both a product and a process. The process incorporates a set of design and behavioural science activities; build, evaluate, justify and theorise (March & Smith, 1995). The products of Design Research can be classified according to the four-type product classification (March & Smith, 1995);

- Constructs are sets of concepts used to define the problems and solutions.
- Models are used to describe a real world situation of the design problem and its solution space.
- Methods are used to provide guidance on how to solve problems using the constructs and models. They are thought of as methodological tools (March & Smith, 1995).
- Instantiations are the implementations of constructs, models and methods allowing actual evaluation, of feasibility and effectiveness, of the Design Research artefact.

Design research must be applied as a search process for an effective solution, utilizing and sustaining laws in the problem space. In order to demonstrate the

effectiveness of the solution, rigorous Design Research evaluation methods from the knowledge space must be executed to evaluate the quality of the artefact (Hevner et al., 2004). Design Research seeks to achieve an appropriate solution to the design problem in an iterative knowledge refinement manner, where each iteration executes build and evaluate cycle, contributing new learning and knowledge that feeds back into consequent iterations.

Ontology learning as a research area is still young; consequently Design Research is employed as the research methodology as it allows learning to evolve as the solution is developed for the problem space (Vaishnavi & Kuechler, 2004). A Design research process is employed as a problem solving method, whereas a valid IS research is achieved through an iterative build and evaluate design cycle of a purposefully designed artefact. The main Design Research phases applied are as follows;

- Problem Awareness: This involves reviewing the literature to analyse the availability of ontology learning techniques and confirmed the lack of automated knowledge acquisition tools in the Semantic Web Services domain.
- Suggestion: This phase involves introducing a tentative idea of how to apply suitable knowledge extraction techniques. The learning techniques are borrowed from the machine learning and natural language processing disciplines to satisfy the aim of learning ontologies from Web Service sources.
- Development: The development of the solution will be achieved by building the design artefact. Here the artefact is a service ontology learning framework (SOLF). By immersing in the build activity the researcher achieves an understanding of the problem space raising new suggestions to improve the next build and evaluate cycle.
- Evaluation: This phase is concerned with the development of an assessment method or metric to assess the quality and effectiveness of the designed artefact (March & Smith, 1995). Synthesising the Design Research evaluation criteria to identify appropriate evaluation methods and metrics from the problem space has lead to identifying the commonly

applied information extraction metrics, precision and recall, to evaluate the ontology learning method. The learned ontology model, SOLF, is evaluated for coverage of the domain and for accuracy.

- Conclusions: This is the final phase of the Design Research cycle, withdrawn from the learning that emerged from understanding how and why the solution works in the problem domain when applied to real sets of services. Limitations of the solution and areas for future work are also provided in the conclusion of the research.

Applying March & Smith's (1995) Design Research product classification to illustrate research contributions leads to identifying the main design artefact as the development of a Service Ontology Learning methodological Framework (SOLF). In order to deliver the final SOLF method the research significance lies in building consequent set of constructs, models, methods and instantiations.. In this research, framework development follows from executing Build and Evaluate activities. These activities are executed in an iterative incremental Design Research manner consisting of three iterations as follows:

- Iteration 1 – Core framework development including service term extraction technique. Automate the framework by implementing an application tool and evaluate the technique and tool by applying them on real sets of Web Services and evaluating the learned ontology model with the identified evaluation metrics.
- Iteration 2 – Extending the framework to incorporate rule based relation extraction techniques. This iteration contributes a secondary Design Research structured interpretation models and a set of transformation rules. A domain ontology model is also produced representing both lexical and structural aspects of the learned ontology of the financial domain.
- Iteration 3 – Validate the framework by applying and evaluating the extraction method across other domains. The generality of the SOLF and tool will be demonstrated through comparing evaluation measures for two different data sets.

The effectiveness of the Design Research problem is in reducing the cost and time of the ontology development process. An instantiation tool is created and applied to real case scenarios of Web Services, to illustrate the effectiveness and provide a live proof of the proposed method (SOLF in this research) and as the means by which deficiencies and improvements are identified (March & Smith, 1995). Determining whether progress is made by the extraction method and tool is evaluated by applying the appropriate metrics from the knowledge base to measure the accuracy and coverage of the learned domain ontology model.

1.4 Thesis Overview

In achieving the objectives of the work, the thesis is structured as follows:

Chapter 2: Drawing extensively from the literature, this chapter presents a review of relevant research articles, giving a general background of Semantic Web Services. Advances and development in the field are also discussed. A broad overview of the required technologies for the Semantic Web Services is introduced, leading to the role of ontologies in the Semantic Web Services. The chapter proceeds by discussing issues and challenges that hamper the ontology development, and by introducing ontology learning as a step towards a faster Semantic Web vision. A background discussion of techniques and tools for ontology learning is presented according to their relevance toward ontology development, and therefore towards Semantic Web Services. Finally, the chapter presents similar approaches that apply Ontology Learning techniques on the Web Services application domain, demonstrating the feasibility and utility of the approach and pointing to the limitations of the state-of-the-art, thereby highlighting the need for this research.

Chapter 3: This chapter proposes Design Research as the research methodology for effectively conducting a valid Information Systems research. It then discusses how Design Research is applied in order to plan and execute the research design problem, by developing a method and a tool for learning ontologies from Web Services. Research iterations are identified and research outputs are categorized according to the Design Research products classification. The chapter discusses issues surrounding OL evaluation and presents a taxonomy of evaluation approaches in

order to derive an appropriate evaluation framework for assessing the effectiveness of the developed methodological framework. Finally, the chapter is summarized.

Chapter 4: This chapter presents the first Design Research iteration, tackling the first task of OL by developing and implementing a service term extraction process. The steps involved in the service term extraction are explained and an implementation of the method is detailed. The output of the iteration is presented as a set of Design Research products. An evaluation of the products is then performed, and finally the learning outcome and discussion of future improvements is presented.

Chapter 5: This chapter presents the implementation of the second Design Research iteration. Here, the initial framework developed in chapter 4 is refined and extended by incorporating the relation extraction technique. This chapter contributes a service relation extraction technique based on a set of structured interpretation patterns. The output of this chapter is evaluated by applying the extended framework and the tool on a real set of Web Services. The learned ontology is evaluated by executing a specifically tailored evaluation framework in order to assess the validity of the relation extraction process.

Chapter 6: The third research iteration is executed here to improve and validate the generality of the framework, by applying the framework and the structured interpretation patterns produced in the previous iteration to different sets of Web Services. Evaluating the automatically learned ontology model against the gold standard ontology, measures its completeness and coverage of the underlying domain. The evaluation is performed and appropriate metrics are used to measure the ontology precision.

Chapter 7: This chapter concludes the research thesis and presents the contributions and key findings. Limitations that were learned from applying Design Research to solve the proposed problem are also explained. An evaluation of the Design Research process is performed against satisfying the research aim and objectives, highlighting the research limitations. Lastly, relevant conclusions will be drawn against the degree to which the proposed approach meets its objectives, while an explanation of the research limitations suggesting future improvements is presented.

A thesis outline diagram is created in Figure 1-1 in order to provide an abstract level structure that maps the Design Research iterations to the thesis chapters and the research objectives.

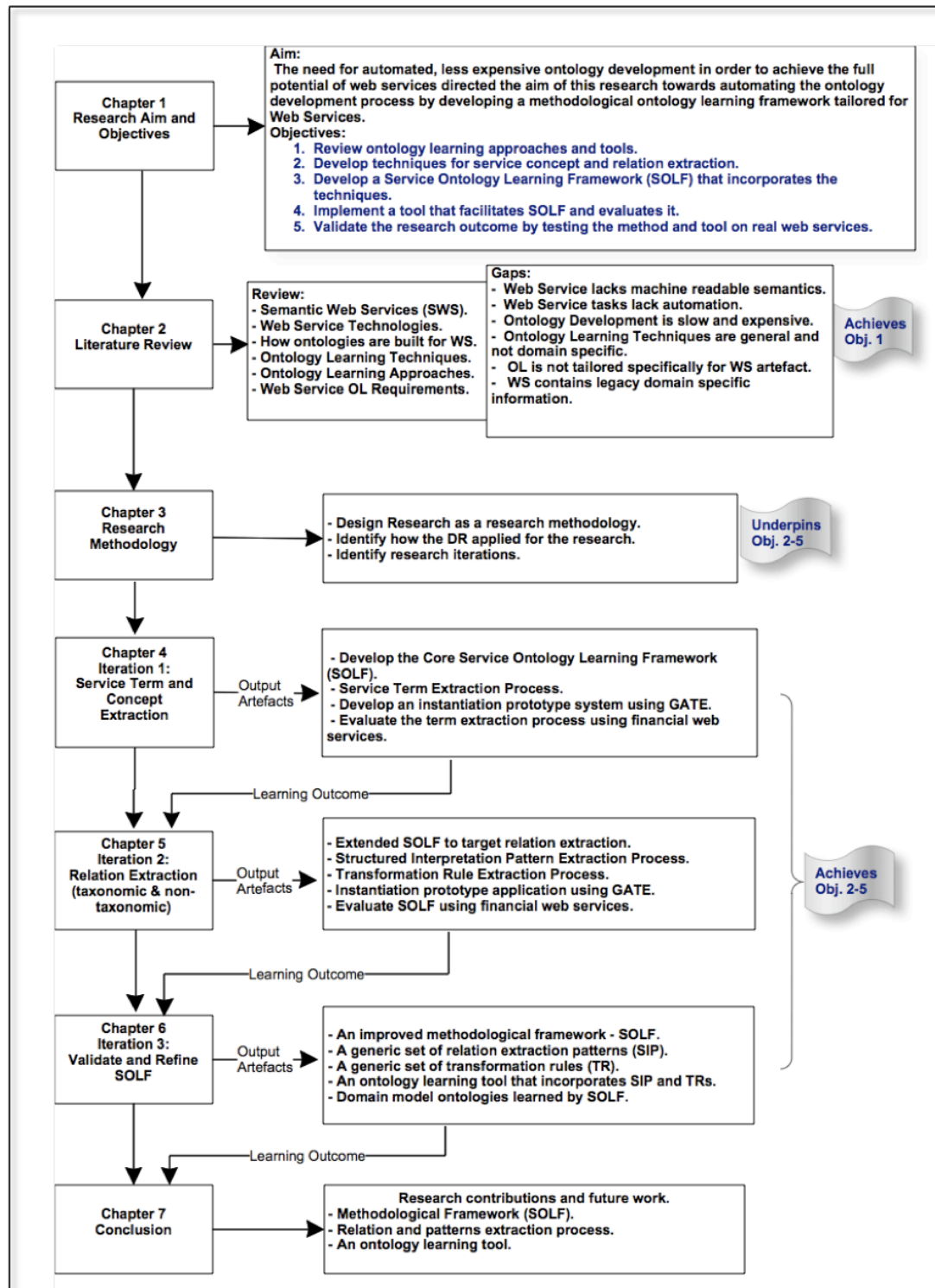


Figure 1-1: Thesis Outline

CHAPTER 2 - LITERATURE REVIEW

2.1 Introduction

Research in accomplishing a decentralised knowledge representation across applications can be achieved by Web Services, which provide an effective way of allowing interoperability across platforms, organizations and operating systems. This chapter looks at the state-of-the-art of current Web Services and discusses how the Semantic Web capacity can bring a new dimension into e-business through current Web Service standards. Literature has shown that by adding semantics into Web Services, automation of enterprise cooperation can be achieved. This chapter reviews the relevant research literature on achieving Semantic Web Services, ontology development challenges are discussed and suggestions on how to improve the ontology development process from the literature are introduced. Existing Web Service sources offer a good starting point for ontology learning and a pragmatic way forward in developing semantics for existing assets. Automating the knowledge acquisition process from different Web sources is discussed and analysed for the purpose of developing an effective approach for adding semantics onto the current Web.

This chapter is structured as follows. Section 2.2 describes a general review of Web Services, introducing the need for adding semantics and the requirements for embedding semantics into Web Services. Section 2.3 presents a broad overview of tools and languages used for ontology engineering. Section 2.4 discusses the challenge of manual ontology development. Section 2.5 presents ontology learning as a way for advancing the ontology development bottleneck and reviews existing literature to present the most important approaches in the field. Section 2.6 classifies existing ontology learning approaches in relation to the techniques applied, and the disciplines from which these techniques are borrowed. Section 2.7 introduces the application of ontology learning in Web Services standards, detailing current work in the area and highlighting issues and challenges and suggesting improvements.

2.2 Achieving Semantic Web Services/ Industry Perspective

Service Oriented Architecture (SOA) is an emerging architectural approach with the potential to better accommodate changing enterprise requirements. SOA unifies business processes by encapsulating modules as well-defined interoperable services delivering large applications as a collection of services (Papazoglou & van den Heuvel, 2007). Currently, Web Services are the predominant technological means of delivering on the SOA ideal and there is a clear increase in organizational interest in both the architecture and delivery mechanism (Azoff, 2007; Heffner & Peters, 2008; Martin, 2007a; Tsai et al., 2006; Yu et al., 2008). Recent surveys, for example Meyer (2006), indicate that Web Service creation and application development using Web Services is under way within 50% and 33% of the US and Western European organizations surveyed respectively. Larger organizations are the primary adopters of SOA, primarily due to a greater need for integrating applications and services to adapt to dynamically changing processes.

Web Services are a collection of application programs that can be accessed remotely using the Web. Therefore, they provide distributed applications with the limitation that these organizations have to follow Web Service standards using Hyper Text Transfer Protocol (HTTP). Once these standards are followed applications can achieve interoperability via the Web (Yu, 2007). Lee, however, suggests that the challenge for the Web is to incorporate a more decentralized knowledge representation system. Semanticising knowledge bases can minimize the need for common standards, hence the Web capacity to achieve the goal of decentralized knowledge representation across applications is greater. In a business environment this implies automatic cooperation between enterprises (Fensel & Bussler, 2002), which is a highly valued goal across organizations (Martin, 2007b; Bruijn et al., 2009).

The literature also identifies a number of technologies for facilitating Web Services that are also essential to cater for SWS. Some of the most commonly adopted standards are SOAP, WSDL and UDDI.

- SOAP (Simple Object Access Protocol) is a lightweight protocol for exchanging structured information in a decentralized environment (W3C).

- WSDL (Web Services Description Language) is an XML-based language used for describing the Web Services.
- UDDI (Universal Description, Discovery, and Integration) is an XML-based registry for worldwide businesses. This service registry is used for service lookup, listing available services and their providers. The UDDI acts as a 'yellow pages' for published services (Berners-Lee, Hendler & Lassila, 2001).

Figure 2-1 illustrates key components, roles and operations in a Web Service environment. Service providers use the Web Service Description Language (WSDL) to provide a syntactic description of service interfaces. Service providers and service requesters are provided with SOAP standards, e.g., as a mechanism for communication description. These two standards are sufficient for enabling the two parties to share and invoke services remotely, but only with a predefined agreement between the provider and the requester. The third component is the service registry (UDDI), which is used to provide a list of businesses and the services they provide. This service registry is unable to achieve its full potential, however, due to the fact that service location, selection and composition (usage tasks) requires extensive human struggle (Bruijn et al., 2009).

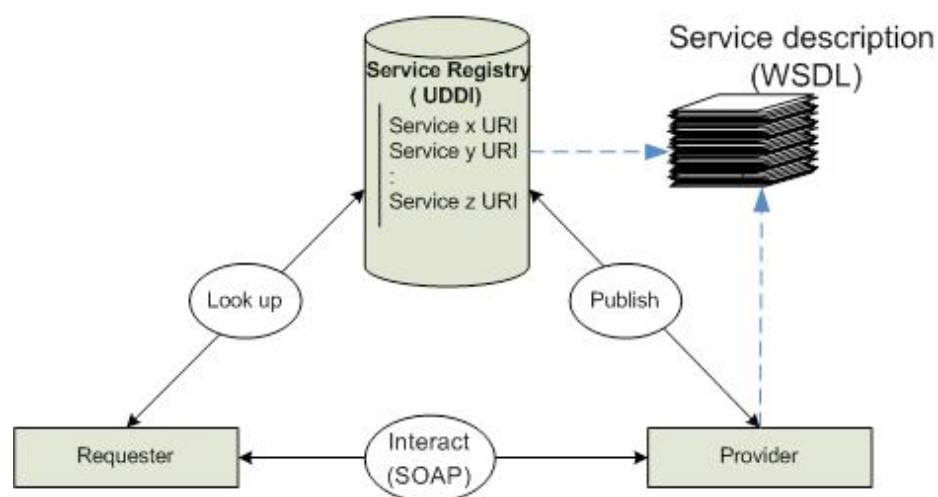


Figure 2-1: Web Service Architecture

Service composition involves service lookup and selection in addition to the act of composing. Although there is an increase in popularity, several barriers to adoption

exist including organizational complexity, the need for manual intervention and a lack of application support (such as easy to adopt tools) (Gedda, 2007). In particular, the need for manual intervention in discovery and adoption stands out as a challenge - Web Services cannot be automatically discovered and composed as the description of those services is not rich enough in its semantics (Martin, 2007a).

Delivering semantics into Web Services can be achieved through annotating a Web Service description to a suitable ontology (Sheth, Verma & Gomadam, 2006) – this is the basis of the so called Semantic Web Services (SWS) (Bruijn et al., 2009). This point is explicitly recognized by the Semantic Web community (Berners-Lee, Hendler & Lassila, 2001; Shadbolt, Hall & Berners-Lee, 2006), who argue that full automation of service discovery and composition is indispensable and is necessary for dynamic, flexible and machine understandable services and, as a consequence, an infrastructure that meets the business ideal (Maedche & Staab, 2003). Embedding semantics on to Web Services implies automation of Web Service tasks, primarily service discovery, execution and composition (McIlraith, Son & Zeng, 2001).

Without the full automation of Web Service tasks (Fensel & Bussler, 2002; Studer, Grimm & Abecker, 2007), Internet-based e-commerce will not reach its full potential in economic extensions of trading relationships. A number of approaches proposed for SWS rely on using ontologies as a core component (Martin, 2007a; Lara et al., 2004; Shafiq, 2007; Bell et al., 2007). As an example, the semantic Web Service framework, introduced by Medjahed, Bouguettaya & Elmagarmid, (2003) uses ontologies for describing semantic and syntactic features of a Web Service and presents a set of compatibility rules for automating service composition. By enabling dynamic and scalable cooperation between different systems and organizations (Davies, Studer & Warren, 2006; Bruijn et al., 2009), the significant impact of the SWS on many Web areas, such as e-Commerce and Enterprise Application Integration, becomes clear.

Services allow organizations to communicate data without the intimate knowledge of each other's IT systems behind the firewall, requiring human intervention in the communication process. Distinctively, SWS are a means for businesses to dynamically communicate with each other and with clients (Papazoglou & van den

Heuvel, 2007; Yu et al., 2008; Martin, 2007b; Bruijn et al., 2009; Sabou & Pan, 2007) whilst overcoming the manual human intervention bottleneck.

Moving towards the Semantic Web can be conceptualized as a semantic layer being added on to the current Web. It intends to give current Web pages a well-defined machine understandable meaning (Berners-Lee, Hendler & Lassila, 2001; Fensel & Bussler, 2002; Medjahed, Bouguettaya & Elmagarmid, 2003; McIlraith, Son & Zeng, 2001). SWS is one important application of the Semantic Web, whereby it intends to provide semantic description to current Web Services, and thereby facilitate the dynamic composition of Web Services. Even though the proposed Web Service standards are essential for Web Services, they are not sufficient to provide the full potential of Web Service (Fensel & Bussler, 2002), due to the fact that the service functionality description is limited to human interpretation to locate, select and compose the service. Consequently, there are certain main components that need to be used in order for the Semantic Web and SWS to evolve. The following sub sections gives a general overview of the core SWS components examining their relevance and how far these components have come to existence, and to what extent they can be applied to date.

2.2.1 Agents

Agents are user-generated code that can be used to surf the Web in order to answer a particular question or collect information. Currently agents are implemented specifically to cater for and access certain Web sites, i.e. a typical agent is assessed by a human (implementer) to connect and interact with the correct Web site. It would be much more beneficial if software agents were written generically as they would then be able to understand and interpret relevant web sites dynamically. To be able to do so, agents need to be able to use the semantic feature of Web pages in order to understand the pages and to perform tasks accordingly (Berners-Lee, Hendler & Lassila, 2001).

The literature elucidates that agents play an important operational role in the Semantic Web in general, and more specifically in SWS (Berners-Lee, Hendler & Lassila, 2001; McIlraith, Son & Zeng, 2001; Sycara et al., 2004; Gibbins, Harris & Shadbolt, 2004). Sycara et al. (2004) introduce the use of a middle agent broker, used

as part of the discovery and mediation mechanism between agents and Web Services. A broker is an important component of Web Service infrastructure as it acts as mediator and service discovery simultaneously. This approach implies that the broker will require a semantic layer to operate on, in order to provide the translation required if the requester and provider are using different languages. Hence, the broker acts as the intermediary to execute a request and sends the response to the requester. This implies that the requester will have a lack of knowledge regarding the service provider. Even though this broker seems tempting, if used, the SWS might lack decentralization. The alternative approach would be to use the matchmaker middle agent for service discovery, and allow the service provider and the requester to handle the translation process, in which case decentralization is expected (Sycara et al., 2004). In each of these two approaches ontologies are employed to provide agents with the required semantic information.

2.2.2 Ontology

Ontologies are the general means by which semantics are added into Web Services (Sheth, Verma & Gomadam, 2006; Akkiraju et al., 2005; Burstein et al., 2005), providing the required semantic layer for agents to operate on. Ultimately, ontologies form a vital component for recognising the SWS. Fensel and Bussler (2002) define ontologies as a formal consensual specification of conceptualization, which can be used to provide a shared and common understanding of a given domain, and is a way of defining concepts and the relationships between them. Ontologies here refer to the computational ontologies, the countable noun (an ontology), as implied in the computer science field (Guarino, 1998; Guarino, Oberle & Staab, 2009)..

The literature clearly identifies that Ontologies form an important component of the Semantic Web (Martin, 2007a; Lara et al., 2004; Shafiq, 2007; Bell et al., 2007). A simple example that illustrates its use is when two communicating organizations refer to the same concept using different names; then if one application needs to access the databases of both organizations, it needs to be able to recognise that those two concepts refer to the same subject. Therefore, this system may need to refer to an ontology file that defines concepts using a logic-based machine-readable format so that the machines would be able to resolve the name mismatch and infer whether the two concepts share the same semantics.

Ontology types can be classified by different criteria. The most prevalent are generality and level of detail (Guarino, 1998; Guarino, Oberle & Staab, 2009). Ontology types based on the level of generality as summarized in Table 2-1 are:

- Top-level ontologies
- Domain ontologies
- Task-based ontologies
- Application ontologies; where ontologies are used to represent a conceptualization of a specific domain and a specific task

Table 2-1: Summarized Ontology Types

Ontology type	Description	Example
Top level ontologies (Foundational ontologies)	Specification of a conceptualization based on linguistics independent of domain specific concepts	<ul style="list-style-type: none"> ▪ SUMO (http://www.ontologyportal.org/) ▪ DOLCE (http://www.loa-cnr.it/DOLCE.html)
Domain ontologies	Provides domain specific model describing domain concepts and relations	<ul style="list-style-type: none"> ▪ Financial system domain ▪ Life science domain
Task-based ontologies (Generic ontologies)	Describes concepts that are specific for a task	<ul style="list-style-type: none"> ▪ Web Service: WSMO ▪ OWL-S
Application ontologies	Combines domain and task specific ontologies	<ul style="list-style-type: none"> ▪ Describing a banking service in the financial domain using domain ontologies and OWL-S

Ontologies are classified by Gomez-Perez, Fernandez-Lopez & Corcho (2003) into two types (according to the level of details of specifications between terms):

- Lightweight ontologies are domain models that include taxonomic hierarchy and properties between concepts.

- Heavyweight ontologies are domain models that add more detail to lightweight ontologies by adding axioms and constraints to explicate terms.

The SWS domain ontologies provide the semantics of business data, processes and services. Ontology allows logic-based reasoning by machines – a necessary step in automating the process of service discovery and composition. This research is concerned with the development of domain specific ontology (referred to in some literature as application ontology) (Guarino, 2009).

Ontologies consist of taxonomies and a set of inference rules (Berners-Lee, Hendler & Lassila, 2001), which can be used to derive the meaning and relationship among objects. This meaning can then be applied during data exchange to result in a more appropriate interpretation for both parties involved. By describing service information using formal languages like description logic, machine processable reasoning capabilities can be used to enable the automation of Web Service usage tasks (Bruijn et al., 2009). For this reason research interests are widening in the ontological engineering community, producing new methods and techniques to assist in the automatic knowledge acquisition process from existing data sources (Gomez & Manzano, 2004; Gasevic, Kaviani & Milanovic, 2009).

A number of proposed approaches seek to add semantics to Web Services either as a formal ontology as in WSMO and OWL-S (Lara et al., 2004; Shafiq, 2007), or by annotating WSDL files with one of the aforementioned formal ontologies as proposed in SAWSDL (Al Asswad, de Cesare & Lycett, 2009). Fensel and Bussler (2002) propose a conceptual Web Service Modelling Framework (WSMF) for developing, describing and composing Web Services. In WSMF, ontologies are presented as an essential element required for the development of a Semantic Web Service framework. Another proposed ontology-based framework for the automatic composition of Web Services is introduced by (Medjahed, Bouguettaya & Elmagarmid, 2003); this contribution focuses on three main steps towards automatic Web Services. The first is a composability model which checks whether two services can interact with each other. The second is an automatic generation of composite services. The third step is a prototype implementation and experiment.

Table 2-2 summarizes the main approaches and presents a general comparison between them as reviewed in Bruijn et al. (2009), Al Asswad, de Cesare & Lycett (2009) and Cabral et al. (2004). A general Semantic Web Service infrastructure categorizes three main elements (Cabral et al., 2004):

- 1. Usage activities:** Define functional requirements that should be supported by any SWS framework.
- 2. Architecture:** Defines components required to undertake the usage activities.
- 3. Service ontology:** Aggregates all concept models that describe SWS. The ontology also contains the knowledge-level model that describes and supports service discovery and composition.

Service ontologies integrate information defined by SWS standards such as UDDI and WSDL with related domain knowledge. This information described by the service ontology can be distributed in different levels of ontologies (Sheth, Verma & Gomadam, 2006); Business level, Physical level and Conceptual level. Service ontology is required to describe the capabilities and restrictions of the service by providing a semantic description for the following service information:

- Functional capabilities
- Inputs/Outputs
- Preconditions/post conditions
- Non-functional capabilities such as category, cost and quality of service
- Provider related information such as company name, address, task or goal related information
- Domain knowledge defining, e.g. the type of service inputs

Table 2-2: Summarized Approaches to SWS

Approach	OWL-S	WSMO	IRS	SWSF	SAWSDL
Stands for	Web Ontology Language for Web Services	Web Service Modelling Ontology	Internet Reasoning Service	Semantic Web Services Framework	Semantic Annotation for WSDL
Based on	DAML-S	WSMF	UPML	SWSO	WSDL-S
	(DARPA Agent Markup Language)	(Web Service Modelling Framework)	(United Problem Solving Method Development Language)	Semantic Web Services Ontology	Web Service Description Language - Semantic
Execution Platform	Works with Protégé as Plug-in Editor.	WSMX (Java)	N/A	N/A	N/A
Concept	Agent oriented approach to SWS. Provides ontology for describing Web Service capability.	Business oriented approach to SWS, focus on set of e-commerce requirements for WS including trust and security.	Knowledge-based approach evolved from reusable knowledge components.	Based on Process Specification Language (PSL), supports reasoning over service description	Lightweight Web Service description that extends WSDL and can be mapped to another task ontology like WSMO
Example Citation	(Martin et al., 2004)	(Fensel & Bussler, 2002)	(Motta et al., 2003)	(Battle et al., 2005)	(Farrell & Lausen, 2007)

An ontology that can be used to describe the functional and non-functional aspects of the Web Service domain remains very expensive to develop, since it has to be derived from business data using domain expert knowledge. Current generic ontologies (the so called Task ontologies), like OWL-S (Sycara et al., 2004), attempt to provide service descriptions at different levels but still need to be linked with domain specific ontologies that describe domain specific concepts and relations. The literature emphasises the use of ontologies as a main component in all of the proposed Semantic Web Service approaches and also that ontology development remains a restricting bottleneck.

2.3 Tools used for Ontology Development

Defining ontologies for SWS requires the use of an appropriate language that provides the capability to describe concepts and relations. A number of ontology

languages and supporting tools are evolving rapidly. Resource Description Framework (RDF) is the first knowledge description standard introduced for the Semantic Web, RDF is the basic building block for supporting the Semantic Web (Yu, 2007) and is based on XML: It uses triples consisting of resource, property and statements to formulate the knowledge that machines can understand (Berners-Lee, Hendler & Lassila, 2001). RDF is extended and followed by a series of ontology languages. The first extension to RDF was the RDFschema (RDFS), but the RDFschema lacks the ability to express complex and richer relationships between classes. The RDFschema is extended to cater for the new features by adding new constructs for expressiveness, thereby leading to a richer ontology language. Hence, a new Web Ontology Language (OWL) (Antoniou & Harmelen, 2009) emerged in three different forms; OWL-Lite, OWL-DL and OWL-full. The different forms were introduced by the W3C as different sublanguages that vary in the expressiveness of the modelling primitives offered and the reasoning capabilities in each form. Typically the choice is made by the user based on the tradeoffs between the expressive power and the efficient reasoning support made in each OWL sublanguage.

Moving on from OWL, there was the need to express Web Services semantic features to allow for the automatic discovery, invocation and composition of Web Services, hence OWL-S was introduced as a Web Service description language with the semantic capability (Sycara et al., 2004) to assist in those tasks. OWL-S is structured into three main parts:

- **Profile:** This part provides the description of the Web Service capabilities.
- **Process model:** The service provider describes its computation, makes it publicly available and provides an interaction protocol used between the provider and a requester
- **Grounding:** This part provides a description of simple process transformation into remote procedure call

Ontology development, however, remains a wide-open research area in which a number of tools and methods have been introduced for the manual acquisition and

construction of ontology models. For example On-To-Knowledge, a process-oriented methodology for introducing and maintaining ontology-based knowledge management systems (Staab et al., 2001). This process is supported by a Tool (OntoEdit). The proposed approaches are considered ontology-engineering tools, developed to manage the construction and visualisation of ontologies, with some differences such as the degree of compatibility, availability of query engines and reasoners. Taniar and Rahayu (2006) state that the most cited ontology-editing tools are OntoEdit, Protégé-2000 and WebODE. Some of the tools are open source and have matured, enabling wider research and a number of plug-ins to be made available. Protégé is an open source ontology development environment and supports different OWL forms. Providing visual support and offering different reasoning and inferencing capabilities, through a number of plug-ins, makes Protégé a preferable ontology development candidate for most of the current research.

2.4 Ontology Development Challenge

Currently, domain ontologies are developed manually through collaboration between highly skilled domain experts and ontology engineers. By its very nature, ontology building is therefore an expensive and time consuming task that lacks the appropriate automated knowledge acquisition support tools (Buitelaar & Cimiano, 2008). In all of the proposed ontology development approaches, manual knowledge extraction from legacy systems and conceptually modelling this knowledge remains a bottleneck, that provides a considerable barrier to adopting SWS, consequently preventing Web Services from reaching their full potential (Martin, 2007a; Martin, 2007b; Gedda, 2007).

The challenge in achieving the SWS is, therefore, to develop ontologies from existing services. Thereby, enabling those ontologies to adapt and evolve in line with the domain and any demands made on it (Cuel et al.2008). Existing Web Service sources offer a good starting point for ontology learning (Sabou et al., 2005) and a pragmatic way forward in developing semantics for existing assets. This avenue is not well explored however. Adopting knowledge extraction techniques in the form of Ontology Learning provides an automated means of dealing with the manual ontology extraction and building, as it allows automatic knowledge acquisition from

different sources of Web content for the purpose of facilitating the process of ontology development (Buitelaar, Cimiano & Magnini, 2007).

Web Services need to be described at different levels; therefore, for ontology engineers to build ontologies that represent faithfully the knowledge embedded in these services, it is important to expose the new legacy systems available in different parts of the Web Services.

The literature highlights the importance of a faster ontology development process. Manual ontology acquisition is a tedious expensive task that can slow down knowledge acquisition (Maedche & Staab, 2001). Ontology learning can be used as an important step in an ontology development cycle. It could add an enormous power to the Semantic Web by contributing towards low cost ontology development (Maedche & Staab, 2001).

2.5 Ontology Learning

Ontology Learning (OL) is an automated or semi-automated process in which ontological elements such as concepts and relations are extracted automatically from different resources (Buitelaar & Cimiano, 2008). Ontology learning is still a long way from being fully automatic, but is now considered as a plug-in in the ontology development cycle (Maedche & Staab, 2001; Buitelaar & Cimiano, 2008; Staab & Studer, 2004; Shamsfard & Barforoush, 2003). Ultimately, it can be used to support ontology engineers in defining the conceptual model of a particular domain (Buitelaar & Cimiano, 2008).

Cimiano (2007) suggests an ontology learning layer cake (as shown in Figure 2-2), contributing to a better understanding of the OL tasks. This ontology learning layer cake as proposed by Cimiano (2007) can be used to classify an OL approach according to the task that it aims at. These tasks are described below:

- **Term extraction**, as shown in Figure 1, is the first task of an ontology learning system. The task here is to determine the relevant phrases and terms for a specific domain. Typically, a textual corpus is used as the input for term extraction.

- **Synonym discovery** consists of finding synonym words for concepts. Here two words are regarded as synonymic if they share a common meaning. This definition is similar to the *synsets* in WordNet, and WordNet is commonly used for this purpose.
- **Concept formation** is defined, for ontology learning, as a set of triples consisting of concept intension, extension and lexical realization in a corpus. Concept extensions are defined as a set of instances for a concept. Whereas concept intensions represent a shallow description of the concepts which could be taken from a dictionary. The lexical realization is the term defining the concept from the corpus.
- **Concept hierarchies** involve putting each concept in the correct place in a hierarchy. This is considered to be an important task in the ontology learning process, since it provides the taxonomic layer of the ontology.
- **Relations** learning involves finding relationships among concepts. There are different types of relations, for example, in the case of binary relations appropriate domain and range have to be identified. These types of relations are commonly referred to as non-taxonomic relations (Cimiano. 2007).
- **Rules** are concerned with the axiomatic definition of concepts. The task in this layer is to learn the rules that apply for concepts and relations. For example, there is a need to learn which pairs of concepts are disjoint, or to learn whether a relation is symmetric or non-symmetric.

The OL tasks are ordered in the way that each layer is built depending on the output of the lower layer, i.e. a concepts hierarchy learning task can only be achieved if the appropriate concepts are first extracted. The same applies for the relations learning task. Any OL methodology typically follows the layers conceptual dependency (Cimiano et al., 2009).

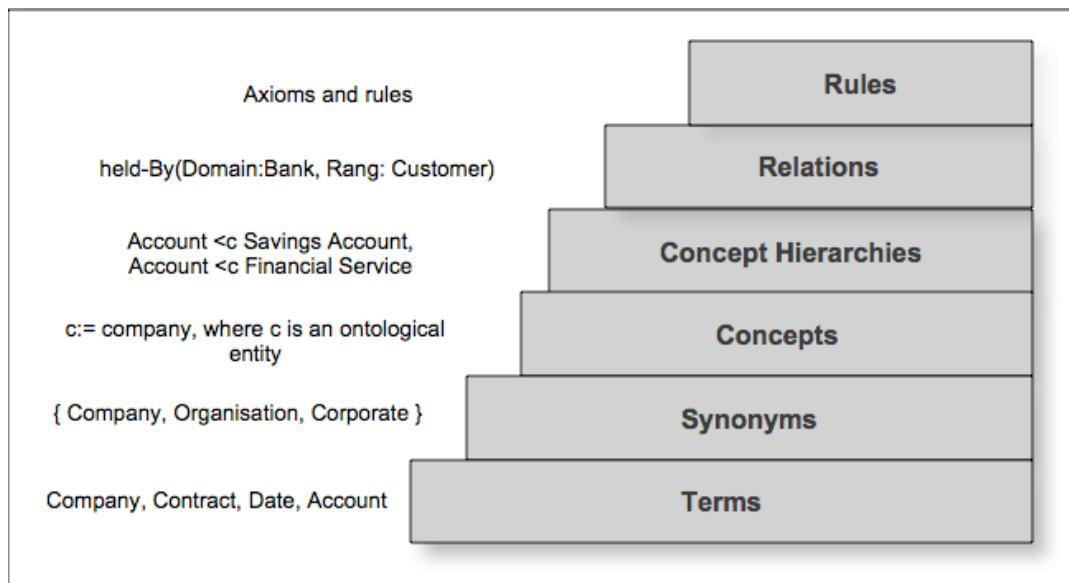


Figure 2-2: Ontology Learning Layer Cake (adopted from Cimiano, 2007)

In broad terms, Ontology Learning (OL) is grounded in a combination of Ontology Learning Techniques (OLT). Most of these techniques are drawn from well-established disciplines such as Machine Learning (ML), Natural Language Processing (NLP) and statistical-based learning (see Gomez & Manzano 2004; Zhou 2007; and Buitelaar & Cimiano 2008, for review). Each of these approaches are mainly aimed at learning the concept, relation and concept hierarchy tasks in the layer cake, but none of the proposed approaches yet tackles all of the tasks identified in the layer cake, requiring human validation or involvement in the ontology development process (i.e. they are considered as being semi-automatic ontology learning and still a long way from being fully automatic).

2.5.1 Text-based Ontology Learning Approaches.

This section explores the learning methods and tools used mainly to learn ontologies from textual unstructured data. Generally, ontology learning can be regarded to some extent as a reverse engineering process. The challenge of ontology learning from text is to derive meaningful concepts, on the basis of the usage of certain words in the text, and to represent them in a hierarchical organization. These approaches usually involve applying a mixture of knowledge engineered rule-based techniques and machine learning techniques in order to learn relations and concepts, thus enabling concepts to be interpreted by defining their relation to other concepts in the form of logical axioms (Cimiano, 2007).

Different learning approaches have been introduced over the last few years that support ontology engineers in developing domain ontologies semi-automatically from textual sources. To name a few, Text-to-Onto (Maedche & Volz, 2001), OntoLt (Buitelaar, Olejnik & Sintek, 2004) and OntoLift (Volz et al., 2003) are all aimed at extracting ontological knowledge from textual sources by applying a mixture of knowledge extraction and text-mining techniques. These approaches can be further classified according to the type of techniques used and in some cases a mixture of more than one can be adopted as discussed in Section 2.6. A number of survey papers and reviews present comparisons between OL textual-based approaches (e.g., Gomez & Manzano, 2004; Zhou, 2007). Each approach shows only limited success (Pivk, Cimiano & Sure, 2005; Pivk et al., 2007), however, and they are far from being capable of tackling all of the tasks in the OL layer cake.

2.5.2 Learning Approaches Based on Semi-structured Data

Here, semi-structured data sources are used to refer to documents that have a mixture of text and template structure, such as tables or XML/HTML schema (Antonacopoulos & Hu, 2004). HTML tables would be considered as semi-structured data since they usually contain a mixture of tabular structure and text (Jung, Kang & Kwon, 2007). Web tables have a tabular structure and an internal hierarchical semantic layer. A number of approaches are proposed that attempt to extract ontology knowledge from data sources that are categorized as semi-structured documents.

Jung, Kang & Kwon, (2007) present an approach that is mainly based on mapping different types of table schemata that are extracted from Web documents belonging to the same domain, into a domain ontology. This approach mainly aims at constructing domain ontologies by combining table schemata extracted from tables belonging to a specific domain where hierarchical clustering is applied for the construction of domain ontologies. Similar work aimed at semi-structured sources was introduced in Pivk, Cimiano & Sure, (2005) and improved by Pivk et al. (2007). This approach analyses the different characteristics of a table and converts the outcome to an F-logic frame. The approach can be considered as a starting point towards extracting ontologies from table structures. This work is limited to being useful as a means of ontology population rather than ontology learning, however.

Approaches that fall under this category are all aimed at mapping the structure (schemata) of a Web document into an ontological hierarchy/taxonomy, but neglect the domain knowledge available as text in such sources. An approach that is targeted at extracting knowledge from document structures as well as from knowledge embedded in the text is therefore required. Web Service artefact sources are rich in semi-structured sources, and if any progress is to be made in domain ontology development for Web Services, it is vital that this area rigorously explored.

2.5.3 Learning Approaches Based on Structured Data

Structured data in this case is used to refer to data which are highly structured and mostly generated from databases. Relational databases are considered to be an essential component in modern Information Systems. Therefore, relational database schemata are considered to be a significant source for ontology extraction. In these types of data sources, data is stored based on logical schemas which provide some conceptualization about the domain in which the given information system operates. Ontologies have been used for mediation between different databases. These types of approaches can be considered as mapping approaches (Li, Du & Wang, 2005), since most of the concepts and relations would already be described in legacy systems.

An interesting method that can be adopted in an ontology learning process can be inferred from Johannesson (1994). In this approach, a method was introduced to extract a conceptual schema from a relational schema. Basically the challenging task was to map concepts and relations from the relational databases conceptual level into an ontological representation. This method can be applied to create a middle model representation of the relational database; an example of an ontology learning approach that applies a middle model as the method is presented by Kashyap (1999). Another approach, introduced by Pan & Pan (2006) which is basically a framework for the data-mining process, is based on using an ontology repository to integrate domain knowledge. Other approaches which are aimed at OL from structured sources by applying learning rules in order to map relational database elements into ontological elements are presented in Li, Du & Wang (2005) and An et al. (2007).

A number of tools and approaches have been developed for this purpose, including RDBToOnto (Cerbah, 2008) and OntoLift (Volz et al., 2003). In the latter tool the

lifting process tries to capture the semantics of the databases by mapping relations to concepts and attributes to roles in the ontology model. Of note, all of the proposed OL approaches apply learning techniques borrowed from existing information extraction and artificial intelligence disciplines. The techniques predominantly applied are discussed in the following section.

2.6 Overview of Ontology Learning Techniques

This section introduces commonly used techniques in ontology learning, classified according to the disciplines from which these techniques are borrowed (Maedche & Staab, 2004). There are a number of surveys and comparison articles on the state-of-the-art in ontology learning (Maedche & Staab, 2001; Shamsfard & Barforoush, 2003; Gomez & Manzano, 2004; Zhou, 2007) each of which provide different comparison criteria. A broad overview of each learning discipline is given in the following subsections.

2.6.1 Machine Learning Techniques

Machine Learning (ML) techniques are used to automatically detect and recognize specific patterns and regularities in example data (Cimiano, 2007), which are then used to make predictions. ML is based on induction or generalization using sample data, with learning typically classified as supervised and unsupervised. Supervised learning requires manually tagged training data and is based on an understanding of the tasks that data are applied to and a given learning paradigm. A popular supervised classifier example is the weather example (Witten & Frank, 2002), where training data is represented as vectors for input data and target values represent outputs, as illustrated in the three given training sets:

(sunny,not-windy,warm)→ play outside.

(rainy,windy,cold) →do not play outside.

(rainy,windy,warm) → play.

These training sets can then be used by the learner to infer certain rules (or mapping functions) such as: IF temperature = warm THEN play. In contrast, unsupervised learning does not require any training data and is mostly applied in discovering taxonomic relationships among concepts in order to classify them into meaningful categories (Witten & Frank, 2002). Importantly, it is this latter type of ML that is

commonly applied in the OL field (Cimiano, 2007). For example, clustering can be applied in unsupervised ML and is basically aimed at grouping similar objects in the data set. If hierarchal clustering is used then groups are organised in a hierarchal structure. A comprehensive review of all available ML approaches and methods is presented in Gomez & Manzano (2004).

2.6.2 Statistical Analysis

A statistical analysis model is usually represented as a network that indicates the probabilistic dependencies between terms (Zhou, 2007). Generally, the statistical information computed from observed frequencies of the term within a corpus is used to detect new concepts and relations relevant to the domain represented in the underlying corpus. A technique used here is frequency analysis of word repetition. Other methods include: (a) Naïve Bayes (Sanderson & Croft, 1999) which is used for learning classifications; and (b) statistical hypotheses testing, which is used for testing whether or not two concepts occur more frequently together (Cimiano, 2007).

