2016

A Run-Time Approach of Combining Ontologies to Enhance Interactive Requirements Elicitation for Software Customization

Shubhrendu Tripathi
University of Windsor

A Run-Time Approach of Combining Ontologies to Enhance Interactive Requirements Elicitation for Software Customization

by

Shubhrendu Tripathi

A Thesis

Submitted to the Faculty of Graduate Studies

through Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2016

A Run-Time Approach of Combining Ontologies to Enhance Interactive Requirements Elicitation for Software Customization

by

Shubhrendu Tripathi

APPROVED BY:

Dr. A. Azab

Dept. of Industrial & Manufacturing Systems Engineering

Dr. S. Goodwin

School of Computer Science

Dr. X. Yuan, Advisor

School of Computer Science

May 20, 2016

Declaration of Co-Authorship / Previous Publication

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research of the author and his supervisor Prof. Xiaobu Yuan. This joint research has been published / submitted to various conferences that are listed below.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from Prof. Xiaobu Yuan to include those materials in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes two original papers that have been previously published / submitted for publication in peer reviewed conferences, as follows:

Thesis Chapter	Publication title / Full citation	Publication status
Chapter 3, Chapter 4, Chapter 5	Yuan X. and Tripathi S., "Combining ontologies for requirements elicitation," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2015 IEEE International*. IEEE, 2015, pp. 1-5.	Conference Proceeding (Published)
Chapter 3, Chapter 4, Chapter 5	Yuan X. and Tripathi S., "An approach of dynamically combining ontologies for interactive requirements elicitation," in *Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on.* IEEE, 2016	Conference Proceeding (In press)

I certify that I have obtained written permissions from the copyright owners to include the above published materials in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that the copyright owners of the two papers, referenced above, allow (IEEE)

copyrighted material to be used in this thesis without additional written permission. [1] [2].

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

[1] ©2015 IEEE [5]
[2] ©2016 IEEE [5]

Abstract

This thesis highlights the recent developments in Requirements Engineering for Software Product Line Engineering, with a focus on the use of ontology in interactive Requirements Elicitation and the existing techniques of ontology operations. Recent research done in Requirements Elicitation has been towards using ontologies as a modeling basis for gathering requirements. A new algorithm has been developed to allow ontologies to be combined at run-time when gathering the requirements of software clients. By harnessing knowledge in other ontologies, a more refined set of requirements can be generated. A scenario illustrating the use of ontology combination towards acquiring requirements for mobile platforms is also provided. The proposed method further enhances the capability of interactive software customization, thus helping to make Software Product Line Engineering a new practice in software development.

Dedication

This thesis is dedicated to my dear mother, sister, and late father for their endless love and support.

Acknowledgements

I would like to sincerely thank and express my deep sense of gratitude for my supervisor, Dr. Yuan. He has been very kind and patient throughout this endeavour. Without his clear and effective guidance, this work would not have been possible. I have been very fortunate to have a supervisor with a great intellect and vast knowledge, who cared so much about my work, and who responded to my questions and queries so promptly. He was consistently supportive and constantly encouraged me to improve and refine my work. He was always available to discuss any issues, big or small, I had during my research.

I also want to thank Dr. Goodwin and Dr. Azab. Their comments and suggestions have been invaluable and have greatly improved the quality of this work.

Table of Contents

List of Tables

List of Figures

List of Acronyms

DL Description Logic

JWNL Java WordNet Library

OWL Web Ontology Language

RE Requirements Elicitation

SOA Service Oriented Architecture

SPL Software Product Line

SPLE Software Product Line Engineering

List of Listings

Chapter 1

Introduction

Software Product Line Engineering (SPLE) is an active area in Software Engineering. It holds the promise of making software customization as successful as the assembly-line process in the automotive industry. By reducing bloat of unwanted code in software systems, customization increases efficiency. In the near future of mobile, wearable and embedded devices [58], the size of a software program takes on an important dimension. By utilizing customization, software modules can potentially be assembled and re-assembled quickly to target different platforms in a cost-effective manner. Considerable progress [51] has been made in recent years for realizing this paradigm of software development.

One of the main subdisciplines of Software Engineering, including SPLE, is Requirements Engineering. Requirements Elicitation (RE) forms an important part of the Requirements Engineering process. A lot of effort [28] [60] has been put in this area of research. From the early 1990s to the present, many techniques have been identified to reduce errors and make the elicitation process work more efficiently. Ontologies have been used to try and ensure that RE is accomplished in a well-defined manner which in turn, ensures a robust implementation of a software system. Considerable progress has been mode towards an interactive mode of RE for software customization [74]. Ontologies have been utilized for providing the foundation for such interactive systems.

They provide an excellent basis for representing concepts and the relationships between them. Due to this, they are being increasingly used across a variety of domains [45] [40].

The existing approach to interactive RE relies on using single ontologies to guide the interaction [73]. It would be more beneficial to harness knowledge available across multiple domains to dramatically improve the scope of interaction. Various operations on ontologies, such as merging, are design time operations and are thus not useful for an interactive system. Recently [73] with dialogue-based RE, an interactive way of gathering requirements has been made possible. This thesis proposes a novel method of Ontology Combinations. It is an approach to obtain knowledge in different ontologies when requirements elicitation is actually performed. The existing interactive approach uses a single ontology to drive the RE process of gathering requirements for building a customized Software Product Line (SPL) application. By bringing together different ontologies at run-time, this methodology promises to strengthen the interactive RE process and enhance it considerably. By combining multiple ontologies dynamically at run-time, a more detailed set of requirements can be obtained. This work defines the methodology for performing ontology combinations and presents a combine algorithm along with scenarios illustrating the approach. The contributions of this thesis are:

- An enhanced interactive RE process in which significantly more requirements are acquired from multiple ontologies through ontology combinations.
- Addition of mobile platform-dependent features to a customized SPL application by the use of ontology combinations.

Related work in the field of Software Customization, RE, Interactive RE and Ontologies is surveyed in Chapter 2. The thesis problem statement, along with the proposed method of Ontology Combination as the solution, is covered in Chapter 3. Chapter 4 covers the details of the implementation and goes over the experiments conducted with the proposed methodology. It also lists the contributions of this thesis. Chapter 5 concludes the thesis and points out some future directions of continuing research.

Chapter 2

Related Work

2.1 Overview

This chapter surveys the various topics pertinent to the thesis and a literature review of related work.

2.2 Software Customization

2.2.1 Overview

This section presents an overview of the research area of Software Customization. It highlights the area of inquiry in the context of background literature.

2.2.2 Software Product Line Engineering

Software Product Line Engineering (SPLE) is defined as a paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customizations [59]. It is divided into two areas: Domain Engineering and Application Engineering. Figure 2.1 summarizes the different processes involved in these two areas.

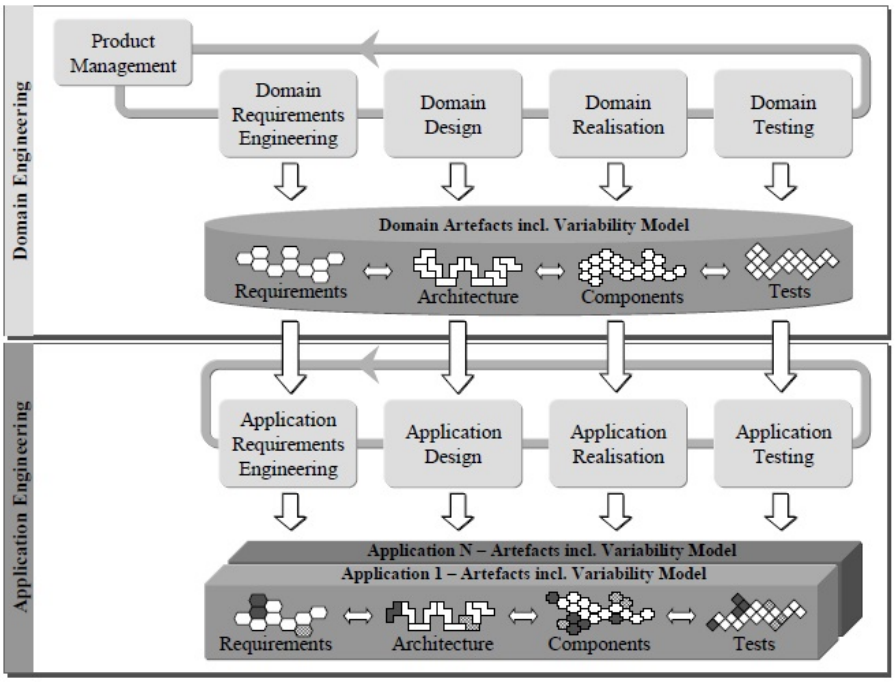


Figure 2.1: Software Product Line Engineering [59]