2.6.3 Linguistic Techniques

Natural Language Processing (NLP) techniques are typically applied as a pre-processing step in any OL system, in which textual input data is semantically analysed and transformed into tagged output using a sequence of pipelined steps. Popular techniques applied for the pre-processing step include tokenization, part of speech tagging, stemming and lemmatization (Buitelaar, Cimiano & Magnini, 2007). Tokenization, for example, is used to identify words and sentences within texts. Typically, with unstructured text this activity involves using obvious word separators including spaces, full stops and commas to split sentences into tokens. Part of speech tagging implies differentiating syntactic categories such as nouns, verbs and adverbs that lead on to semantic analysis. In broad terms, these syntactic techniques are able to identify different ontological elements, with proper nouns, for example, being used to identify instances. The pre-processing step is essential for all OL approaches, especially if the source data is a textual document (Maedche & Staab, 2001). A number of the learning approaches apply linguistic techniques have been previously discussed, which are summarized and compared in Gomez & Manzano (2004), Zhou (2007) and Cimiano (2007).

2.6.4 Rule-based Techniques

Rule-based techniques typically involve matching predefined rules or heuristic patterns in order to extract relative ontological elements, mostly terms and relations. In the OL application area these techniques usually rely on knowledge engineers to identify lexical patterns and hand-crafted rules as applied in Text-To-Onto (Maedche & Volz, 2001). Rule-based techniques are widely applied as pattern-based matching information extraction methods. These methods are widely used for the extraction and transformation of concepts and relations from unstructured sources (Buitelaar & Cimiano, 2008; Cimiano, 2007; Borislav et al., 2004).

Lexico-syntactic patterns as introduced by Hearst (1992), are often applied in relation extraction from textual sources, e.g. finding semantic relations between noun phrases in the text can be achieved by finding matches to lexico-syntactic patterns in the form of regular expressions as in the following pattern:

$$NP_0 \text{ such as } \dots \{NP_1, NP_2 \dots (\text{and|or}) \} \dots NP_n$$

Here, a noun phrase (NP) is identified as a hyponym within a corpus – one example being animal and horse. Hearst's (1992) work aims at identifying patterns leading to hyponymy relation extraction. Examples of how this work has been extended and applied include: First, identifying patterns that target taxonomic knowledge (Iwanska, Mata & Kruger, 2000). Second, extracting part-of relations (Berland & Charniak, 1999). Third, investigating texts surrounding images (Ahmad et al., 2003).

Lexical syntactic pattern identification has been widely reported (Buitelaar & Cimiano, 2008; Cimiano, 2007; Borislav et al., 2004; Giovannetti, Marchi & Montemagni, 2008), including syntactic patterns in OL from specific Web Service domains. Such patterns are applied extensively in OL from unstructured sources of Web Services as proposed by Sabou (2005). The rule-based techniques are widely applied in information extraction providing accurate and promising results leading to increased precision (Cimiano, 2007; Buitelaar & Cimiano, 2008; Giovannetti, Marchi & Montemagni, 2008). These pattern-based techniques are classified as knowledge engineering approaches requiring domain engineers to analyse the textual

sources to identify patterns and engineer transformation rules, in which the difficulty remains in finding the patterns that frequently and unfailingly denote the relation.

Unsurprisingly, there is often a significant overlap between these disciplines in practice. For example, statistical techniques are combined with machine learning and classified as such in some literature (Cimiano, 2007). Linguistic-based methods are commonly applied with statistical approaches to calculate the relevance of the concept to the given domain, these methods include techniques based on linguistic patterns, pattern-based extraction, methods that measures the semantic relativeness between terms within a domain, etc. (Gomez & Manzano, 2004; Cimiano, 2007; Zhou, 2007). In some approaches a combination of all three types are applied. Text-To-Onto (Maedche & Volz, 2001) and OntoLearn (Navigli & Velardi, 2004), for example, use statistical techniques applied with machine learning algorithms. Other approaches combine linguistic analyses methods and machine learning algorithms, including OntoLt (Buitelaar, Olejnik & Sintek, 2004) and ASIUM (Gacitua & Sawyer, 2008).

One important point of note, however, is that it is clear that most comparative surveys compare text-based approaches and that there is little work focusing on comparing learning from unstructured sources versus learning from structured sources. Web Service sources resemble a specific domain in which an effective OL approach needs to be tailored to cater for the specific nature of these sources. This tailoring involves applying a combination of techniques, including a pre-processing step to produce syntactically analysed data, followed by the application of an efficient combination of ML and statistical techniques that are applicable in the Web Service domain. Determining a suitable OL technique applicable on the Semantic Web Service sources is discussed in the next section.

2.7 Related Work / Ontology Learning for Web Services

Very little work exists that aims at ontology learning from Web Service sources. Work found on OL from Web Service sources can be classified in two forms; the first is one that investigates structural aspects of structured sources. The second form is work that is aimed at learning from textual sources of Web Services. It is clear that

most of the OL approaches are based on the general OL framework presented by Maedche & Staab (2001).

In light of this, the approach introduced by Sabou et al. (2005) applies NLP to textual description, and therefore learns Web Service ontologies from textual descriptions attached to implementation files (i.e., Javadoc). Noun phrases and service functionality are learnt from verbs by applying a preprocessing pipeline on textual descriptions of Web Services. Linguistic techniques are then applied in order to extract syntactic patterns and apply dependency parsing. The limitation of this work is that it is confined to Javadoc files, which are not a common means of description in Web Services (Guo et al., 2007). The focus on extracting concepts and service functionality from textual description only, whilst ignoring the structural aspect of the Javadoc file, can be improved and extended by considering other Web Service sources, such as structured sources as in WSDL and XSD documents.

On the other hand, using the structural aspect of Web Service sources that maps WSDL schema onto ontologies are attempted in some approaches, such as the method proposed by ASIUM (Faure & Nédellec, 1998); nevertheless, the relation extraction is restricted to learning taxonomic relations from the WSDL structure only. This can limit the learning to service functionality rather than the domain specific non-taxonomic relations. These relations implicitly exist in the method names or input/output parameter names in WSDL and XSD files. This area still needs to be explored and is mainly addressed by this research.

Capturing the relationships between WSDL elements and transforming them into ontological concepts and relationships, by looking only at simple pattern detection, is shallowly attempted in Guo et al. (2007), where a limited number of simple transformation rules are applied only on the source WSDL documents. Although WSDL documents provide important application level service descriptions, they alone are not sufficient for OL as: (a) They provide technical descriptions only; and (b), in many cases Web Services use XSD files to provide data type definitions. The need to include other Web Service resources in the OL process is therefore an important one that has not yet been achieved. Most work of this nature is aimed at Web Service matching rather than the domain ontology learning itself.

Other reported work that attempts to combine different input sources to learn domain ontology is Latino (Bontcheva & Sabou, 2006). The method applied in Latino is based on creating a document network ontology where concepts are learned from classes in Java code. This work is potentially useful as a conceptual search in a search engine like Google. The method does not apply any pattern-based knowledge extraction to extract text in semi-structured sources.

Given the aim of automatically learning ontologies from Web Services, this review illustrates two main points:

- There is a need to clarify and address the demands on OL in light of the mix of (semi-) structured elements that typically accompany Web Services.
- There is a need to investigate the appropriate mix(es) of OL techniques in meeting those demands.

Both points are illustrated in Figure 2-3 – highlighting a need to identify techniques for effectively combining a range of Web Service software artefacts with appropriate OL methods.

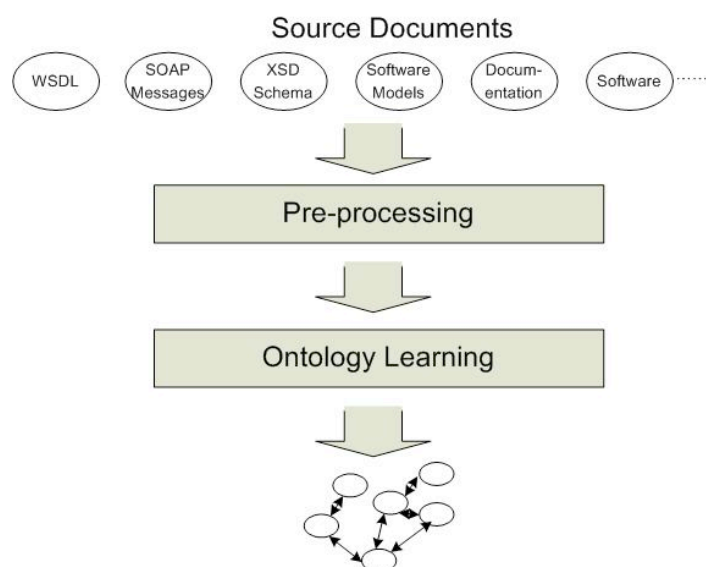


Figure 2-3: Ontology Learning from Web Service Source Artefacts

The choice of an ontology learning strategy, whether it is bottom-up or top down, can be identified based on the data sources and domain (Zhou, 2007). Web Service sources are diverse in a number of areas, containing both structured and unstructured data and generating both static and dynamic sources. WSDL and XSD files are examples of static data sources, with WSDL files providing a usable source of service interface information, including inputs, output and basic service functionality. SOAP messages, dynamically generated by Web Services and client applications in use, contain instances of server requests issued by clients and instances of service responses issued by service providers. Messages are created when a service is invoked and are an example of a dynamic source. Extending the work by Guo et al. (2007) to include XSD schema and SOAP messages may offer a number of interesting opportunities – revealing additional concepts and relations through more complex transformation rules. For example, WSDL structures may be transformed into ontological relationships, elements are analysed so that the “message : parts” relationship is transformed into “has property”. Applying similar, but more extensive, transformation rules to XSD and SOAP may result in more effective methods. Possible opportunities include: (1) domain specific rules, (2) advanced source document pre-processing heuristics and (3) source document bootstrapping approaches. WSDL files alone are typically limited to only providing a technical description of the underlying service.

Support for a variation in Web Service style is also appropriate. When interpreting document style Web Services, a major part of the service description is found within the referenced XSD schema (Curbera et al., 2002). Interpreting the underlying schema in unison with other Web Service artefacts would result in a considerable increase in the number of identified concepts (when compared to interpreting WSDL in isolation). Moving beyond the service description and exploring dynamic SOAP analysis allows executing services to be interpreted and opens further avenues for ontology learning. Service invocation and messaging, via SOAP messages, provides related instance data for each service description. It is this instance data that has the potential to provide opportunities for revealing additional relations, axioms and patterns (Daga et al., 2005).

Current OL approaches are in the most part general, and need to be specialised to cater for both the technology of the Web Service domain and the business domain in which these services operate. Identifying efficient learning techniques that are applicable in the Web Service domain is a challenging task. Learning techniques from different paradigms need to be combined and tested on varied sources in order to identify effective multidisciplinary techniques aimed at ontology learning from Web Service artefacts. A number of research questions arise and can be categorized according to Web Service source documents, pre-processing requirements and Ontology learning techniques. In order for any progress to be made in achieving the SWS, domain ontologies need to adopt and evolve with legacy systems, dealing with current Web Services standards.

2.8 Summary

The literature has illustrated the need for Semantic Web Services, indicating the realization of the importance of Web Services and its capability of reaching its full potential through the SWS. Understanding the varieties of Web Service sources and analysing the role of OL in the Semantic Web have provided a deeper understanding of the need to apply OL on Web Services in order to advance the SWS uptake. The literature review classified OL techniques and approaches and identified applicability on different data sources. It is clearly confirmed in the literature that ontology development is a costly and time consuming process, requiring the services of highly qualified expertise both in ontology engineering and the domain of interest. A wide spread adoption of ontology development can be very difficult to achieve. Ontology learning can assist in this direction by introducing some sort of semi-automatic knowledge extraction that can be used by ontology engineers for speeding up the process of ontology construction (Davies, Studer & Warren, 2006). Web Service artefacts form a vital source of domain knowledge. For progress to be made in the SWS, it is fundamental to rigorously explore OL from these sources. Since most of the research is carried out on ontology learning from text, there has been less work completed on mixing techniques and developing ontology learning methods for combining Web Service data sources. Consequently, combining OL techniques and approaches that deals with the differing characteristics of these Web Service sources remains an open research area.

CHAPTER 3 – DESIGN RESEARCH METHODOLOGY

3.1 Introduction

In any given discipline the research community agrees upon the set of systematic activities considered suitable to the production and validation of knowledge. In a multidisciplinary paradigm like Information Systems there exist a number of research methods. These methods differ in fundamental ways, among them the phases employed, techniques, philosophical aims and structure of those phases. This chapter investigates and presents Design Research as the chosen methodology to execute this research, detailing the phases, techniques and philosophical background behind Design Research. Design Research employs a set of techniques to implement research in Information Systems. Normally this entails analysing the use and potential of the designed artefact. Discussing Design Research as a valid and legitimate IS research demonstrates the justification behind choosing Design Research as the framework that guides the research execution.

In this chapter, Section 3.2 introduces the background to Design Research with reasoning behind the validity of design as a research method. Design Research in general as a methodology for Information systems research is described in Section 3.3, giving a broad review of major Design Research frameworks in IS and detailing the main strategy in those frameworks. Section 3.4 presents Design Research evaluation criteria associated with Design Research artefacts and typical evaluation methods. While Section 3.5 presents the design plan for this thesis and explains how Design Research is applied for the execution of the research, Section 3.6 introduces the research evaluation giving a general background of OL evaluation. Section 3.7 illustrates the three Design Research iterations for the thesis, and finally, section 3.8 summarizes the chapter.

3.2 Design Research Background

Information Systems design is defined as “the purposeful organization of resources to accomplish a goal”, (Hevner et al., 2004). It is important to discuss how design can be incorporated as a research method. Hevner et al. (2004) categorize research as

an innovative way of solving a problem, where Edelson (2002) and Winter (2008) distinguish Design Research by the generality of the proposed solution in a sense that it can be applied to a wider class of situations therefore leading to design science.

Simon (1996) makes a valid differentiation between behavioural science and design science, in unfolding the science of the artificial, Simon introduced the notion of an artefact, viewed as a link between the inner and outer environment in the search for a solution that fulfils the desired goal in the search for a satisfactory design rather than an optimal one. Design is a learning process through which the underlying artefact development process is observed differently and learned from.

Design Research as presented by March & Smith (1995) marked a new research era where it enabled research to achieve both relevance and effectiveness by combining research output (product) and research processing (activities) from behavioural and design science in a two-dimensional framework, as presented in Figure 3-1. The four research activities drawn from design science and natural science are Build, Evaluate, Justify and Theorize. These four processes are applied in IT research to produce different types of artefacts; constructs, models, methods and instantiations, and these artefacts are employed to ensure the utility and efficiency of the produced Information System. Design Research achieves an optimal solution to the design problem in an iterative knowledge refinement manner.

		Research Activities			
		Build	Evaluate	Theorize	Justify
Research Outputs	Constructs				
	Model				
	Methods				
	Instantiation				

Figure 3-1: A Research Framework (March & Smith 1995)

Categorising design artefacts using March and Smith's (1995) research outputs classification can help in identifying an appropriate procedure to build, evaluate,

theorize and justify the research. The four types of research artefacts are described below.

- **Constructs:** Constructs are sets of concepts or vocabulary that form specialized knowledge within a domain; they are used to define problems and solutions (Hevner et al., 2004).
- **Models:** Models use constructs to describe a real world situation of the design problem and its solution space (Hevner et al., 2004); models can be used to express relationships between constructs (March & Smith, 1995).
- **Methods:** Methods are a set of steps that defines the solution space. They provide guidance on how to solve problems using the constructs and the models. Methods can be thought of as methodological tools that are created by design science and applied by natural scientist (March & Smith, 1995).
- **Instantiation:** Instantiations are the implementation of constructs, models or methods within a working system. They prove the feasibility and effectiveness of the models, methods and constructs allowing actual evaluation (March & Smith, 1995). Instantiation plays an important role in enabling researchers to learn about the working artefact in a real world scenario. As Newell & Simon (1976) explain, the significance of instantiations is providing a better understanding of the problem domain and consequently to offer better solutions.

According to Owen (1998) and Takeda, Veerkamp & Yoshikawa (1990), knowledge can be generated and accumulated through a process that iterates through knowledge using and knowledge building activities. Consequently, design is considered as a process, and the steps involved in the design process are clearly identified by Vaishnavi & Kuechler (2004). Design can be employed as a research that generates knowledge. A number of research attempts to link theories and design to justify Design as a research approach leading to theories (Brown, 1992; Kelly & Lesh, 2000) while others attempt to put emphasis on the learning aspect of Design Research and identify types of learning that can evolve when a researcher emerges in the design process as demonstrated by Edelson (2002).

A general Design Research methodology that incorporates five phases of design and motivates an iterative design cycle in which learning is a key attribute is proposed by Vaishnavi & Kuechler (2004) adopted from Takeda, Veerkamp & Yoshikawa (1990). Problem awareness in this method is the initial step in Design Research, followed by a suggestion, producing a proposal and a tentative design. The third step is artefact development that may result in learning and improvement being fed back through circumscription into the first step. The fourth and most important step is the evaluation of an artefact, in which performance measures from the knowledge base could be applied to test the utility of the artefact in the problem domain. The fifth step is the conclusion, which involves highlighting the results of the Design Research adding knowledge to the solution space or feeding back to consequent cycles. Nunamaker, Chen & Purdin, (1990) agree that system development (artefact construction) is considered as a research methodology that can lead to an improved, and more effective design when applied in conjunction with other research methodologies, whilst at the same time making a rigour contribution to knowledge.

In accordance with utility and truth as two important aims of Design Research and behavioural science respectively, Design Research is proposed by March & Smith (1995) and Hevner et al. (2004) as a research framework where IT research can occur by integrating two complementary disciplines. The first of these is behavioural science where research is more focused on theorize and justify, and the second is design science research, where the research is more focused on the build and evaluate process.

3.3 Design as an IS Research methodology

Design Research frameworks attempt to provide the IS community with a Design Research methodology (Hevner et al., 2004; Nunamaker, Chen & Purdin, 1990/91; March & Smith, 1995). In those attempts, a common process is an iterative design cycle employed as a problem solving process where a valid IS research is achieved through the building and evaluation of purposefully designed artefacts. Importantly, research in Information Systems (IS) is not any different from any other research. where Blake (1978) defined research as “...systematic, intensive study directed toward fuller scientific knowledge of the subject studied”. IS Research is considered a multi-inter-related disciplinary field, made up of social and natural sciences

management and engineering, bound by an overlap in methods of research, in which continued improvement is necessary to meet the complicated dual nature of the IS field (Nunamaker, Chen & Purdin, 1990/91; Purao, 2002).

A typical research in Information Technology is one that is commonly categorized as one of two types; the first being a knowledge using action where research is aimed at improving IT performance, whilst the second type is a knowledge producing action where the research is aimed at understanding the nature of IT (March & Smith, 1995). In both cases IS research takes place as a juncture connecting people, organizations and technology, therefore, IS definitely incorporates IT research.

Simon (1996) made a clear distinction between natural science and science of the artificial (design science), where the first is concerned with naturally occurring phenomenon whilst the second relates to artificial human made artefacts. With this distinction being made clear, it has led the IS community to realize and justify the need for design as a research discipline that combines the two (Hevner et al., 2004; Edelson, 2002; Winter, 2008; Nunamaker, Chen & Purdin, 1990/91; March & Smith, 1995).

Design Research (Design Research) as an Information Systems valid research methodology, is formulated by integrating two complementing disciplines (design and behavioural science), in a way that provides the means by which an IS researcher engages in designing an artefact, hence the design science aspect, while at the same time learning is emphasized during the development process, therefore, the implication of utility on people and organization, and hence the behavioural science aspect (Hevner et al., 2004). In design science research, truth and utility are considered to be vital elements, gained through an implicit cycle between design science and behavioural science, where truth is provided by IS theories and utility is provided by IS artefacts (Hevner et al., 2004). The design cycle is executed in an iterative incremental process that can be initiated by simple conceptualization providing the necessary learning that feeds into consequent iterations, where the final iteration results in an improved product that satisfies the problem requirements and constraints. An earlier Design Research framework presented by Nunamaker, Chen & Purdin (1990/91) that connects aspects of design and design science. In their

framework, Nunamaker, Chen and Purdin (1990/91) assign system development a central role in the research life cycle, again showing an integrated approach that includes design science as a core component in an Information Systems methodological research framework. The process for conducting the research is left for the researcher to infer

Hevner et al. (2004) on the other hand propose a descriptive Design Research framework as illustrated in Figure 3-2 that satisfies both natural science and design science. Research rigour can be achieved by effectively applying knowledge (theories) from the knowledge base to develop and build an IS artefact, while relevance can be accomplished by assessing whether the artefact satisfies business needs. The justify-evaluate process is used to assess the artefact applicability in the appropriate environment.

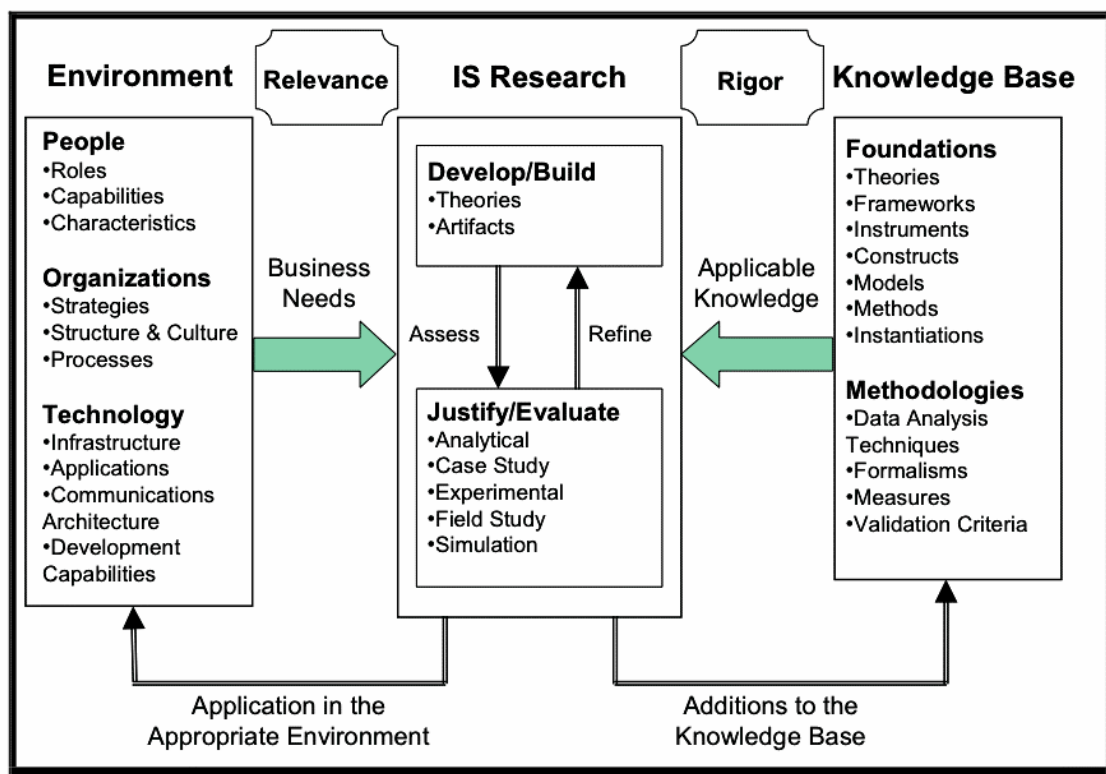


Figure 3-2: IS Research Framework (Hevner et al., 2004)

In Hevner et al. (2004) a concise IS research framework is presented and used to induce Design Research methodological guidelines that can be followed to identify, execute and evaluate IS research. Build and evaluate are considered to be an iterative process through which both method and product are carefully assessed by the

researcher and used to assess and refine the developed product. This evaluate process typically applies measures from the knowledge base to assess the utility, efficacy and quality of the designed artefact. Hevner et al. (2004) proposes a set of evaluation methods that can be used to evaluate the designed artefact discussed in the next section.

3.4 Design Research Evaluation

Evaluating a Design Research artefact is a vital phase; its importance resides in the need to determine artefact performance and measure progress according to well-defined metrics (March & Smith, 1995). Assessing the progress made in the problem space when the artefact is built to perform a specific task demonstrates its utility, and therefore, validates the research. On the other hand, evaluation plays a fundamental role on iterative research (design science) where knowledge generated from the evaluation phase can be fed back into consequent iterations. Hence, developing appropriate evaluation metrics to assess artefact performance for proving the evaluation criteria (March & Smith, 1995) is critical. Here an evaluation criteria of the so called quality attribute is identified based on artefact type as proposed by March & Smith (1995), and is summarized in Table 3-1. Generally, evaluation is concerned with answering the important question “How well does the artefact work?” (March & Smith, 1995). This can be answered by applying a suitable evaluation metric or measure from the knowledge base, thereby proving the appropriate evaluation criteria. For example, a search algorithm instantiation in the information extraction field can be evaluated by a mathematical metric such as precision and recall (Hevner et al., 2004). Therefore, these metrics can be used to prove the efficiency and effectiveness of the algorithm.

Table 3-1: Summarized Evaluation Criteria with Artefact Types (Hevner et al., 2004)

Artefact Type	Evaluation Criteria
Constructs	Completeness, simplicity, elegance, understandability and ease of use.
Model	Fidelity with real world phenomena, completeness, level of detail, robustness and internal consistency.
Method	Operationality (ability of others to efficiently use the method), efficiency, generality and ease of use.
Instantiations	Efficiency, effectiveness and impact on an environment and its users.

Once the evaluation metrics and criteria are identified an empirical work is applied (March & Smith, 1995), where an evaluation method is chosen appropriately. Hevner et al. (2004) emphasize that the selection of the evaluation method should be carefully considered, and when matched with the suitable artefact and evaluation metric evaluation methodologies are typically withdrawn from the knowledge base. An inclusive set of evaluation methodologies is summarized in Table 3-2, adopted from Hevner et al. (2004). The classifications represent the most common evaluation methods from which a suitable method/s can be applied based on the type of artefact and the evaluation metrics used.

Table 3-2: Design Evaluation Methods (Hevner et al., 2004)

Design Research Evaluation Method Types and their Description	
1. Observational	Case Study: Study artefact in depth in business environment.
	Field Study: Monitor use of artefact in multiple projects.
2. Analytical	Static Analysis: Examine structure of artefact for static qualities (e.g., complexity).
	Architecture Analysis: Study fit of artefact into technical IS architecture.
	Optimization: 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).
3. Experimental	Controlled Experiment: Study artefact in controlled environment for qualities (e.g., usability).
	Simulation: Execute artefact with artificial data.
4. 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.
5. 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.

3.5 Applying Design Research

The research contribution is the development of a methodological ontology learning framework for SWS and a tool resulting from instantiating the framework. To meet the research aim, Design Research is adopted from Vaishnavi & Kuechler (2004) as an overall research methodology. March & Smith's (1995) research products classification is adopted to illustrate the research output. Research products are identified in the form of constructs, models, methods and instantiations. The Design Research methodology employed for developing the research artefacts is an iterative design cycle (build and evaluate). In design science build is concerned with the development of the artefact, and evaluation is concerned with the development of an assessment method or metric to assess the quality and effectiveness of the artefact in its context (March & Smith, 1995). The main design artefact is a methodological

ontology learning framework, an iterative process involving the five design process steps; awareness, suggestion, development, evaluation and conclusion, as elaborated upon in Figure 3-3.

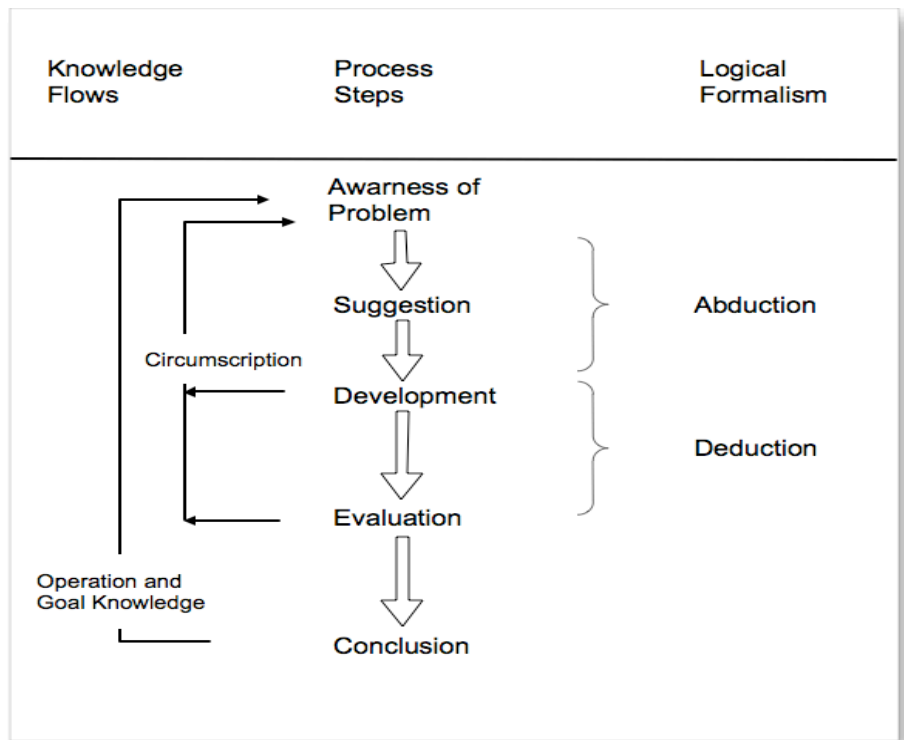


Figure 3-3: Steps of Design Research (Vashnavi & Kuhler, 2004)

An Awareness of the problem was achieved in Chapter 2. This involves reviewing the literature and analysing existing ontology learning techniques, in addition to recognising the importance of faster ontology development for SWS. It also incorporates finding suitable ontology learning techniques appropriate for developing an ontology learning framework (as detailed in Chapter 2), by comparing existing OL approaches and highlighting weaknesses.

Suggestion involves introducing a tentative idea of how the problem might be solved by signifying appropriate learning techniques (Alfaries, Bell & Lycett, 2009). This step forms Iteration 1, which develops an appropriate service term and concept extraction method, and then new suggestions arise for relation extraction in consequent iterations. As new knowledge is gained during development and evaluation of the developed method, new suggestions from the build and evaluate cycles are used to initiate subsequent iterations.

Development is carried out by building the research artefact as an ontology learning framework (SOLF). The framework consists of phases and steps that adopt the relevant machine learning and NLP techniques. SOLF is aimed to automate domain knowledge extraction from Web Services and the building of a domain specific ontology. SOLF is subsequently automated by creating an instantiation as an ontology learning tool.

Evaluation is carried out through an evaluation strategy that measures the effectiveness of the research based on the significant performance improvement of the developed framework over existing ontology learning methods and approaches. An evaluation of the automatically learned domain ontology against manually produced gold standard ontology in order to illustrate the effectiveness of the method is performed. Evaluation is carried out using Design Research evaluation criteria to examine the efficiency and generality of the framework. Automating the process of applying the method (SOLF) on a realistic Web Service scenario taken from the financial domain, resulted in the development of a tool that served as an instantiation of SOLF. Evaluating the efficiency and effectiveness of the tool developed as an instantiation of SOLF is also performed. This tool is used to validate SOLF in an experimental evaluation over different set of Web Services and gold standard in iteration three.

Conclusion is where the research output is summarized and the results of the evaluation are identified and future improvement is highlighted towards improving ontology learning from Web Services.

3.6 Research Evaluation

Two common evaluation metrics for Design Research are novelty and effectiveness (Edelson, 2002). The novelty of this work lies in developing a new framework model, designed to extract ontological knowledge from Web Service artefacts and bring Web Services to their full potential. In evaluating the novelty and effectiveness of the research, Design Research artefacts will need to be formally evaluated to determine whether progress have been made in the ontology development process within the Web Service domain.

The effectiveness of this framework is in reducing the cost and time of the ontology development process. When the research objective is to achieve intelligent behaviour, instantiations are used to illustrate the effectiveness and provide a live proof of the proposed method (SOLF in this research). It is the means by which deficiencies and improvements are identified (March & Smith, 1995). Determining whether progress is made in the OL requires applying the appropriate metrics from the knowledge base. Due to the fact that OL is a new machine learning application domain, as yet there is no optimal evaluation framework for ontology learning approaches (Dellschaft & Staab, 2008). Typically, OL evaluation methods can be classified according to the different scenarios into two main evaluating methods [ibid]. Those methods are mainly aimed at evaluating structural and functional aspects of an OL method. The evaluation methods can primarily be classified in two main types: (1) quality assurance during ontology engineering, which can be further classified into task-based, corpus-based or criteria-based evaluation approaches as depicted in Figure 3-4, and (2) comparing OL algorithms which can be either manual evaluation by a domain expert or Gold Standard-based evaluation.

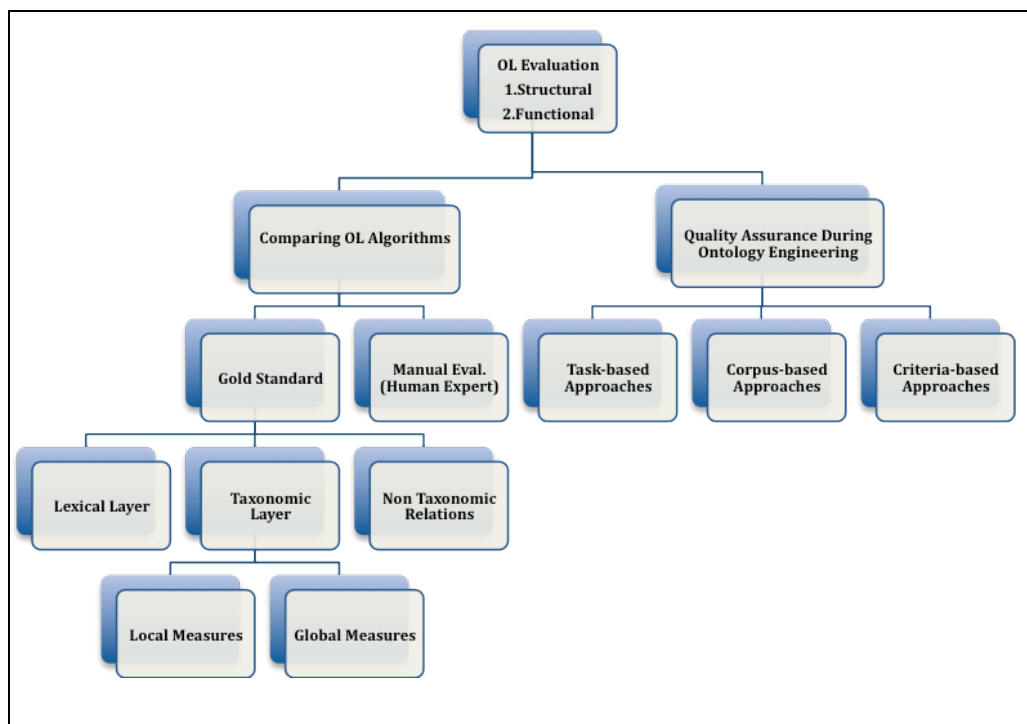


Figure 3-4: Taxonomy of OL Evaluation Approaches

Evaluation approaches can be further subcategorized according to the measure used and what they intend to measure in terms of the functional and structural aspects as summarised in Table 3-3. Generally speaking, precision and recall are the end metrics used when evaluating OL approaches either by gold standard or manual evaluation by domain expert. Here the evaluation is performed over a subset of real world commercial services. The qualitative measures are borrowed from the information extraction field, applied here to measure the accuracy and precision of automatically extracted information in comparison with manual extraction.

Table 3-3: Comparison of OL Evaluation Methods

Method/ Scenarios	Purpose/ Evaluation criteria	Description & Purpose		Pros.	Cons.
A. Quality Assurance During Ontology Engineering	Consistent,	Structural and	Whether it improves the task the ontology is engineered to cater for.		
	Complete,	Functional dimension			
	Concise and	of ontologies and their			
	Expandable.	usability profile.			
A.1. Task-based Approaches		Functional dimension.	Information Retrieval metrics. Because every task-based evaluation is individual, no finite set of well-suited measures can be defined.	Improving results in task-based evaluation is most important goal.	No specific measures defined.
A.2. Corpus-based Approaches		Coverage of domain.	Compare ont. With textual content corpus representing domain using IE & OLTs.	Use of natural language techniques to extract list of term then compare it with ontology.	Can't be used, if learning algorithm is similar (based on NLP or rule based).
A.3. Criteria-based Approaches		Measures how far ont. adheres to a certain criteria; By measuring: 1. Structure. 2. More sophisticated measures (e.g. philosophical taxonomy)	1. Average depth of paths from root to leaf nodes.	Can be fully automatic evaluation.	
			2. Define measures	Can be sophisticated partially automatic e.g. OntoClean.	
B. Compare OL Algorithm	Measures quality of OL algorithm.	Structural and Functional aspects. By comparing input with output By: 1. Domain expert. 2. Gold standard.	Precision and recall used to measure the coverage and accuracy for both cases (1 & 2)	Can be used to improve a Learning Algorithm.	Depending on the scenario. i.e. B.1 or B.2 (see below)
B.1. Manual Evaluation by Human Experts.		Judges how far the extracted information is correct.	Ont. is presented to one or more human experts to mark correct terms (i. e. the precision is measured)		Subjective to human expert. Impossible to measure recall.
B.2. Gold Standard based approaches.		Compare onto. With idealized previously created gold standard;	1. Term precision (Lexical precision),	Can be used to directly measure prec. & recall. Gold standard created only once and used thereafter.	Where to get or how to create such a gold standard? Again subjective to Human expert. Can be expensive if more experts involved. To ensure less subjectivity- more than one domain expert should be involved in the design of the Gold Standard.
		1. Lexical layer.	2. Term recall (Lexi. Recall) and Fmeasure		
		2. Taxonomic layer	1. Local measures (compares the similarity of the positions of two concepts in the learned and the reference hierarchy- taxonomic precision and recall). 2. Global measures (by averaging the results of the local measure for concept pairs from the reference and the learned ontology).		
		3. Non taxonomic relations	Local and Global Measures.		

The evaluation framework for this research is a combined method of experimental and testing simulation using real data, in which SOLF is tested on real data (Web Services) and a detailed scenario is constructed to formulate the evaluation of the output ontological model. Qualitative evaluation measures such as precision and recall are applied to evaluate the model using gold standard-based evaluation and domain expert manual evaluation. Recall is used to measure the number of correctly identified concepts by the system as follows:

$$recall = \frac{correctlyExtractedConcepts}{totalAvailableConcepts}$$

For example, if 10 concepts are identified manually in the corpus and the system has automatically identified 7 of these 10 then 70% would be the recall figure. An ideal scenario for recall calculation is to either use a gold standard ontology (existing ontology) or use a domain expert to extract concepts and relations manually from the input sources upfront (pre-create an ontology). Evaluation using gold standard and automatically produced ontology can be misleading however. Typically, an exact match is employed to compare and produce the results as a binary decision of correctness. When attempting a complex business area (such as that found in global banking) it is not possible to deploy a domain expert on all input sources. This is due in part to the size of the input sources and variation in these domains. It is feasible, however, to utilize domain expert knowledge to evaluate concepts and relations produced by SOLF. Therefore, a hybrid approach has been adopted in order to better account for the domain complexity and availability of evaluative artefacts. The domain expert participates in evaluating the extracted concepts and relations, combined with a similarity-based evaluation for calculating the recall metric between Reference (manually extracted concepts) and Response (the output of SOLF); the reference ontology is one produced manually for the same Web Services by previous work.

Precision is used to measure the accuracy of the obtained concepts as:

$$precision = \frac{correctlyExtractedConcepts}{totalExtractedConcepts}$$

where the number of correctly extracted

concept is divided by the total number of automatically extracted concepts by the learning algorithm. For example, if SOLF found a total of 10 concepts, 8 of which are correct then the precision is $8/10 = 80\%$. Precision is calculated here with the aid of a domain expert in order to evaluate the learned relations more directly.

3.7 Research Design Iterations

Design Research is performed through iterative design cycles, which can be improvement iterations or improvement and incremental iterations (Hevner et al.,

2004). This research is implemented as iterative incremental iterations where each iteration (see below) is used to extend and refine the design problem (SOLF).

1. Develops the core ontology learning framework. Ontology is automatically learned as a set of domain specific concepts, automatically extracted from Web Service sources.
2. The second iteration refines the framework and extends it by developing techniques to automatically extract ontological relations between the extracted concepts.
3. Finally, the third iteration refines the SOLF by generalizing and validating the developed structure interpretation patterns (SIP) and transformation rules (TR).

Three design iterations are used to deliver the final artefact as illustrated in Figure 3-5. In each iteration the artefact refinement process is formed as a mini Design Research cycle of build and evaluate, following Vashnavi & Kuhler's (2004) design cycle steps.

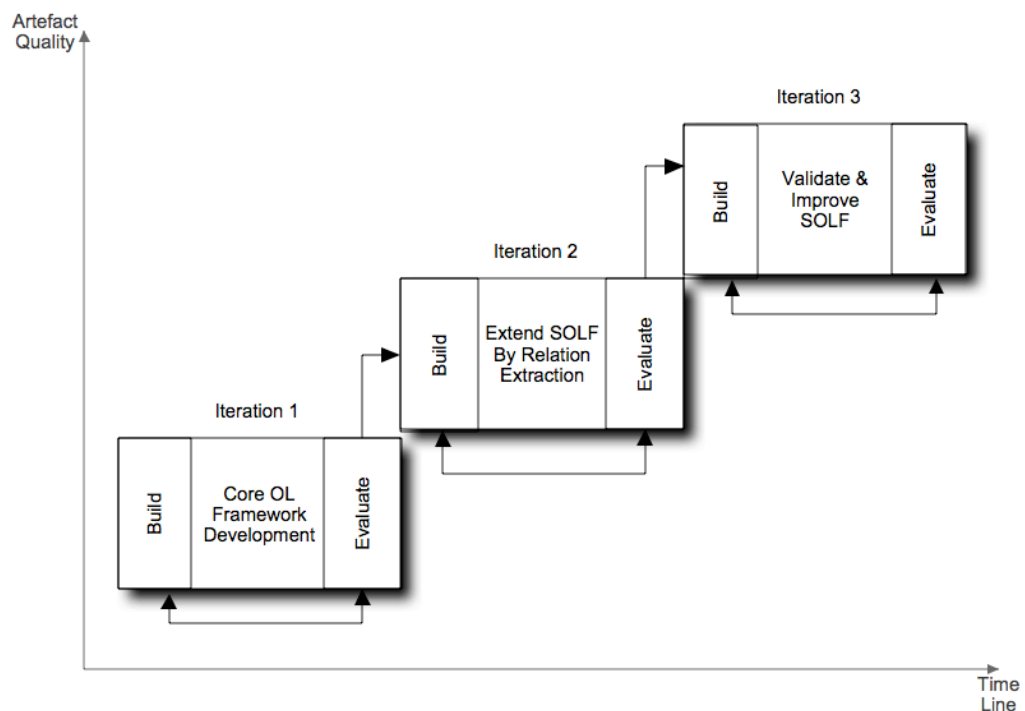


Figure 3-5: Research Iterations

Interestingly, Design Research motivates knowledge generation as part of the design problem, here new awareness is generated and suggestions are made during the build

and evaluate cycle. The learning outcome for each iteration is used to refine the explanatory hypothesis and feeds back into subsequent iterations.

The main Design Research outcome is the development of a methodological framework (SOLF), where framework is defined in the Oxford dictionary as “a basic structure underlying a system, concept, or text: the theoretical framework of political sociology”. Methodology is defined by Checkland (1981) as “a set of principles of method, which in any particular situation has to be reduced to a method uniquely suited to that particular situation”. SOLF incorporates aspects of both a methodology and a framework.

Iteration 1:

This iteration aims at analysing, understanding and testing the applicability of existing ontology learning techniques, more specifically textual-based information extraction techniques on Web Service semi-structured sources. This is achieved by comparing and testing similar approaches on Web Service artefacts (WSDL and XSD documents). The output of this iteration is a set of constructs that identify the appropriate OL techniques. An initial Service Ontology Learning Framework (SOLF) consisting of a Service Term Extraction (STE) phase and an ontology building method. A prototype application is created as an instantiation of SOLF. The method is evaluated for its operationality, efficiency, generality and ease of use, by applying it using the instantiated application on a real set of financial Web Services. A domain ontology model is produced as an output artefact from this iteration consisting of a set of domain concepts. The learned ontology model is evaluated for fidelity, completeness and level of detail by using an evaluation framework that compares the produced ontology model with models from other approaches.