Both, Domain and Application Engineering, gather requirements for which some aspects of Requirements Engineering are needed. Domain Engineering is the process of SPLE in which the commonality and the variability of a SPL are defined and realised [59]. It is comprised of five sub-processes: Product Management, Domain Requirements Engineering, Domain Design, Domain Realisation, and Domain Testing. The Domain Requirements Engineering sub-process covers "all activities for eliciting and documenting the common and variable requirements of the product line" [59] whereas the Domain Design sub-process covers activities for defining the reference architecture [59].

Application Engineering is the process of SPLE in which applications of the SPL are built by reusing domain artefacts and exploiting the product line variability [59]. In contrast to Domain Engineering, one of the main goals of Application Engineering is to make use of the commonality and variability of a SPL to develop a customized product line application [59]. Application Engineering is comprised of four sub-processes: Application Requirements Engineering, Application Design, Application Realisation and Application

Testing. The Application Requirements Engineering sub-process contains activities that are needed for developing the application requirements specification [59].

Considerable research has been done in the field of SPLE in the past few years [51]. Integrating SPLE and Software Oriented Architecture (SOA) paradigms has also been an important focal point for researchers, more of which will be covered in a later section. Various open research challenges can be found for topics encompassing SPLE [51]. Software factory automation has been proposed [15], analogous to manufacturing factory automation, for managing reusable assets across distinct SPLs. This model is based on an architecture-driven meta-model which is customized to create applications directly. A systematic overview of research literature for product derivation in SPLE has also been done [60], where requirements are identified and validated for this purpose.

2.2.3 Service Oriented Architecture

Service Oriented Architecture (SOA) is a software model in which automation logic is decomposed into smaller, distinct units of logic [22]. These units are collectively used to create a larger piece of business automation logic. Figure 2.2 provides an overview of this model.

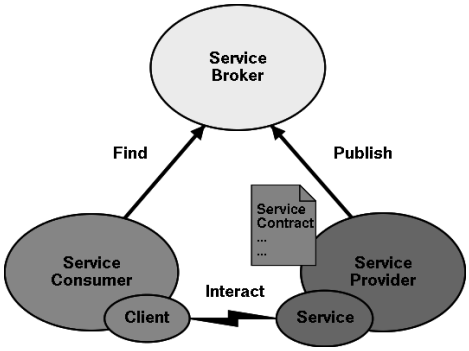


Figure 2.2: Service Oriented Architecture [22]

Services can assume different roles when involved in different scenarios [22]. The three main roles are, as shown in Figure 2.2, Service Broker, Service Consumer, and Service Provider. In the role of a Service Provider, a service exposes a public interface

through which it can be invoked by requestors of the service [22]. A Service Consumer is the sender of a service message requesting a specific service [22]. A Service Broker acts as a registry of services, and stores information about what services are available and who may use them. Universal Description, Discovery and Integration (UDDI) is an example of a Service Broker.

The core concept in SOA is that these units can be distributed. They don't need to reside on the same machine but can be spread across an intranet or even the Internet.

2.2.4 Integrating SPL and SOA

SPL and SOA integration is an active area of research. The various studies done in this combined field over the last decade have been surveyed [52]. The studies have been classified according to research focus, types of research and contribution, along with the various fields of ongoing research.

The concepts of SPL, SOA and component frameworks have been compared [32], concluding with the assertion that while there are differences between them, these concepts are in fact complementary to one another. An approach of a service-oriented architecture in which product lines are regarded as services which are then used to combine together into another, distinct product line has been presented [67]. A web product line to showcase this approach has also been provided there. An approach for reusing and combining services into service oriented product line applications has also been proposed [43]. Various issues such as identification of services are resolved by using feature-oriented product line engineering. Another method has been proposed [37] in which services and their level of granularity are identified by using ontologies in product lines. A way of grouping features and evaluating services, along with a case study, has also been provided there.

Developing SOA applications as SPLs has been attempted [49]. A combination of these two concepts is shown to provide advantages such as improved reuse and production of customized applications for specific clients. The issue of service identification for service-oriented product lines has been explored [36]. An approach has been de-

fined which bridges Feature Models (FMs) in SPLs and Business Process Models (BPMs) in SOAs by using a BPM workflow model to identify services.