Iteration 2:

This iteration aims at applying the learning from the first Iteration to improve and extend the developed SOLF. The SOLF improvement includes extending the concept pattern extraction to relation extraction. It also includes developing a method for identifying transformation rules. The ontology model from the first iteration is a set of automatically extracted domain specific concepts without any relations between

them. This iteration applies an unsupervised pattern-based relation extraction method to learn relations between those concepts. The method is aimed at finding patterns between concepts formulating a rule-based pattern extraction process from Web Service artefacts, mainly WSDL and XSD files. The application of this process to the set of Web Services contributed a number of secondary Design Research products including constructs, models and methods as illustrated in Table 3-5. A domain ontology model is automatically learned by the improved and refined SOLF. The learned model now consists of domain concepts and taxonomic and non-taxonomic relations between these concepts. A number of SIP patterns as well as a set of TRs; models also considered secondary Design Research output of the iteration.

The evaluative framework for this iteration is aimed at evaluating the efficiency and operability of the method (SOLF), by applying the instantiated application on real Web Services from the financial business domain. Evaluating the completeness and level of detail of the learned ontology is based on employing the evaluation metrics precision and recall. Precision here is calculated by scoring the learned relations and concepts by a domain expert and pattern recall is calculated manually by comparing the learned concepts to a previously created manual ontology (Gold Standard).

Iteration 3:

The aim in this iteration is towards validating, improving and extending SOLF to include more specific domain relations. Applying the SIP and TR on other sets of Web Services to test the generality of SIP and TR produced by the previous iteration, facilitates validating the patterns and extending them to add new ones and refine SOLF. This iteration uses the learning (formed by evaluate, theorize and justify activities), shaped by Iteration 2, to suggest improvement of the models (SIP) and the TR and SOLF method. This leads to developing the final products of the research consisting of a Web Service ontology learning methodological framework (SOLF), a set of SIP patterns, and a set of TRs and an ontology model representing the underlying domain.

Applying SOLF to real Web Services results in a number of secondary Design Research products including constructs, models, methods and instantiations. Measuring significant improvement of the research requires careful evaluation in

order to prove efficiency (March & Smith, 1995) and assess the progress made in the problem domain is done by applying the developed products into real Web Service artefacts and applying OL evaluation methods. The research significance lies in building consequent constructs, models, methods and instantiations addressing the same service ontology learning task. March and Smith's (1995) 16 cell Design Research grid relating a product to a process, is used to highlight and summarize the overall products and processes of the research in an integrated and coherent framework as Table 3-4 illustrates the first activity is meant to provide an understanding and proper explanation of how or why the Design Research products works within a live experiment using real case scenarios (here financial domain Web Services) and the second activity serves to prove or disprove the theory scientifically. Iteration 1 and Iteration 2 are mainly design science, those build and evaluate activities are considered by the research alongside each of the four Design Research product types in those chapters.

Theorize and justify as identified by March & Smith (1995), are mainly behavioural science activities, where, theorizing the SOLF implies understanding how and why it can be applied in real case scenarios. And Justification of SOLF implies proving its applicability across different sets of Web Service domains. Therefore theorize and justify, are only reflected upon in Chapter 6.

Table 3-4: Research Products Versus Research Processes

Research Activities					
		Build	Evaluate	Theorize	Justify
Research Outputs	Constructs	Extraction of Terms (STE). Learning Framework for Services (SOLF). Patterns for Term Extraction Process. Pattern for relation Extraction (SIP). Rules for Transforming Patterns (TR).	Completeness. Simplicity. Elegance. Ease of use.	Are reflected upon in Chapter 6 & 7.	
	Models	Model for Term Extraction Process. Model for the Learning Framework (SOLF). SIP Patterns. Set of Rules (TR). Domain Ontology Model.	Fidelity. Completeness. Level of detail. Robustness. Internal consistency.		
	Methods	Term Extraction Process (STEP). SIP Extraction Process. TR Development Process. SOLF Framework.	Operationality Efficiency Generality Ease of use		
	Instantiation	SOLF Application.	Effectiveness Efficiency Impact on environment		

Executing the research in a Design Research incremental iterative manner enabled learning to emerge from the first iteration by applying and testing techniques from the knowledge base on Web Services. Table 3-5 summarizes the three Design Research iterations illustrating the objectives and output artefacts of each. Research iterations are described in more detail in the following chapters.

Table 3-5: Summary of Research Iterations

Iteration	Activities	Output	Artefact Type
1.	A. Test existing approaches and compares them (part of obj. 1).	Identified appropriate Natural Language processing techniques.	Constructs.
	B. Develop an automated process for service term extraction process (part of obj. 2 & 3).	Service Term Extraction Pattern process.	Method. Model.
	C. Automate method by building a prototype application to test STE using a real case scenario from the financial domain (part of obj. 2).	STE Application. Ontology building algorithm.	Instantiation.
	D. Evaluate STE by comparing it to other similar approaches (obj. 4).	Ontology as a set of domain concepts.	Model.
	E. Suggest an improvement and extension of existing techniques.	List of requirements to improve the approach in the next iteration.	Theories.
2.	A. Develop a relation extraction method for Web Service artefacts (part of obj. 2 & 3).	A structured interpretation pattern process (SIP). Transformation Rule (TR) Extraction Process.	Constructs. Method.
	B. Extend the prototype application to include relation extraction (part of obj. 4).	A set of Structured Interpretation Patterns (SIP). A set of Transformation Rules (TR).	Model. Instantiation.
	C. Evaluate the improved framework (part of obj. 4).	Ontology representing financial domain using sample services.	Model.
	D. Suggest an improvement and extend existing relation extraction patterns.	Suggestions for future improvements.	Theories.
3.	A. Validate research by testing SIP patterns and SOLF application on other Web Services (obj. 5).	Extended set of SIP. Extended set of TR.	Model.
	B. Extend SOLF and application (part of obj. 3 & 4).	Improved SOLF.	Method. Instantiation.
	C. Evaluate SOLF.	Domain Ontology.	Model.

3.8 Summary

This chapter set out the research methodology in accordance with the tenets of Design Research. The methodology is executed in five Design Research steps as adopted from Vaishnavi & Kuechler (2004): (1) Problem awareness, (2) suggestion of suitable OL techniques from the knowledge space, (3) development of the main Design Research artefact (SOLF), (4) evaluation of the artefact is based synthesising Design Research evaluation methods to the OL field and (5) conclusions. In order to achieve the research aim and objectives the research is executed in three incremental Design Research iterations. Each of the iteration is used to build and evaluate a set of artefacts aimed at the OL task from the Web Services domain. In the first iteration a pattern based service term extraction method is developed and evaluated on real Web Services. The second stage extends the method to include relation extraction techniques. And finally the third iteration proves SOLF by applying the learning method and tool to other application domain to prove it generality. Hevner's (2004) Design Research products classification is adopted to illustrate the research outputs produced from iteration. The Research products are identified in the form of consequent constructs, models, methods and instantiations.

An OL evaluation taxonomy and background illustrates that efficiency of OL approaches is determined by assessing the accuracy and coverage of the automatically learned ontology model. Accordingly, two main evaluation scenarios are typically applied; first is a gold standard based scenario, the second is a domain expert evaluation. These two evaluation methods are commonly applied to compute the standard metrics precision and recall.

CHAPTER 4 - ITERATION I

4.1 Introduction

This iteration addresses the term extraction task of the ontology learning layer cake (Cimiano, 2007, p.23). Different NLP techniques for term extraction are applied on Web Service resources, more specifically WSDL and XSD files. Term extraction implies applying linguistic pre-processing techniques. As discussed in Chapter 2, these techniques are commonly applied on unstructured documents. This chapter applies an innovative pattern based term extraction method, that applies pre-processing techniques, which are normally used on textual data sources, on semi-structured Web Service sources, namely WSDL and XSD files. The development of an application prototype as an instantiation artefact is used to evaluate the method and apply it on the financial Web Services taken from commercial organisations.

The rest of the chapter is organised as follows. To begin with, Section 4.2 discusses how Design Research is applied for this iteration. Design Research artefacts are identified along with the iteration plan and research products. Section 4.3 introduces the building stage of the Design Research problem, presenting a method for service term extraction and explaining the steps involved in the method. Section 4.4 develops a prototype that implements the suggested method and presents the outcome of applying the prototype on sample files from the financial domain. Section 4.5 presents the experimental data and evaluates the iteration outputs and the method. Finally the research concludes in Section 4.6 by discussing the iteration feedback and presenting the learning outcome. The chapter is summarized in Section 4.7.

4.2 Design Research and Output Artefacts

This iteration applies Design Research as a miniature iterative process through which learning of the problem space is achieved through artefact development and evaluation. A method can be seen as a set of steps that can be followed to accomplish a certain task

(March & Smith, 1995). Here, a method for Service Term Extraction (STE) is proposed, an instantiation is then developed as a prototype that implements the STE method. This iteration is used to produce an initial Service Ontology Learning Framework (SOLF) comprising the STE and an ontology building algorithm. As illustrated in Figure 4-1, an iterative cycle of artefact building, development and evaluation is employed, adopted and based on the general methodology of Design Research by Vaishnavi & Kuechler (2004).

As discussed in Chapter 2, a number of Web Service sources characteristics are identified that necessitate the development of a tailored OL process to deal with the special characteristics of Web Service resources. The applicability of term extraction techniques, commonly used with unstructured data sources, on WS semi-structured sources, requires analysis and testing to determine their tailoring ability to extract semantic information. It is the aim of this iteration to adopt and modify existing learning techniques that deal with these semi-structured sources using real examples taken from the financial Web Service domain. A typically applied OL scenario (Maynard, Li & Peters, 2008) starts with term extraction as a first step. This iteration targets term extraction as a pre-processing stage involving a sequence of NLP techniques. This stage is considered as a starting point to provide an understanding and an experimentation environment for the Design Research cycle and OL framework to evolve.

Term Extraction involves applying information extraction techniques to extract possible terms from Web Service resources. Identifying words that are possible candidates for concepts and relationships in the underlying context implies collecting and analysing available Web Service resources and employing text analysis techniques to them.

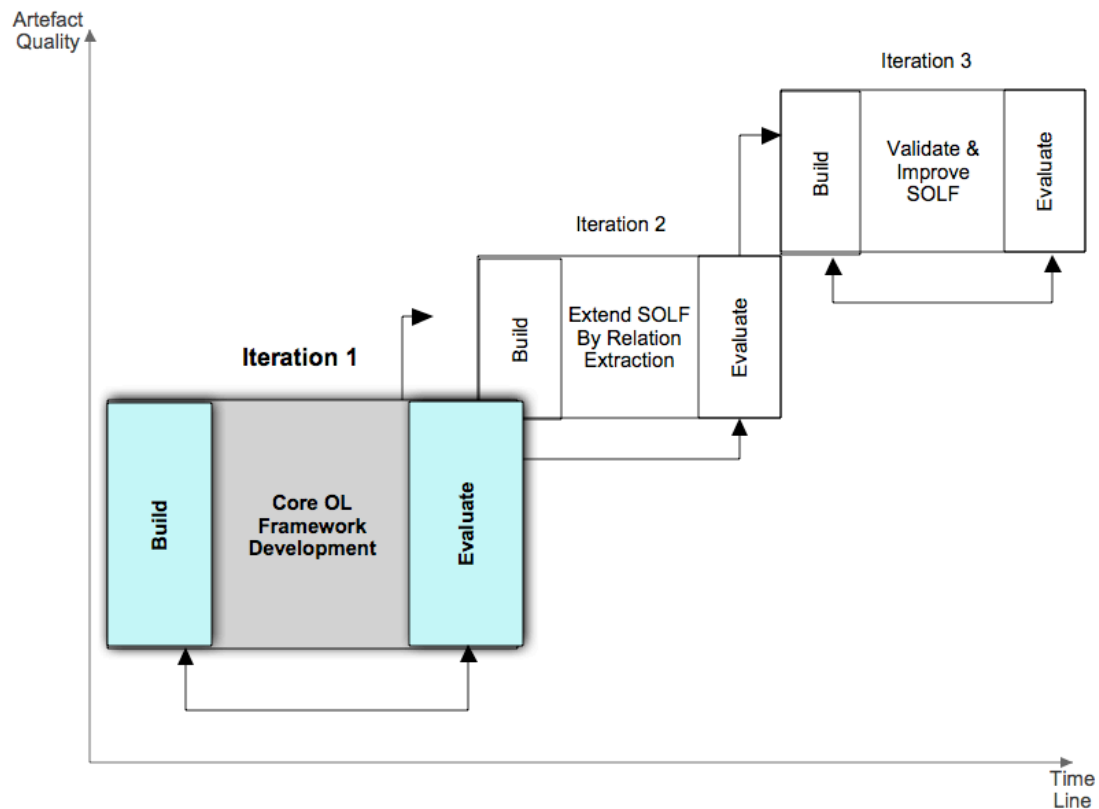


Figure 4-1: Iteration 1 Overall Framework

The novelty of this method is that it is applied on semi-structured data sources consisting of XML files. Pattern based term extraction is commonly applied on unstructured textual sources (Buitelaar & Cimiano, 2008). The innovation of this approach is to adopt and apply pattern based term extraction to extract knowledge from technical semi-structured sources.

4.2.1 Design Research Artefacts

The aim of this iteration is to develop the core SOLF that embodies the service term extraction (STE) technique, automates the technique and evaluates the process. The technique involves applying a process consisting of a sequence of steps and results in a number of outputs. As illustrated in Table 4-1, each step applies a natural language processing method on an input artefact and results in an output that is used as input for the next step. Applying the methods in the consequent steps results in a pipeline process, which is then implemented as a pipeline application using the GATE development environment.

This iteration extends the pattern based knowledge extraction in two ways: First, a dynamic process for deriving term extraction patterns. Applying this process on the sample set of services contributes a set of patterns. Second, applying the patterns on the WSDL and XSD sources of industrial Web Services to evaluate the extraction outcome.

Table 4-1: Iteration Steps – Input Output Model

Steps	Method	Input Artefact	Output Artefact
1. Develop WSDL and XSD model tokenizer method.	WSDL & XSD Tokenizer	WSDL & XSD files	WSDL & XSD-Term Model
2. Decide a suitable Part Of Speech (POS) identifier method for WSDL and XSD models.	POS Tagger	WSDL/XSD-Term Model	POS-Term Model
3. Identify concept patterns for concept extraction from WSDL and XSD models.	Pattern Term Extraction Process	POS-Term Model	Pattern Term Extraction Models
4. Build Service Term Extraction (STE) method.	Build GATE Application	Web Service Artefacts	Prototype Application (using GATE)

Evaluation of the iteration is aimed at evaluating the following output artefacts:

- The initial STE method is evaluated using the instantiation prototype created as a GATE application pipeline, in which real Web Service resources are used.
- The Concept and Relation Pattern Model; which links tokenised concepts via relationships, are evaluated by running the method on the real case example and ensuring that all relevant names are picked up by the identified patterns. The Lucena Data Store viewer is a GATE plug-in typically used for analysis and testing of the results over the real Web Services.
- Evaluating the learned ontology model involves the evaluation of the quality of the STE method by measuring the coverage and precision of the learned concepts.

4.3 Artefact Building and Development

The Building stage involves problem awareness and suggestion. This implies identifying the initial steps for the process and explaining what each step involves. This stage involves reviewing and analysing existing OL approaches, finding suitable techniques for WSDL and XSD files, and suggesting appropriate tools and techniques. Testing current similar work enabled a deeper understanding of the limitations of current approaches and suggested improvements to overcome the limitation on current approaches, which has eventually led to identifying appropriate techniques and tools for concept and term extraction from WS sources. Term extraction involves applying document pre-processing techniques to allow for lexical and semantic analyses of the input sources. This is achieved by applying a tokenization step followed by a POS tagging.

4.3.1 Tokenization

Pre-processing involves tokenization as a first step. Default tokenizers are designed to parse natural language text using typical tokenization techniques, which are reliant on assuming that token separators are based on natural language separators like spaces, commas, full stops, etc; whereas, Web Service sources are semi-structured and in some cases, like WSDL and XSD files, relevant ontological concepts can be found only in tag names. Figure 4-2 shows a sample WSDL file illustrating the structure and character of the content of a WSDL document, e.g a sample line of a WSDL is `<xs:element name="checkInDate">`. In such cases tokenization should be based on capitalization of the first letter. By analyzing Web Service sources, it can be clearly seen that the name attributes are a common venue for ontological concepts. In this example 3 tokens can be extracted using capitalization of the first letter.

Another naming scheme that can be found in such sources is `<xs:element name="company_search_response">` in which an underscore character is used as a token separator. For this kind of text a tokenizer is implemented to deal with these cases. The WSDL and XSD tokenizer is adopted from the GATE built-in default tokenizer and modified to suit the described characteristics. Tokenization produces a tokenized WSDL and XSD model, in which restrictions to limit the extracted concepts,

relies on lexical analysis of the document and deriving patterns based on tokens lexical category.

```
<xs:complexType name="CheckAvailability">
  <xs:sequence>
    <xs:element name="checkInDate" type="xs:date"/>
    <xs:element name="checkOutDate" type="xs:date"/>
    <xs:element name="roomType" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Figure 4-2: WSDL sample file

4.3.2 POS Tagging

Applying shallow semantic analysis involves categorizing words based on their meaning, and a POS tagger serves this purpose. Part Of Speech (POS) Tagging involves identifying and adding parts of speech tags to the WSDL tokenized model, i.e. identifying verbs, nouns, adjectives and other parts of speech for each token. POS tagging is a step commonly applied as a second step on unstructured sources (Maynard, Li & Peters, 2008; Sabou, 2005) as part of the term extraction process. Since WSDL and XSD contain semi-structured data, words that appear in operation names such as "*checkAvailability*" are considered to be the only source of domain information available in these sources. Therefore, this information needs to be analysed and examined for domain concept extraction. The tokenized terms need to be tagged by applying a POS tagger, which will identify the type of each word using their basic dictionary meaning regardless of their context. Hence, *check* should be identified, as a verb and *Availability* should be tagged as a noun.

Off-the-shelf techniques are sufficient for this purpose. The Brill-style tagger, offered by GATE, uses basic Part-of-Speech information, and is selected as the POS tagger method employed for this step (Cunningham et al., 2002). A POS tagged WSDL model enables the researcher to identify patterns of concepts and relations based on semantic analysis of the words identified by the POS tagger.

4.3.3 Pattern Extraction

Rule-based information extraction uses domain specific handcrafted rules that describe patterns to be matched. This step involves finding appropriate patterns that detect concept related terms in WSDL elements, for example, the name attribute in the WSDL line `<xs:complexType name="CheckAvailability"> .` *CheckAvailability* provides the most likely domain concepts; therefore the ultimate goal would be to identify patterns that will extract all such WSDL entries. All possible patterns can be identified by following an iterative pattern identification process, as depicted in Figure 4-3; the process is based on analyzing the commonly applied naming convention used in method names, input and output parameters and discovering all of the possible pattern combinations based on the semantic and syntactic analysis information produced by previous step, in order to ensure that all of the possible patterns in the WSDL and XML files are identified and therefore extracted. The process starts by identifying an initial set of patterns, analyzing the pattern matches on WSDL and XSD files, evaluating their coverage and detecting any missing patterns, and adding new patterns if required. This process stops when no more new patterns were found in the chosen sample files.

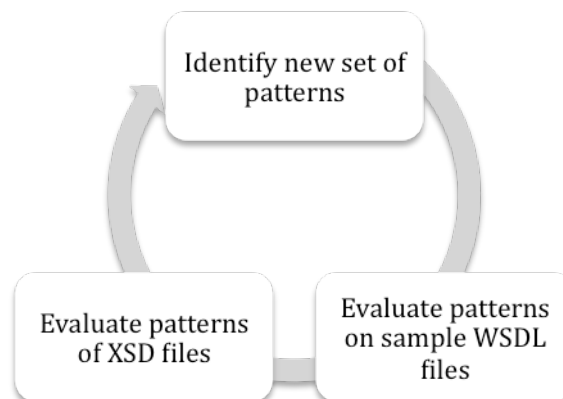


Figure 4-3: Pattern Extraction Process

Given the interest here of extracting domain knowledge rather than service functionality, the concepts identification query employed is based on identifying different forms of nouns in Web Service sources (WSDL and XSD). Therefore, this step leads to identifying patterns for extracting service concepts based on extracting matches to different types of nouns as classified by the POS tagger. Appendix A contains a list of POS tags used by the GATE Brill tagger.

4.3.4 Ontology Building

This step involves bootstrapping the concepts identified in the input sources to construct a lexical layer of the domain ontology model. The model is produced using a Web ontology language commonly supported by most ontology editors. The output is a lightweight ontology that represents the domain covered by the input semi-structured data sources. Concepts identified by the patterns in the previous step are matched and annotated using regular expression matching (Bontcheva et al., 2004), and then ontological concepts are created according to the annotated terms in the service artefacts.

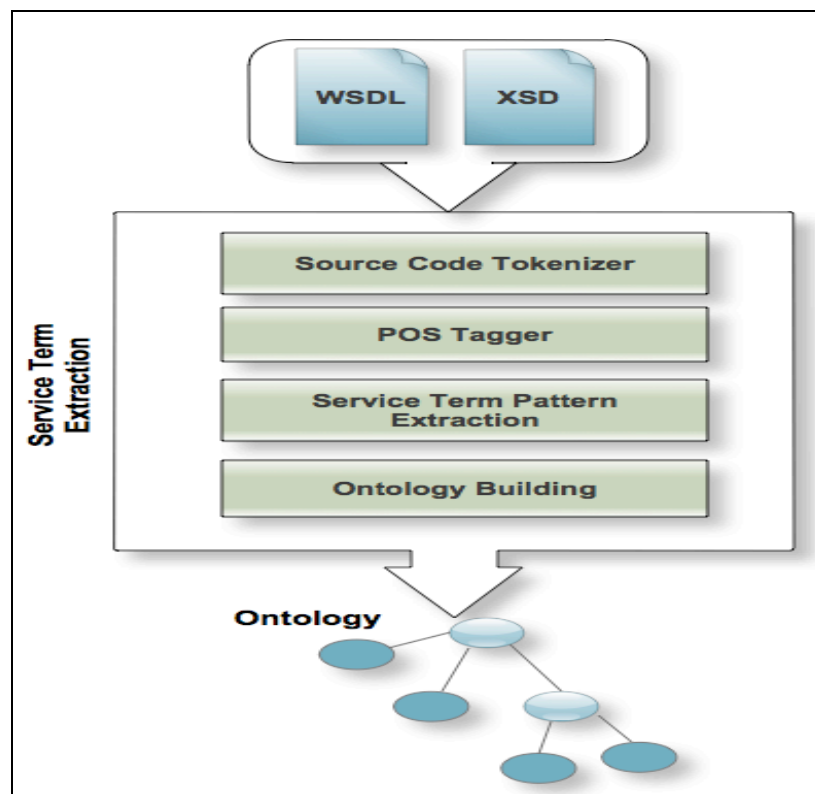


Figure 4-4: Service Term Extraction (STE)

4.4 Framework Prototype Implementation

The search for a well established open source tool that can be used for Term extraction has lead to choosing GATE 5.0 beta version. GATE stands for General Architecture for Text Engineering, and provides the researcher with an integrated infrastructure for experimentation with modifiable built-in tools for Computational Linguistics, Natural Language Processing (NLP) and language engineering (GATE User Guide 2008).

The GATE platform is chosen as it provides a flexible platform with the required language engineering and ontology building tools, for example:

- The use of off-the-shelf NLP techniques.
- A Java Annotation Pattern Engine (JAPE) (Cunningham et al., 2002) that facilitates the development of pattern identification rules and TRs.
- The GATE Ontology API (Bontcheva et al., 2004) based on the OWLIM model, which supports the OWL-Lite standard (see <http://www.w3.org/TR/owl-features/>).

The developed application reads a corpus of WSDL files and runs a sequence of processing resources over the corpus, extracting concepts from the input files. It then produces an ontology as an output of the system. The algorithm is based on pattern matching using JAPE regular expression matching; first, a JAPE file that finds and annotates concepts in the input documents, then another JAPE file finds the annotated concept and creates the ontological concept accordingly. Figure 4-5 illustrates a snapshot of the prototype implementation of the STE application pipeline.

GATE Processing Resources (PR) are specifically tailored for the needs and requirements of an application domain. In this case GATE PR are modified to the requirements of the underlying WSDL and XSD files. Service Term Extraction in this research applies a sequence of processes over Web Service artefacts. A pipeline application is created in GATE that performs Term Extraction as the first stage of any OL system. The pipeline consists of a number of GATE's Processing Resources (PR), reflecting the steps described in this section; the first PR is the WSDL and XSD tokenizer, which is implemented to deal with the characteristics of these sources, as discussed earlier in this chapter.

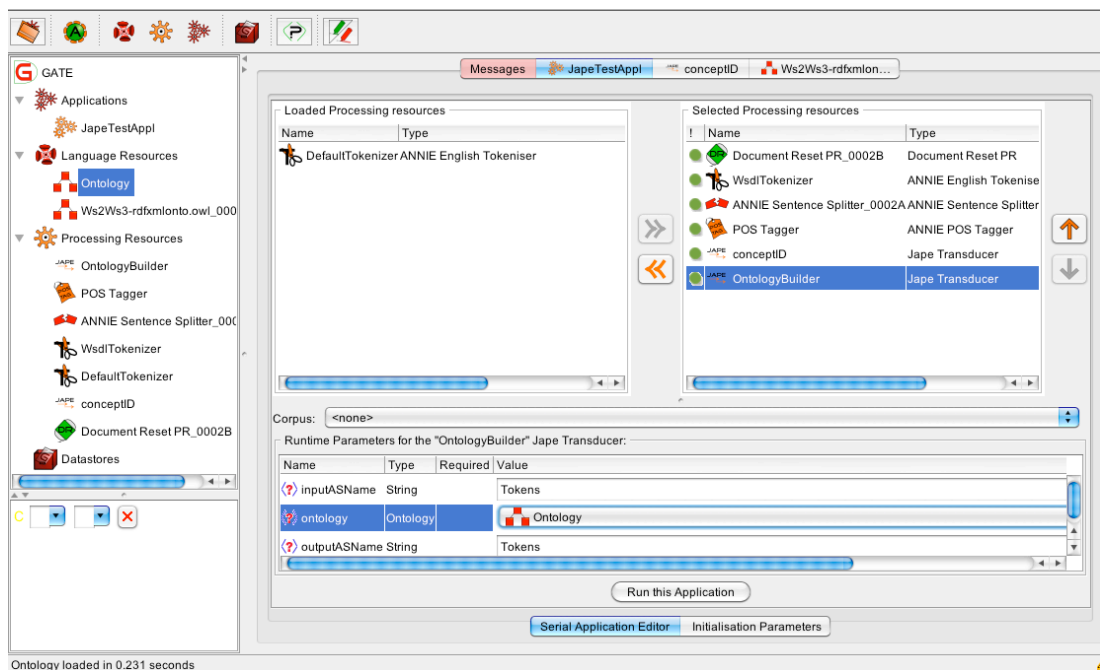


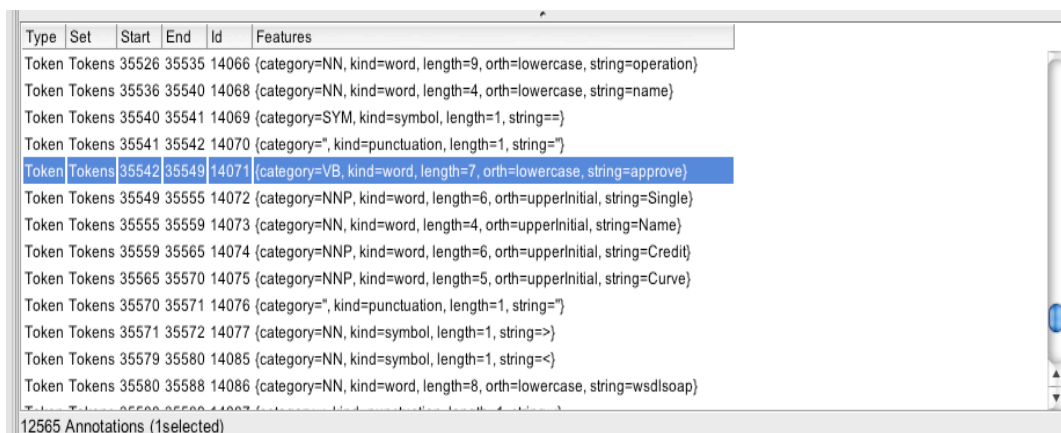
Figure 4-5: SOLF Application Pipeline

First, a WSDL tokenizer is developed to tokenize the input files into simple tokens, dealing with compound words and tokenizing WS1 phrases such as “*unwindTradeExtResponse*” into four distinct tokens instead of one. Table 4-2 illustrates a WSDL tokenised model representing a sample output of a WSDL tokeniser step, where each word is identified as a token. This table is used to analyse the output of the tokenizer. It can be clearly seen that the tokenization of the element *name=“roomType”* produced two tokens that are very good concept candidates.

Table 4-2 : WSDL Tokenized Model

No.	Document ID	Annot. set	Left Context	Word Tokens	Right context
192	hotelWsdITst__12416 19854614_2774	Tokens	element name="check	In	Date" type="
193	hotelWsdITst__12416 19854614_2774	Tokens	element name="check	Out	Date" type="
194	hotelWsdITst__12416 19854614_2774	Tokens	name="checkIn	Date	" type="xs
195	hotelWsdITst__12416 19854614_2774	Tokens	name="checkOut	Date	" type="xs
196	hotelWsdITst__12416 19854614_2774	Tokens	element name="room	Type	" type="xs
197	hotelWsdITst__12416 19854614_2774	Tokens	Type name="t	Check	Availability"> \f
198	hotelWsdITst__12416 19854614_2774	Tokens	name="tCheck	Availability	"> \f1

The second step requires applying POS tagger that identifies the POS of each token. ANNIE POS tagger, which is based on the Brill tagger (Cunningham et al., 2002), is applied for implementing this step, adding part of speech tags to each token as a new feature. The output from this phase, as Table 4-3 illustrates, enables patterns to be identified based on the category feature added here. For example, the POS tag of each token in the phrase *“unwindTradeExtResponse”* is added as a category feature, where *Trade* is tagged as *NNP*, and denotes a singular proper noun according to the ANNIE POS tagger. Other tags such as *NN* and *VB* would have a different meaning, where the first is used to denote a singular or mass noun and the second denotes a verb in its base form (Cunningham et al., 2002). Figure 4-6 illustrates a snapshot taken from GATE GUI, in which a category feature *“VB”* is added to the string token *“Approve”*. The category feature is assigned different values such as *VB* (Verb), *NN* (Noun) or *NNP* (Proper Noun) according to the Part of Speech type of each token.



Type	Set	Start	End	Id	Features
Token	Tokens	35526	35535	14066	{category=NN, kind=word, length=9, orth=lowercase, string=operation}
Token	Tokens	35536	35540	14068	{category=NN, kind=word, length=4, orth=lowercase, string=name}
Token	Tokens	35540	35541	14069	{category=SYM, kind=symbol, length=1, string=}
Token	Tokens	35541	35542	14070	{category=", kind=punctuation, length=1, string="}
Token	Tokens	35542	35549	14071	{category=VB, kind=word, length=7, orth=lowercase, string=approve}
Token	Tokens	35549	35555	14072	{category=NNP, kind=word, length=6, orth=upperInitial, string=Single}
Token	Tokens	35555	35559	14073	{category=NN, kind=word, length=4, orth=upperInitial, string=Name}
Token	Tokens	35559	35565	14074	{category=NNP, kind=word, length=6, orth=upperInitial, string=Credit}
Token	Tokens	35565	35570	14075	{category=NNP, kind=word, length=5, orth=upperInitial, string=Curve}
Token	Tokens	35570	35571	14076	{category=", kind=punctuation, length=1, string="}
Token	Tokens	35571	35572	14077	{category=NN, kind=symbol, length=1, string=>}
Token	Tokens	35579	35580	14085	{category=NN, kind=symbol, length=1, string=<}
Token	Tokens	35580	35588	14086	{category=NN, kind=word, length=8, orth=lowercase, string=wsdlsoap}

12565 Annotations (1selected)

Figure 4-6: WSDL POS Model

Thirdly, a Pattern Extraction process follows, that identifies concept extraction patterns. ANNIC (ANNotations In Context) plug-in, is a GATE plug-in that offers applying pattern extraction using the Lucena Data Viewer tool (Aswani et al., 2005). ANNIC is used in this step to view and analyse the output of the lexical and semantic analysis steps, and the results are exported to an html file. The initial pattern is drawn from Cimiano (2007) and Hearst (1992) as VB + Noun (verb followed by one noun, e.g. CancelTrade or GetTrade).

Table 4-3 illustrates the Lucena Data Viewer model of the identified patterns for a sample WSDL file. Following the process illustrated in Figure 4-3 Using ANNIC enabled instantaneous evaluation and refinement of patterns. A sample table produced that represents a *VB+NNP+NNP* pattern model. Notice that the *"GetCreditDefault"* that appears in the pattern column matches the *VB+NNP+NNP* pattern. This illustrates that all element names in Web Service sources that match the identified patterns are extracted automatically by the system. The aim of this step is to identify all of the possible patterns that will lead to candidate ontological concepts.

Table 4-3: Pattern Extraction Model

No.	Document ID	Annotation Set	Left Context	Pattern	Right Context
1	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapFromSingleDay"
2	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapFromSingleDayB y
3	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapFromSingleDayB y
4	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapFromMultipleDay s"
5	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapFromMultipleDay sBy
6	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapForDateRangeBy
7	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapByTargetSystemT rade
8	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetCreditDefault	SwapBySummitTradeld
9	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	GetTradeAudit	History" style="
10	Trdport2_12363550 59316_2719	Tokens	/tradecapture/wsd/	CreateDefaultedTrade	" style="document
::	::	::	::	::	::
21	Trdport2_12363550 59316_2719	Tokens	"impl:to	DoBlotterRequest	" name="to
22	Trdport2_12363550 59316_2719	Tokens	"impl:to	DoBlotterResponse	" name="to

Table 4-4 represents a set of identified patterns that can be used to determine relevant phrases as terms and is therefore applied by an ontological transformation process to transform automatically extracted terms to ontological concepts.

Table 4-4: Summarized Generic Patterns

Pattern	Pattern Match Sample
Verb + Noun	CancelRequest
Verb + Noun + Noun (2 or more nouns up to 10)	DoBlotterRequest
Noun + Noun + Noun (2 or more nouns up to 10)	PendingRefEntities

Building the pattern for regular expression matching is achieved using JAPE Transducers (Cunningham et al., 2002; Bontcheva et al., 2004). These transducers are developed to perform rule-based pattern extraction. Rule definition is carried out using regular expressions over annotations. A JAPE rule consists of two parts, as illustrated in Figure 4-7; the left hand side (LHS) and the right hand side (RHS). The LHS of the rule (shown to the left of the arrow in the Figure 4-7) identifies the patterns to be matched based on information generated by the previous steps (tokenization and POS tagging). The RHS of the JAPE rule identifies the annotation set to be created for the text that matches the pattern on the LHS. The result of executing this JAPE rule on the input files is that each token that matches the pattern is annotated with a concept annotation. Another JAPE rule is then created to find annotated concepts in the text and create ontological concepts accordingly. The ontology is created using the GATE OWLIM API.

```

Phase: locationcontext1
Input:  Lookup Token Tokens
Options: control = applet
//rule identifies concepts of type NN or NNP
Rule: ConceptIdentification1
Priority:50
(
    {Token.kind == word, Token.category == NNP}|
    {Token.kind == word, Token.category == NNPS}|
    {Token.kind == word, Token.category == NN}
):concept
-->:concept.Concept = {rule= "ConceptIdentification1" }

```

Figure 4-7: JAPE Sample Code

The second JAPE file is created to add new concepts, as they are found, to the existing ontology, as illustrated in Figure 4-8.

```
// Concept ID creation Rule
Rule: ConceptID
({ConceptID}):relationIden
-->

:relationIden{
    Annotation theInstance = (Annotation)relationIdenAnnots.iterator().next();
    //get the concept strings from the features of Annot
    String Concept = theInstance.getFeatures().get("ConceptString").toString();

    // Create URI for the new concept
    gate.creole.ontology.OURI classURI = ontology.createOURI(
http://example.com/classes# + Concept);

    // Add new concept to ontology
    gate.creole.ontology.OClass Concept = ontology.addOClass(domclassURI);
}
```

Figure 4-8: JAPE Rule for Concept Creation

Executing the application pipeline on a corpus of Web Services consisting of WSDL and XSD files produced an ontology model representing the financial Web Services employed for the experiment. The model represents the automatically created financial ontology model. Figure 4-9 depicts a snapshot of the produced ontology as the final product of the application.

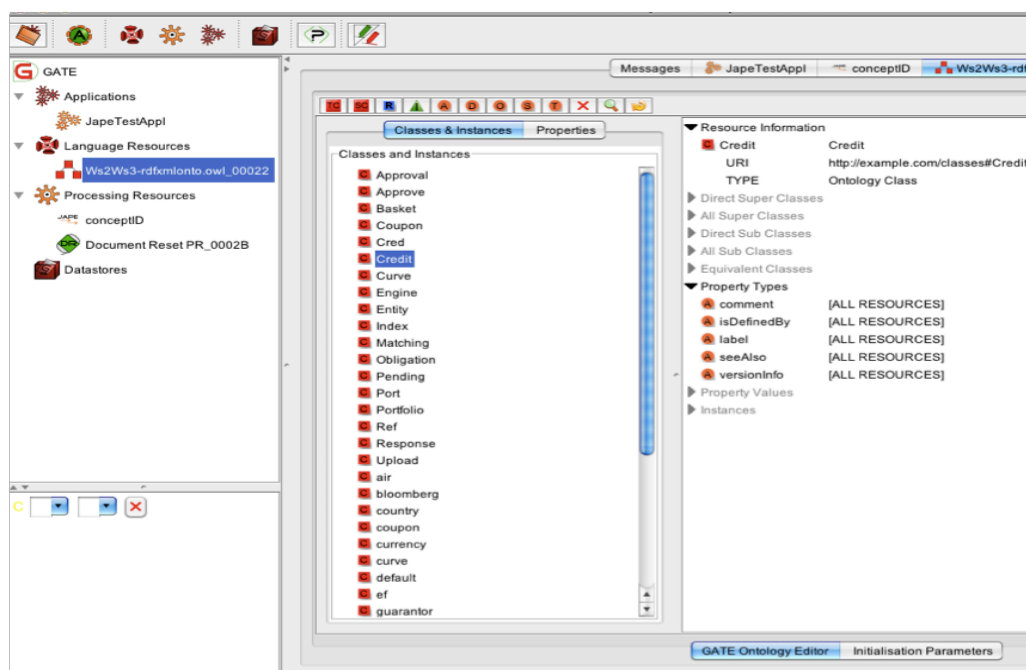


Figure 4-9: Snapshot of the Learned Domain Ontology Model

4.5 Evaluation

Instantiations can be viewed as existing implementations, and are used to evaluate constructs, models and methods (March & Smith, 1995). For meeting the objectives of this iteration, a prototype system was developed and implemented that operationalized the proposed method using GATE 5.0 beta1 version. Evaluation of this iteration is achieved through assessing the performance of the system in extracting domain relevant terms, consequently leading to domain concepts. Importantly, the information extraction performed here is ontology-based information extraction that needs to be evaluated differently from normal IE in the sense that, misclassifying a term as a concept rather than a relation is preferable to misidentifying the term in the first place (Maynard, Li & Peters, 2008).

Commonly applied IE metrics are precision and recall. As discussed in Chapter 3, these metrics are used to evaluate the accuracy and coverage of the learned ontology model. Precision and recall are typically calculated either by comparing outputs to manually extracted data, or by involving a domain expert. The expert role is in validating the accuracy of the extracted terms, concept by concept, i.e. to evaluate the learned concepts and relations by presenting them to a human assessor who can verify their correctness and relevance to the domain using a certain grade given to different concepts (Cimiano

2007). Here, precision is used to assess the accuracy of the STE calculated according to the formula:

$$precision = \frac{NoOfCorrectConcepts}{TotalNoOfConcepts}$$

Where *NoOfCorrectConcepts* is the number of scored concepts validated as correct by the domain expert, and *TotalNoOfConcepts* is the total number of concepts extracted by the system.

4.5.1 Experimental Data

Due to the large size and commonality of the structure and content of WSDL and XSD files, a decision was made to use a realistic number that would allow practical and accurate evaluation when presented to a domain expert. Therefore, three Web Services are taken from the financial domain. The Web Services are used to evaluate the Design Research output artefacts outlined in Section 4.2. The WSDL and XSD files are grouped and categorized according to the Web Service to which each files belongs.

A summary of the Web Service resources used for this iteration is presented in Table 4-5. The details of the three ‘real world’ Web Services are described below, though some details are omitted for reasons of confidentiality. Each service differs in its complexity and style, both in the Web Service usage and the specific design decisions taken by the respective development groups:

- *Trading (WS1)*. This Web Service provides an interface from the Front and Middle offices (traders and risk managers) to a back office processing system. The interface provides access to core trade data as well as market specific measures that are added to the trade over its life (i.e. affecting its risk profile). The Trading Web Service follows a document binding style and consists of 774 lines and its size is 30506 bytes.
- *Matching engine (WS2)*. This Web Service supports a fixed income business with Bond and Repo product types, in particular, processes where a trader and salesman enter separate trade details, which are subsequently matched and integrated. The

matching process is carried out by this service. The Matching Engine Web Service has a smaller description than the Trading Web Service, consisting of 64 lines and with a size of 2086 bytes. Primarily the interface is being a document that is detailed in the associated schema XSD files. This service adopts an RPC Web Service style.

- *Credit service (WS3)*. This Web Service is part of a trading system that supports a range of derivative instruments. The system is used globally by various trading departments. The service again follows a document-based binding style and consists of 423 lines and has a size of 40434 bytes.

Table 4-5: Summary Information Representing Used Web Services

Web Service Name	No. Of WSDL files	No. Of XSD files	Total No. of lines (WSDL Code only)
Web Service 1 <i>TradePort</i>	1	6	774
Web Service 2 <i>MatchingEngine</i>	1	10	64
Web Service 3 <i>SOLService</i>	1	N/A	423

Given the size and the structure of these files manual extraction is time consuming, expensive and inapplicable; therefore, a more appropriate and practical evaluation strategy is designed for evaluating the coverage and accuracy of the extracted terms. The adopted evaluation strategy is aimed at evaluating the performance of the implemented STE method against similar research efforts and targets the gaps discussed in Chapter 2. The evaluation is performed against an unstructured approach and another structured approach (based on WSDL only), in order to determine the validity of the STE in extracting the required terms. Then a domain expert, with experience in working with financial banking industry, is used to validate the concepts and calculate the precision according to their scoring of correct concepts. Lastly, analysing the results of the evaluation leads to reaching a conclusion and learning from the developed artefact for future improvements of the method for the next iteration.

4.5.2 STE Performance

Due to the fact that this research is aimed at ontology related term extraction, only candidate terms are considered for evaluation. The evaluation of term extraction in this iteration is carried out using the GATE plug-in the Lucena Data Viewer that enabled the analysis of the extracted terms using pattern recognition and determining the domain coverage of the method. Tokenization produces all file contents as Tokens, in which case symbols and tags are tokenised, and for this stage are considered irrelevant due to the fact that they only present XML code. To filter out irrelevant Tokens from the Tokenised WSDL and XSD model, a self-evident pattern is applied for the purpose of producing the Web Service Term Model (WSTM). A query is formulated using JAPE patterns (GATE 5.0 User Guide 2008) that is based on pattern extraction in order to extract relevant terms for the purpose of evaluating the STE system. Relevant terms can only be words that are either verbs or nouns. Therefore, the applied query to produce the WSTM is given below:

```
{Token.kind=="word",Token.category=="NN"}|{Token.kind=="word",Token.category=="NNP"}|{Token.kind=="word",Token.category=="VB"}
```

The output produced from executing the query containing the STE pattern is uniquely filtered and a WSTM is produced for each service accordingly. A sample of the WSTM is illustrated in Table 4-6, and represents the WSTM for WS3. A full list of extracted terms can be found in Appendix D.

Table 4-6: WSTM Extracted from WS3

Concept List 1	Concept List 2
Coupon	series
Date	currency
Sequence	bloomberg
Target	ticker
Curve	issuer
Market	issue
Guarantor	summit
Maturity	org
Redemption	equity
Obligation	credit

The sample files are run three times using three term extraction methods taken from three different approaches. In line with the literature review, the first approach is taken

from previous work by Sabou (2005), which employs unstructured term extraction techniques. The second approach employs semi-structured tokenization but is applied only to WSDL files, i.e. it doesn't include any XSD files. The third method uses the STE term extraction method, as developed in this iteration. The STE method targets gaps found in both approaches and therefore the results are expected to be better than the other two approaches chosen for this evaluation in terms of providing better coverage of the domain concepts and increased accuracy in concept extraction.

The produced result representing the evaluation model consists of three columns representing the extracted concepts from each method, which are analyzed and then presented to the domain expert for validation. Table 4-7 represents a concept evaluation model, which gives an overview of the experimental settings used for evaluation. Analyzing the outcome of this model revealed that better extraction performance was achieved with the STE method, due to a number of reasons: (1) although Method 1 produced more terms, most of the terms were compound terms that were unlikely to serve as domain concepts. (2) Method 2 improved the term extraction over Method 1 in the sense that those terms were better suited as candidate domain concepts, but are quantitatively less than the terms produced using Method 3. (3) Method 3 provided better domain coverage since it produced an improved intensive list of terms that are more likely to serve as domain concepts.

Table 4-7: Concept Evaluation Model

	Web Service	XSD only	WSDL only	Both
Method 1 Default Tokeniser	Web Service 1	Terms: 2598 Unique: 283	Terms: 2574 Unique: 172	Terms: 5172 Unique: 455
	Web Service 2	Terms: 2397 Unique: 149	Terms: 181 Unique: 44	Terms: 2578 Unique: 193
	Web Service 3	N/A	Terms: 3090 Unique: 247	Terms: 3090 Unique: 247
Method 2 Based on WSDL files only	Web Service 1	N/A	Terms: 3670 Unique: 112	Terms: 3670 Unique: 112
	Web Service 2	N/A	Terms: 203 Unique: 47	Terms: 203 Unique: 47
	Web Service 3	N/A	Terms: 4741 Unique: 183	Terms: 4741 Unique: 183
STE Method (Improved version of 1 and 2)	Web Service 1	Terms: 3887 Unique: 239	Terms: 3670 Unique: 112	Terms: 7557 Unique: 351
	Web Service 2	Terms: 2924 Unique: 126	Terms: 203 Unique: 47	Terms: 3127 Unique: 173
	Web Service 3	N/A	Terms: 4741 Unique: 183	Terms: 4741 Unique: 183

Now, to determine whether the extracted concepts forms a good source for building lexical layer of domain ontology. Evaluation measures need to be calculated based on expert scoring of each automatically extracted concept. Therefore, for practical reasons, this procedure is performed only on WS2. A WSTM (as illustrated in Table 4-6) is presented to the domain expert to score each concept. The scoring system employed, is a lenient system in the sense that each concept is scored with 1, 0.5 or 0, such that 1 indicates a correct concept, 0 indicates that it is an incorrect domain concept, and half-weight indicates partially correct concepts. The results of the domain expert evaluation have shown an improvement with the STE method over the other two approaches. The summarized precision is presented in Figure 4-10, and illustrates a 67% precision for the STE method.

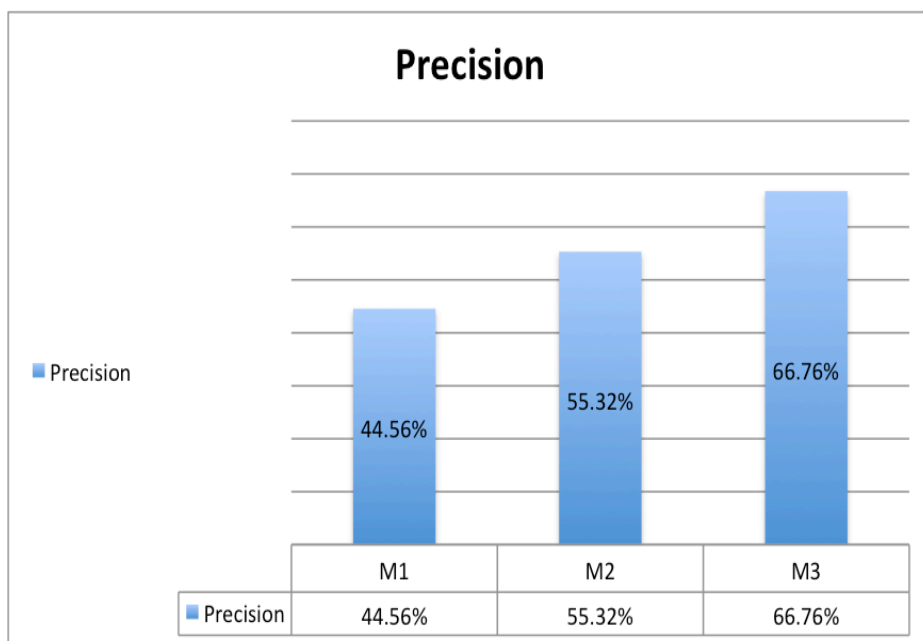


Figure 4-10: WS2 Precision

4.5.3 Pattern Evaluation

The evaluation at this stage will involve coverage and specificity of patterns, ensuring that they cover all existing concepts and relations in the Web Service artefacts. The process followed embodies the notion of saturation in grounded theory (Bernstein, 1999), where the cyclic pattern extraction process ensured the refinement and identification of new patterns. This process has lead to the discovery that all candidate

terms in the input files are classified mainly into either noun or verb. Here, a verb is more likely to determine service functionality. ANNIC provided instantaneous evaluation of pattern extraction and evaluation (Maynard, Li & Peters, 2008). ANNIC is used to replace the identified pattern with live validation on the tokenised WSDL and XSD models. All of the identified patterns are tested directly on the input models to ensure pattern coverage of all existing concepts and relations in the source files. Completeness is evaluated by comparing the ANNIC results of the identified matched patterns against all of the element names that exist in the source data files, automatically extracting all of the element names. The sample output is illustrated in Table 4-8; the table is produced by executing the query on ANNIC that results in producing all of the element names before any tokenization is performed on them, thereby ensuring that the researcher has a list of all element names that exist in the source files, which are then used to validate the pattern extraction process.

Table 4-8: Default Tokenizer WSDL Model

No	Document ID	Annotation Set	Left Context	Pattern	Right Context
1	WSDL_WS1_124 3994895541_928 4	Tokens	—	Query	"
2	WSDL_WS1_124 3994895541_928 4	Tokens	—	Request	"
3	WSDL_WS1_124 3994895541_928 4	Tokens	—	Response	"
4	WSDL_WS1_124 3994895541_928 4	Tokens	—	Response	"
5	WSDL_WS1_124 3994895541_928 4	Tokens	—	Request	"
6	WSDL_WS1_124 3994895541_928 4	Tokens	—	Query	"
7	WSDL_WS1_124 3994895541_928 4	Tokens	:	InsertTradeRequest	"
8	WSDL_WS1_124 3994895541_928 4	Tokens	:	MirrorTradeRequest	"
9	WSDL_WS1_124 3994895541_928 4	Tokens	:	AmendTradeRequest	"
10	WSDL_WS1_124 3994895541_928 4	Tokens	:	CancelTradeRequest	"
11	WSDL_WS1_124 3994895541_928 4	Tokens	:	MatureTradeRequest	"
12	WSDL_WS1_124 3994895541_928 4	Tokens	:	SingleDayTradeQueryRequest	"
13	WSDL_WS1_124 3994895541_928 4	Tokens	:	SingleDayTradeQueryRequestByTradeDate	"
14	WSDL_WS1_124 3994895541_928 4	Tokens	:	MultipleDayTradeQueryRequest	"
15	WSDL_WS1_124 3994895541_928 4	Tokens	:	MultipleDayTradeQueryRequestByTradeDate	"
16	WSDL_WS1_124 3994895541_928 4	Tokens	:	DateRangeTradeQueryRequestByTradeDate	"
17	WSDL_WS1_124 3994895541_928 4	Tokens	:	TargetSystemTradeIdQueryRequest	"
18	WSDL_WS1_124 3994895541_928 4	Tokens	:	SummitTradeIdQueryRequest	"
19	WSDL_WS1_124 3994895541_928 4	Tokens	:	TradeAuditHistoryRequest	"
20	WSDL_WS1_124 3994895541_928 4	Tokens	:	VerifyTradeRequest	"

4.6 Specifying the Learning

By evaluating the output of this iteration, the automatically extracted terms from each source revealed some motivating conclusions;

- The extracted list of terms presented to the ontology engineer forms a high-density list of domain specific concepts that would be harder to extract from textual sources.

- The domain concepts are very likely to be linked by non-taxonomic relations. Linking pattern structures to relations can lead to an effective way of extracting these relations automatically, which could result in an effective relations extraction from software artefacts, and would be very desirable to the ontology engineer. A list of condensed domain concepts is extracted automatically and presented to the domain engineer.
- Concept extraction as defined by Cimiano (2007) and Buitelaar, Cimiano & Magnini (2005) requires finding a concept extension (a set of concept instances), which can be found in SOAP messages. It is noticed from the output of this iteration that concept extraction can be emphasised by extracting the instance data from SOAP messages since they have information regarding service execution. Therefore, they are a suitable venue for the instance data.
- It is significant at this stage to build concept hierarchies linking the extracted concepts by taxonomic relations. It is observed from analysing the output from the STE method that some patterns can successfully lead to specific relations. Therefore, identifying patterns leading to concept hierarchies is an essential improvement to the system and require a new iteration to be initiated.

4.7 Summary

This iteration was intended to develop a service term extraction method by applying NLP techniques. The STE method is used to develop an initial SOLF that builds an initial ontology model consisting of automatically extracted domain concepts, has provided a conceptual understanding of IE constructs and their applicability on the OL, by demonstrating the feasibility of automatic ontology acquisition especially when the data sources are software artefacts like WSDL and XSD files. The contribution made here is the development of an initial ontology learning method. The method applies IE techniques, and starts by applying syntactic analysis as a pre-processing stage. The pre-processing is then used to identify patterns and perform concepts extraction based on the identified pattern. The process is automated by building a prototype application in GATE that implements the steps identified in the framework.

As a result of processing WSDL and XSD files, a list of concepts are automatically identified within these input files. The developed SOLF and the tool are evaluated by comparing the outputs to other similar methods. The outcome of this iteration illustrates that there is a sufficient amount of domain specific concepts in WSDL and XSD files that can be effectively extracted automatically by the STE method, since manual ontology acquisition from domain is a daunting task, engineers can benefit greatly from the lexical ontology model produced by the proposed OL approach. Automatically extracted service concepts can be used as a starting point in an ontology development process. There is a need to further investigate how to extract relations between these concepts, to allow for the automatic extraction of ontological relations between the identified domain concepts. Identifying patterns for concept and relation extraction is brought forward for the next Design Research iteration.

CHAPTER 5 - ITERATION 2

5.1 Introduction

Relation and concept taxonomy extraction forms two important layers of the ontology learning layer cake (as detailed in Chapter 2). Most OL research targets relation learning that is often from unstructured data sources. The aim of this iteration is to refine the SOLF developed in Chapter 4 by extending the framework to include techniques for concept taxonomy and relation extraction, where the research focus is to extract relations from Web Service artefacts that are classified as semi-structured data sources.

Extending the pattern-based ontology learning in Chapter 4 to include pattern-based relation learning can be achieved by applying a Structured Interpretation Patterns (SIP) extraction process. Here patterns are identified based on the output produced by applying the steps in Iteration 1, as presented in Chapter 4. Syntactically and semantically analyzed documents produced by the previous iteration are used as input to the SIP extraction process in this iteration. These SIP patterns are then integrated into the service ontology-learning framework (SOLF), by applying specifically tailored transformation rules to automatically produce ontological relations depicted in attributes and concept taxonomies. SOLF is instantiated in an ontology learning tool that can be used to learn a domain ontology model from Web Service artefacts.

This chapter is structured as follows. Section 5.2 provides the research design and the research outputs of this iteration. Section 5.3 presents the building and development of the design artefact (SOLF) – illustrating and detailing the newly incorporated relation extraction technique; including a rigorous pattern extraction method and the transformation rules development process followed by the last 2 steps of the framework; ontology building and ontology validation. Section 5.4 describes the implemented SOLF tool illustrating the application of each of the framework steps using a sample set of financial Web Services. Section 5.5 illustrates the evaluation of the research outputs using the appropriate evaluation metrics, with details of the experimental settings. The learning outcome of this iteration is presented in section 5.6 and finally the chapter is summarized in section 5.7.

5.2 Design Research and Output Artefacts

The purpose of this Design Research iteration is to build a relation extraction technique and incorporate the technique in SOLF. Relation extraction involves finding semantic relations between concepts. As noted in Chapter 2, two commonly applied Information Extraction approaches, related to relation extraction, are rule-based and machine learning IE systems. The first is based on the manual design of lexical patterns, which relies on implementing pattern-matching algorithms over linguistic annotations. The second type of IE system is the machine-learning system, in which the system is trained over manually annotated data to automatically learn new rules. This chapter proposes a method for the relation extraction task based on the first approach due to its simplicity and accuracy when rules are designed for specific domains (Sabou, 2005b; Cimiano et al., 2005).

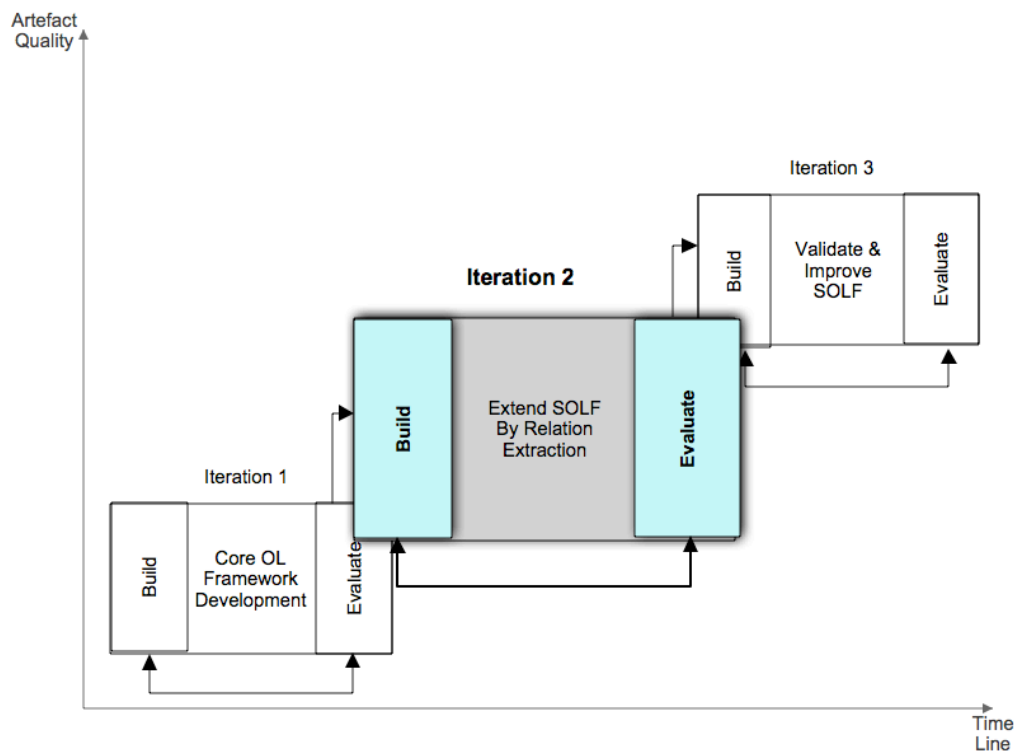


Figure 5-1: Research Iterations

5.2.1 Design Research Artefacts

This iteration introduces an automatic approach to apply pattern-based IE techniques to learn semantic relations between concepts in the semi-structured Web Service data sources (WSDL and XSD files), ultimately improving the developed framework (as discussed in Chapter 4) to include the ontological relation extraction technique. To achieve the aim of the research, this iteration executes the following steps (see Table 5-1).

Table 5-1: Iteration Steps Input Output model

Steps	Method	Input Artefact	Output Artefact
1. Identify Structured Interpretation Patterns (SIP)	SIP Extraction Process	POS-Term Model	SIP (Models) & (Method)
2. Develop transformation rules	TR Development Process	SIP Models	TRs (Models)
3. Refine and extend SOLF by incorporating Relation Extraction Process (REP)	Service Term Extraction Framework	OLD SOLF	Improved SOLF (Method)
4. Develop a prototype tool that implements SOLF	Build GATE Application	Web Service Artefacts	Prototype Application (Instantiation)

5.3 Artefact Building and Development

This section presents the building and development of a refined SOLF as illustrated in Figure 5-2. Each step in the SOLF is further described in the following subsections which integrate STE and the Relation Extraction process to learn a domain ontology model representing the underlying domain. The methodological framework using real sample set of financial Web Services. The application of SOLF on the sample set of services is detailed and demonstrated in the following subsections.

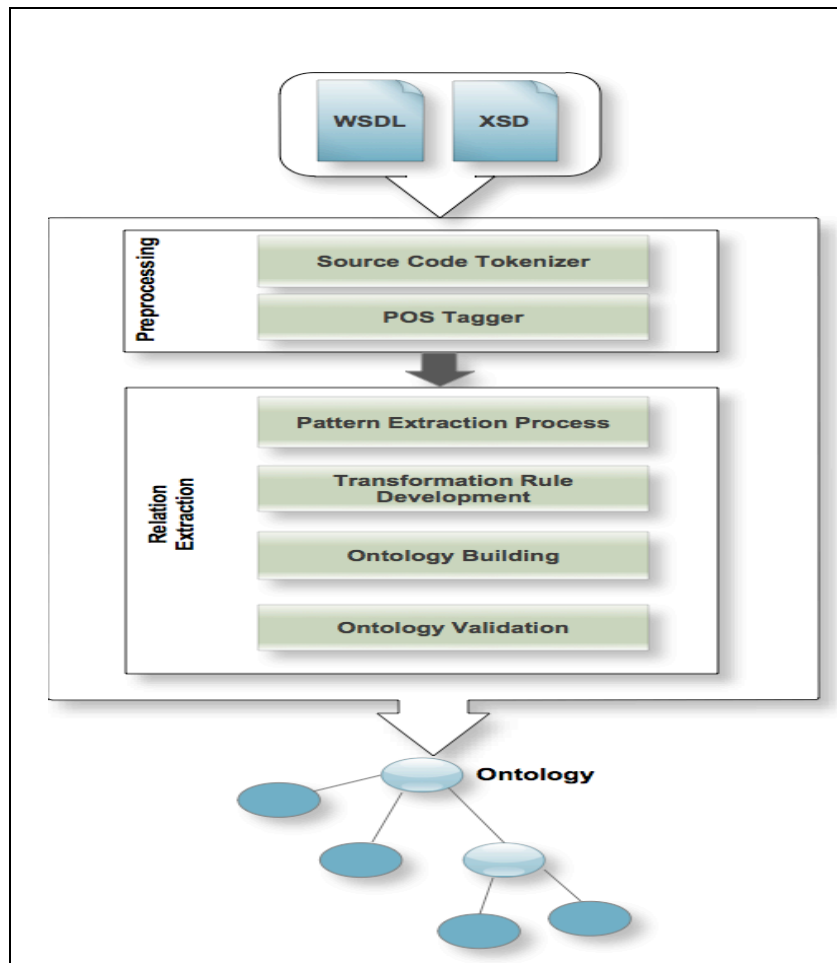


Figure 5-2: Service Ontology Learning Framework (SOLF)

5.3.1 Document Pre-processing Phase

In this opening phase, the Web Service artefacts are pre-processed by applying Natural Language Processing (NLP) techniques in order to linguistically analyze the input sources. This phase employs pre-processing as presented and discussed in Chapter 3. The tokenizer splits these 297 semi-structured files (WSDL and XSD) in the same manner as detailed in Alfaries, Bell & Lycett (2009). Application of these techniques enables rule-based extraction methods to be used on textual sources (Maedche & Volz, 2001; Maedche & Staab, 2004; Gacitua, Sawyer & Rayson, 2008; Gacitua & Sawyer, 2008).

5.3.2 Relation Extraction

The relation extraction technique adopted here is a pattern based relation extraction that targets both taxonomic and non-taxonomic relations. The technique requires the careful identification of patterns and transformation rules as described in the following sub sections.

Pattern Extraction Process

A particular relation can be automatically extracted by applying a set of structure interpretation patterns to identify that relation. In this phase language engineers/analysts identify relationships between concepts and identify associated patterns – known as Structured Interpretation Patterns (SIP). SIP are found in element and method names within the program code: They are similar to lexical syntactic patterns in IE in that they are based on the syntactic analysis of the corpus and they differ in the fact that they are not formed out of normal textual data sources. Here, patterns are identified using an efficient automated process based on the frequency analysis of automatically extracted terms.

The automated process is aimed at accurately deriving patterns (determined by pattern recurrence) that, when applied in a rule-based OL algorithm, results in higher precision. Identifying patterns extracted from semi-structured data sources, where domain knowledge exists, adopts Hearst's (1992) criteria and term frequency analysis.

Transformation Rule Development

This phase involves developing a set of Transformation Rules (TR), which are used to identify an appropriate ontological element for each SIP identified in the pattern extraction phase. For example, a subclass TR can be applied to map a term such as *"MatchingEnginePort"* to concepts and relations in OWL ontology. An illustration is provided in Figure 5-3.

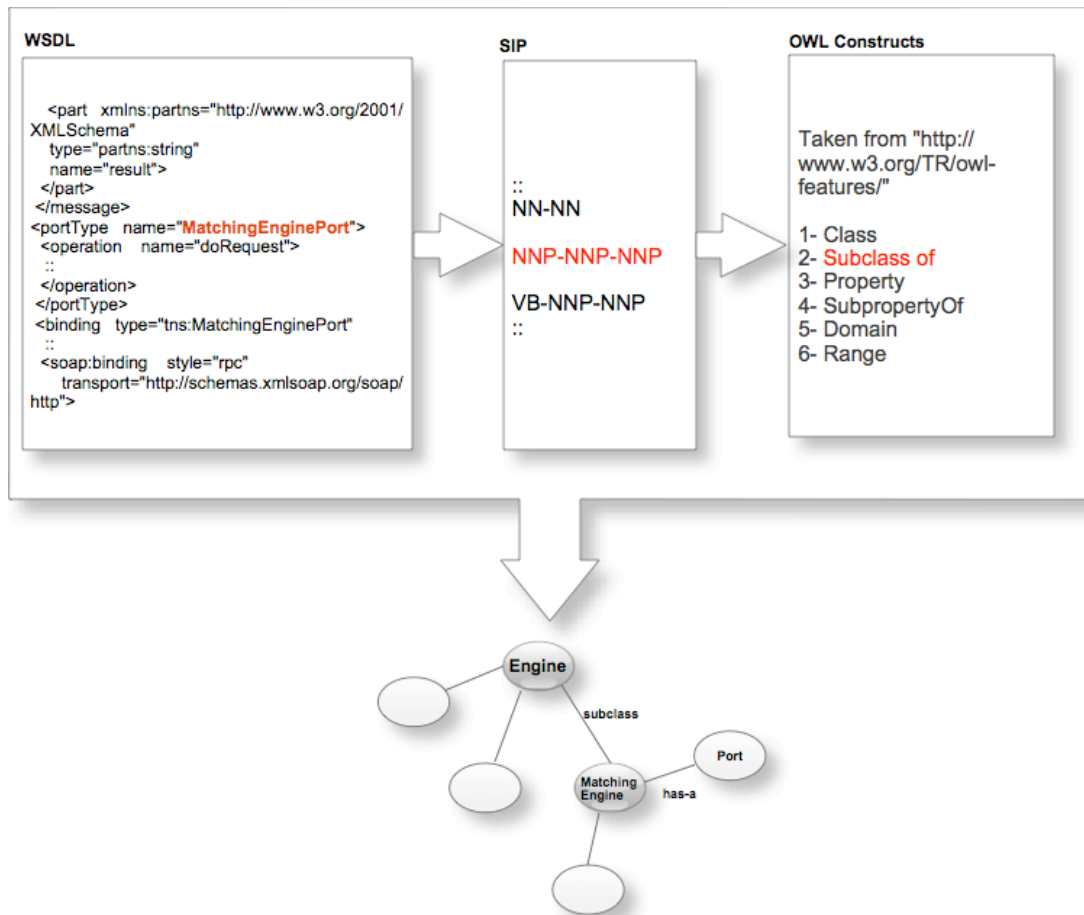


Figure 5-3: WSDL to OWL SIP Mapping

It is important to emphasize that rule development is likely to lead to different mapping possibilities depending on the underlying domain of study. TR development thus follows an automated process that aims to ensure optimal accuracy and limits subjective analysis. The process is capable of identifying the most appropriate mapping between the patterns and the underlying ontological element.

5.3.3 Ontology Building

Ontology building involves bootstrapping SIP and TRs by applying an appropriate rule-based pattern-matching algorithm. That algorithm searches for and annotates relations and concepts in the input sources and creates the corresponding ontological elements according to the TRs developed in the previous phase.

5.3.4 Ontology Validation

A domain expert is typically used to validate and modify the resulting domain ontology and filter out any irrelevant relations or concepts. The user is then able to view the automatically generated ontology and make any further changes or amendment to the rules or ontology.

A prototype implementation of SOLF in action, described in the next section, is created in GATE, General Architecture for Text Engineering (Cunningham et al., 2002), which provided the required development environment for implementing the SOLF tool. A set of three real-world Web Services taken from the financial domain are used for the pattern extraction, testing and evaluation of the framework. The chosen services (and their underlying descriptions) vary in complexity and style and are described in more detail in Section 4.5.1.

5.4 Application and Implementation of SOLF

Here, the same set of Web Services introduced in section 4.5 is used for the pattern and TR extraction process, where pre-processing is first performed consisting of two steps that are both implemented in GATE as two processing resources in the application pipeline. First, a WSDL tokenizer is developed to tokenize the input files into simple tokens, dealing with compound words and tokenizing WS1 phrases, such as “unwindTradeExtResponse”, into four distinct tokens instead of one. As Table 2 illustrates the output of a WSDL tokenizer step, where each word is identified as a token.

Table 5-2: Output of WSDL (WS1) Tokenizer Step

Annotation	Features
Token	{kind=word, length=6, orth=lowercase, string=unwind}
Token	{kind=word, length=5, orth=upperInitial, string=Trade}
Token	{kind=word, length=3, orth=upperInitial, string=Ext}
Token	{kind=word, length=8, orth=upperInitial, string=Response}

The second step uses the ANNIE POS tagger (Cunningham et al., 2002), adding part of speech tags to each token as a new feature. The output from this phase, as Table 5-3 illustrates, enables a pattern to be identified based on the category feature added here. For example, the POS tag of each token in the phrase “*unwindTradeExtResponse*” is added as a category feature, where Trade is tagged as *NNP*, as it denotes a singular proper noun according to the ANNIE POS tagger. Other tags such as *NN* and *VB* would have a different meaning, where the first is used to denote a singular or mass noun and the second denotes a verb in its base form (Cunningham et al., 2002).

Table 5-3: Output of the WSDL (WS1) POS Tagger

Annotation	Features
Token	{category=VB, kind=word, length=6, orth=lowercase, string=unwind}
Token	{category=NNP, kind=word, length=5, orth=upperInitial, string=Trade}
Token	{category=NNP, kind=word, length=3, orth=upperInitial, string=Ext}
Token	{category=NNP, kind=word, length=8, orth=upperInitial, string=Response}

5.4.1 Pattern Extraction

The approach adopted in order to improve the effectiveness of semi-structured artefact processing is now introduced. Each WSDL file is lexically analyzed by the previous phase, producing candidate terms with POS tags added to each term. Patterns are identified using these POS tags (initially ordering patterns by frequency). Typically, the identification of patterns starts by following a heuristic approach as detailed in previous research (Hearst, 1992; Berland & Charniak, 1999; Guo et al., 2007; Sabou, 2005a). The process is aimed at ensuring accuracy, specificity and coverage of patterns in a semi-structured data source as in WSDL or XSD files. The rationale behind SIP is to identify patterns that can be applied in a pattern matching based OL algorithm to extract suitable concepts and their taxonomic and non-taxonomic relations.

Initially, patterns are discovered by querying the underlying text, using GATE's ANNIC plug-in. The tool provides enhanced querying of input files with more flexibility than a simple search - especially if the files have been pre-processed, thereby allowing the search to be based on part-of-speech tags. Automating the pattern extraction process involves employing ANNIC and frequency analysis to produce a Web Service pattern extraction model for each service. Here, ANNIC is used to perform a live analysis and test each pattern directly on the input sources enabling the specificity and coverage to be assessed almost instantly (Maynard, Li & Peters, 2008).

The pattern extraction process consists of three main steps that are applied to all of the WSDL files that describe the services used for this experiment. **Firstly**, a generic query was written in ANNIC that produces a sequence of compound words extracted from the input sources. It is clearly noticeable that candidate domain concepts can be found in element names in WSDL files. The obvious query that returns all possible patterns from these element names would be a generic query that matches any sequence of words. Following the pattern extraction process proposed in Chapter 4 has lead to the following query, which is executed, on all Web Services as given below:

```
({Token.kind = word}) + 11.
```

This query extracts up to eleven tokens of type word (i.e. a sequence of letters followed by a word terminator). The output of this step is used to assess the coverage and preciseness of the overall extracted patterns by running the same query for each of the Web Services. Since WSDL files are a form of software artefact, it became very obvious from the preliminary analysis that candidate concepts and relations typically appear as a sequence of word tokens, e.g. operation names such as *SingleDayTrade* or *VerifyTrade*. All other text is XML related tags and symbols.

Secondly, the output from the first step is analysed to derive patterns corresponding to a semantic structure interpretation for each service. The frequency analysis of patterns is calculated as the number of occurrences of each pattern in each of the input sources. Consequently, a pattern extraction model is required for each service and is detailed in the next section. This is achieved by implementing a more specific query that produces matches of almost all-possible candidate patterns. This is directed from the analysis of

the output from the previous step. The result set is then exported as an html file to be filtered and analyzed in order to decide frequent patterns in each WSDL document. The executed ANNIC query for this step is shown in Figure 5-4.

<pre>({Token.kind=="word",Token.category=="VB"} {Token.kind=="word",Token.category=="VBG"} {Token.kind=="word",Token.category=="VBP"}) * 3 ({Token.kind=="word",Token.category=="NNP"} {Token.kind=="word",Token.category=="NN"} {Token.kind=="word",Token.category=="NNS"}) * 3</pre>

Figure 5-4: ANNIC Pattern Extraction Query

This query returns matches to compound noun phrases, formed of any sequence of verbs and nouns (up to three). For practicality of analysis and evaluation purposes it was decided to limit the query in this step to find up to 3 tokens of each type (verb and noun). In WS1, a total number of 29 unique patterns were found, some of which were frequently repeated giving a total sum of 383 occurrences in the WSDL file. A snapshot of the pattern extraction model for this service is produced in Table 5-4. In the matching engine (WS2), the pattern extraction in Table 5-5 shows less complex and fewer patterns - but similar in type to Web Service 1, patterns are found to be relatively frequent. Processing the WSDL file for the matching engine Web Service produced a total of 83 phrases (patterns matched) and a total number of 18 patterns. The Credit service is recognized to be a complex Web Service due to the fact that it contains more complex and varied functionality, resulting in more complex patterns than the other two Web Services. A phrase can be composed of up to 11 terms. The pattern extraction model for this service is shown in Table 5-6.

Table 5-4: Web Service 1 Pattern Extraction Model

Filtered Pattern Matches	Pattern	Count
tradecapture	NN	67
VerifyTrade	NNP+NNP	63
ICTML	NNP	63
SingleDayTrade	NNP+NNP+NNP	47
submitTrade	NN+NNP	19
verifyTradeRequest	VB+NNP+NNP	16
security policy	NN+NN	16
DateRangeTrade	NN+NNP+NNP	11
amend	VB	10
CPSign	NNP+NN	10
GetTrade	VB+NNP	9
services	NNS	9
TradeIdQuery	NNP+NN+NNP	8
TimeTradeDate	NNP+NNP+NN	6
mirrorTradeResponse	VBP+NNP+NNP	4
GetCreditDefaultSwap	VB+NNP+NNP+NNP	4
targetNamespace	VBP+NNP	3
mature	VBP	3
submitTradeId	NN+NNP+NN	3
encodingStyle	VBG+NN	2
Using	VBG	2
UsingPolicy wsdl	VBG+NNP+NN	1
UsingPolicy	VBG+NNP	1
using security policy	VBG+NN+NN	1
GetTradeAuditHistory	VB+NNP+NNP+NN	1
address location	VB+NN	1
ServiceName Name	NNP+NN+NN	1
schema xmlns	NN+NNS	1
operation soapAction	NN+NN+NNP	1
Total Matches 29	Total No. of patterns = 29	Sum = 383

Table 5-5: Web Service 2 Pattern Extraction Model

Sample Pattern Matches	Pattern	Frequency
body	NN	39
Action	NNP	10
body namespace	NN+NN	9
definitions	NNS	5
EnginePort	NNP+NNP	3
portType	NN+NNP	3
encoding	VBG	2
Type name	NNP+NN	2
doRequestResponse	VBP+NNP+NNP	1
doRequest	VBP+NNP	1
do	VBP	1
encodingStyle	VBG+NN	1
address location	VB+NN	1
address	VB	1
definitions xmlns	NNS+NNS	1
MatchingEnginePort	NNP+NNP+NNP	1
part xmlns	NN+NNS	1
portType name	NN+NNP+NN	1
Total Matches = 18	Total No. of patterns = 18	Sum = 83

Table 5-6: Web Service 3 Pattern Extraction Model

Sample Pattern Matches	Pattern	Frequency
solservice	NN	100
Upload	NNP	93
EntityType	NNP+NNP	61
NameCredit	NN+NNP	53
UploadCurveException	NNP+NNP+NNP	46
BloombergId	NNP+NN	41
CouponDate	NN+NN	34
defaultObligationName	NN+NNP+NN	16
NameCreditCurve	NN+NNP+NNP	16
CreditCurveName	NNP+NNP+NN	13
BloombergIdResponse	NNP+NN+NNP	12
obligations	NNS	11
approvePortfolio	VB+NNP	5
approveBasketCreditCurve	VB+NNP+NNP+NNP	5
owning	VBG	5
schema targetNamespace	NN+NN+NNP	4
approveBasketCredit	VB+NNP+NNP	4
owningTrader	VBG+NNP	4
ObligationsDescribers	NNS+NNP	3
approve	VB	3
pendingCurvesRequest	VBG+NNP+NNP	3
eportingGroupName	NNP+NN	2
REDPairId	NNP+NN+NN	2
docsEntityType	NNS+NNP+NNP	2
SettleDate	VB+NN	2
pendingRefEntitiesRequest	VBG+NNP+NNP+NNP	2
useParagonRatings	NN+NNP+NNS	1
ParagonRatings	NNP+NNS	1
approveSingleName	VB+NNP+NN	1
approveSingleNameCredit	VB+NNP+NN+NNP	1
target	VBP	1
targetNamespace	VBP+NNP	1
Total Matches =32	Total No. of patterns = 32	Sum=548

After analyzing each of the Web Services a **Third** step is undertaken, generalizing patterns over the sample Web Services by deriving the average relative frequency of each pattern across the three Web Services. Pattern frequency is used to ensure the discovery of as many instances of a relation. Due to the varying size and nature of the Web Services, a relative frequency is identified for the three Web Services for the most frequent patterns.

Table 5-7: Relative Frequency of SIP Across Three Web Services

Pattern	Frequency			Relative-Frequency		
	WS1	WS2	WS3	WS1	WS2	WS3
				Freq./229	Freq./25	Freq./335
NNP+NNP	63	3	61	27.51%	12.00%	18.21%
NN+NNP	19	3	53	8.30%	12.00%	15.82%
NNP+NNP+NNP	47	N/A	46	20.52%	N/A	13.73%
NNP+NN	10	2	41	4.37%	8.00%	12.24%
NN+NN	16	9	34	6.99%	36.00%	10.15%
NN+NNP+NN	0	N/A	16	N/A	N/A	4.78%
NN+NNP+NNP	11	N/A	16	4.80%	N/A	4.78%
NNP+NNP+NN	6	N/A	13	2.62%	N/A	3.88%
NNP+NN+NNP	8	N/A	12	3.49%	N/A	3.58%
VB+NNP	9	N/A	5	3.93%	N/A	1.49%
VB+NNP+NNP	16	N/A	N/A	6.99%	N/A	N/A
VBP+NNP+NNP	4	N/A	N/A	1.75%	N/A	N/A
VB+NNP+NNP+NNP	4	N/A	5	1.75%	N/A	1.49%

Table 5-7 summarizes the relative frequency. Due to the specificity of the financial domain and to ensure coverage and generality of the SIP the following criteria are adopted:

- The top 10 frequently occurring patterns are chosen.
- Patterns that occur only once are ignored.
- Patterns that represent a single term are eliminated since they only represent concepts not relationships.

To select the top 10 frequent patterns, the relative pattern frequency is calculated across the three Web Services according to the formula:

$$\text{Relative Pattern Frequency} = \frac{PO}{TP}$$

Relative Pattern Frequency = $\frac{PO}{TP}$, where PO is the number of occurrences of a pattern and TP is the total number of all patterns excluding the one-term pattern. Applying this formula has resulted in generating a relative pattern frequency as illustrated in Table 5-7. From this table the top patterns can then be selected for TR development as detailed in the next step. This will ensure that the patterns selected lead to relation extraction based on frequency analysis.

5.4.2 Transformation Rule Development

A particular ontological relation can be automatically extracted using the previously identified patterns to represent a particular relation. The output of the previous pattern extraction phase is analyzed and relations for each pattern are identified by the researcher and validated by a domain expert. For this process, a pattern relation identification model is generated for each of the patterns as exemplified in Table 5-8. Deciding a suitable transformation rule for each pattern is critical. Transformation rules are the result of implementing appropriate codified mappings between the pattern and the ontological relation/element that can be extracted. To ensure accuracy of the transformation rules, an automated extraction process is followed.

Table 5-8: Pattern Relation-Identification Model

Pattern Matches	Relation	Pattern	Total Matches
portType name	NN Has-A (NNP+NN)	NN+NNP+NN	16
ieldCurveld			
efEntityName			
currencyISOCCode			
issuerLegalName			
guarantorLegalName			
couponCurrencyName			
couponFreqName			
couponAccrualDate			
industrySectorName			
industrySectorId			
parentLegalName			
refEntityId			
defaultObligationName			
defaultObligationId			
ratingTierId			

This process involves identifying specific relations and finding patterns that indicate its existence. The research targets the two popular ontological relations *has-A* and

subClass-of. Here taxonomic relations are identified as subclass relations, representing the taxonomic layer of ontology models. Non-taxonomic relations are relations that are used to represent a relation between two concepts, where one is the domain and the other is the range (Cimiano, 2007, p.10). For the purpose of fully automating the extraction process, a decision was made to identify those relations with *has-A* relations and to associate the domain and range with the relation name.

There are some cases where more than one relation may apply, thus requiring a decision by the researcher as to the best fit for the patterns. For example, “*CreditCurve*” is a match for a pattern of type *NNP–NNP*, where both cases of relation may apply. The decision as to best fit was made based on the work of Hearst (1992) on pattern discovery criteria: that is, to choose the relation that covers most of the matches of a single pattern. So for the case above, although both relations are valid, the one that most accurately represents most matches is *subClass* (see Table 5-9).

Table 5-9: Sample Pattern-Relation Identification Model

Sample Matches	Possible Relation	Pattern
CreditCurve	subClass	NNP+NNP
EntityRequest	subClass/has-A	
EntitiesResponse	subClass/has-A	
ApprovalException	subClass	
PendingEntity	subClass	
PortfolioCredit	subClass	
IndexCredit	subClass	
ApproveBasket	subClass	
PendingCurves	subClass	
ApprovePortfolio	subClass	
ApproveSingle	subClass	
ApproveIndex	subClass	
CurvesType	has-A	

A number of patterns are found to have conflicting relations for the matches they represent. In some of these cases the conflicting terms are found to be non-domain terms - typically these words are found to be Web Services keywords such as Request or Response (e.g. “*EntityRequest*”) that matches the pattern *NNP–NNP*. Therefore, some form of filtering is required to deal with this issue. For these cases, conflicting relations for a single pattern are encountered, in which it is found that it most likely that

the match will be covered by a more complex pattern, for example, “*CurveUpload*” is matched by 2 patterns - *NNP–NNP*, *NNP–NNP–NNP* in “*CreditCurveUpload*”. This can be dealt with by processing the complex patterns first in order to ensure that these concepts will be created according to the more appropriate rule (i.e. more complex pattern). In order to apply these criteria for each of the three Web Services the following transformation rules are identified. Ontological relations are manually identified by the researcher and validated by a domain expert. The implemented transformation rules are summarized in Table 5-10.

Table 5-10: Summarized Transformation Rules

Rule	Pattern	Relation	Sample	OWL Construct
R1	NN+NN	Has-a	CouponDate	Coupon has-a Date
R2	NN+NNP+NN	Has-a	issuerLegalName	Issuer has-a LegalName
R3	NNP+NNP+NNP	Has-a	BasketCreditCurve	Basket has-a CreditCurve
R4	NNP+NNP	subClass	CreditCurve	CreditCurve SubClassof Curve

5.4.3 Ontology Building

Now that concepts and relations have been identified, it is possible to produce an explicit representation in ontological form. Ontology building is undertaken by implementing a GATE pipeline (see Figure 5-5) consisting of a sequence of JAPE transducers that apply a pattern-based matching algorithm to find and annotate concepts and relations, and then create the appropriate OWL construct accordingly. A JAPE transducer is created for each rule and another is created for each TR in the order illustrated in Figure 5-5. First a JAPE transducer implements the first rule (R1), as illustrated in Figure 5-6, and finds and annotates the pattern *NN–NN* with the appropriate tag. In this JAPE rule, domain and range concepts are annotated as such and a *has–A* rule is created. Then, a second JAPE Transducer performs the associated transformation rule, (see Figure 5-7) then finds that where the object property is created to represent the *has–A* relation that two OWL concepts are created, if they do not already exist, which are then associated with the newly created relation, thereby resulting in an OWL ontology model to be produced accordingly. See Appendix A for the remaining JAPE files.

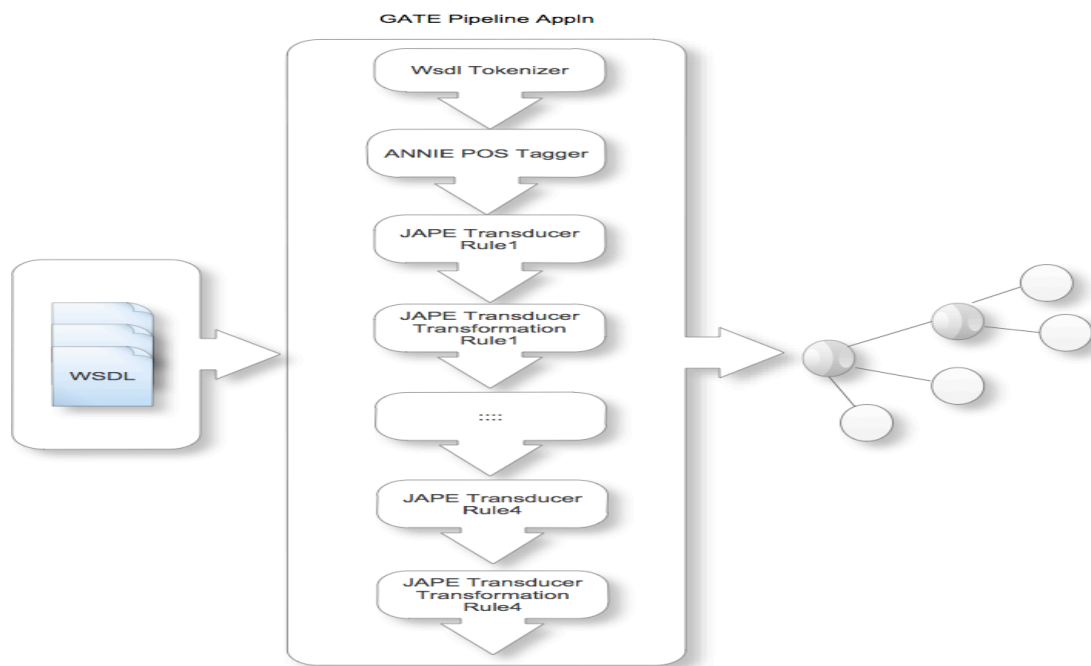


Figure 5-5: Application Pipeline Processing Steps

JAPE Rule 1
<pre> {({Token.kind==word,Token.category == NN}):domain ({Token.kind==word,Token.category == NN}):range):hasA --> :hasA.RelationIden={domain=:domain.Token.string,range=:range.Token.string,relation= "hasA-Rule1"}, :domain.Domain = {rule="Rule1 C1has-aC2"}, :range.Range = {rule="Rule1 C1has-aC2"} </pre>

Figure 5-6: JAPE Rule 1

JAPE Transformation Rule1
<pre> Rule: TransRule1 ({RelationIden}):relationIden --> :relationIden{Annotation theInstance = (Annotation)relationIdenAnnots.iterator().next(); String kind = theInstance.getFeatures().get("domain").toString(); gate.creole.ontology.OURI classURI = ontology.createOURI("http://example.com/classes#" + kind); gate.creole.ontology.OClass oClass = ontology.addOClass(classURI);} </pre>

Figure 5-7: JAPE Transformation Rule 1

The output produced in this phase is an OWL ontology consisting of concepts and taxonomic relations (*subClass*) between class and subclass concepts. Non-taxonomic relations (*has-A*) are also created between domain and range concepts using the GATE Ontology API. A sample snapshot, as presented in Figure 5-8, is visualised using Protégé 4.1, where straight arrow lines are used to symbolize taxonomic relations between concepts and dotted arrow lines correspond to the *has-A* relation between domain and range concepts

Figure 5-8: A Sample of the Learned Domain Ontology Model

An instantiation of the framework was developed using the GATE GUI as a prototype tool that enabled live evaluation of SOLF on the real set of Web Services (as presented in Section 4.5.1).

The evaluation is carried out to evaluate the generality of the produced patterns. Clearly, the frequency of the patterns and their being apparent in all three Web Services implies generality. Due to the different nature and complexity of the chosen Web Services and each being from the same domain, this fact ensures that if a pattern appears at the top of the list of each Web Service then it should be a generic pattern in other Web Services. The Domain expert and ontology engineers are involved in this process to assist in the

pattern extraction process and to evaluate the results of relation identification for each pattern in the TR step.

The SIP Patterns are evaluated for their coverage and preciseness, to ensure that the patterns cover all available terms in the corpus. The output produced from step1, as discussed in Section 5.4.1, produces all possible phrases from the data source, and by comparing that output with the output extracted using the identified patterns from step 1 should lead to the missing unidentified phrases that are available in the corpus. The Domain expert and the ontology engineer are used to validate the patterns and the extraction process thereby ensuring accuracy of the patterns. Applying the Brill tagger (offered by GATE) enabled a fully automatic tagging of tokens leading to accurate extraction. Although some inaccuracy occurred due to the fact that this POS tagger usually uses context information to detect the POS of a word, here only compound terms are identified as nouns and some nouns are identified as verbs. These are rare cases detected by the domain expert during validation, e.g. as in `targetNamespace` where `target` is tagged as a verb rather than as a noun. In such cases a minor error rate is expected, as the POS tagger applied is an off-the-shelf one that is mainly developed for textual sources.

5.5.2 Precision and Recall Evaluation Measures

As noted in Chapter 3, metrics for evaluating the learned ontology for its coverage and accuracy are borrowed from the IE field. These metrics are typically applied to evaluate automatically extracted information in comparison with manual extraction (Van Rijsbergen, 1979). Recall is used to measure the number of correctly identified concepts by the system for example, if 10 concepts are identified manually in the corpus and the system has automatically identified 7 of these 10 then 70% would be the recall figure. An ideal benchmark scenario for recall calculation is to use either a gold standard ontology (existing ontology) or a domain expert to extract concepts and relations manually from the input sources upfront (pre-create an ontology). Evaluation using a gold standard and automatically produced ontology can be misleading however (Sabou, 2005). Typically, an exact match is employed to compare and produce the results as a binary decision of correctness. When attempting a complex business area (such as that found in global banking) it is not possible to deploy a domain expert on all input sources.

This is due in part to the size of the input sources (in this case three software artefact files consisting of over 1200 lines of code). It is feasible, however, to utilize domain expert knowledge to evaluate concepts and relations produced by SOLF. Therefore, a hybrid approach has been adopted in order to better account for the domain complexity and availability of evaluative artefacts. The domain expert participates in evaluating the extracted concepts and relations, combined with a similarity-based evaluation for calculating the recall metric, between Reference (manually extracted concepts) and Response (the output of SOLF) the reference ontology is one that was produced manually for the same Web Services by previous work (Bell, Ludwig & Lycett, 2007).

It is noted that only four patterns are implemented due to time restrictions, which has limited the coverage of the produced model to cover fewer domain concepts than there are available. Consequently, it would only be reasonable to compare the learned ontology with a similar ontology covering the same part of the input sources. Therefore, the reference ontology is used to calculate pattern recall rather than a general recall. The evaluation here is designed to create the domain ontology in an incremental iterative manner. First, an ontology is created by executing the first pattern then the recall of this is calculated forming the first pattern recall. A second run is to incorporate the second pattern extraction to the first ontology and again the recall for the first and second pattern is calculated and so on. Adding one pattern extraction at a time, and calculating the recall each time a new pattern is added leads to evaluate how recall increases, as more patterns are included in the extraction process.

Table 5-11: Pattern Recall Summary

Patterns	Recall
Pattern 1	4.8%
Pattern 1&2	9.1%
Pattern 1,2&3	20.6%
Pattern 1,2,3&4	30.3%

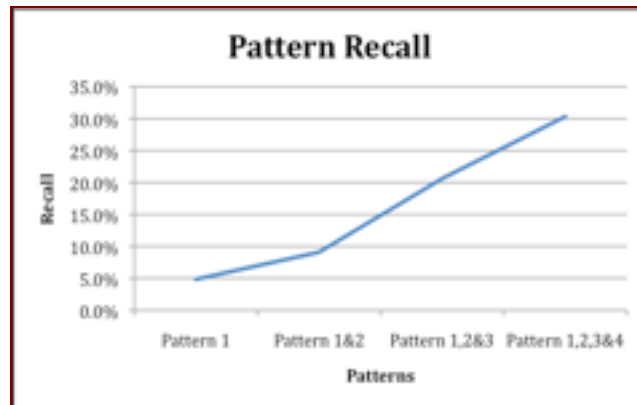


Figure 5-9: Pattern Recall Chart

Pattern recall is used to measure the number of concepts extracted from the corpus using the 4 patterns described earlier. Unsurprisingly, it is clear from the results presented in Figure 5-9 and Table 5-11 that recall increased as more patterns are added to the extraction process. Pattern 1 extracted 4.8% of the correct concepts, Pattern 2 increased the number of correct concepts to 9.1%, Pattern 3 further increased the number of concepts to 20.6% and lastly Pattern 4 reached 30.3% of correct domain concepts. It is clearly evident that Patterns 3 and 4 produced more concepts than Patterns 1 and 2; this could be related to the fact that patterns for implementation are randomly chosen for running the experiment on the four implemented rules. An interesting observation is that patterns that generate high recall might not necessarily generate high precision, and vice versa. In examining pattern recall and precision, it is clear that pattern-based ontology-learning leads to higher precision and lower recall. It is likely that the results would improve by implementing additional patterns to increase overall recall. It should be noted, however, that this research has targeted higher frequency patterns. More complex, less frequent patterns may yield some interesting results, i.e increased precision.

Table 5-12: Summarized Results for Precision

	Domain	Range	SuperClass	SubClass
Total Concepts	55	67	39	56
Correct Concepts	43	41	20	37
Precision %	78.18	61.19	51.28	66.07

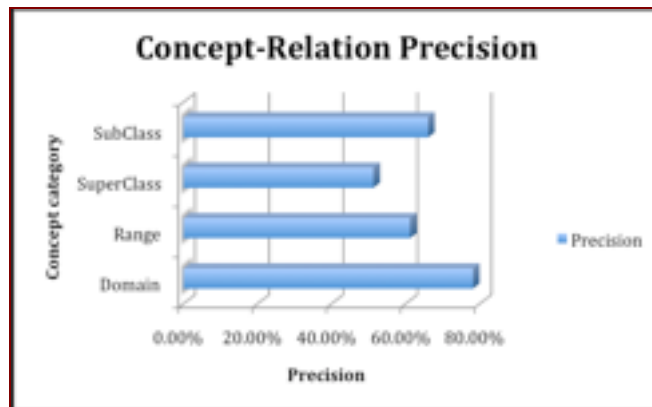


Figure 5-10: Concept-Relation Precision Chart

The results produced for this purpose are a list of domain and range concepts that represent *has-A* relations and a further list of super-class and sub-class relations. The domain expert then scored each list to represent domain, range, super- and sub- classes, identifying the correct and incorrect concepts in each list. See Table 5-12 and Figure 5-10 for the summarized results (See Appendix D for full list of scored concepts).

Running the prototype application over the three Web Services also revealed some insights regarding eliminating some of the spurious matches that might lead to WSDL related terms rather than domain terms. For example, the extracted pattern *NN-NN* was

produced as a result of `<wsdl:part name="amendTradeRequest"...>` - part name being detected. These WSDL specific concepts were discarded.

5.5.3 Qualitative Evaluation

A qualitative evaluation for OL (Sabou, 2005) is one that assesses the sufficiency of ontology as a conceptualization of a certain domain. In addition to the quantitative evaluation, several interesting insights have arisen from deriving and executing the pattern extraction process (and subsequently confirmed by a domain expert during the quantitative evaluation). Due to the complex nature and complexity of the chosen Web Services and the domain in which they reside, commonality in domain terminology ensures that if a pattern appears popular in each Web Service then it is likely to be a generic pattern in other Web Services. The systematic way in which patterns were derived ensures an ongoing evaluative process. The frequency of each pattern with distinct Web Services also indicates a measure of generality.

The extraction process was an iterative evaluative process in its formation from undertaking the process steps. Patterns are evaluated for their coverage and preciseness as they are identified, in order to balance between specificity and coverage of patterns. ANNIC has enabled the testing and assessment of pattern coverage almost instantaneously (allowing micro-level tests to drive process adaption). To ensure the patterns cover all available relations in the corpus, the output produced from initial queries produces all possible phrases from the data sources. Comparing this output with the output extracted using the identified patterns has lead to the identification of missing phrases that are available within the corpus.

The precision of concept extraction achieved here (78%) is considered promising when compared to the results of Sabou (2005), who achieved up to 54% extractable concepts from service description in Javadoc files. It would seem natural, however, to combine methods in a complementary manner - both the methods themselves and the source software artefacts (WSDL, Javadoc, Schema etc.). It is claimed that pattern-based extraction approaches typically achieve low recall and higher precision (Cimiano et al., 2005). The approach presented here has the potential to overcome the low recall drawback due to the fact that patterns are automatically extracted by applying frequency analysis to ensure patterns with higher frequency are used for relation extraction.

5.6 Specifying the Learning

The learning outcome of this iteration is as follows:

It is observed that patterns that appeared with high frequency in large WSDL files (i.e. those that did not have an accompanying XSD) did not appear at all in other Web Services, which raises an important question about the effect of the type and size of the WSDL on the pattern extraction process (weighting of patterns), and more specifically on the correlation between frequency/popularity and precision. This can be addressed by applying the extracted patterns on another set of Web Services (including different types of WSDL and including XSD files). Investigating the effect of the WSDL file size and style on the pattern extraction process is therefore an important area to investigate.

The existence of one pattern as part of another more complex pattern, i.e. *NNP-NNP* is part of *NNP-NNP-NNP*, which might lead to having to make a choice as to which one is more appropriate. Hence, the observation in TR development, that when a pattern contradicts a relation suitable for other matches of the same pattern, leads to the fact that the pattern either consists of WSDL keywords rather than domain concept, or the fact that the pattern is part of another more complex pattern.

In order to take this research forward, the following issues initiate a new Design Research iteration:

- The generality of the extracted SIP patterns and TRs across different domains needs further examination. i.e. test the applicability of SOLF and the extracted patterns from Iteration 2 on different domains using a sample set of Web Services.
- More complex patterns need to be included. In this iteration only up to 3-term patterns were extracted. It is established that in more complex services patterns of up to 11 terms exist. Complex patterns have less frequency and might therefore reveal more important/specific relations. Including more complex patterns, as part of the learning process might enable wider coverage of relations and concepts.
- The possibility of generalizing existing patterns needs further investigation, i.e. *NN*, *NNP* and *NNS* are all different types of nouns. Is it possible to include these under the one category of type *NOUN*? - i.e. will the same patterns give the same results?

- Identifying more domain specific relations needs further analyses and investigation. Relations identified are: subclass and *has-A* relations. Is it possible to define patterns that will lead to more specific relations? Will more complex patterns lead to more specific relations?
- The ability to incorporate WSDL structure with SOLF, to identifying and add new domain specific relations needs to be tested. Is it possible to use the WSDL structure to lead to other relations?, e.g. to attribute the relation *has-A* between complex types and sub-elements.

5.7 Summary

The work here presents a Service Ontology Learning Framework (SOLF), the core aspect of which extracts Structured Interpretation Patterns (SIP). These patterns are used to automate the acquisition of ontological concepts and the relations between those concepts. Identifying patterns is an important step that requires rigour, and the use of the framework ensures accuracy, generality and coverage of SIP. Three real-world Web Services from global banking systems were used for pattern extraction and rule development as the means to evaluate the framework. The output of the SOLF process is an automatically generated OWL domain ontology, which presents a number of financial domain concepts extracted from the Web Services.

It can be seen that the automatically learned ontology, moves beyond basic taxonomy – extracting and relating concepts at a number of levels. Evaluation of applying SOLF of the set of services used for pattern extraction raised a number of issues that direct further improvements. More importantly, the precision achieved by the Domain expert evaluation directs the next framework improvement towards testing the generality of the extracted SIP and TRs across other domains. This requires a applying a more rigorous independent evaluation measures to prove the generality and effectiveness of SOLF.

CHAPTER 6 - ITERATION 3

6.1 Introduction

Automatically extracting domain specific non-taxonomic relations is one of the challenging tasks of OL (Weichselbraun, Wohlgenannt & Scharl, 2010; Snow, Jurafsky & Ng, 2006; Manine, Alponse & Bessières, P, 2008). The results achieved in the last iteration from applying the SOLF framework on the sample set of financial Web Services further developed a SOLF tool, a set of SIP patterns and transformation rules that can be applied to extract taxonomic and non-taxonomic relations. The automatically extracted SIP patterns from WSDL files of the sample set of services. This chapter aims at proving SOLF and generalizing the SIP patterns and the transformation rules by validating and evaluating their applicability across other domains. This involves a thorough evaluation of the taxonomic and non-taxonomic relation extraction, requiring a set of carefully selected Web Services with gold standard ontology specifically built for those services. The literature, as discussed in Chapter 3 presents theoretical definitions for performing non-taxonomic evaluation measures, such as the taxonomic and non-taxonomic overlap, but lacks the illustration of how these measures can be practically applied (Velardi et al., 2005; Cimiano, 2007; Dellschaft & Staab, 2008). This iteration contributes a detailed practical evaluation addressing the different layers of ontology.

The chapter is structured as follows. Section 6.2 presents how Design Research is applied to execute this iteration as two evaluative mini iterations. Section 6.3 describes the first mini iteration that applies a gold standard based evaluation on the dataset. A refined and extended SOLF is presented that incorporates new relations extraction technique. Then evaluation measures are applied to evaluate different aspects of the ontology models. Section 6.4 presents the domain expert evaluation of the learned models. The learning outcome of this iteration is discussed in section 6.5. Finally the chapter summary is presented in Section 6.6.

6.2 Design Research and Output Artefacts

The learning outcome of Chapter 5 has directed the SOLF improvement in this iteration towards proving its efficiency across other domains. In essence providing the theoretical ground for the research to illustrate how and why the approach proposed in the SOLF can provide an efficient solution to the problem space. The application of SOLF on the set of Web Services from which the patterns and transformation rules were extracted, achieved the promising precision cover of up to 79% in the previous iteration. Intuitively, in order to take this research to the next level, it is vital to validate the generality of the SOLF tool and the developed SIP patterns by understanding how and why they are applicable across other domains. This iteration aims at developing and applying a more rigorous evaluation framework that satisfies OL evaluation criteria as suggested by Dellschaft & Staab (2008). An iterative Design Research process is aimed at developing a thorough evaluation of the research. As Ontology learning is considered a recent research area where the knowledge base is still raw, the evaluation poses a challenging task as the knowledge base lacks well-defined practical evaluation methods. Therefore, Design Research iterative process forms a suitable method to expose and develop a practical and thorough evaluation method. The process executed here involves two mini Design Research iterations. The purpose of this iteration as a whole is to effectively utilize SOLF to learn domain ontology models from new sets of Web Services. Evaluating the OL approach is achieved by applying rigorous evaluation measures and methods from the knowledge base as presented in Chapter 3.

The first mini iteration executes a build and evaluate cycle suggesting new refinement to the research artefacts (SOLF and patterns). Two sets of Web Services (Books and Financial domains) are operated on by SOLF, producing two automatically built domain models. The sets of services are accompanied by manually built gold standard ontology (GSO). Those GSO models are developed specifically for the accompanying service by other research projects (ISLAB and LSDIS). In both cases the GSO are built for the task of service matching. A gold standard based evaluation method is applied to evaluate the automatically built SOLF ontology (SOLFO). The evaluation criteria are to determine the preciseness (accuracy) and coverage of the learned SOLF ontology at three different levels; (1) Lexical Layer, (2) Taxonomic layer and (3) Non-taxonomic layer. The evaluation metrics of precision and recall are again applied. Precision and recall metrics

are applied to evaluate each of the three layers of the SOLFO model. For example lexical precision is implemented to evaluate the accuracy of the lexical layer of SOLFO as detailed in the following section. As there is no clearly defined practical way of calculating those measures this iteration makes another Design Research contribution in the form of an evaluation model for the Non-taxonomic relation evaluation.

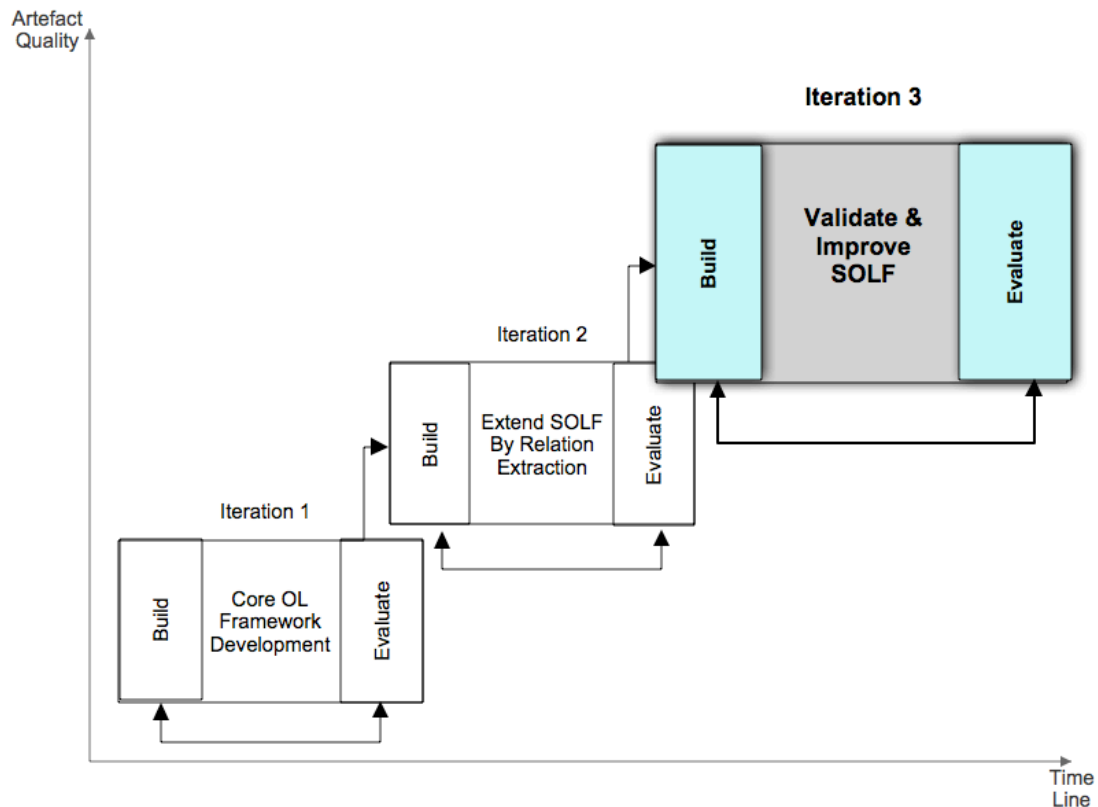


Figure 6-1: Overall Design Research Iterations Framework

6.3 SOLF Refinement and Gold Standard Evaluation

A preliminary analysis of applying the SOLF on the new sets of services has identified the need to extend it to adopt relations embedded in the WSDL structure. Hence, the first improvement to SOLF would be to allow for amending the pattern extraction and transformation rules to incorporate the WSDL structure. Hence, the final refined SOLF is illustrated in Figure 6-2.

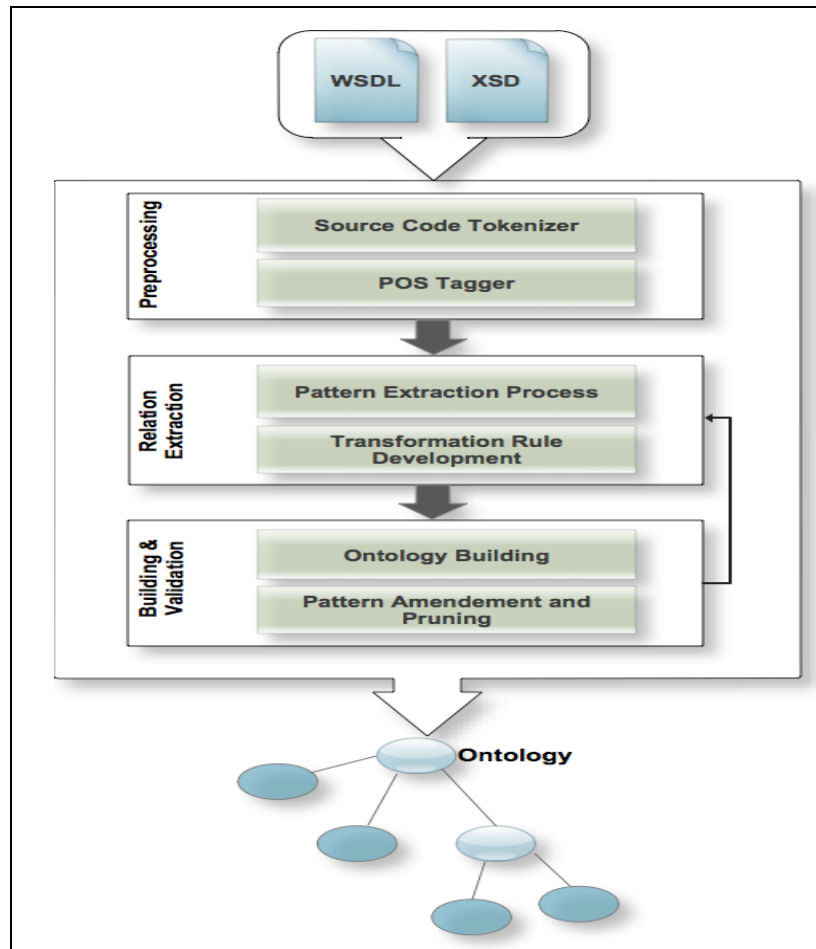


Figure 6-2: Service Ontology Learning Framework

The new improved steps are discussed in more detail in the next subsections, where the first improvement is to improve the validate ontology step and refine the patterns step to allow new patterns to be developed and added as required and as decided by the domain engineer. The second improvement is to employ an ontology pruning step allowing domain expert interaction to improve and finalize the ontology. The final refined framework can be summarized in five main phases, as illustrated in Table 6-1. The table presents a formal definition of the output of each phase.

Table 6-1: Formal Definition of SOLF Output Phases

Phase	Name	Description	Output
1	Pre-processing Phase	Apply linguistic techniques on Web Service artifacts.	T is a set of tokens, that is $T = \{ t: t \text{ is tagged with POS} \}$.
			$C = \{ c: c \text{ is a concept } \subseteq O \}$ where O is an ontology.
2	Pattern Extraction	Automated Structure Interpretation Pattern Extraction Process.	$S = \{ s: s \text{ is a pattern for a concept } c \text{ where } c \in C \}$.
3	Transformation Rule Development	Develop transformation rules for each pattern.	$TR = \{ f: S \rightarrow R \}$ where $R = \{ r: \exists \text{ a term with two concept } c_1, c_2 \text{ such that } c_1 R c_2 \}$.
4	Ontology Building	Apply rule based pattern-matching algorithm to automatically build ontology.	$O = (C, \leq_c, R)$ As defined by Cimiano (2007), Where \leq_c is a semi-upper lattice on C with top element $root_c$, called concept hierarchy or taxonomy.
5	Ontology Validation and Pruning	Allow user to prune and modify ontology.	$O_U = \{ O: O \text{ is user pruned} \}$.

6.3.1 Validate Ontology and Amend Patterns

Incorporating ‘validate and amend’ step in SOLF enables going back to pattern extraction step to add new patterns. The pipeline can be regenerated to incorporate new patterns and rules to extract new concepts and relations and add them to the ontology model. This step is necessary to allow for the flexibility of the framework and enable the ontology engineer to go back to the pattern extraction phase to add new patterns and create new TR. Adding this step after the ontology building allows the developer to validate the ontology first and consider adding new patterns or removing rules if necessary.

6.3.2 Incorporating WSDL Structure in SOLF

An initial pattern analysis was performed using the ANNIC GATE plugin to get an insight into discovering links between the patterns and structure. The WSDL structure is therefore analysed to discover new relations. The obvious pattern structure is the complex type structure, which might reveal an object property relation linking domain and range concepts with a *has-A* relation. This resulted in the identification of new patterns and creation of the necessary JAPE code to identify and create the *has-A*

relation as an OWL object property. The new relation links complex types and their inner elements, e.g. *MarketNews* and *Time* as illustrated in Figure 6-3. In some Web Services where complex patterns are used less frequently, the WSDL structure, between the complex type name and the sequence elements, revealed an important relation addition to SIP extractions.

```
<s:complexType name="MarketNews">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Headline"/>
    <s:element minOccurs="0" maxOccurs="1" name="Time"/>
    <s:element minOccurs="0" maxOccurs="1" name="Source" />
    <s:element minOccurs="0" maxOccurs="1" name="Url"/>
    <s:element minOccurs="0" maxOccurs="1" name="Summary"/>
  </s:sequence>
</s:complexType>
```

Figure 6-3: Financial WSDL Code Sample

The adding of JAPE transducers to implement patterns to map the complex type structural aspects of WSDL is implemented as an important extension of the relation extraction phase. This involves creating new constructs to parse the WSDL files and annotate complex types and attributes as Domain and Range concepts. This is achieved by adding the necessary JAPE rules to the SOLF tool. The first JAPE rule is designed to first parse the WSDL files and identify the name attribute in the complex type tag as the domain concept, and then identify the inner elements name as the range of the relation. The JAPE rules added to SOLF tool are illustrated in Figure 6-4.

```

Phase: ComplexRelation
Input: Token Tokens COTag InnerElement SeqTag
Options: control = applet
Macro: EndElementTag
Rule: fullElementTag

Priority: 50

(
//Finds the name attribute of the complexType tag

    ((({COTag}

        {Token.string == "name"}
        {Token.string == "="}
        {Token.kind == punctuation})
        ({Token.kind == word}):className
        ({Token.kind == punctuation}
            {Token.string == ">"})
        ({Token.kind == punctuation}
            {Token.string == ">"})
        ({COTag}
            {Token.string == ">"})
    )
    ({SeqTag})
    // This looks for the first element.
    ((({InnerElement}
        ({Token.kind == word}
            ):elementAtt1
        {Token.kind == punctuation}
        (EndElementTag)?
        (minmax)?
        ({Token.kind == punctuation})?
        ({Token.string == "/"})
        ({Token.string == ">"})
        )?
    ):att1
):complexRelation
-->
:complexRelation.ComplexRelation = {rule = "ComplexRelation", Class =
:className.Token.string , Attribute=:elementAtt1.Token.string },
:att1.Att1 = { Class = :className.Token.string,
Attribute=:elementAtt1.Token.string},

```

Figure 6-4: Sample Complex Relation JAPE Rule

Another JAPE rule is added to perform the transformation from the complex type (WSDL structure) in order to formulate an owl object property representing the *has-A* relation between the complex type name attribute and the inner elements. The developed rule is presented in Figure 6-5.

```

Phase: RuleNNNN
Input: Att1
Options: control = applet
// Complex Type Transformation Rule
Rule: ComplxTypeTR
({Att1}):relationIden
-->
:relationIden{
    System.err.println("Hello ComplexTypeTransRule");
    Annotation theInstance =
(Annotation)relationIdenAnnots.iterator().next();
    //get the domain and range strings from the features of Annot
    String domain =
theInstance.getFeatures().get("Class").toString();
    String range =
theInstance.getFeatures().get("Attribute").toString();
    String hasA = "-Has-A-";
    String Oproperty = domain + hasA + range;
    // Create URI for domain and range 24
    gate.creole.ontology.OURI domclassURI =
ontology.createOURI("http://example.com/classes#" + domain);
    gate.creole.ontology.OURI rngclassURI =
ontology.createOURI("http://example.com/classes#" + range);
    // Add domain and range concept to ontology
    gate.creole.ontology.OClass Domain =
ontology.addOClass(domclassURI);
    gate.creole.ontology.OClass Range =
ontology.addOClass(rngclassURI);
    //check if property exist then
    gate.creole.ontology.ObjectProperty xxx =
ontology.getObjectProperty(ontology.createOURI("http://example.com/c
lasses#" + Oproperty));
    if ( xxx==null)
    {
        // Create Domain and Range Sets and add Domain and Range
classes
        Set<gate.creole.ontology.OClass> theDomain = new
HashSet<gate.creole.ontology.OClass>();
        Set<gate.creole.ontology.OClass> theRange = new
HashSet<gate.creole.ontology.OClass>();
        theDomain.add(Domain); // the class you have for the
domain
        theRange.add(Range); // the class you have for the range
        ontology.addObjectProperty(
            // create the URI for the new property:
            ontology.createOURI("http://example.com/classes#" +
Oproperty),
            theDomain,
            theRange);
    }
    else
    {
        Set<gate.creole.ontology.OResource> theDomain=
xxx.getDomain();
        theDomain.add(Domain);
        Set<gate.creole.ontology.OResource> theRange= xxx.getRange();
        theRange.add(Range);

        System.err.println("hello object property has A exist");
    }
}

```

Figure 6-5: Complex Relation Transformation Rule

A sample ontology model is automatically built, as illustrated in Figure 6-6, using SOLF. The model illustrates the integration achieved by the improved framework; SOLF integrates the SIP pattern based extraction techniques and structural techniques, revealing valuable structural additions between the domain concepts. Here, ontology engineers can benefit from the variety of automatically extracted relations from the structured WS artifacts, where both taxonomic and non-taxonomic relations are automatically extracted and added to the domain ontology representing the underlying Web Services. Here, as the diagram illustrates, the concept *News* and its subclasses *StockNews* and *MarketNews* are successfully extracted using SIP patterns. Extending SOLF to cater for the structural aspect of WSDL and XSD resulted in new object properties (*has-A*) being added to the ontology model. These additions are illustrated by the new dotted arrows linking *MarketNews* to new concepts like *Source*, *Time* and *Headline*. Another interesting observation from the sample produced in Figure 6-6, is that new concepts are revealed that are domain specific and subject to deep domain expert understanding of the underlying domain. For example, *High* is a domain concept in the financial domain that represent different types of subclasses. The concept and its subclasses, such as *DayHigh* and *WeekHigh*, are automatically learned by SOLF.

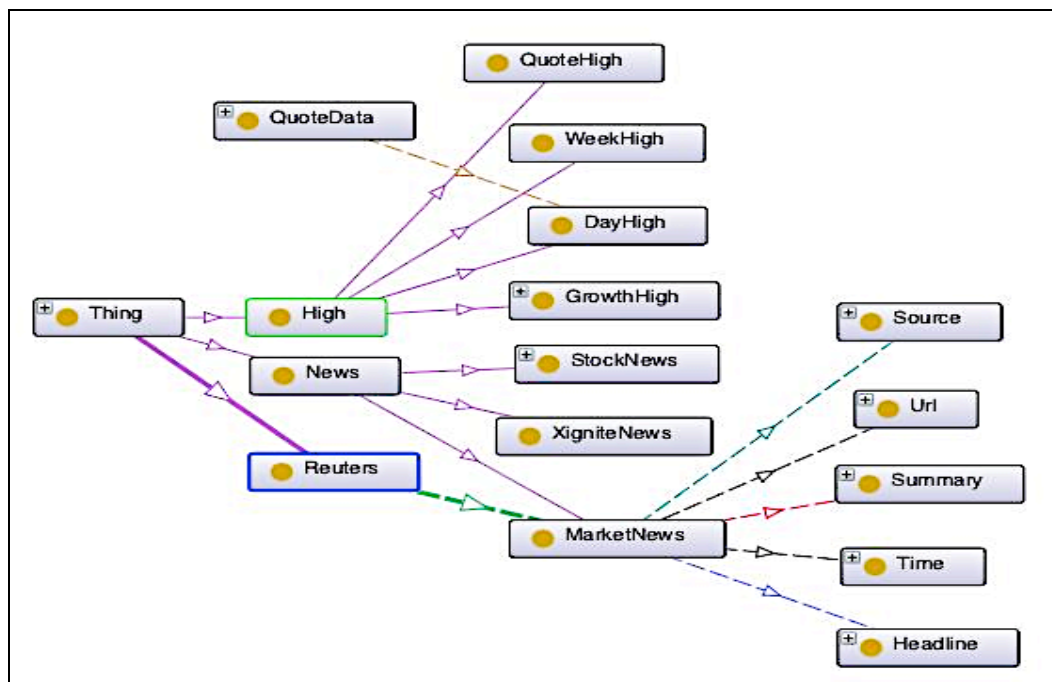


Figure 6-6: Sample SOLF Ontology model (Group 2)

6.3.3 Ontology Pruning

Validation of the learned model requires the expertise of both domain experts and ontology engineers. The process presents an initial domain ontology model consisting of both lexical and structural layers to the domain expert for manual validation. Importantly, this phase enables the expert feedback to direct the restructuring of the pipeline by the ontology engineer where necessary. An Ontology Pruning step is needed to allow the ontology engineers to filter out any irrelevant concepts or relations. Domain experts can apply this step in either a strict or lenient manner. The pruning strategy applied here is a strategy that eliminates concepts that are not domain specific, such as Web Service keywords or XML tags. It is noted by the domain expert that there are duplication in concepts. Where these concepts differ only spelling or abbreviation. These concepts can only be removed if the domain expert decides that they refer to the same concept. A basic pruning step would include eliminating the duplicate concepts that vary in case letters, such as Publisher and publisher. It is important that the pruning step is carefully executed in order to allow the domain expert to learn synonyms and concept extensions during the pruning step.

6.3.4 Experimental Data and Evaluation

The evaluation of the first mini-iteration is to measure the learned model for accuracy and coverage of the underlying domain. The evaluation of this iteration follows a gold standard based evaluation method as noted in Section 6.2. This type of evaluation is typically based on performing a manual comparison between the learned ontology (SOLFO) and the gold standard ontology (GSO) (Dellschaft & Staab, 2008). The evaluation is performed at three different layers, as applied and detailed in the next subsections. The measures applied here are carefully designed to be independent of each other, based on Dellschaft & Staab (2008) suggested evaluation criteria.

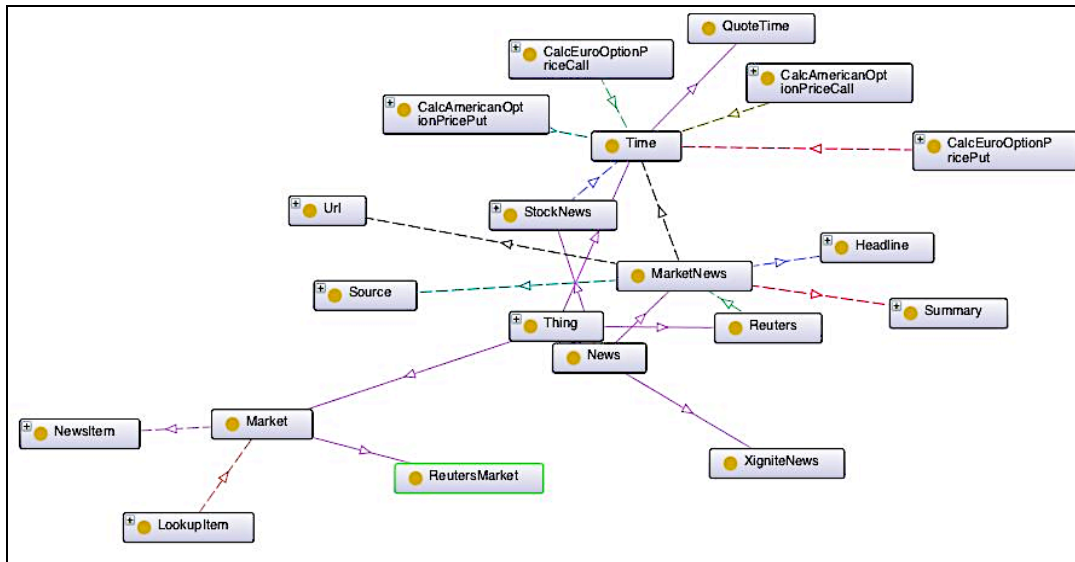


Figure 6-7: Sample of the Financial Learned Ontology (SOLFO)

An ideal scenario in which to perform the experiment is to be able to find real sets of services with accompanying Gold standards ontology models built specifically to represent the services. Since ontology development is still a difficult and expensive task these sets are not widely available. Two sets of services from different domains (Books and Financial) are used for this experiment because they were made available by previous research. Each set consists of 5 Web Services and a gold standard ontology model built for those Web Services. The steps followed to prepare the data for the evaluation are:

- Create a corpus consisting of 5 Web Services in GATE
- Run the SOLF application with the existing SIP and TRs produced from the previous iteration. (Here, use original pipeline first, then add the complex relation JAPE files)
- Save the automatically learned ontology by the system as an owl file.
- Create a table that consists of two columns, representing the Gold standard ontology (GSO) and the SOLF ontology (SOLFO) respectively. This step ensures a practical way of managing the evaluation of the different layers of the ontology model.

The first data set (Group1) represents 5 Books Services and an accompanying GSO the sample set of services and the ontology are provided in Appendix C. The second data set (Group 2) represents 5 stock exchange financial services and an accompanying ontology, again as a GSO (see Appendix C for the set of web financial services and the GSO). It is clearly indicated in the literature that two ontologies can be compared at three different levels: (1) lexical layer evaluation, (2) taxonomic relation evaluation and (3) non-taxonomic evaluation. The first is used to determine the similarity of the two ontologies at the lexicon level (concepts). The second and third are used to determine the structural similarity of the two ontologies. In the following section precision and recall are used to indicate the accuracy and coverage of the learned ontology (SOLFO) by comparing it to the gold standard ontology (GSO).

Generally, the precision and recall metrics are used to measure the performance of the OL approach, where precision is used to judge the accuracy of the learned ontology model and recall is used to judge the coverage of the domain by the learned ontology model as there is no exact method or guidelines for how to actually calculate those measures. The suggested method by Dellschaft & Staab (2008) is that the gold standard based evaluation is the ideal scenario. In the next subsections the GSO is used as a benchmark for scoring the accuracy of concepts and relations.

6.3.5 Domain Coverage - Lexical Layer

SOLF performance is determined by evaluating the domain coverage of the lexical layer. The lexical precision (LP) and lexical recall (LR) are calculated according to the following definition as adopted from Dellschaft & Staab (2008):

$$LP(O_{SOLFO}, O_{GSO}) = \frac{|C_{SOLFO} \cap C_{GSO}|}{|C_{SOLFO}|} \quad (1)$$

Where O is an ontology, C is the set of concepts, SOLFO is the ontology learned by SOLF and GSO is the gold standard ontology.

$$LR(O_{SOLFO}, O_{GSO}) = \frac{|C_{SOLFO} \cap C_{GSO}|}{|C_{GSO}|} \quad (2)$$

The F1 measure is normally used to give a summarized value of precision and recall.

$$F1(O_{SOLFO}, O_{GSO}) = \frac{2 * LP(O_{SOLFO}, O_{GSO}) * LR(O_{SOLFO}, O_{GSO})}{LP(O_{SOLFO}, O_{GSO}) + LR(O_{SOLFO}, O_{GSO})} \quad (3)$$

The comparison is carried out manually as the sample evaluation model illustrates in Figure 6-8. This model is used to identify correct, incorrect and total number of concepts in each model. Then the precision and recall are produced accordingly. The model is used to manually analyse each concept in the learned ontology against its existence in the Gold Standard Ontology (GSO). As the figure illustrates, the first and second columns represent the list of pruned SOLFO concepts. The third column represents the set of concepts from the GSO. Here, it is important to understand that the lexical layer represents the set of all concepts of an ontology, including super and subclasses regardless of their position in the concept hierarchy. For illustration purposes the subclass concepts are right justified in each column.

Lexical Layer Evaluation Model		
Pruned SOLFO	Pruned SOLF Continued	GSO
Account	infoResponse	Availability
subAccount	Search	Book
Array	KeywordSearch	Category
BookArray	BookService	Chapter
ChapterArray	ServiceWSS	CoverImage
Author	Status	Currency
Availability	Title	Description
Binding	Type	DiscountPercent
Book	Vendorprice	Editor
LookyBook	infoVendorprice	Id
BookInfoResponseType	BookInfo	ImageUrl
Chapter	BookInfoVendorprice	Inventory
CustomerAccount	format	KeywordSearch
CustomerSubAccount	imgUrl	Marc
Default	name	Money
FormDefault	numpages	Name
DiscountPrice	page	Ontology
DoKeywordSearch	price	PagesNumber
DoKeywordSearchEx	pricePrefix	PricePrefix
Edition		Property
GetBookInforByISBN	siteUrl	PublicationDate
GetBookInforByISBNResponse	source	PublicationPlace
GetBookInforByISBNResult	timestamp	Restriction
GetInfo	toc	Seq
ISBN		ShortDescription
		SiteUrl
InfoService		Source
Information		Status
Keyword		String
LoginName		Author
Looky		DeliveryDate
Marc		edition
Password		FiveStarRating
LoginPassword		Format
Place		Isbn
PublicationPlace		Keyword
PublicationDate		Price
Publisher		PublicationData
Request		Publisher
BookRequest		Title
CategoryRequest		Subject
Response		TableOfContents
BookResponse		TimeStamp
CategoryResponse		Vendor
Total Concepts = 66		Total Concepts = 44

Figure 6-8: Sample of Lexical Layer Evaluation Model

Applying the LP and LR, as defined in formula (1) - (3), on the sets of Web Services chosen for the experiment, produced the results that are summarized in Table 6-2 (See appendix D for evaluation sheets). The results illustrate a lexical precision of 37% and lexical recall of 57% for the group 1 (Books) Web Services. The results are much improved for the group 2 (Financial) Web Services. As with prior iterations, it is possible that this is an indication that implementing more patterns would yield higher recall.

The low precision can be clearly justified by a number of reasons:

More concepts appear in the SOLFO than the GSO, highlighting the question of whether the GSO has all of the possible domain concepts, i.e. there could be several concepts that are correct but not counted as such, because they do not exist in the GSO. It can be clear that the SOLFO has brought new concepts from the input sources that were not present in the GSO. These new concepts could be important new additions that have been missed by the GSO. For example in the book services data set, *LoginName* and *Customeraccount* are concepts that exist only in SOLFO (See Figure 6-8 shown highlighted in blue text). Nevertheless they appear to be valid domain concepts. Therefore, another method of evaluation may be viable to judge the accuracy of those concepts, which might result in an increased precision value.

Some concepts that appear in SOLFO can lead to service functionality or functional hierarchy as addressed by other research (Sabou, 2005) such as *DoKeywordSearch* or *GetBookInfor*. This is clearly not the aim of the learning algorithm proposed here, which is to build a domain ontology rather than a service functionality ontology. Nonetheless this observation can lead to further research in that direction.

The researcher performed the pruning step superficially to the best of their domain knowledge, i.e. pruning only trivial non domain concepts or duplications which differ in spelling. A stricter pruning step can lead to higher precision. The results of how pruning can increase precision are clearly shown in other research, as in Sabou (2005).

Significantly, the aim of SOLF is to semi automate the ontology development process for Web Services, where it can be used as a plugin tool by ontology engineers or domain experts (Buitelaar, Cimiano & Magnini, 2007). This highlights the importance of a domain coverage evaluation and a domain expert evaluation in order to judge and validate the newly extracted concepts by SOLF. It is important, at this point, to remember that only partial rules were implemented for the purpose of implementing the SOLF tool. Therefore, it would be pertinent to implement and test other rules in the future, as this might lead to higher recall.

It is interesting to compare our results with other OL approaches. Where Rule-based OL normally leads to higher precision and lower recall as shown by Sabou (2005), here the SOLF has managed to achieve a higher recall. This leads to the important observation that the SIP extraction process yielded a higher pattern recall performance, which is of particular relevance for domain engineers when building ontologies for Web Services by using existing legacy systems and software artefacts (Buitelaar, Cimiano & Magnini, 2007), thereby proving the adequacy of SOLF to be embedded in an ontology engineering process.

Table 6-2: Summarised Precision and Recall for Group 1 and Group 2

	Group 1: Books Services	Group 2: Financial Services
GSO Total Concepts	44	171
SOLFO Total Concepts	66	247
Lexical Precision (LP)	38%	43%
Lexical Recall (LR)	57%	63%
F1 Measure	45.45%	51.20%

It is important, therefore, to explore the nature of the Gold Standard Ontology (GSO). The gold standard is developed to perform the task of service matching. Hence, here it is used as a benchmark mainly for calculating the recall, which is not often possible for domain experts to produce manually. Accordingly, applying the precision and recall metrics by using the GSO can only give an accurate evaluative insight in regard to the recall, whereas precision should be more accurate if produced by using the domain expert scoring. The higher recall validates the SOLF in a way that demonstrates its ability to automatically extract concepts.

6.3.6 Non Taxonomic Layer – Structural Evaluation

As indicated in Chapter 3, the non-taxonomic relations refer to semantic relations linking domain and range concepts, usually mapped in OWL as an object properties. (NonT) layer evaluation is not well defined in the literature. Although there exist some attempt to define those measures based on precision and recall as non-taxonomic precision

(NonTP) and non-taxonomic recall (NonTR) (Dellschaft & Staab, 2008; Buitelaar, Cimiano & Magnini, 2007), none of these actually illustrate how to calculate those measures. The majority of the evaluation attempts perform only lexical layer evaluation and omit the structural evaluation as in Sabou (2005). Therefore, this iteration contributes in this area. Design Research is employed to develop a practical detailed evaluation model that executes a gold standard based evaluation method, and shows the detailed steps of how the results are calculated.

NonTP and NonTR are generally defined, as the intersection of the non-taxonomic relations between the GSO and the SOLFO, as follows:

$$NonTP = \frac{|R_{SOLFO} \cap R_{GSO}|}{|R_{SOLFO}|} \quad (4)$$

where NonTP is the non-taxonomic precision and R is the set of non-taxonomic relations.

$$NonTR = \frac{|R_{SOLFO} \cap R_{GSO}|}{|R_{GSO}|} \quad (5)$$

where NonTR is the non-taxonomic recall.

Performing NonT relation evaluation is used to measure the structural aspects of the learned model. It is found to be a significant and complicated procedure that needs to be carefully designed and executed to ensure accurate evaluation. Since presenting information is fundamental, therefore, comparing relations between SOLFO and GSO was a difficult time consuming task and could not be easily performed without presenting the information in a comparative visual model. Applying a trial and error strategy resulted in developing an effective, easy to use evaluation model that allows visual identification of overlapping relations. The evaluation here is designed to satisfy Dellschaft & Staab's (2008) evaluation criteria, that is to ensure that the influence of one dimension of error doesn't exceed one measure; i.e, the influence of the lexical precision evaluation on taxonomic and non-taxonomic layers is minimized by combining different evaluation methods, that is the measures should be applied to minimize the dependency

between lexical layer evaluation measure and the non-taxonomic evaluation. The following criteria are applied when implementing the NonT evaluation model, as shown in Figure 6-9:

- An evaluation model is developed that ensures the adequacy and accuracy in calculating the global taxonomic precision. This is achieved by modelling the local NonT intersection between the learned ontology and the gold standard ontology using an evaluation model that allows visual interpretation of the local and taxonomic overlap. The sample model is presented in Figure 6-9. As the significance of this research lies in relation extraction, it is essential to thoroughly evaluate this aspect of the learning approach.
- The local NonTP values are calculated so that the influence of the lexical precision evaluation measure is minimized. Therefore the common set of concepts $C_{SOLFO} \cap C_{GSO}$ is preferred over the learned ontology set of concepts when determining the NonT relations overlap.
- The first column of the model represents the list of concepts in the learned SOLFO. The header row represents a list of range only concepts. Generally, each cell is divided into three sub-cells to represent the presence (indicated by 1) or absence of a relation between the intersecting concepts C_i and C_j , where C_i refers to the i^{th} concept in the domain set and C_j refers to the j^{th} concept in the range set. The first sub cell of an intersection represents whether a relation exists in the SOLFO, the second sub-cell represents whether the relation exist in the GSO, and the third sub-cell is used to calculate the intersection, i.e the local NonT overlap (highlighted in green in Figure 6-9). This model is used to calculate the global NonT overlap as the sum of all NonT overlaps.

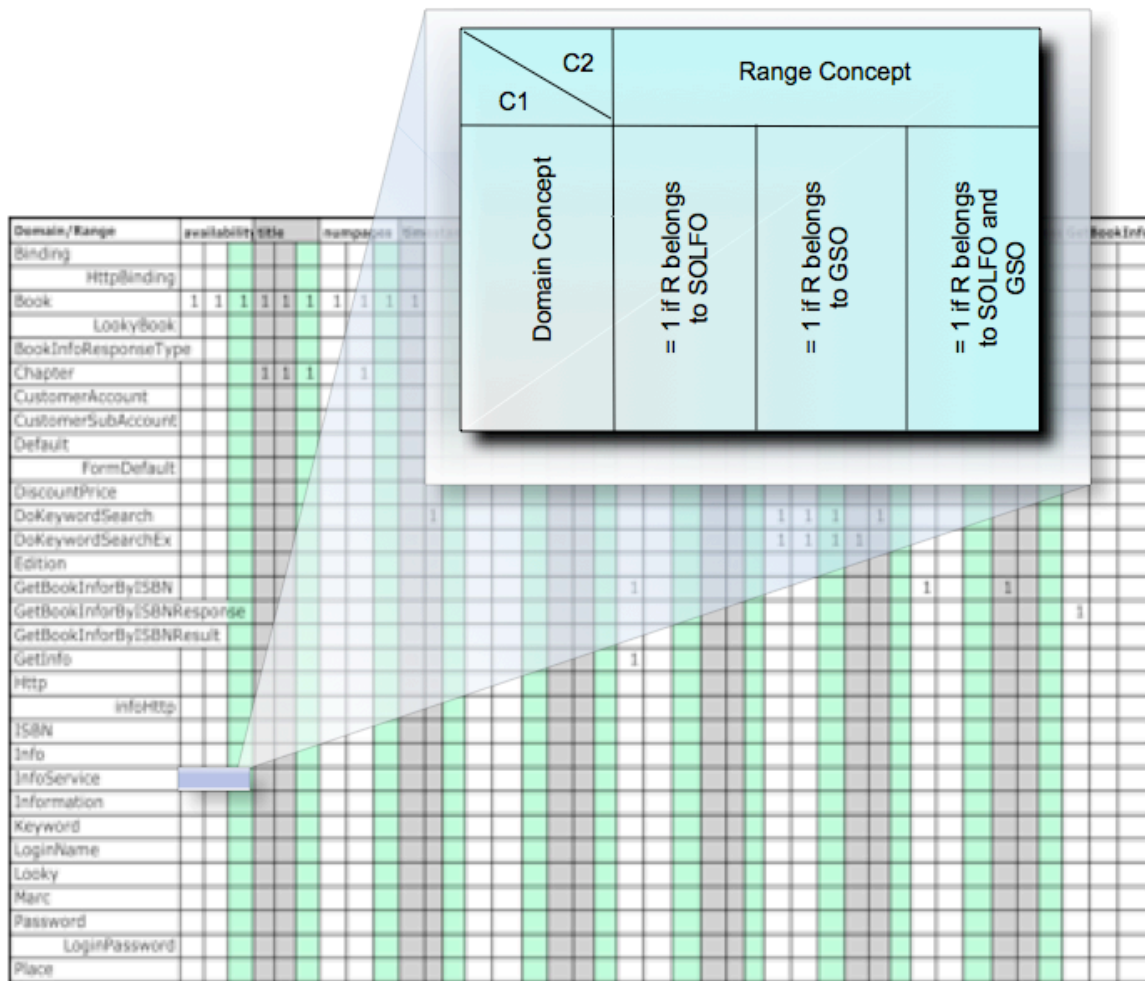


Figure 6-9: NonTP Evaluation Model

The model presents a measurable impact of people's understanding allowing for visual identification of where relations are condensed according to the GSO and SOLFO. Where, consecutive one's in a row shows that the relation exists in both models. Visual deduction of precision and recall is possible.

The result for the NonTP and NonTR are calculated for the two sets of data using the proposed evaluation model. The summarized results are presented in Table 6-3 illustrating similar precision of 49% and 50% for the two datasets respectively. A very high recall is also achieved for the two datasets of 95% and 100%, which clearly validates the completeness of the pattern extraction process in selecting higher frequency patterns. The results achieved by the SOLF are encouraging when compared to other work (40% relation precision achieved by (Ciminaio, 2007 p. 138). The fact that only some of the patterns are implemented could be an explanation for not achieving a higher

recall in the lexical layer evaluation. It is apparent from the recall evaluation results produced in Chapter 5 and the results achieved here, that implementing more patterns might increase the precision and recall dramatically.

It is important for this evaluation to consider the presence of other NonT relations, which are not part of the GSO. Although some of these relations can be counted as correct, they are not included here due to the fact that the evaluation method here is a gold standard based evaluation. This clearly indicates that if the learned ontology does not reflect the gold standard, it does not necessarily imply that the learned ontology is inaccurate, as also noted by Sabou (2005). On the other hand, these new concepts and relations might lead ontology engineers to identify important new relations to the domain. Therefore, further evaluation requiring the domain expert to validate these relations, by scoring each and every relation, is vital to calculate the precision of the ontology faithfully. The precision values indicate that the SOLF proved to have a reasonably accurate relation extraction rates as compared to other research, 20% and 40% for Sabou (2005) and Cimiano (2007) respectively, thereby demonstrating that SOLF can effectively assist domain engineers in the ontology development process.

Table 6-3: Summarized NonTP and NonTR Results

	Group 1: Books Web Services	Group 2: Financial Web Services
GSO Total Non Taxonomic Relations	20	89
SOLFO Total Non Taxonomic Relations	39	175
NonTP	49%	50%
NonTR	95%	100%
F1 Measure	64.5%	67.4%

6.3.7 Taxonomic Layer – Structural Evaluation

The taxonomic layer evaluation of the first group (Books WS) revealed no correlation between the two taxonomies. Interestingly from the produced SOLFO model that there exists a valid hierarchy in SOLFO that did not exist in the GSO. For example, subclasses of *String* concept in GSO, such as *Author* and *keyword*, does not really correlate to similar relation in SOLFO. Whereas clearly, there appears to be conflicts between what

is represented and the representation, due in large to the fact that the GSO was built to perform the task of service matching, whereas the SOLFO model is built to conceptualize the underlying domain more faithfully. Subsequently, performing a taxonomic evaluation for the financial Web Services appeared to be an excessive process. This indicates that there are obvious differences between the ontology models; consequently, a more accurate precision evaluation measure would be to perform a domain expert evaluation of the taxonomic layer to effectively determine the accuracy of the taxonomic relation extraction.

6.4 Domain Expert Evaluation and SOLF Refinement

The GSO contains less non-taxonomic relations and fewer concepts. The learned ontology contains more taxonomic and non-taxonomic relations. Therefore, calculating the precision using the domain expert in scoring the correct relations would result in higher precision, but recall would be impossible to calculate, since it would require the domain expert to analyse the input sources and manually extract all available concepts and relations. Analysis of the results revealed that there are new concepts in the learned ontology, SOLFO; interestingly the GSO missed these concepts. Sabou (2005) defines the new concepts as Ontological Improvements

On the other hand, evaluation measures should be chosen so that they are independent of each other (Dellschaft & Staab, 2008). Here, an expert evaluation is used to evaluate the structural layer of the learned ontology to produce the taxonomic and non-taxonomic precision. This involves assessing the usefulness of the extracted relations, by allowing a domain expert to carry a concept-by-concept analysis to judge the newly extracted concepts and relations. In this case the domain expert knowledge is used to score the new concepts as well as the relations, and judge whether the new concepts are either correct or spurious.

$$\text{Lexical Precision} = \frac{\text{Correct} + \text{NewConcepts}}{\text{AllConcepts}}$$

$$\text{Taxonomic Precision} = \frac{\text{Correct} \dots \text{Subclass} \dots \text{Relations}}{\text{All} \dots \text{Subclass} \dots \text{Relations}}$$

$$\text{Non Taxonomic Precision} = \frac{\text{Correct...nonTaxonomic...Relations}}{\text{All...nonTaxonomic...Relations}}$$

Table 6-4: Summarized Domain Expert Precision

	Group 1: Books Web Services	Group 2: Finance Web Services
SOLFO Total Taxonomic Relations	19	78
SOLFO Pruned Taxonomic Relations	10	50
Taxonomic precision	21%	42.31%
Taxonomic Precision (Pruned)	40%	66%

The results produced in this iteration clearly indicate that evaluation methods can be effectively combined to produce more accurate evaluation measures, where the domain expert evaluation can be applied effectively to determine the accuracy of the learned ontology. On the other hand the gold standard based evaluation can be used efficiently to evaluate the domain coverage of the learned ontology, i.e. the gold standard based evaluation can be used to calculate lexical and structural recall, whereas the domain expert evaluation method can be used to calculate lexical and structural precision more effectively. Here a pruning step is performed to remove any technical or WSDL related subclassing by performing a quick scan “and eliminate”, of redundant relations, in the financial domain case. Although, the pruning is performed superficially, it is clearly seen that the pruning step increased the precision in both data sets adding an extra 20% precision. Which can be considered relatively good compared to the small time and effort required to prune the relations.

6.5 Specifying the learning

The primarily points of learning are:

- Verb relations may lead to identifying the functional service hierarchy in OL from textual sources (Sabou et al., 2005); Verb terms in SIP e.g *CalculateInterestRate* or *GenerateInterestPayments*. A number of OL approaches adopt the hypothesis that ontological relations are mostly represented by verbs within an argument, for learning from textual sources (Völker, Haase & Hitzler, 2008; Sabou et al., 2005; Navigli & Velardi, 2008). A preliminary analysis to adopt the verb to relation hypothesis has lead to identifying functionality hierarchy rather than domain concepts hierarchy. Following this line and the fact that lower frequency SIP patterns consist of verb terms followed by nouns (the learning outcome of Iteration 2), this direction can be further investigated to learn domain specific relations through implementing the more complex SIP patterns.
- Different kinds of information appear in different parts of the Web Service. As the learning indicates from Iteration 1. For some cases, where the service contains accompanying XSD files, the XSD files might be potential venues for domain specific concepts. It is essential to include these as inputs for the system as analysing the SIP extracted from the XSD files is potentially an important extension to the pattern extraction steps in SOLF (Papazoglou & van den Heuvel, 2007; Sabou & Pan, 2007).
- Domain specific information is distributed in different parts of a Web Service according to the structure of the Web Service. Relating structure to SIP may lead to different ontological domain specific relations (Alfaries, Bell & Lycett, 2009; Yu et al., 2008; Bell et al., 2007; Sabou & Pan, 2007).
- It is clearly evident that there are duplications, which can be dealt with at different stages of the OL life cycle. A **first** option would be to introduce a pruning step at different stages of the ontology learning life cycle; i.e involving the user at an earlier stage to resolve name mismatches before the ontology was actually built can result in higher accuracy. The same argument applies if the concepts are pruned in a stricter manner, after they are extracted but before they are added to the ontology as new concepts. Then the mapping and ontology building is based on unified names for the concepts, thereby eliminating most of the duplicity created in the relations and concepts. This could potentially result in achieving a higher precision ontology, as illustrated in the sample taken from the Books ontology (Group 1) in Figure 6-10.

Here *book* as a concept and information related to book like *author*, *title* etc. is modelled more than once. The domain expert can easily eliminate this duplication, i.e. the learned ontology model may serve ontology engineers in a powerful way, linking concepts in different ways can bring different modelling possibilities to the domain engineer attention. A **second** option would be to include other NLP techniques such as a lemmatizer or stemmer step in order to eliminate the redundancy before concept creation. Lemmatizer and stemmers are normally used for the purpose of getting the basic form of the word. This step can be seen as a filtering step that can be definitely applied as a pre-processing step. A **third** refinement would be to apply techniques that are usually employed to perform deeper ontology merging to accurately check for the existence concepts before they are added into the ontology, since SOLF checks for the existence of concepts before adding them based on exact string matching. This behaviour is expected since the system automatically builds the ontology and the user is involved in the pruning stage. On the other hand, the domain expert identified that this can be an important advantage of the system, since it extracts all of the possible concepts, depending on the naming and spelling used in the Web Service. At the end of the process, however, it is clear that these concepts are the same and can therefore be merged into one. Hence, integrating the SOLF with ontology matching techniques before the new concepts are added to the ontology is a desirable improvement that should lead to eliminating the majority of the redundant concepts. This should also reduce the effort of the ontology engineer in the pruning step.

- Synonym learning is an OL task, as illustrated in the OL layer cake. Cimiano (2007 p.24) regards two concepts as synonyms if they share a common meaning. An interesting observation made by the domain expert is that multiple names used to represent the same concept can be the best fit for the synonym learning task. Synonyms can be easily modelled as equivalent classes in OWL. The pattern-based extraction process applied in SOLF, extracted concepts and possible synonyms that are made available for the domain experts for them to make into equivalent classes. The domain expert can easily identify the synonyms during the pruning step. Alternatively some concepts are found to be good candidates for identifying lexicons of a concept as identified by Cimiano (2007 p.22).

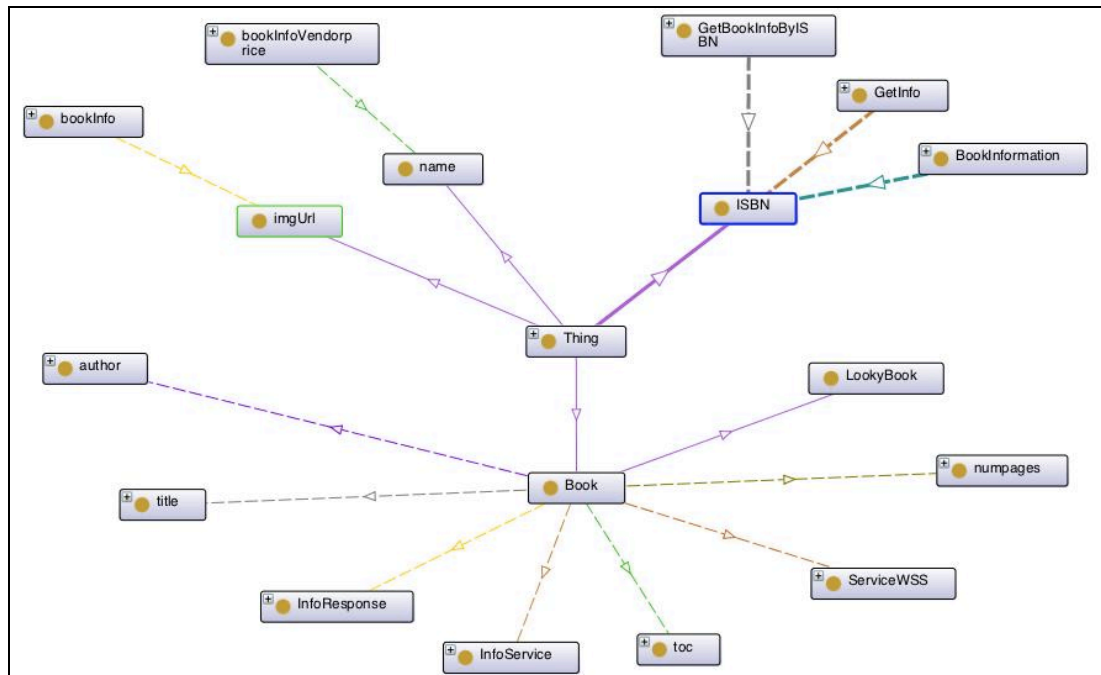


Figure 6-10: Sample Group 1 (Book) Ontology

- The evaluation of the learned ontology against the gold standard required the manual identification of correct and incorrect concepts. In the lexical layer case it was a simple task of concept-by-concept comparison, taking up to 2 working days for the books ontology, and another 5 working days for the financial services, since the ontology consists of 247 concepts. The amount of time and effort required to perform the non-taxonomic layer evaluation was extremely time-consuming, since there is no direct way of performing the comparison automatically. Although the developed evaluation model allowed for accuracy, and visualised the evaluation analysis by representing the compared relations in adjacent cells, it required long, condensed working hours to complete the evaluation for both sets, taking up to two whole weeks to produce the NonTP and NonTR final results.

6.6 Summary

The chapter validates the theory of this research, that SOLF is capable of automatically extracting domain knowledge, including concepts and semantic relations, from Web Service artefacts by applying pattern based IE techniques. This iteration contributes an improved service ontology learning framework and tool. A formal definition of the output of the phases consisting the framework is provided. Another main contribution of

this chapter is a thorough evaluation process to prove the SOLF, despite having to overcome the problem of the OL evaluation. This iteration combines two OL evaluation methods effectively. The evaluation method is illustrated through the application of a detailed experiment and has demonstrated that there is enough domain knowledge in Web Service artefacts, from which an initial ontology can effectively be learned. The approach adopted in the SOLF proved to be efficient in extracting domain concepts and linking them with relations based on pattern-based information extraction techniques, thus proving reasonable preciseness and coverage. Domain expert evaluation proved that the automatically learned ontology recommends a new set of additions, including taxonomic and non-taxonomic relations that can be used to supplement the manual ontology. Overall, the method proved efficiency by introducing new relations and concepts that had not been included in the GSO. Finally, the learning that emerged from this iteration highlights a number of issues and challenges that can be employed to direct future research.

CHAPTER 7 - CONCLUSION

7.1 Research Summary

Web Services typically contain domain knowledge that can be semantically annotated through the use of domain ontologies. These domain ontologies are considered to be the standard form of providing shared knowledge representation, providing a solution to more widespread of functional interoperability via SWS (Buitelaar, Cimiano & Magnini, 2007). Manual ontology development, however, is an expensive, time consuming and error prone process, requiring the services of highly qualified expertise, both in ontology engineering and in the domain of interest (Staab & Maedche, 2001; Ding & Foo, 2002). Therefore, the widespread adoption of ontology development can be very difficult to achieve in practice. Given the vital role that the Semantic Web can play in achieving the full potential of Web Services, a faster, less expensive ontology development process is clearly required (Medjahed, Bouguettaya & Elmagarmid, 2003; Davies, Studer & Warren, 2006).

To make Semantic Web Services a practical reality, ontologies need to evolve from sources with embedded business knowledge - Web Service artefacts. Consequently, this thesis has sought to assisting ontology engineers in building and maintaining low cost domain ontologies from Web Services. This aim was achieved by developing a service ontology learning framework to automatically extract ontological knowledge from existing legacy systems. The objectives as set out in chapter 1 are summarised below:

Objective 1: Review the available OL approaches to provide an understanding of the state-of-the-art of ontology learning and Web Services.

Objective 2: Develop ontology learning techniques for service concept and relation extraction and to automate these techniques by building a prototype application to test the applicability of the techniques using real Web Services.

Objective 3: Develop a methodological Service Ontology Learning Framework (SOLF) that incorporates the techniques for concept and relation extraction.

Objective 4: The implementation of a tool that facilitates the framework and evaluating the application of the framework, by assessing its impact on the state-of-the-art of ontology learning.

Objective 5: Validate the research outcome by testing the generality of the extracted patterns and rules on other sets of services representing varying domains.

In achieving the aim and objectives of the work, Chapter 2 reviewed the varieties of Web Service sources and the applicable techniques for each source by providing an understanding of the theory and practice of currently available OL techniques. In the context of this research, the literature provided the basis for proving how OL can assist in faster, less expensive ontology development processes (Buitelaar, Cimiano & Magnini, 2007; Zhou, 2007; Buitelaar & Cimiano, 2008). Although applying OL techniques is predominantly limited to learning from textual sources, the Web Service application domain contains a mixture of structured and unstructured sources, where the available sources are predominantly categorised as semi-structured. Current research is mainly focused on learning from textual sources; there has been much less work completed on developing techniques and tailoring ontology learning methods aimed at semi-structured sources (Buitelaar, Cimiano & Magnini, 2005; Zhou, 2007). Interestingly those semi-structured sources represent domain knowledge embedded in technical, rich sources of data (Sabou, 2005). Consequently, an opportunity for contribution lies in introducing automatic knowledge extraction techniques to extract domain specific concepts and semi-automating ontology development (Davies, Studer & Warren, 2006).

Chapter 3 set out the means for achieving the objectives via Design Research. This approach provides a means by which to engage in the design problem - providing the necessary learning to improve the proposed solution, whilst, at the same time enriching the solution space with the Design Research output. The main Design Research artefact is a service ontology learning methodological framework (SOLF). The overall research methodology is executed as Design Research incremental iterations, where each of the three iterations forms a design problem that executes the build and evaluates design activities (Vaishnavi & Kuechler, 2004). The iterations were designed such that; Iteration 1 develops the core framework including a service term extraction technique,

Iteration 2 extends the framework by adding a relation extraction method, and Iteration 3 validates and generalises the design artefact by applying the SOLF on other sets of carefully selected Web Services with an accompanying gold standard ontology. Given that the literature review demonstrated limited understanding and work in the problem space, Design Research is particularly appropriate, allowing an iterative learning process to feed ongoing understanding of the design problem. More specifically in the case of the OL field, evaluation is identified as an important stage at the end of each cycle. Practical evaluation methods are not yet well defined, thereby posing another learning challenge in the knowledge space.

The products of Design Research included constructs, methods and models in order to facilitate the framework development. The build and evaluate design activities are applied in incremental iterations to build and effectively evaluate each of the Design Research products as illustrated in Table 7-1. The evaluation for the Design Research products is achieved by synthesising the Design Research evaluation criteria, as the table illustrates, to create the suitable evaluation method derived from the OL knowledge base as presented in Chapter 3. The evaluation demonstrates the successful application of each product in the final SOLF method and tool.

Table 7-1: Design Research Products X Activities

		Research Activities			
		Build	Evaluate	Theorize	Justify
Research Outputs	Constructs	STE SOLF SIP TR	Completeness Simplicity Ease of Use	Explain why and how constructs work by employing them to describe real case scenarios (addressed in Ch5)	Prove that constructs work scientifically by applying them in models and methods (addressed in Ch4, 5 & 6)
	Model	STE SOLF SIP TR SOLF Domain Ontology	Fidelity Completeness Internal Consistency	Adapting theories from the current OL discipline, and Hypothesising that those models are true (achieved by theorising SOLF in Ch 6)	Test the models on a real life example to prove them (addressed in Ch4, 5 & 6)
	Methods	STE Process SIP Process TR Development Process SOLF Framework	Operationality Efficiency Generality Ease of Use	Explain why and how methods are applied using real WSS (achieved in Ch5 &6)	Prove the methods work formally by instantiating them using real examples (achieved Ch 6)
	Instantiation	SOLF Application	Effectiveness Efficiency Impact on Environment	Understanding how and why application works across other domains (achieved in Ch6)	Prove that SOLF works by testing it across different domains. (achieved in Ch6)

Chapter 4 described the first iteration, which concentrated on developing a service term extraction technique based on NLP methods. The STE technique was used to build the core SOLF, by automatically extracting an initial ontology model consisting of automatically extracted domain concepts. An initial set of constructs, models and a method was built and evaluated, meeting Objectives 2, 4 and part of Objectives 1 and 3. The service term extraction technique formed the pre-processing stage of the learning framework. The first stage laid out the foundation of the ontology learning framework

by accomplishing the first ontology learning task. The rule-based IE technique applied, started by applying syntactic analysis as a pre-processing stage to identify patterns and perform concepts extraction based on the identified pattern. The successful automation of the method was achieved through building a prototype application in GATE that implemented the steps identified in the framework. As a result of processing WSDL and XSD files, a list of concepts were automatically identified within these input files, contributing another Design Research product in the form of an initial financial domain ontology model representing the sample set of services.

This early form of SOLF and tool were evaluated by comparing the output to other term extraction methods, where the learning outcome of the first iteration directed the next iteration towards adding structure to form another dimension of the domain model. This observation highlighted the need to further investigate how to extract relations between these concepts and initiated another Design Research iteration that is to allow for the automatic extraction of ontological relations between the identified concepts.

Chapter 5 extends SOLF with a pattern-based relation extraction technique. This second iteration contributes another set of Design Research products facilitating the extraction of relations based on identifying Structured Interpretation Patterns (SIP). The structural aspect of domain ontology was learned through applying a rule based IE approach, where identifying patterns is an important step that requires rigour and use of the framework to ensure accuracy, generality and coverage of SIP. Transformation rules were used to identify mappings between SIP patterns and OWL constructs. Three real-world Web Services from global banking systems were used for pattern extraction and transformation rule development to demonstrate the completeness, efficiency and effectiveness of SOLF.

An instantiation of SOLF as a prototype tool was developed and used to prove and evaluate the framework. The output of the SOLF process was an automatically generated OWL domain ontology, which presented a number of financial domain concepts extracted from the Web Services. It was clearly visible that the automatically built domain model moved beyond basic taxonomy – extracting and relating concepts at a number of levels. More importantly, the approach provided integrated knowledge (represented by the individual WSDL documents) from a number of services across a

group of banks. It was clear at end of the second iteration (Chapter 5) that in order to justify and theorize the SOLF a further iteration was required to take the research to the next level, by proving that SOLF is practically applicable across other domains.

Chapter 6 addressed all of the research objectives, showing that the SOLF is capable of automatically extracting domain knowledge from WS artefacts. The extraction included concepts and semantic relations from Web Service artefacts garnered by applying pattern-based IE techniques. The SOLF has demonstrated that there is enough domain knowledge in Web Service artefacts from which an initial ontology can effectively be learned. The approach adopted in the SOLF proved the efficiency in extracting domain concepts and linking them with relations based on pattern-based information extraction techniques. The automatically learned ontology recommended new sets of additions, including new domain concepts and relations, which could be used to enhance and update the manual ontology. Overall, the method proved efficiency by introducing new relations and concepts that were not included in the GSO.

This last iteration used the learning produced by evaluate, theorize and justify activities from 2, to suggest improvements for the models (SIP and TR) and the SOLF method. This led to producing the final products of the research, consisting of a Web Service ontology learning methodological framework (SOLF), including a formal definition of the output of the phases that constitute the framework, a set of SIP patterns and a set of TRs. Applying the SOLF on the two groups of the selected Web Services resulted in another set of Design Research products (ontology models).

Besides overcoming the challenge of the OL evaluation, this iteration combines two OL evaluation methods effectively. The evaluation method is illustrated by its application in a detailed experiment. The gold standard based evaluation method is complemented with a domain expert evaluation to judge the taxonomic layer. The integrated evaluation proved that the automatically learned ontology recommends a new set of additions, including taxonomic and non-taxonomic relations that can be used to supplement the manual ontology.

A deeper understanding of how and why the SOLF works was achieved in the last iteration, by performing a thorough evaluation that enabled knowledge and learning to

emerge whilst the SOLF was applied and allowed to be refined iteratively. Finally, the learning that emerged from the third iteration highlighted a number of issues and challenges that could be employed to direct future research.

7.2 Contributions and Conclusions

Research contributions are categorized according to Design Research product classification (March & Smith, 1995). In overall terms, the major contribution is a novel OL approach that applies textual IE techniques to automatically extract knowledge from semi-structured Web Service sources, mainly WSDL and XSD files. Within the literature, a number of proposed classifiers apply rule-based algorithms that identify different types of taxonomic and non-taxonomic relations. Recent relation learning approaches that showed success can be found in Cimiano (2007); Buitelaar, Cimiano & Magnini (2007); Sabou & Pan (2007); Buitelaar & Cimiano (2008). All of these approaches, however, are aimed at learning from textual sources. WSDL and XSD are semi-structured data sources, thereby posing an even greater challenge for domain experts to be able to read and manually extract knowledge from such sources.

More specifically, the main research contributions and their value are detailed below:

- **The SOLF methodological framework (method)** is the main contribution made by this research and can be applied in different scenarios in an ontology development lifecycle. Typically, in other OL approaches, pattern-based OL is applied as a first step in a more integrated ontology development process. Therefore, this approach has the potential to be integrated as a first step of a more complex ontology engineering process. The SOLF can be used to automatically extract semantic information from Web Service artefacts and is capable of building a domain ontology model representing the knowledge embedded in semi-structured Web Service sources. The SOLF targets different ontology learning tasks; (1) Domain Concept Extraction, (2) Concept taxonomy and (3) Non-taxonomic relations.
- **The SIP extraction process (method)** is a novel generic method that enables pattern extraction from Web Services artefacts. This method contributes a generic structured interpretation pattern extraction process that can be effectively

applied in a rule-based IE algorithm to identify and extract semantic relations from semi-structured software artefacts. The literature typically applies a heuristic pattern extraction strategy as per Cimiano (2007) and Sabou (2005), which normally apply generic patterns that result in lower recall. The method contributed by this research is a systematic, frequency based, pattern extraction process. The process is aimed at extracting high frequency patterns from the corpus, thereby guaranteeing higher recall.

- **The TR development process (method)** is an effective method that can be easily applied to identify semantic relations in SIP patterns. The process was aimed at developing a set of transformation rules that can be easily applied in a rule-based ontology building algorithm to automatically map SIP patterns to semantic relations. Transformation rule development is a novel method specifically tailored to map compound words in Web Services sources to a suitable OWL relation. The efficiency of this method was demonstrated by the non-taxonomic F1-Measure value of 67% achieved in Iteration 3, which is considered promising compared to the similar measure of 33% obtained by Cimiano (2007, p.114).
- **The SOLF tool (Instantiation)** is an application prototype that implements the SOLF, the set of SIP patterns and the transformation rules (TRs). The tool can be generally applied to efficiently extract domain specific concepts and relations from Web Service artefacts successfully producing an initial domain ontology model. The learned model can be easily pruned and modified by domain engineers. The generality and effectiveness of the SOLF tool in extracting non-taxonomic relations, is clearly demonstrated by achieving similar evaluation results for both data sets, achieving an F1-Measure of 64% and 67% for the Books and Financial Web Services respectively.
- **More general learning over the course of the research:** First, for the rigorous evaluation of the SOLF, a practical evaluation framework is contributed in Chapter 6 to prove the validity and generality of the SOLF across other domains. The evaluation constitutes a detailed step-by-step evaluation method that integrates gold standard based and domain expert evaluation as illustrated in

chapter 6. The evaluation framework is designed to effectively provide an understanding of why and how the OL method works and to prove SOLF utility in OL for building domain specific Web Service ontology. The non-taxonomic evaluation framework applied contributes a rigorous visual structural evaluation model.

Second, an evaluation taxonomy and model; the need for an effective evaluation model surfaced from the evaluation taxonomy, and its background illustrates the typically applied evaluation metrics for OL approaches. Accordingly, an evaluation framework based on precision and recall is selected in order to evaluate the research products, providing another contribution as detailed and theorized in Chapter 6. The comprehensive evaluation method is designed to ensure efficient and effective evaluation of the structural and lexical aspects of an OL approach. The model details a process for calculating local and global non-taxonomic precision and recall as defined in Dellschaft & Staab (2008).

Third, the STE method is a service term extraction method that can be applied to extract candidate domain concepts representing the underlying domain. The method showed improved performance compared to other approaches, when extracting domain concepts from Web Service artefacts (WSDL and XSD files). The method provided better domain coverage by producing a rich list of terms that are more likely to serve as domain concepts as representing semi-structured data sources. The extracted list of terms presented to the ontology engineer forms a high-density list of domain specific concepts that would be harder to extract from textual sources. The method proved efficient in concept term extraction by achieving 67% precision as demonstrated by the evaluation in Iteration 1.

Fourth, A set of SIP patterns and TRs models are contributed which can be expanded to form a library of SIP patterns. Once a set of patterns and TRs are available the tool can be applied to any set of WS to learn a first cut domain ontology model easily, allowing ontology engineers to adopt and amend patterns according to domain needs. The effectiveness of these models and TRs are illustrated by the similarity of precision and recall results achieved when applied to two different sets of services each representing different domains. On the other

hand, the SOLF learned models certainly represent the domain more faithfully by introducing new additions to the GSO. This is demonstrated by the domain expert evaluation results of the taxonomic layer evaluation of the two groups of services, where a precision of up to 66% is achieved.

7.3 Limitations and Areas for Future Research

Though the research has made a number of valuable contributions to the ontology learning domain both in the process and the tools, a number of limitations and challenges may be noted:

- The SOLF can be considered an initial machine learning algorithm, in which manual pattern extraction is the main extraction technique for automatically learning ontological relations. Supervised machine learning algorithms (Buitelaar & Cimiano, 2008) require manually trained data to initiate the automatic learning process -which can be considered a drawback in supervised learning approaches. The approach presented by the SOLF would benefit greatly from applying machine learning algorithms to learn these patterns. From one perspective, machine learning can be used to learn new SIP patterns, where the contributed patterns in this research can serve as the training data for the algorithm. From a second perspective the output ontology model produced here can in itself be used as training data and allow the ML algorithm to learn new ontology models when applied to new set of services.
- Chapter 4 noted that the concept learning task as defined by Cimiano (2007) and Buitelaar, Cimiano & Magnini (2005) consists of finding concept extensions (a set of concept instances), intensions and lexical realization in the corpus. SOLF has successfully extracted lexical realisations of concepts from the WSDL files, such as *Book* and *Author*. Identifying certain instances of book or author leads to identifying concept extensions referred to in the literature as ontology population. SOAP messages, as discussed in Chapter 2, contain information about service invocation. Where instances of *Books* and *Authors* can be found. This area is not explored in this research and can be further investigated.
- Chapter 5 noted several limitations. First, it is observed that patterns that appeared with high frequency in large WSDL files (i.e. those that did not have an

accompanying XSD) did not appear at all in other Web Services, which raises an important question about the effect of the type and size of the WSDL on the pattern extraction process (weighting of patterns), and more specifically on the correlation between frequency/popularity and precision. This can be addressed by applying the extracted patterns on another set of Web Services (including different types of WSDL and including XSD files). Investigating the effect of the WSDL file size and style on the pattern extraction process is therefore an important area to investigate. Second, the existence of one pattern as part of another more complex pattern, i.e. *NNP-NNP* is part of *NNP-NNP-NNP*, might lead to having to make a choice as to which one is more appropriate. Third, more complex patterns can be included. In this iteration only up to 3-term patterns were extracted. It is established that in more complex services patterns of up to 9 terms exist. Complex patterns have less frequency and might therefore reveal more important/specific relations. Including more complex patterns and analysing how this benefits the recall is highly recommended. Fourth, the possibility of generalizing existing patterns needs further investigation, i.e. *NN*, *NNP* and *NNS* are all different types of nouns. Is it possible to include these under the one category of type NOUN? - i.e. will the same patterns give the same results?. Finally, identifying more domain specific relations needs further analyses and investigation. Relations identified are: subclass and *has-A* relations. The possibility of defining patterns that will lead to more specific relations is recommended.

- Chapter 6 also noted several areas in which the approach may be improved. First, verb relations may lead to identifying the functional service hierarchy in OL from textual sources (Sabou et al., 2005); Verb terms in SIP e.g CalculateInterestRate or GenerateInterestPayments. Those structures were not exposed by this research due to the fact that the extraction process was based on frequency analysis. Therefore, higher pattern frequency is used as the selection criteria. But it was clearly evident that there are fewer pattern starts with verbs tokens, those patterns can be investigated and analysed in more detail. Second, Different kinds of information appear in different parts of the Web Service. As the learning indicates from Iteration 1. Domain specific information is distributed in different parts of a Web Service according to the structure of the Web Service. Relating structure to SIP may lead to different ontological domain specific relations (Alfaries, Bell & Lycett, 2009; Yu et

al., 2008; Bell et al., 2007; Sabou & Pan, 2007). Third, the learned ontology model showed a number of duplicate concepts appearing to be representing the same concepts but differ in the names, although this seems to be of advantage to the domain engineer, highlighting different names or illustrating different structural possibilities. These duplications need to be dealt with at different stages of the OL life cycle. Investigating applying lemmatizers or ontology matching techniques would be beneficial. On the other hand investigating how Synonym learning task might benefit from these duplications would be advantageous.

- Unexpectedly, the evaluation of the learned ontology against the gold standard was a time consuming task. Which required the manual identification of correct and incorrect concepts. In the lexical layer case it was a simple task of concept-by-concept comparison, taking up to 2 working days for the books ontology, and another 5 working days for the financial services, since the ontology consists of 247 concepts. The amount of time and effort required to perform the non-taxonomic layer evaluation was time-consuming, since there is no direct way of performing the comparison automatically. An automated evaluation tool that can be used to compare two ontology models at the different evaluation levels would be beneficial.

BIBLIOGRAPHY

- Ahmad, K., Tariq, M., Vrusias, B., & Handy, C. 2003. Corpus-based thesaurus construction for image retrieval in specialist domains. In Proceedings of the 25th European Conference on Advances in Information Retrieval (ECIR), Pisa, Italy, April 14-16pp. 502–510.
- Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.T., Sheth, A. & Verma, K. 2005, "Web Service Semantics-WSDL-S", W3C Member Submission, [Online]. Available at: <http://www.w3.org/Submission/WSDL-S/> (accessed on 24/8/2009).
- Al Asswad, M.M., de Cesare, S. & 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.
- Alfaries, A., Bell, D. & Lycett, M. 2009, "Ontology Learning for Semantic Web Services", Proceedings of the 14th Annual UK Association of Information Systems Conference (UKAIS), Oxford University, Oxford, U.K, 31st March - 01st April, pp. 27-36.
- An, Y.J., Geller, J., Wu, Y.T. & Chun, S. 2007, "Automatic Generation of Ontology from the Deep Web", 18th International Conference on Database and Expert Systems Applications, Regensburg, 3-7 Sept. 2007 pp. 470-474.
- Antonacopoulos, A. & Hu, J. 2004, "Web Document Analysis: Challenges and Opportunities" Google Book Search [Homepage of World Scientific Press], [Online]. Available: <http://books.google.com/books?id=ubs2mwNIHnEC&printsec=frontcover&sig=ACfU3U1r5r0cV5dWuAhUUvxYFJ0LNpZHXQ> (accessed on 9/4/2008).
- Antoniou, G. & van Harmelen, F. 2009, "Web Ontology Language: OWL", in S. Staab and R. Studer (eds.), *Handbook on Ontologies*, International Handbooks on Information Systems, Berlin/Heidelberg: Springer-Verlag, pp.91-110.
- Aswani, N., Tablan, V., Bontcheva, K. & Cunningham, H. 2005, "Indexing and Querying Linguistic Metadata and Document Content", Fifth International Conference on Recent Advances in Natural Language Processing Borovets, Bulgaria.
- Azoff, M. 2007, Application Development End-User Survey, Butler Group.
- Bell, D., de Cesare, S., Iacovelli, N., Lycett, M. & Merico, A. 2007, "A framework for deriving Semantic Web Services", *Information Systems Frontiers*, vol. 9, no. 1, pp. 69-84.
- Berland, M., & Charniak, E. (1999). Finding parts in very large corpora. In Proceedings of the 37th Annual Meeting of the Association For Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, pp. 57-64

- Berners-Lee, T., Hendler, J. & Lassila, O. 2001, "The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", *Scientific American*, vol. 284, no. 5, pp. 34-43.
- Bernstein, D.S. 1999, "On bridging the theory/practice gap", *IEEE Control Systems Magazine*, vol. 19, no. 6, pp. 64-70.
- Blake, S.P. 1978, *Managing for responsive research and development*, W.N. Freeman & Co. San Francisco.
- Bontcheva, K. & Sabou, M. 2006, Learning Ontologies from Software Artifacts: Exploring and Combining Multiple Sources, EU-IST Strategic Targeted Research Project (STREP) IST-2004-026460 TAO, Sheffield, UK.
- Borislav, P., Atanas, K., Damyan, O., Dimitar, M. & Angel, K. 2004, "KIM – a semantic platform for information extraction and retrieval", *Natural Language Engineering*, vol. 10, no. 3-4, pp. 375-392.
- Brown, A.L. 1992, "Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings", *Journal of the learning sciences*, vol. 2, no. 2, pp. 141-178.
- Bruijn, J.d., Kerrigan, M., Zaremba, M. & Fensel, D. 2009, "Semantic Web Services", in S. Staab and R. Studer (eds.), *Handbook on Ontologies*, International Handbooks on Information Systems, Berlin/Heidelberg, Springer-Verlag, pp. 617-636.
- Buitelaar, P. & Cimiano, P. (eds) 2008, *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Amsterdam, The Netherlands, IOS Press.
- Buitelaar, P., Cimiano, P. & Magnini, B. 2005, "Ontology Learning from Text: An Overview" in *Ontology Learning From Text: Methods, Evaluation and Applications*, B.P. Buitelaar, P. Cimiano & B. Magnini (eds.), Amsterdam, Netherlands: IOS Press, pp. 3.
- Buitelaar, P., Cimiano, P. & Magnini, B. (eds) 2007, *Ontology Learning From Text: Methods, Evaluation and Applications*, 2nd edn., Netherland: IOS Press
- Buitelaar, P., Olejnik, D. & Sintek, M. 2004, "A Protege Plug-In for Ontology Extraction from Text Based on Linguistic Analysis", *Proceedings of the 1st European Semantic Web Symposium (ESWS)*, Heraklion, Greece, 10-12 May 2004, PP. 31-44.
- Burstein, M., Bussler, C., Zaremba, M., Finin, T., Huhns, M.N., Paolucci, M., Sheth, A.P. & Williams, S. 2005, "A Semantic Web Services Architecture", *IEEE Internet Computing*, vol. 9, no. 5, pp. 72-81.
- Cabral, L., Domingue, J., Motta, E., Payne, T. & Hakimpour, F. 2004, "Approaches to Semantic Web Services: an Overview and Comparisons", *1st European Semantic Web Symposium*, Heraklion, Greece, May10-12, pp. 225-239.
- Cerbah, F. 2008, "Learning highly structured semantic repositories from relational databases: the RDBToOnto tool", *ESWC'08: Proceedings of the 5th European*

- semantic web conference on the semantic web, Tenerife, Canary Islands, Spain, June 1-5, pp. 777-781.
- Checkland, P.B. 1981. *Systems Thinking, Systems Practice*. Chichester, UK. John Wiley & Sons. 330 pp.
- Cimiano, P., Maedche, A., Staab, S. & Volker, J. 2009, "Ontology Learning", S. Staab and R. Studer (eds.) *Handbook on Ontologies*, International Handbooks on Information Systems, Berlin/Heidelberg, Springer-Verlag, pp. 245-267.
- Cimiano, P. 2007, *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. New York: Springer.
- Cimiano, P., Pivk, A., Schmidt-Thieme, L. & Staab, S. 2005, "Learning taxonomic relations from heterogeneous sources of evidence", in P. Buitelaar, P. Cimiano & B. Magnini (eds.), *Ontology Learning from Text: Methods, evaluation and applications*, Frontiers in Artificial Intelligence, IOS Press vol. 123, July, 2005, pp. 59–73.
- Cuel, R., Delteil, A., Louis, V. & Rizzi, C. "The Technology Roadmap of the Semantic Web", Knowledge Web, [Online], Available: <http://knowledgeweb.semanticweb.org>. (Accessed on November 2008).
- 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.
- 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.
- Daga, A., de Cesare, S.d., Lycett, M. & Partridge, C. 2005, "An Ontological Approach for Recovering Legacy Business Content", *IEEE Computer Society*, Washington, DC, USA, pp. 224.
- Davies, J., Studer, R. & Warren, P. 2006, *Semantic web Technologies Trends and Research in Ontology-based Systems*, John Wiley & Sons, Ltd.
- Dellschaft, K. & Staab, S. 2008, "Strategies for the Evaluation of Ontology Learning", in P. Buitelaar & P. Cimiano (eds.) *Bridging the Gap between Text and Knowledge Selected Contributions to Ontology Learning and Population from Text* Amsterdam: IOS Press.
- Ding, Y. & Foo, S. 2002, "Ontology research and development. Part I - a review of ontology generation", *Journal of Information Science*, vol. 28, no. 2, pp. 123-136.
- Edelson, D. 2002, "Commentary: Design Research: What We Learn When We Engage in Design", *Journal of the Learning Sciences*, vol. 11, no. 1, pp. 105.

- Faure, D. & Nédellec, C. 1998, "ASIUM: Learning sub categorization frames and restrictions of selection", *Proceedings of 10th Conference on Machine Learning (ECML 98): Workshop on Text Mining*, Germany, Chemnitz, pp. 410-417.
- Farrell, J., & Lausen, H. 2007, "Semantic annotations for WSDL and XML schema". W3C Recommendation 28 August 2007. [Online] Available: <http://www.w3.org/TR/sawSDL/> (Accessed on September 2008).
- Fensel, D. & Bussler, C. 2002, "The Web Service modeling framework WSMF", *Electronic Commerce Research and Applications*, vol. 1, no. 2, pp. 113-137.
- Gacitua, R. & Sawyer, P. 2008, "Ensemble Methods for Ontology Learning - An Empirical Experiment to Evaluate Combinations of Concept Acquisition Techniques", *Seventh IEEE/ACIS International Conference on Computer and Information Science*, Portland, OR, 14-16 May 2008, pp. 328-333.
- Gacitua, R., Sawyer, P. & Rayson, P. 2008, "A flexible framework to experiment with ontology learning techniques", *Knowledge-Based Systems*, vol. 21, no. 3, pp. 192-199.
- Gasevic, D., Kaviani, N. & Milanovic, M. 2009, "Ontologies and Software Engineering", S. Staab and R. Studer (eds.), *Handbook on Ontologies*, International Handbooks on Information Systems, Berlin/Heidelberg: Springer-Verlag, pp. 593-615.
- Gedda, R. 02/10/2007 16:39:35-last update, SOA Uptake Still Split A Mid Market Confusion [Homepage of Computerworld], [Online]. Available: <http://www.computerworld.com.au/index.php/id;1744558846> (19/11/2008).
- Gibbins, N., Harris, S. & Shadbolt, N. 2004, "Agent-based Semantic Web Services", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 2, pp. 141-154.
- Giovannetti, E., Marchi, S. & Montemagni, S. 2008, "Combining Statistical Techniques and Lexico-Syntactic Patterns for Semantic Relations Extraction from Text", *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP) Rome, Italy, 15-17 December*. pp. 10.
- Gomez-Perez, A. & Manzano, M., D. 2004, "An overview of methods and tools for ontology learning from texts", *Knowledge Engineering Review*, vol. 19, no. 3, pp. 187-212.
- Gomez-Perez, A., Fernandez-Lopez, M. & Corcho, O. 2003, "Ontological Engineering." *Advanced Information and Knowledge Processing*. Berlin/Heidelberg: Springer.
- Gruber, T.R. 1993, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220.
- Guarino, N. 1998, "Formal ontology in information systems", In N. Guarino, (ed.), *Proceedings of FOIS98*. FOIS, IOS Press, pp. 3-15.

- Guarino, N., Oberle, D. & Staab, S. 2009, "What Is an Ontology?". in S. Staab and R. Studer (eds.), *Handbook on Ontologies*, International Handbooks on Information Systems, Berlin/Heidelberg : Springer-Verlag, pp.1-17.
- Guo, H., Ivan, A., Akkiraju, R. & Goodwin, R. 2007, "Learning ontologies to improve the quality of automatic Web Service matching", IEEE International Conference on Web Services, Salt Lake City, UT, 9-13 July, pp. 118-125.
- Hearst, M.A. 1992, "Automatic acquisition of hyponyms from large text corpora", Proceedings of the 14th conference on Computational linguistics, Association for Computational Linguistics Morris town, NJ, USA, Vol. 2, pp. 539-545.
- Heffner, R. & Peters, A. 2008, Topic Overview: Service-Oriented Architecture For CIOs, Forrester.
- Hevner, A.R., March, S.T., Park, J. & Ram, S. 2004, "Design science in information systems research", MIS Quarterly: *Management Information Systems*, vol. 28, no. 1, pp. 75-105.
- Hristoskova, A., Volckaert, B., Turck, F.D. and Dhoedt, B. 2010, "Design of a Framework for Automated Service Mashup Creation and Execution Based on Semantic Reasoning", International Conference on Internet and Web Applications and Services, Barcelona, 9-15 May, pp. 149-154.
- Iwanska, L., Mata, N. & Kruger, K. 2000, "Fully Automatic Acquisition of Taxonomic Knowledge from Large Corpora of Texts: Limited-Syntax Knowledge Representation System based on Natural Language", In L.M. Iwanska and S.C. Shapiro (ed.), *Natural Language Processing and Knowledge Processing*, MIT/AAAI Press, pp. 335.
- Johannesson, P. 1994, "A method for transforming relational schemas into conceptual schemas", *Proceedings 10th International Conference Data Engineering*, Piscataway, N.J: IEEE Press, pp. 190-201.
- Jung, S., Kang, M. & Kwon, H. 2007, "Constructing Domain Ontology Using Structural and Semantic Characteristics of Web-Table Head", *Lecture Notes In Computer Science*, vol. 4570, pp. 665-674.
- Kashyap, V. 1999, "Design and creation of ontologies for environmental information retrieval", Proceedings of the 12th workshop on knowledge acquisition, modelling and management, KAW'99, Banff, Canada. October 1999.
- Kelly, A.E. & Lesh, R.A. 2000, *Handbook of research design in mathematics and science education*, Mahwah/US, Lawrence Erlbaum Associates Inc.
- Lara, R., Roman, D., Polleres, A. & Fensel, D. 2004, "A conceptual comparison of WSMO and OWL-S", in Zhang, L.-J.; Jeckle, M. Hrsg (eds.), *Web Services: Lecture Notes in Computer Science*, Berlin, Heidelberg: Springer, vol. 3250, pp. 254-269.
- Li, M., Du, X.Y. & Wang, S. 2005, "Learning Ontology from Relational Database", In: The 4th International Conference on Machine Learning and Cybernetics, *IEEE explorer*, Guangzhou, China, pp. 3410-3415.

- Maedche, A. & Staab, S. 2001, "Ontology learning for the semantic web", *IEEE Intelligent Systems and Their Applications*, vol. 16, no. 2, pp. 72-79.
- Maedche, A. & Volz, R. 2001, "The ontology extraction and maintenance framework text-to-onto", Proceedings of the ICDM'01 Workshop on Integrating Data Mining and Knowledge Management, IEEE International Conference on Data Mining, California.
- Maedche, A. & Staab, S. 2004, "Ontology learning" in S Stabb and R Studer (eds.) *HandBook on Ontologies*, International Handbooks on Information Systems Series. Berlin: Springer. pp. 173-190.
- Maedche, A. 2002, *Ontology learning for the Semantic Web*, Boston, Kluwer Academic Publishers.
- Maedche, A. & Staab, S. 2003, "Services on the Move: Towards P2P-Enabled Semantic Web Services" in: Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January, pp. 124-133.
- Manine, A., Alponse, E. and Bessières, P. (2008), Information Extraction as an Ontology Population Task and Its Application to Genic Interactions, 20th IEEE International Conference on Tools with Artificial Intelligence, vol 2, pp 74-81
- March, S. & Smith, G. 1995, "Design and natural science research on information technology", *Decision Support Systems*, vol. 15, no. 4, pp. 251-266.
- Martin, D. et al. OWL-S: Semantic markup for Web Services. W3C Member Submission 22 November 2004. [On line] Available at: <http://www.w3.org/Submission/OWL-S/> (Accessed: August 2009).
- Martin, D. 2007a, "Semantic Web Services, Part 1", *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 12-17.
- Martin, D. 2007b, "Semantic Web Services, Part 2", *IEEE Intelligent Systems*, vol. 22, no. 6, pp. 8-15.
- Maynard, D., Li, Y. & Peters, W. 2008, "NLP Techniques for Term Extraction and Ontology Population", in P. Buitelaar & P. Cimiano (edt.) *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Amsterdam, The Netherlands: IOS Press, pp. 107-127.
- McIlraith, S.A., Son, T.C. & Zeng, H.L. 2001, "Semantic Web Services", *IEEE Intelligent Systems & Their Applications*, vol. 16, no. 2, pp. 46-53.
- Medjahed, B., Bouguettaya, A. & Elmagarmid, A. 2003, "Composing Web Services on the Semantic Web", *Vldb Journal*, vol. 12, no. 4, pp. 333-351.
- Meyer, M. 2006, *The adoption of SOA among US and Western European enterprises (Customer Focus)*, Butler Group.

- Motta, E., Domingue, J., Cabral, L., Gaspari, M., 2003, "IRS-II: A Framework and Infrastructure for Semantic Web Services", *The Semantic Web - ISWC 2003: Lecture Notes in Computer Science*, Berlin / Heidelberg: Springer, Vol. 2870, pp. 306-318
- Navigli, R. & Velardi, P. 2004, "Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites", *Computational Linguistics*, vol. 30, no. 2, pp. 151-179.
- Navigli, R. & Velardi, P. 2008, "From Glossaries to Ontologies: Extracting Semantic Structure from Textual Definitions" in P. Buitellar & P. Cimiano (eds.) *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Amsterdam: IOS Press, The Netherlands, pp. 71-87
- Newell, A. & Simon, H.A. 1976, "Computer science as empirical inquiry: symbols and search", *Communications of the ACM*, vol. 19, no. 3, pp. 113-126.
- Niles, I. & Pease, A. 2001, "Towards a standard upper ontology", FOIS '01: *Proceedings of the International Conference on Formal Ontology in Information Systems ACM*, New York, NY, USA, pp. 2-9.
- Nunamaker Jr., J.F, Chen, M. & Purdin, T.D.M. (1990/91). "Systems development in information systems research", *Journal of Management Information Systems*, vol. 7, no. 3, pp. 89-106.
- Owen, C.L. 1998, "Design Research: building the knowledge base", *Design Studies*, vol. 19, no. 1, pp. 9-20.
- Pan, D. & Pan, Y. 2006, "Using Ontology Repository to Support Data Mining", *Intelligent Control and Automation, 2006.(WCICA 2006): The Sixth World Congress* vol. 2, June 2006, pp. 5947-5951.
- Papazoglou, M. & van den Heuvel, W. 2007, "Service oriented architectures: approaches, technologies and research issues", *The VLDB Journal*, vol. 16, no. 3, pp. 389-415.
- Battle, S., Bernstein, A., Boley, H., Grosz, B., Gruninger, M., Hull, M., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Jianwen, S., Said, T., "Semantic Web Service framework". W3C Member Submission 9 September 2005. [Online] Available at: <http://www.w3.org/Submission/SWSF/> (Accessed in November 2008).
- Pivk, A. Cimiano, P. Sure, Y. 2005, "From tables to frames", *Web Semantics*, vol. 3, no. 2-3, pp. 132.
- Pivk, A., Cimiano, P., Sure, Y., Gams, M., Rajkovič, V. & Studer, R. 2007, "Transforming arbitrary tables into logical form with TARTAR", *Data & Knowledge Engineering*, vol. 60, no. 3, pp. 567-595.
- Purao, S. 2002, Design Research in the technology of information systems: Truth or dare. Unpublished paper available at www.purao.ist.psu.edu/working-papers/dare-purao.pdf

- Reed, S. & Lenat, D. 2002, "Mapping Ontologies into Cyc", Proceedings of American Association for Artificial Intelligence (AAAI). Technical Report WS-02-11, pp.1-7. [Online] Available at: <https://www.aaai.org/Papers/Workshops/2002/WS-02-11/WS02-11-010.pdf> (Accessed in June 2010).
- Sabou, M. & Pan, J. 2007, "Towards semantically enhanced web service repositories", *Journal of Web Semantics*, vol. 5, no. 2, pp. 142-150.
- Sabou, M., Wroe, C., Goble, C. & Stuckenschmidt, H. 2005, "Learning domain ontologies for semantic web service descriptions", *Journal of Web Semantics*, vol. 3, no. 4, pp. 340-365.
- Sabou, M. 2005, "Learning web service ontologies: an automatic extraction method and its evaluation", in P. Buitelaar, P. Cimiano & B. Magnini (eds.) *Ontology Learning from Text: Methods, evaluation and applications*, Amsterdam, Netherlands: IOS Press, pp. 125-139.
- Sanderson, M. & Croft, B. 1999, "Deriving concept hierarchies from text", Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM , pp. 206-213.
- Shadbolt, N., Hall, W. & Berners-Lee, T. 2006, "The Semantic Web revisited", *IEEE Intelligent Systems*, vol. 21, no. 3, pp. 96-101.
- Shafiq, O. 2007, "Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools". 2nd International Conference on Internet and Web Applications and Services (ICIW '07), Morne, 13-19 May , pp. 31-37.
- Shamsfard, M. & Barforoush, A.A. 2003, "The state of the art in ontology learning: a framework for comparison", *Knowledge Engineering Review*, vol. 18, no. 4, pp. 293-316.
- Sheth, A., Verma, K. & Gomadam, K. 2006, "Semantics to energize the full services spectrum", *Communications of the ACM*, vol. 49, no. 7, pp. 55-61.
- Simon, H.A. 1996, *The Sciences of the Artificial* (3rd ed.), Cambridge, MA: MIT Press.
- Snow, R., Jurafsky, D. & Ng, A.Y. 2006, "Semantic taxonomy induction from heterogenous evidence", Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL, Sydney, Australia, July 17-18, pp. 801-808.
- Staab, S. & Maedche, A. 2001, "Knowledge portals: Ontologies at work", *AI MAG*, vol. 22, no. 2, pp. 63-75.
- Staab, S., Studer, R., Schnurr, H. & Sure, Y. 2001, "Knowledge processes and ontologies", *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 26-34.
- Staab, S. & Studer, R. (eds) 2004, *Handbook on Ontologies*. International Handbooks on Information Systems. Berlin/Heidelberg: Springer-Verlag.

- Staab, S. & Studer, R. 2009, *Handbook on ontologies*, International Handbooks on Information Systems, Second Edition, Berlin/Heidelberg: Springer-Verlag.
- Studer, R., Grimm, S. & Abecker, A. 2007, *Semantic Web Services Concepts, Technologies, and Applications*, Springer-Verlag, Berlin/Heidelberg.
- R. Studer, R. Benjamins, and D. Fensel. "Knowledge engineering: Principles and methods". *Data & Knowledge Engineering*, 25, no 1–2, pp. 161–198, 1998.
- Sycara, K., Paolucci, M., Soudry, J. & Srinivasan, N. 2004, "Dynamic discovery and coordination of agent-based Semantic Web Services", *IEEE Internet Computing*, vol. 8, no. 3, pp. 66-73.
- Takeda, H., Veerkamp, P. & Yoshikawa, H. 1990, "Modeling design process", *AI magazine*, vol. 11, no. 4, pp. 37.
- Tsai, W.T., Malek, M., Chen, Y. & Bastani, F. 2006, "Perspectives on service-oriented computing and service-oriented system engineering", *Proceedings - Second IEEE International Symposium on Service-Oriented System Engineering, SOSE 2006, Shanghai, China, 25-26 October*, pp. 3-8.
- Van Rijsbergen, C.J. 1979, *Information retrieval* (2nd edn.), London, Butterworth.
- Vaishnavi, V. and Kuechler, W. (2004/5). "Design Research in Information Systems" January 20, 2004, [Online] Available at: URL:<http://desrist.org/design-research-in-information-systems>. (August 16, 2009).
- Velardi, P., Navigli, R., Cucchiarelli, A., Neri, F., Buitelaar, P., Cimiano, P. & Magnini, B. 2005, "Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies", in P. Buitelaar, P. Cimiano & B. Magnini (eds.) *Ontology Learning from Text: Methods, evaluation and applications*, Amsterdam, Netherlands: IOS Press, pp. 92–106.
- Völker, J., Haase, P. & Hitzler, P. 2008, "Learning Expressive Ontologies" in P. Buitelaar & P. Cimiano (eds.) *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, IOS Press, pp. 45-67.
- Volz, R., Handschuh, S., Staab, S. & Studer, R. 2003, "OntoLiFT Demonstrator" WonderWeb: Ontology Infrastructure for the Semantic Web", IST Project 2001-33052 WonderWeb. [Online] Available at: <http://wonderweb.semanticweb.org/deliverables/documents/D11.pdf> (Accessed February 2009).
- Wang, X.H. 2004, Ontology based context modeling and reasoning using OWL, *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, Singapore, Singapore, 14-17 March, pp. 18-22.
- Weichselbraun, A., Wohlgenannt, G. & Scharl, A. 2010, "Refining non-taxonomic relation labels with external structured data to support ontology learning", *Data & Knowledge Engineering*, vol. In Press, Corrected Proof.

- Winter, R. 2008, "Design science research in Europe", *European Journal of Information Systems*, vol. 17, no. 5, pp. 470-475.
- Witten, I.H. & Frank, E. 2002, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, ACM, New York, USA.
- Yu, L. 2007, *Introduction to the Semantic Web and Semantic Web Services*, Boca Raton, FL Chapman & Hall/CRC.
- Yu, Q., Liu, X., Bouguettaya, A. & Medjahed, B. 2008, "Deploying and Managing Web Services: issues, solutions, and directions", *The VLDB Journal The International Journal on Very Large Data Bases*, vol. 17, no. 3, pp. 537-572.
- Zhou, L. 2007, "Ontology learning: state of the art and open issues", *Information Technology and Management (Bussum)*, vol. 8, no. 3, pp. 241 (12 pages).

APPENDICES

Appendix A - POS tagger

A.1 Hepple Part-of-Speech Tags used by GATE POS tagger.

Part-of-Speech Tags used in the Hepple Tagger

CC - coordinating conjunction: "and", "but", "nor", "or", "yet", plus, minus, less, times (multiplication), over (division). Also "for" (because) and "so" (i.e., "so that").

CD - cardinal number

DT - determiner: Articles including "a", "an", "every", "no", "the", "another", "any", "some", "those".

EX - existential there: Unstressed "there" that triggers inversion of the inflected verb and the logical subject; "There was a party in progress".

FW - foreign word

IN - preposition or subordinating conjunction

JJ - adjective: Hyphenated compounds that are used as modifiers; happy-go-lucky.

JJR - adjective - comparative: Adjectives with the comparative ending "-er" and a comparative meaning. Sometimes "more" and "less".

JJS - adjective - superlative: Adjectives with the superlative ending "-est" (and "worst"). Sometimes "most" and "least".

JJSS - -unknown-, but probably a variant of JJS

-LRB- - -unknown-

LS - list item marker: Numbers and letters used as identifiers of items in a list.

MD - modal: All verbs that don't take an "-s" ending in the third person singular present: "can", "could", "dare", "may", "might", "must", "ought", "shall", "should", "will", "would".

NN - noun - singular or mass

NNP - proper noun - singular: All words in names usually are capitalized but titles might not be.

NNPS - proper noun - plural: All words in names usually are capitalized but titles might not be.

NNS - noun - plural

NP - proper noun - singular

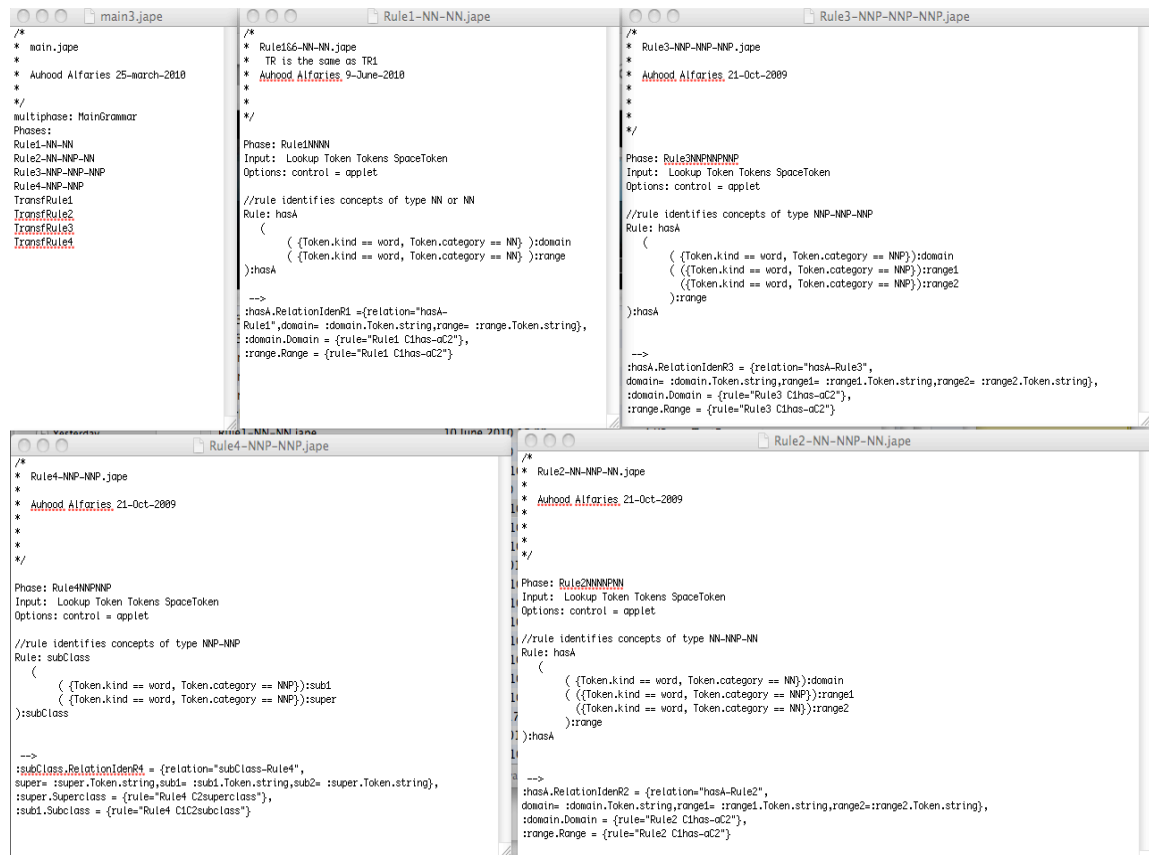
Figure 0-1: Part-Of-Speech Tags (from GATE user Guide)

NPS - proper noun - plural
 PDT - predeterminer: Determinerlike elements preceding an article or possessive pronoun; "all/PDT his marbles", "quite/PDT a mess".
 POS - possessive ending: Nouns ending in "'s" or "'".
 PP - personal pronoun
 PRPR\$ - unknown-, but probably possessive pronoun
 PRP - unknown-, but probably possessive pronoun
 PRP\$ - unknown, but probably possessive pronoun, such as "my", "your", "his", "his", "its", "one's", "our", and "their".
 RB - adverb: most words ending in "-ly". Also "quite", "too", "very", "enough", "indeed", "not", "-n't", and "never".
 RBR - adverb - comparative: adverbs ending with "-er" with a comparative meaning.
 RBS - adverb - superlative
 RP - particle: Mostly monosyllabic words that also double as directional adverbs.
 STAART - start state marker (used internally)
 SYM - symbol: technical symbols or expressions that aren't English words.
 TO - literal to
 UH - interjection: Such as "my", "oh", "please", "uh", "well", "yes".
 VBD - verb - past tense: includes conditional form of the verb "to be"; "If I were/VBD rich...".
 VBG - verb - gerund or present participle
 VBN - verb - past participle
 VBP - verb - non-3rd person singular present
 VB - verb - base form: subsumes imperatives, infinitives and subjunctives.
 VBZ - verb - 3rd person singular present
 WDT - wh-determiner
 WP\$ - possessive wh-pronoun: includes "whose"
 WP - wh-pronoun: includes "what", "who", and "whom".
 WRB - wh-adverb: includes "how", "where", "why". Includes "when" when used in a temporal sense.

Figure 0-2: Part-Of-Speech Tags (from GATE User Guide)

Appendix B - JAPE code

B.1 JAPE files created for each rule and transformation rule.



```
/*
 * main.jape
 *
 * Auhood Alfaries 25-march-2018
 *
 */
multiphase: MainGrammar
Phases:
Rule1-NN-NN
Rule2-NN-NNP-NN
Rule3-NNP-NNP-NNP
Rule4-NNP-NNP
TransfRule1
TransfRule2
TransfRule3
TransfRule4

/*
 * Rule1-NN-NN.jape
 *
 * TR is the same as TR1
 * Auhood Alfaries, 9-June-2018
 *
 */
Phase: Rule1NNNN
Input: Lookup Token Tokens SpaceToken
Options: control = applet

//rule identifies concepts of type NN or NN
Rule: hasA
(
  (Token.kind == word, Token.category == NN):domain
  (Token.kind == word, Token.category == NN):range
)hasA

-->
:hasA.RelationIdenR1 = {relation="hasA-
Rule1", domain= :domain.Token.string, range= :range.Token.string},
:domain.Domain = {rule="Rule1 C1has-aC2"},
:range.Range = {rule="Rule1 C1has-aC2"}

/*
 * Rule3-NNP-NNP-NNP.jape
 *
 * Auhood Alfaries, 21-Oct-2009
 *
 *
 */
Phase: Rule3NNPNNPNNP
Input: Lookup Token Tokens SpaceToken
Options: control = applet

//rule identifies concepts of type NNP-NNP-NNP
Rule: hasA
(
  (Token.kind == word, Token.category == NNP):domain
  ((Token.kind == word, Token.category == NNP):range1
  ((Token.kind == word, Token.category == NNP):range2
  ):range
)hasA

-->
:hasA.RelationIdenR3 = {relation="hasA-Rule3",
domain= :domain.Token.string, range1= :range1.Token.string, range2= :range2.Token.string},
:domain.Domain = {rule="Rule3 C1has-aC2"},
:range.Range = {rule="Rule3 C1has-aC2"}

/*
 * Rule4-NNP-NNP.jape
 *
 * Auhood Alfaries, 21-Oct-2009
 *
 *
 */
Phase: Rule4NNPNNP
Input: Lookup Token Tokens SpaceToken
Options: control = applet

//rule identifies concepts of type NNP-NNP
Rule: subClass
(
  (Token.kind == word, Token.category == NNP):sub1
  (Token.kind == word, Token.category == NNP):super
):subClass

-->
:subClass.RelationIdenR4 = {relation="subClass-Rule4",
super= :super.Token.string, sub1= :sub1.Token.string, sub2= :super.Token.string},
:super.Superclass = {rule="Rule4 C2superclass"},
:sub1.Subclass = {rule="Rule4 C1C2subclass"}

/*
 * Rule2-NN-NNP-NN.jape
 *
 * Auhood Alfaries, 21-Oct-2009
 *
 *
 */
Phase: Rule2NNPNNPNN
Input: Lookup Token Tokens SpaceToken
Options: control = applet

//rule identifies concepts of type NN-NNP-NN
Rule: hasA
(
  (Token.kind == word, Token.category == NN):domain
  ((Token.kind == word, Token.category == NNP):range1
  ((Token.kind == word, Token.category == NN):range2
  ):range
)hasA

-->
:hasA.RelationIdenR2 = {relation="hasA-Rule2",
domain= :domain.Token.string, range1= :range1.Token.string, range2= :range2.Token.string},
:domain.Domain = {rule="Rule2 C1has-aC2"},
:range.Range = {rule="Rule2 C1has-aC2"}}
```

Figure 0-3: JAPE code snippet illustrating code for Rules 1-4

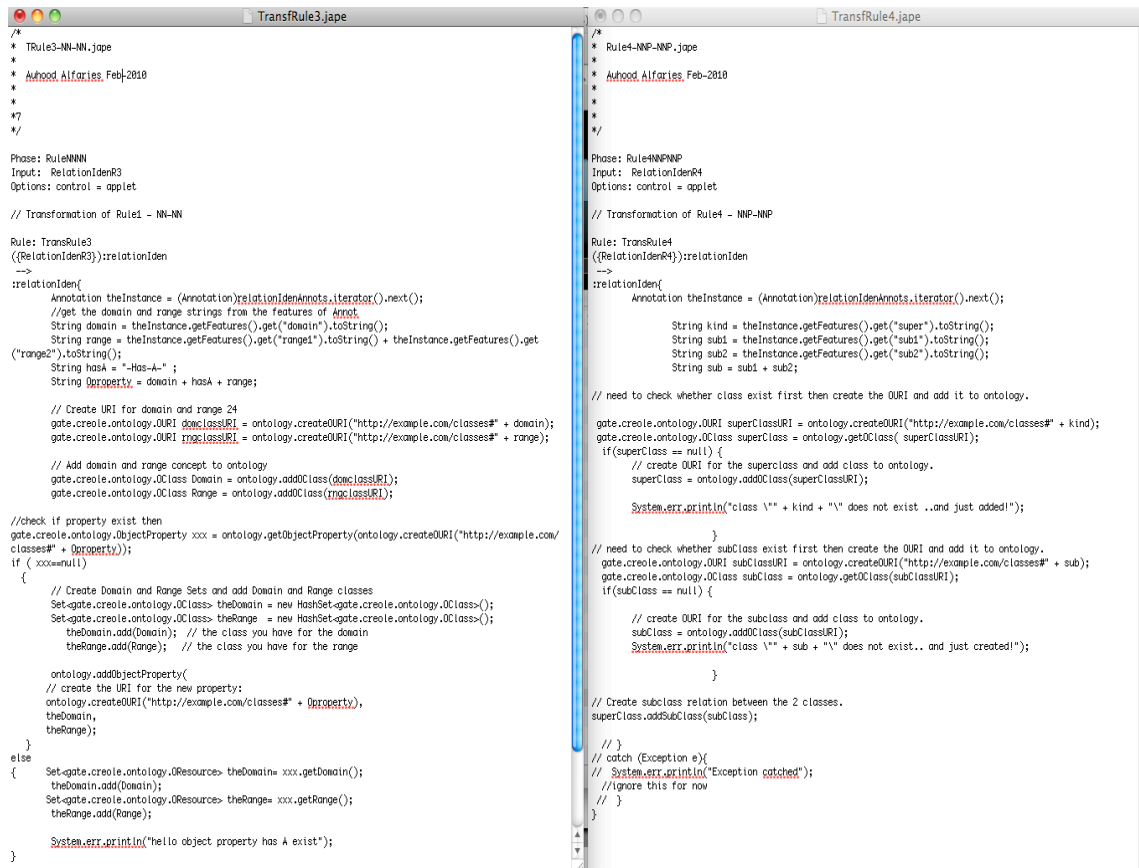


Figure 0-4: JAPE Snippet, illustrating code for transformation rules TR3 and TR4

C.1 First set sample: Financial Web Services and the SOLF learned ontology model (Used for Iteration 1 and 2)



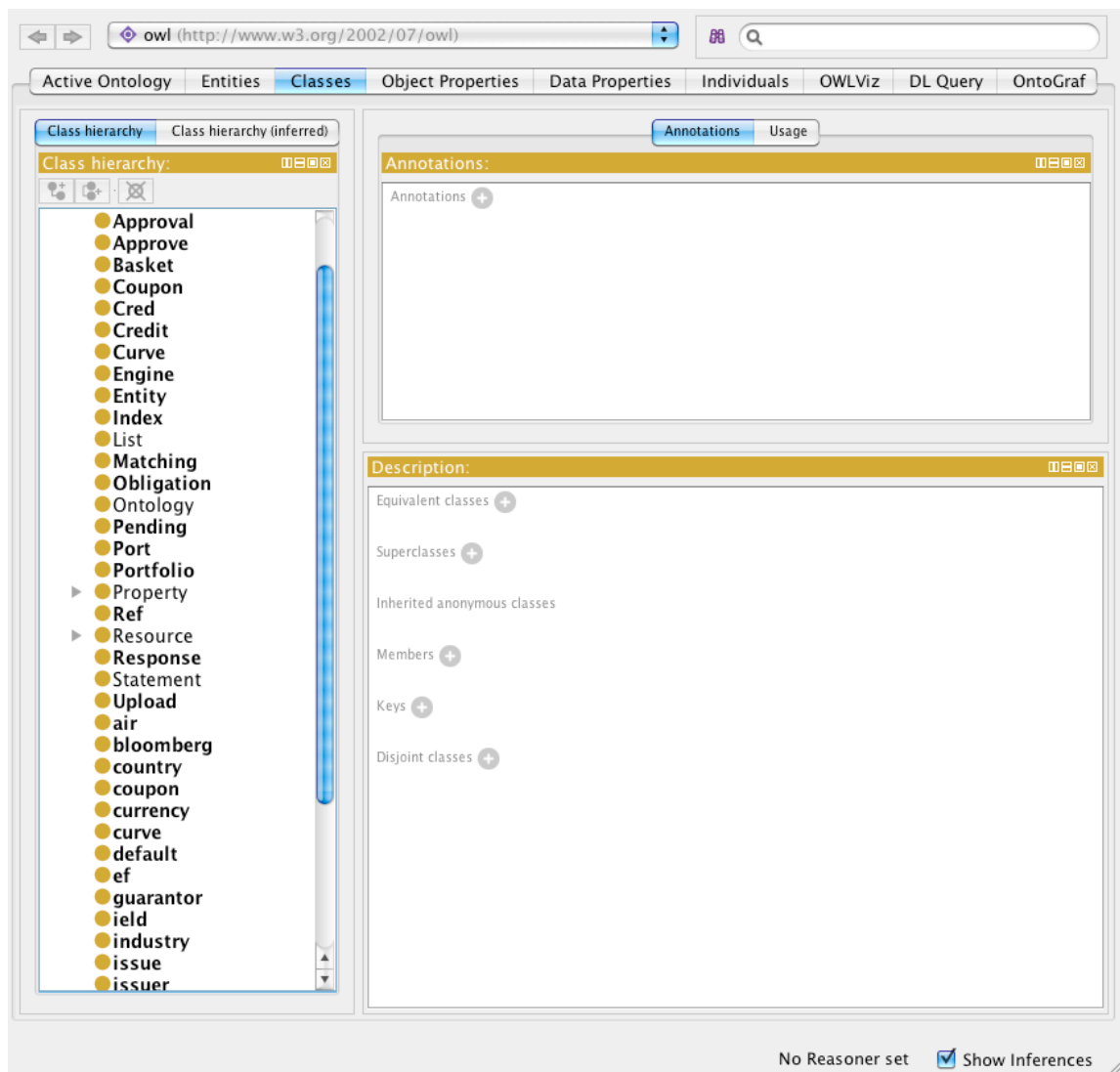


Figure 0-6: Financial Ontology Model (Iteration 1)

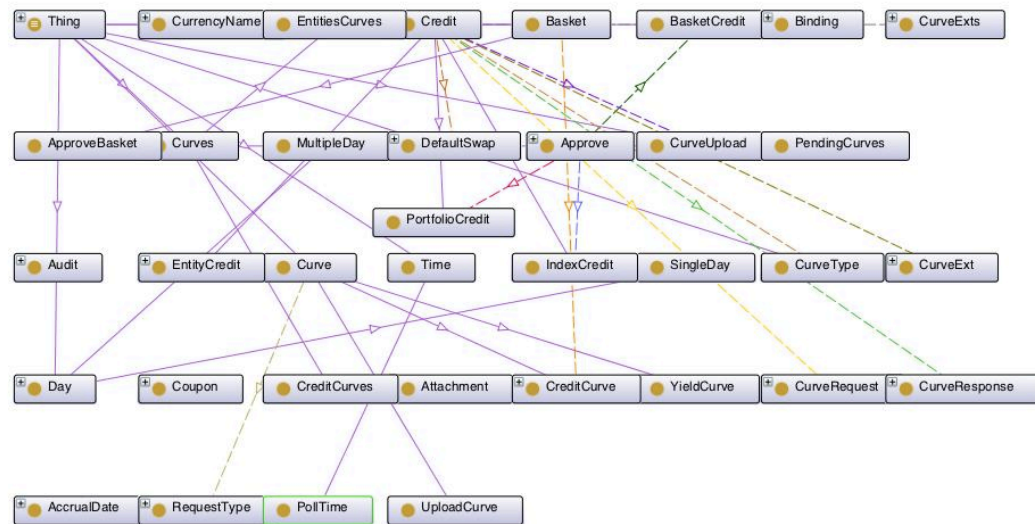
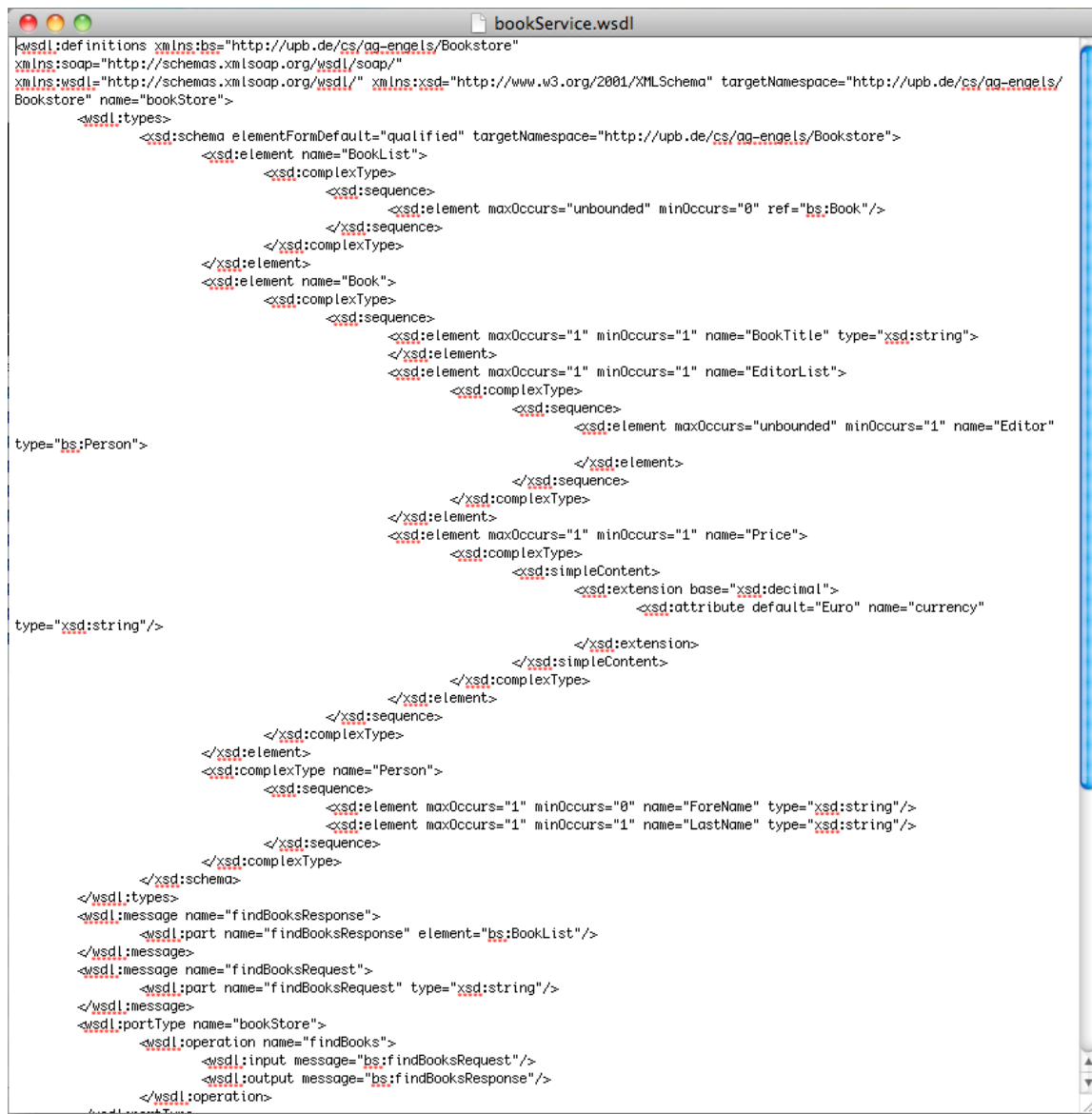


Figure 0-7: Financial Ontology Model (Iteration 2)

C.1 C.2 Second set sample: Books Web Services, SOLFO and GSO (Iteration 3)

The image shows a screenshot of a text editor window titled "bookService.wsdl". The editor contains XML code for a WSDL. The code defines a namespace "http://upb.de/cs/ag-engels/Bookstore" and includes several schema elements. It defines a "BookList" element as a sequence of "Book" elements. The "Book" element is a complex type containing a "BookTitle" string, an "EditorList" sequence of "Editor" elements, and a "Price" decimal with a "currency" attribute. The "Editor" element is a complex type containing a "ForeName" string and a "LastName" string. The WSDL also defines two messages: "findBooksResponse" and "findBooksRequest". The "findBooksResponse" message has a part named "findBooksResponse" of type "BookList". The "findBooksRequest" message has a part named "findBooksRequest" of type "string". Finally, it defines a port type named "bookStore" with an operation named "findBooks" that takes the "findBooksRequest" message as input and returns the "findBooksResponse" message as output.

```
<?xml:definitions xmlns:bs="http://upb.de/cs/ag-engels/Bookstore"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://upb.de/cs/ag-engels/Bookstore" name="bookStore">
  <wsdl:types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://upb.de/cs/ag-engels/Bookstore">
      <xsd:element name="BookList">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" ref="bs:Book"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Book">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="BookTitle" type="xsd:string">
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" name="EditorList">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element maxOccurs="unbounded" minOccurs="1" name="Editor"
type="bs:Person">
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" name="Price">
              <xsd:complexType>
                <xsd:simpleContent>
                  <xsd:extension base="xsd:decimal">
                    <xsd:attribute default="Euro" name="currency"
type="xsd:string"/>
                  </xsd:extension>
                </xsd:simpleContent>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="Person">
        <xsd:sequence>
          <xsd:element maxOccurs="1" minOccurs="0" name="ForeName" type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="1" name="LastName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="findBooksResponse">
    <wsdl:part name="findBooksResponse" element="bs:BookList"/>
  </wsdl:message>
  <wsdl:message name="findBooksRequest">
    <wsdl:part name="findBooksRequest" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="bookStore">
    <wsdl:operation name="findBooks">
      <wsdl:input message="bs:findBooksRequest"/>
      <wsdl:output message="bs:findBooksResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Figure 0-8: Books Service Sample 1 Snippet

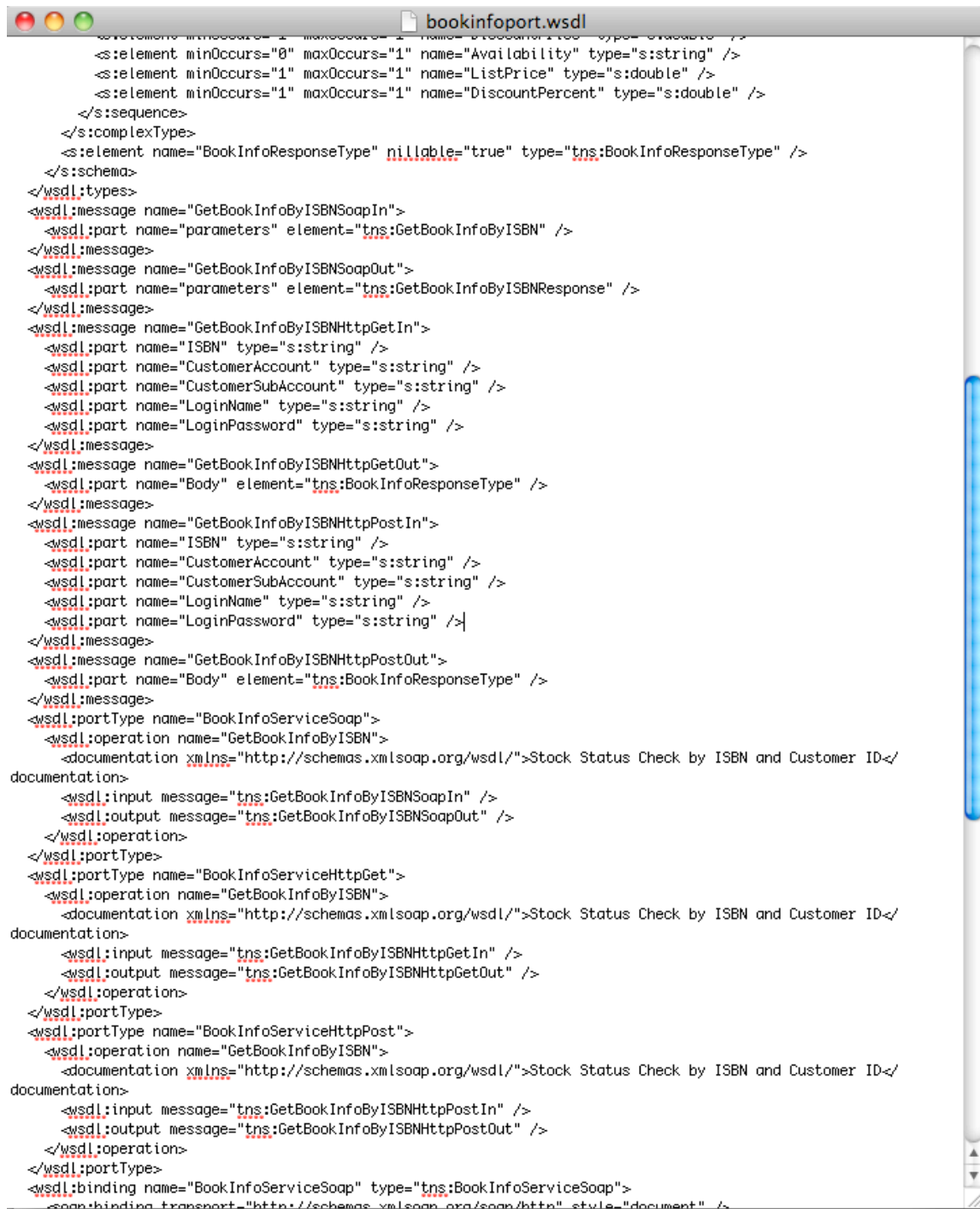


Figure 0-9: Books Service Sample 2 Snippet

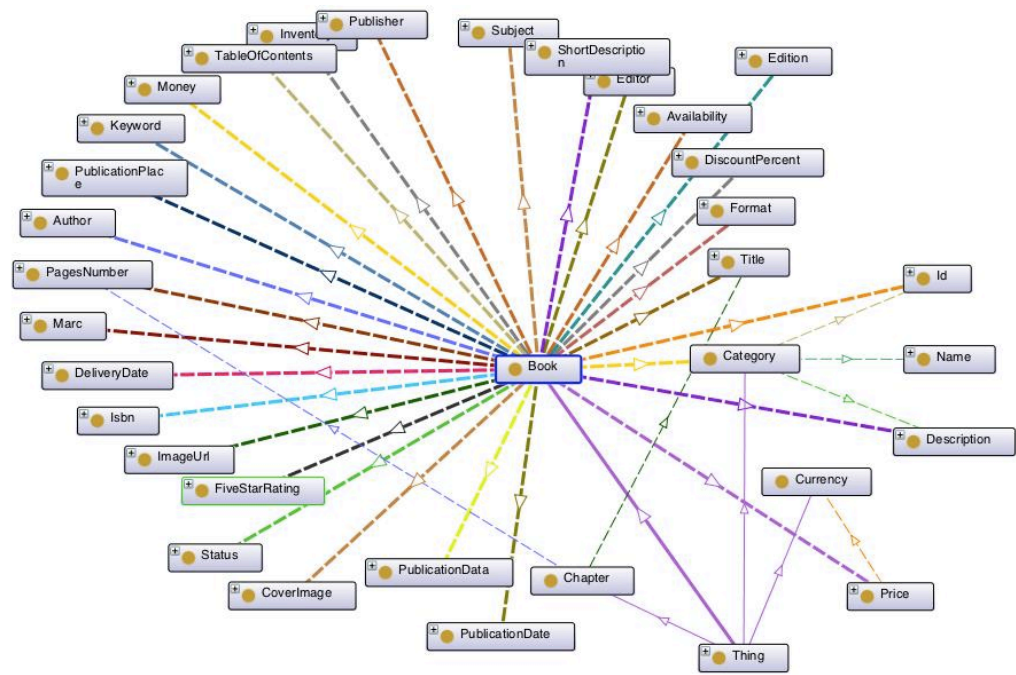


Figure 0-10: Books GSO Snippet

C.2 C.3 Third set sample: Financial services, SOLFO and GSO (Iteration 3)

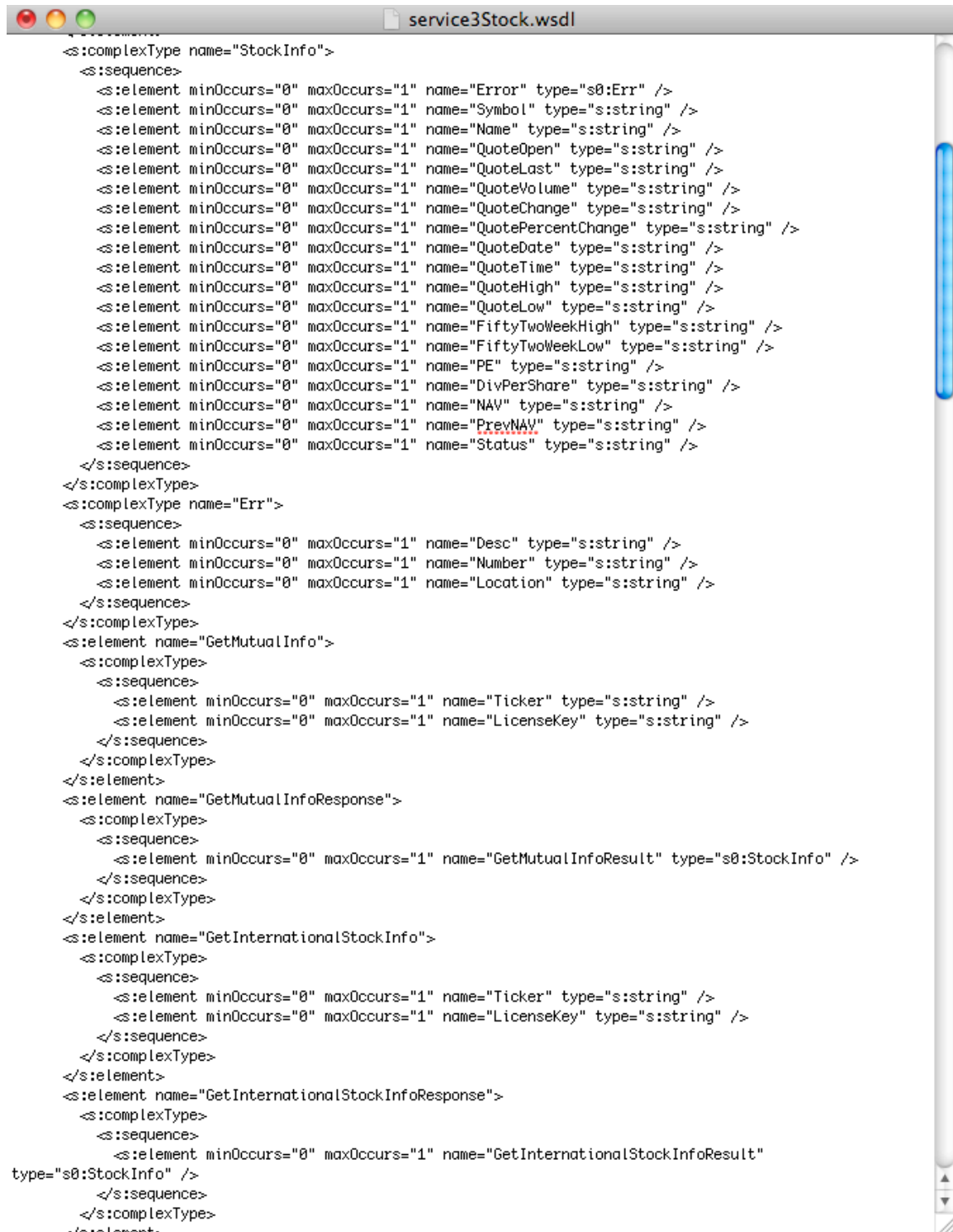


Figure 0-12: Finance Sample 1 Snippet

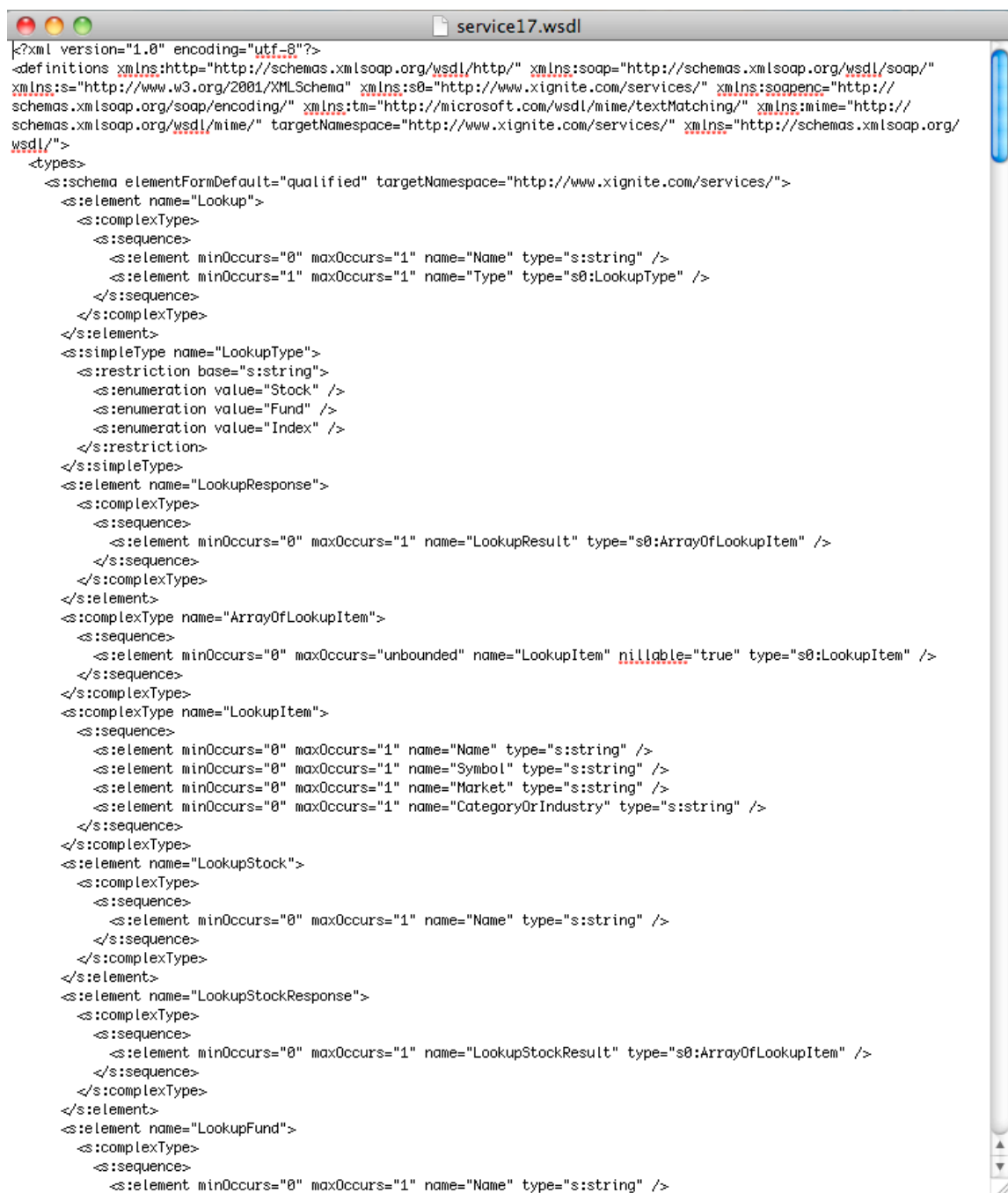


Figure 0-13: Finance Sample 2 Snippet

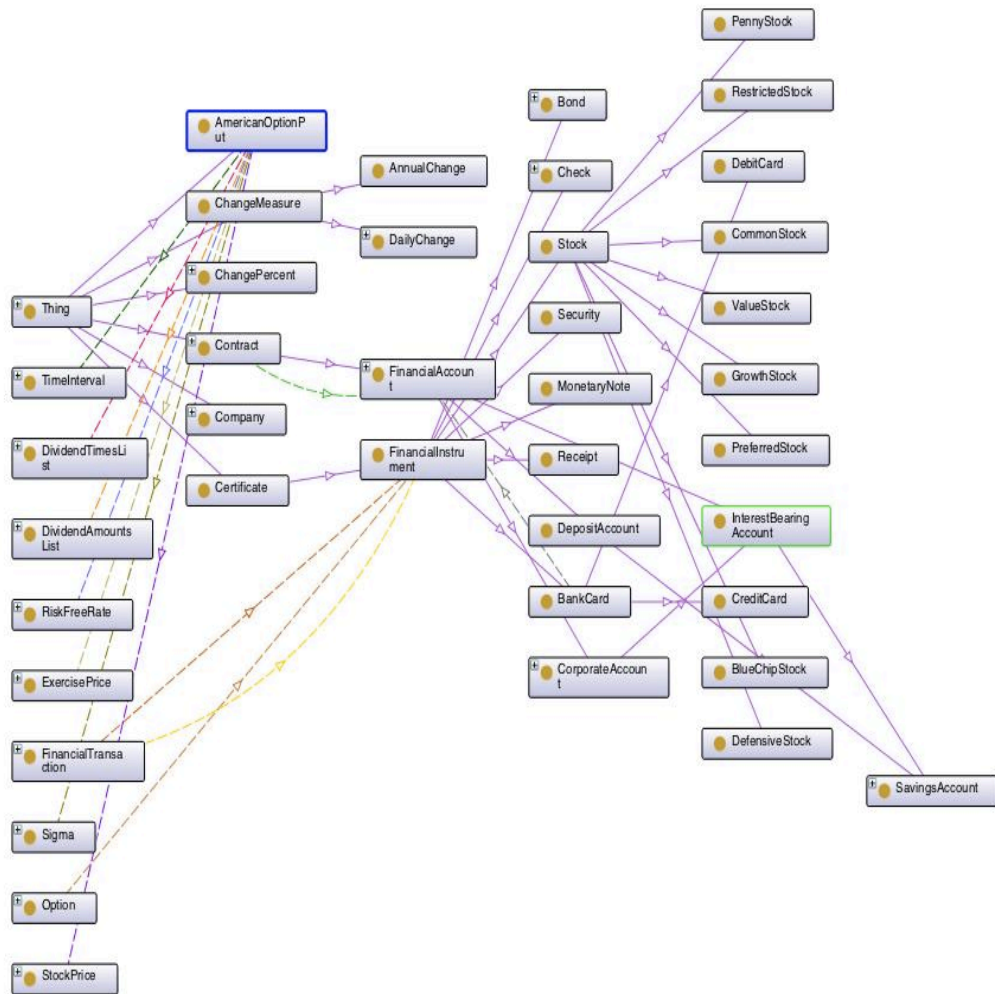


Figure 0-14: Snippet Of Financial GSO

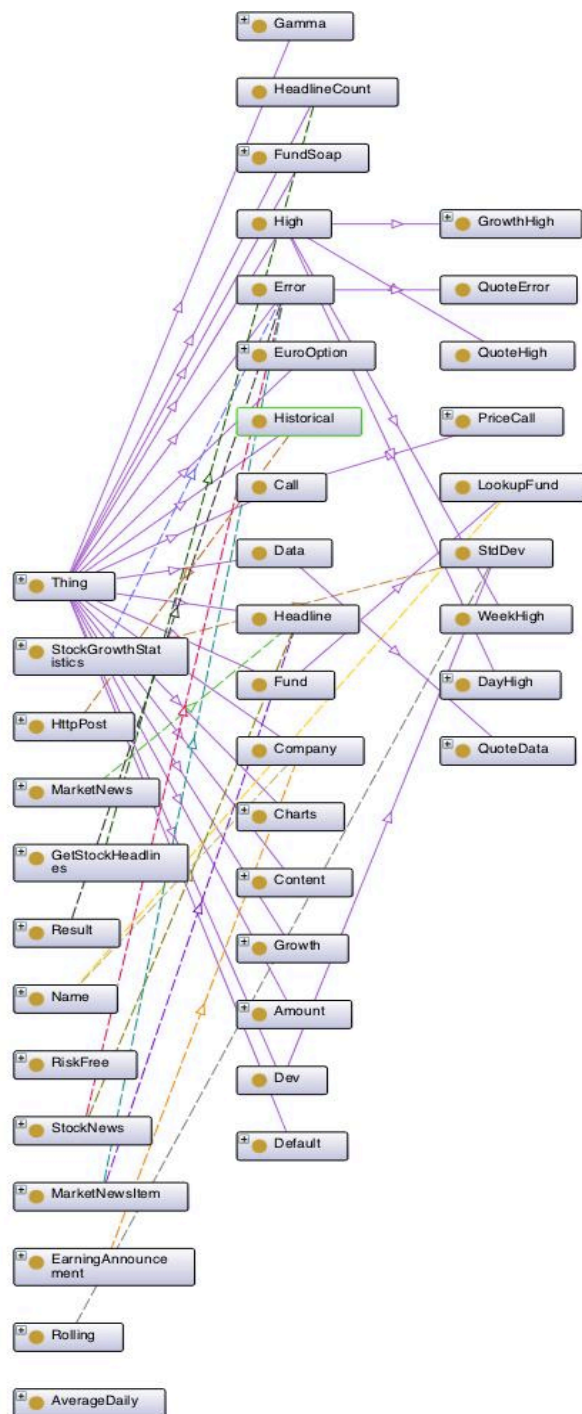


Figure 0-15: Snippet of Financial SOLFO

Appendix D - Evaluation Spread Sheets

D.1 Iteration 1 Evaluation Sheets

Domain Expert (D.E.) Evaluation of Iteration 1: The following tables illustrate the Web Service Term Model (WSTM) for Web Service 2, for the three methods as described in Chapter 4.

WSTM-WS2-Method1								
No.	XSD-Concepts	D.E. Score	No.	XSD-Concepts	D.E. Score	No.	XSD-Concepts	D.E. Score
1	name	1	52	traderTrade	1	103	MatchingRequestHeader	0
2	value	1	53	matchInfo	0	104	MatchingResponseHeader	0
3	base	1	54	updatedTraderTrade	1	105	MatchingBroadcastHeader	0
4	schemaLocation	0	55	closeMatch	0	106	MatchingRequestBody	0
5	targetNamespace	0	56	validate	1	107	QueryTradeMatchStatusRequ	0
6	version	0	57	insertRule	1	108	ForceTradeMatchRequestBod	0
7	www	0	58	updateRule	1	109	ForceTradeNoMatchRequestB	0
8	xsd	0	59	deleteRule	1	110	RematchTradesRequestBody	0
9	xml	0	60	evalData	0.5	111	AttemptTradeMatchRequestB	0
10	lehman	0	61	Thing	1	112	MatchingResponseBody	0
11	http	0	62	rosetta	0	113	ForceTradeMatchResponseBo	0
12	History	1	63	field	1	114	ForceTradeNoMatchResponse	0
13	header	1	64	dateTime	1	115	QueryTradeMatchStatusResp	0
14	winston	0	65	success	1	116	AttemptTradeMatchResponse	0
15	body	0	66	tradeId	1	117	RematchTradesResponseBod	0
16	logQuery	0	67	domain	1	118	MatchingBroadcastBody	0
17	tradeMatchingContext	0	68	org	1	119	TradeMatchBroadcastBody	0
18	tradeValidationContext	0	69	attributeFormDefault	0	120	MatchStatus	0
19	dataId	1	70	elementFormDefault	0	121	TradeMatchStatus	0
20	startDate	1	71	element	1	122	MatchedInfo	1
21	endDate	1	72	date	1	123	TradeMatchedInfo	0
22	rule	1	73	sequence	1	124	TradeNotMatchedInfo	0
23	id	1	74	complexType	0	125	MatchingContext	1
24	failCode	1	75	schema	0	126	ValidationRequestHeader	0
25	failMessage	1	76	simpleType	0	127	ValidationResponseHeader	0
26	context	1	77	restriction	1	128	ValidationRequestBody	0
27	severity	1	78	maxLength	1	129	TradeValidateRequestBody	0
28	evalType	1	79	boolean	1	130	ValidationResponseBody	0
29	definition	1	80	enumeration	0	131	TradeValidateResponseBody	0
30	path	1	81	complexContent	0	132	ValidationContext	1
31	type	1	82	extension	1	133	AdminRequestHeader	0
32	TRADE	0.5	83	choice	1	134	AdminResponseHeader	0
33	productType	1	84	string	0	135	AdminRequestBody	0
34	envelope	0	85	anyType	0	136	RuleAdminRequestBody	0
35	validationResponse	0	86	positiveInteger	1	137	AdminResponseBody	0
36	validationRequest	0	87	Pattern	1	138	RuleAdminResponseBody	0
37	matchingResponse	0	88	UTF-	0	139	SimpleTrade	1
38	matchingRequest	0	89	HistoryRequestHeader	0	140	TradeValidationLog	1
39	matchingBroadcast	0	90	HistoryResponseHeader	0	141	TradeMatchingLog	1
40	historyResponse	0	91	HistoryRequestBody	0	142	KeyValuePair	1
41	historyRequest	0	92	HistoryResponseBody	0	143	XMLSchema	0
42	adminResponse	0	93	WARNING	1	144	VALIDATION	1
43	adminRequest	1	94	COMPLEX	1	145	MATCHING	1
44	forceMatch	0	95	SIMPLE	1	146	Identifier	1
45	forceNoMatch	0	96	LOOKUP	1	147	HistoryLog	1
46	rematch	0	97	RuleContext	1	148	log	1
47	attemptMatch	0	98	TradeRuleContext	1	149	live	1
48	request	1	99	CDS	1	Total	149	71
49	salesTradeId	1	100	CDO	1			
50	traderTradeId	1	101	Errors	1			
51	salesTrade	1	102	Standard	1			

Figure 0-16: Method1-WS2 (XSD) Domain Expert Scoring

WSTM-WS2-Method1			WSTM-WS2-Method2		
No.	WSDL-Concept	D.E. Score	No.	WSDL-Concept	D. E. Scoring
1	location	1	1	location	1
2	style	1	2	style	1
3	namespace	0	3	namespace	0
4	version	0	4	version	1
5	www	0	5	www	0
6	lonlxwebhost	0	6	lonlxwebhost	0
7	targetNamespac	0	7	target	1
8	use	1	8	use	1
9	message	1	9	chema	1
10	part	1	10	message	1
11	portType	0	11	part	1
12	operation	1	12	port	0
13	input	1	13	operation	1
14	output	1	14	input	1
15	soap	0	15	output	1
16	service	1	16	soap	0
17	port	0	17	service	1
18	xml	0	18	xml	0
19	type	1	19	type	1
20	name	1	20	name	1
21	winston	0	21	winston	0
22	http	0	22	http	0
23	doRequest	0	23	string	1
24	string	1	24	result	1
25	doRequestRespd	0	25	rpc	0
26	result	1	26	wsdl	0
27	rpc	0	27	org	1
28	wsdl	0	28	transport	1
29	org	1	29	xmlsoap	0
30	transport	0	30	body	1
31	soapAction	0	31	reliability	1
32	xmlsoap	0	32	enc	1
33	body	0	33	mime	0
34	reliability	1	34	conversation	0
35	enc	0	35	openuri	0
36	encodingStyle	0	36	w	0
37	mime	0	37	wsr	0
38	conversation	0	38	soapenc	0
39	openuri	0	39	conv	0
40	w	0	40	Request	1
41	wsr	0	41	Engine	0
42	soapenc	0	42	UTF	0
43	conv	0	43	Matching	1
44	UTF-	0	44	XMLS	0
Total	44	15	45	Response	1
			46	Action	1
			47	address	1
			Total	47	26

Figure 0-17: Method1& 2-WS2 (WSDL) Domain Expert Scoring

WSTM-WS2-Method3						WSTM-WS2-Method3		
No.	XSD-Concepts	D. E. scorir	No.	XSD-Concepts	D. E. scoring	No.	WSDL-Concepts	D. E. Scoring
1	location	1	65	min	0.5	1	location	1
2	style	1	66	credit	1	2	style	1
3	namespace	0	67	fid	0	3	namespace	0
4	name	1	68	transport	0	4	version	1
5	value	1	69	xmlsoap	0	5	www	0
6	base	0	70	reliability	1	6	lonlwebhost	0
7	schema	1	71	enc	0	7	target	1
8	target	1	72	element	1	8	use	1
9	version	1	73	mime	0	9	chema	1
10	www	0	74	conversation	0	10	message	1
11	lonlwebhost	0	75	w	0	11	part	1
12	use	1	76	openuri	0	12	port	0
13	chema	1	77	wsr	0	13	operation	1
14	Id	1	78	soapenc	0	14	input	1
15	Rule	1	79	conv	0	15	output	1
16	Date	1	80	sequence	1	16	soap	0
17	xsd	0	81	choice	1	17	service	1
18	message	1	82	extension	1	18	xml	0
19	part	1	83	restriction	1	19	type	1
20	port	0	84	boolean	0	20	name	1
21	operation	1	85	enumeration	1	21	winston	0
22	input	1	86			22	http	0
23	output	1	87	Pattern	1	23	string	1
24	soap	0	88	Response	1	24	result	1
25	service	1	89	Content	1	25	rpc	0
26	xml	0	90	Time	1	26	wsdl	0
27	type	1	91	Data	1	27	org	1
28	lehman	0	92	Log	1	28	transport	1
29	http	0	93	Status	1	29	xmlsoap	0
30	History	1	94	Info	1	30	body	1
31	header	0	95	Engine	1	31	reliability	1
32	winston	0	96	Broadcast	0	32	enc	1
33	body	1	97	Length	1	33	mime	0
34	trade	1	98	Matching	1	34	conversation	0
35	end	1	99	Matched	1	35	openuri	0
36	string	1	100	Match	1	36	w	0
37	result	1	101	Code	1	37	wsr	0
38	rpc	0	102	Trades	1	38	soapenc	0
39	updat	1	103	UTF	0	39	conv	0
40	context	1	104	ERROR	1	40	Request	1
41	severity	0	105	WARNING	1	41	Engine	0
42	eval	0.5	106	COMPLEX	1	42	UTF	0
43	defintion	1	107	LOOKUP	1	43	Matching	1
44	path	1	108	CDS	1	44	XMLS	0
45	product	1	109	CDO	1	45	Response	1
46	query	1	110	Errors	1	46	Action	1
47	force	1	111	Standard	1	47	address	1
48	rematch	0	112	XMLS	0	48	47	26
49	attempt	1	113	Form	1			0.6676301
50	request	1	114	Default	1			66.76%
51	trader	1	115	Natch	0			
52	validate	1	116	Pair	1			
53	envelope	0	117	Occurs	1			
54	validation	1	118	Action	1			
55	admin	1	119	Identifier	1			
56	success	1	120	Key	1			
57	rosetta	0	121	start	1			
58	Thing	1	122	insert	1			
59	Simple	1	123	delete	1			
60	field	1	124	live	1			
61	max	0.5	125	fail	1			
62	wsdl	0	126	close	1			
63	org	1	127	address	1			
64	tc	0	Total	126	89.5			

Figure 0-18: Method3-WS2 (XSD & WSDL) Domain Expert Scoring

D.2 Iteration 2 Evaluation Sheets

SuperClass		SubClass		Domain Only		Range Only	
Match	Score	Match	Score	Match	Score	Match	Score
Abb	x	Amend		air	x	AccrualDate	
Attachment	x	Approval		Amend		AuditHistory	
Audit		Approve		Approval		BasketCredit	
Basket		Assign		Approve		CreditCurve	
Binding	x	Basket		Assign		CurrencyName	
Code		Blotter		Basket		CurveException	x
Credit		Cancel		bloomberg		CurveExt	
Curve		Cred	s	Cancel		CurveExts	
Curves		Credit		country		CurveId	
Day		Curve		coupon		CurveRequest	
Default		Curves		Cred	s	CurveResponse	
Descriptor	x	Day		Credit	s	CurvesRequest	
Descriptors	x	Defaulted		currency		CurveType	
Drill	x	Endpoint	x	curve		CurveUpload	x
Engine	x	Entities		Day		Date	
Entities		Entity		default		DayServer	
Entity		Exception	x	Defaulted		DayTrade	
Event		Ext	s	ef	x	DefaultSwap	
Exception	x	Exts	s	Entity		DescriptorsType	
Ext	x	Fire	x	guarantor		DescriptorType	
Index		Index		ield	x	EnginePort	x
Legal		Insert		Index		EntitiesCredit	
Message	x	Matching		industry		EntitiesRequest	x
Port	s?	Mature		Insert	?	EntitiesResponse	x
Portfolio		Mirror		issue		EntityCredit	
Query	?	Multiple	x	issuer		EntityCurve	
Ref	s	Novate		loomberg	x	EntityId	
Reference		Obligation		Matching		EntityName	
Request	x	Officer		Mature		EntityRequest	x
Response	x	Pending		maturity		EntityResponse	x
Server	x	Policy		Mirror		EntityRetrieval	x
Single	x	Poll		Multiple		EntityUpload	x
Soap	x	Port	s?	Novate		EventType	
System	x	Portfolio		Obligation		ExceptionFault	x
Time		Qte	x	org	?	ExceptionType	x
Token	?	Query	?	owner		ExtRequest	x
Trade		Range		parent		ExtResponse	x
Type		Ref	s	Pending		ExtTrade	
WSR	x	Request	x	Poll	x	FreqName	
39		Reset		Port		Id	
		Retrieval	x	Portfolio		IndexCredit	
		Security		Qte	x	ISOCCode	
		Server	x	Range		LegalName	
		Single	x	rating		Name	
		Soap	x	ref		ObligationId	
		Static		Server	x	ObligationName	
		Summit		Single	x	PortfolioCredit	
		System	x	summit		PortSoap	x
		Target	?	supplement		RefEntities	
		Token	?	Target	x	RefEntity	
		Trade		trade		RequestExt	
		Unwind		Unwind		RequestType	x
		Upload	x	Upload		SectorId	
		Username		Verify		SectorName	
		Verify		55		ServerExt	x
		Yield				ServerResponse	x
		56				SoapBinding	x
						SystemTrade	x
						TierId	
Score	Meaning					TimeTrade	
x	not exist					TradeExt	?
blank	yes					TradeId	x
S	Synonym/ Abreviated					TradeQuery	?
?	Not sure					TradeRequest	x
						TradeResponse	x
						TradeServer	?
						67	

Figure 0-19: Iteration 2 Financial Ontology Domain Expert Scoring

D.3 Iteration 3 Lexical Layer Evaluation Sheets

LSDIS_Finance.owl: GoldStandard Ontology			
Class-Level1,2	Class-Level3,4&5		
1 AmericanOptionPut		98 Option	
2 AverageDailyVolume		99 AmericanStyleOption	
3 Briefing		100 CallOption	
4 CallResult		101 ConventionalOption	
5 Category		102 EquityOption	
6 Certificate		103 EuropeanStyleOption	
7 FinancialInstrument		104 IndexOption	
8 ChangeMeasure		105 Leaps	
9 AnnualChange		106 OptionStrategy	
10 DailyChange		107	SingleOption
11 ChangePercent		108	SpreadOption
12 Company		109 PutOption	
13 Content		110 StockOption	
14 Contract		111 Ordering	
15 FinancialAccount		112 Call	
16 CorporateAccount		113	AmericanPriceC
17 InterestBearingAccount		114	EuroPriceCall
18 SavingsAccount		115 PerShareDivision	
19 MoneyMarket		116 PerShareEarn	
20 MoneyMarket		117 PreviousCls	
21 TraditionalSavingsAccou		118 PriceEasningRatio	
22 DepositAccount		119 Procedure	
23 SavingsAccount		120 Index	
24 MoneyMarket		121	InflationIndex
25 InterestBearingAccount		122	StockIndex
26 SavingsAccount		123 QuoteChange	
27 MoneyMarket		124 QuoteDate	
28 LiabilityAccount		125 QuoteVolume	
29 CreditAccount		126 RegulatoryProcess	
30 Loan		127 Audit	
31 PersonalAccount		128 ProspectusIssuance	
32 Financial contract		129 ActionAuthorization	
33 CusipStock		130	CardCodeVerific
34 Date		131	CheckProcessing
35 DayHight		132	PinEntry
36 DayLow		133 Rho	
37 Delta		134 RiskFreeRate	
38 Description		135 RollingMeanMean	
39 DividendAmountsList		136 RollingStddevMean	
40 DividendTimesList		137 RollingStddevStddev	
41 EarningAnnouncement		138 Sigma	
42 EndDate		139 Source	
43 EpsEstimate		140 SplitRatio	
44 Error		141 StartDate	
45 EuroOptionPut		142 StationaryArtifact	
46 ExercisePrice		143 AtmMachine	
47 FiftytwoWeekHigh		144 Status	
48 FiftytwoWeekLow		145 Stddev	
49 FiftytwoWeekRange		146 StockChange	
50 FinancialTransaction		147 StockGrowthStatistics	
51 AutomaticTransaction		148 StockHeadlines	
52 CommercialService		149 StockInformation	
53 FinancialService		150 StockLookup	
54 OpeningAnAccount		151 StockNews	
55 Investing		152 StockPrice	
56 NonQualifiedInvestme		153 StockVolume	
57 QualifiedInvestment		154 Summary	
58 Liquidation		155 Symbol	
59 OpeningAnAccount		156 Text	
60 Payment		157 BankStatement	
61 Prepayment		158 Bill	
62 Refinancing		159 EquityStatement	CreditCardBill
63 Rollover		160 SecuritiesStatement	
64 SecuritiesTransaction		161 Theta	
65 StockQuote		162 Ticker	
66 StockMarketTransaction		163 TimeMeasure	
67 Withdrawal		164 TimeDuration	
68 FundLookup		165 TimeInterval	
69 Gama		166 TimePosition	
70 GrowthStatisticsHistorical		167 Title	
71 HeadlineCount		168 Type	
72 HighGrowth		169 Url	
73 HighQuote		170 Vega	
74 HistoryMonths		171	
75 IntentionalProcess			
76 ItemLookup			
77 LastPrice			
78 LastQuote			
79 LastTradeAmount			
80 LicenceKey			
81 Location			
82 Lookup			
83 LookupResult			
84 LowGrowth			
85 LowQuote			
86 MarketCap			
87 MarketNews			
88 MarketNewsItem			
89 Mean			
90 Message			
91 Name			
92 Nav			
93 Number			
94 Offering			
95 LoanCommitment			
96 OpenAmount			
97 OpenQuote			

Figure 0-20: Iteration 3 Financial Gold Standard Ontology

SOLFO: Financial Ontology Produced from 5 WS Given from Muhammad Research with GSO					
Concepts-Level1 & 2					
1	Aires	N	51	Desc	Y
2	BuenossAires	N	52	Details	N
3	Amount	N	53	Dev	Y
4	OpenAmount	Y	54	StdDev	Y
5	TradeAmount	Y	55	Dividend	Y
6	AmountList	N	56	DividendAmountList	N
7	Announcement	Y	57	DividendTimeList	N
8	AnnouncementDate	Y	58	EPSEstimate	Y
9	AnnouncementTime	N	59	Earning	Y
10	Announcements	N	60	EarningAnnouncement	Y
11	AnnouncementsResult	N	61	Err	Y
12	AsOfDate	N	62	Error	Y
13	Briefing	Y	63	QuoteError	N
14	Briefings	Y	64	Euro	Y
15	Calc	n	65	CalEuro	N
16	CalcAmericanOptionPriceCall	N	66	EuroOption	Y
17	CalcAmericanOptionPriceCallRes	N	67	Free	Y
18	CalcAmericanOptionPricePut	N	68	RiskFree	Y
19	CalcAmericanOptionPricePutResp	N	69	FreeRate	Y
20	CalcEuroOptionPriceCall	N	70	FromDate	N
21	CalcEuroOptionPriceCallResponse	N	71	Fund	Y
22	CalcEuroOptionPricePut	N	72	LookupFund	Y
23	CalcEuroOptionPricePutResponse	N	73	FundHttp	N
24	Calculator	N	74	FundResponse	N
25	Call	Y	75	FundSoap	N
26	PriceCall	Y	76	Gamma	N
27	CallHttp	N	77	GetCUSIPFund	N
28	CallResponse	N	78	GetCUSIPFundResponse	N
29	CallResult	Y	79	GetCUSIPFundResult	N
30	CallSoap	N	80	GetCUSIPStock	N
31	Cap	Y	81	GetCUSIPStockResponse	N
32	MktCap	Y	82	GetCUSIPStockResult	N
33	CategoryOrIndustry	N	83	GetEarningAnnouncements	N
34	Change	Y	84	GetGrowthStatistics	Y
35	QuoteChange	Y	85	GetGrowthStatisticsHistorical	Y
36	StockChange	Y	86	GetInternationalStockInfo	N
37	Charts	N	87	GetMutualInfo	N
38	BigCharts	N	88	GetQuickQuoteResponse	N
39	Cls	Y	89	GetQuickQuoteResult	N
40	PrefCls	N	90	GetQuote	N
41	Company	Y	91	GetQuoteDataSet	N
42	Content	Y	92	GetReutersMarketNewsDetails	N
43	Daily	Y	93	GetSplitRatio	N
44	AverageDaily	Y	94	GetStockHeadlines	N
45	Data	N	95	GetStockInfo	N
46	QuoteData	N	96	Growth	Y
47	Date	Y	97	StockGrowth	Y
48	Default	N	98	Headline	Y
49	FormDefault	N	99	HeadlineCount	Y
50	Delta	Y	100	Headlines	Y
102	High	Y	101	StockHeadlines	Y
103	DayHigh	Y	155	Nav	Y
104	GrowthHigh	Y	156	PrevNAV	n
105	QuoteHigh	Y	157	Name	Y
106	WeekHigh	N	158	News	Y
107	Historical	Y	159	MarketNews	Y
108	Html	n	160	StockNews	Y
109	Http	n	161	XigniteNews	n
110	AnnouncementsHttp	n	162	NewsHttp	n
111	BriefingHttp	n	163	NewsItem	Y
112	BriefingsHttp	n	164	NewsSoap	n
113	CalculatorHttp	n	165	Number	Y
114	DetailsHttp	n	166	Open	Y
115	HeadlinesHttp	n	167	QuoteOpen	n
116	HistoricalHttp	n	168	Option	Y
117	LookupHttp	n	169	Post	n
118	QuoteHttp	n	170	HttpPost	n
119	Important	n	171	Price	Y
120	ImportantMessageResonse	n	172	ExcercisePrice	n
121	ImportantMessageResult	n	173	OptionPrice	n
122	Info	n	174	StockPrice	Y
123	MutualInfo	n	175	Pricing	n
124	StockInfo	n	176	Quick	n
125	InfoHttp	n	177	Quote	Y
126	InfoResponse	n	178	FastQuote	n
			179	QuickQuote	n

Figure 0-21: Iteration 3 Financial SOLFO Gold Standard Evaluation

127	InfoSoap	n	180	StockQuote	y
128	International	n	181	Range	Y
129	Item	Y	182	WeekRange	Y
130	LookupItem	Y	183	Ratio	Y
131	Key	Y	184	SplitRaio	n
132	LicenceKey	y	185	RatioHttp	n
133	LastPrice	y	186	RationResponse	n
134	LastTradeAmount	y	187	RationSoap	n
135	LastTradeDateTime	n	188	Reference	n
136	Location	y	189	Response	n
137	Lookup	y	190	AnnouncementResponse	n
138	Low	Y	191	BriefingRespns	n
139	DayLow	y	192	BriefingsRespns	n
140	GrowthLow	Y	193	ClassResponse	n
141	QuoteLow	Y	194	DetailsResponse	n
142	WeekLow	Y	195	HeadlinesResponse	n
143	Market	Y	196	HistoricalResponse	n
144	ReutersMarket	n	197	LookupResponse	n
145	MarketNewsItem	y	198	QuoteResponse	n
146	Mean	y	199	Result	Y
147	MeanOfRollingMean	n	200	Reuters	n
148	Message	y	201	Rho	y
149	ImportantMessage	n	202	Risk	Y
150	MessageHttp	n	203	RiskFreeRate	y
151	MessageResponse	n	204	Rolling	Y
152	MessageSoap	n	205	Security	n
153	MonthsOfHistory	n	206	XigniteSecurity	n
154	Mutual	n	207	SecurityHttp	n
208		n			
209	SecuritySoap	n			
210	Set	n			
211	DataSet				
212	Sigma	y			
213	Soap	n			
214	AnnouncementsSoap	n			
215	BriefingSoap	n			
216	BriefingsSoap	n			
217	CalculatorSoap	n			
218	ClassSoap	n			
219	DetailsSoap	n			
220	HistoricalSoap	n			
221	LookupSoap	n			
222	QuoteSaop	n			
223	Source	y			
224	Split	Y			
225	Std	Y			
226	RollingStd	Y			
227	StdDevOfRollingMean	n			
228	Stock	Y			
229	InternationalStock	n			
230	LookupStock	n			
231	StockGrowthStatistics	y			
232	StockHttp	n			
233	StockResponse	n			
234	StockSoap	n			
235	StockVolume	y			
236	Summary	y			
237	Symbol	y			
238	StockSymbol	n			
239	Symbols	n			
240	StockSymbols	n			
241	Text	y			
242	Theta	y			
243	Ticker	y			
244	Time	Y			
245	QuoteTime	n			
246	TimesList	Y			
247	Title	y			
248	ToDate	n			
249	Type	y			
250	LookupType	n			
251	Url	y			
252	Vega	y			
253	Xignite	n			
254	tock	n			
255	und	n			

Figure 0-22: Iteration 3 Financial SOLFO Gold Standard Evaluation