A model using SOA architecture derived from current software artefacts has been defined [57]. There the focus has been on the reuse of these artefacts as SOA components and the derivation process that assembles products out of services automatically. This proposed approach has been implemented in the form of the Software Product Line Integration Tool (SPLIT) [56], which has been used to develop modular services obtained automatically from existing software artefacts. Then out of these services, products are assembled using a variability-driven derivation process.

2.3 Requirements Elicitation

2.3.1 Overview

This section goes over the relevant research work done in the field of Requirements Elicitation (RE). It also covers the use of Ontologies in RE.

2.3.2 Requirements Elicitation

Requirements Engineering is comprised of activites related to the development and agreement of the final set of Requirements Specifications [68]. The various processes in Requirements Engineering are outlined in Figure 2.3. The main processes used for a majority of projects are: Requirements Elicitation, Requirements Analysis and Requirements Specification. Other processes, such as Requirements Prototyping, are also done for projects where it is feasible to do so. Requirements Elicitation (RE) is defined as the process of discovering the requirements for a system by communicating with customers, system users, and others who have a stake in the development of the system [63]. It requires specific knowledge of the problem along with application domain and organizational knowledge. RE plays an important part in Requirements Engineering.

Traditionally, human communication has been the method of acquiring requirements

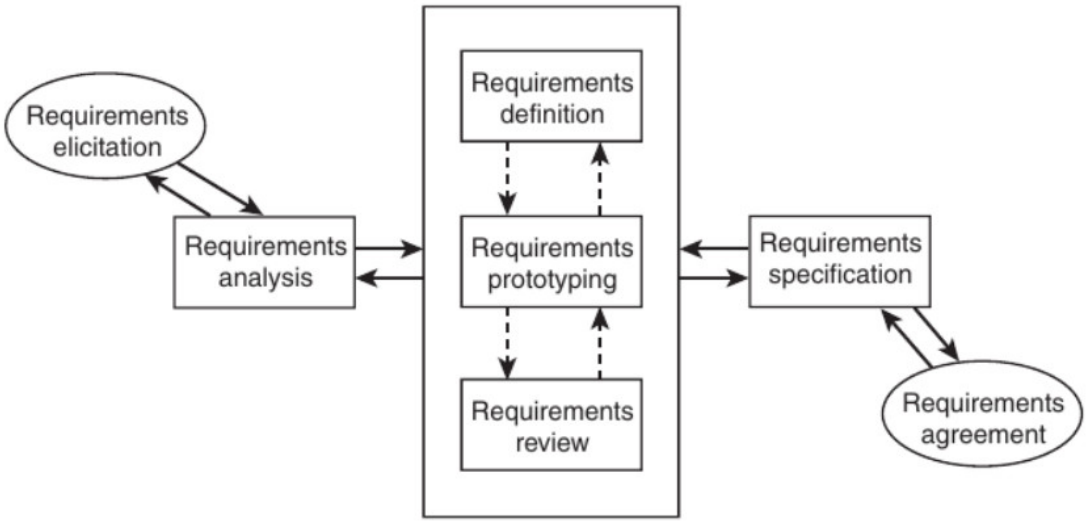


Figure 2.3: Requirements Engineering processes [68]

[47]. However, this mode of collecting requirements is ambiguous and a primary source of errors which leads to flawed and incomplete Requirements Specifications. Recognizing this, attempts have been made to use computer-assisted tools to gather requirements [46]. Extending this paradigm, a human-machine dialogue interface using natural language promises to reduce errors in the RE process.

In an early work [28], various approaches to obtain requirements were presented using insight gained from social science paradigms. A prototype automated SPL engineering environment has been presented which utilizes a product line repository [29]. Multiple-view models of SPLs were then used with a Knowledge Based RE Tool to derive a software product. An approach of interactive RE to build customized software based on a SPL has been presented recently [74]. An ontology model comprising of knowledge of common and variable assets has been developed, which is then used to obtain abstract requirements models for specific domains. A case study of an online book shopping system has also been incorporated into that study to illustrate the approach [74].

2.3.3 Ontologies in RE

Ontologies have been defined as "a formal, explicit specification of a shared conceptualization" [65]. They began to be used in Requirements Engineering in the early 1980s [20]. They were used in a variety of domains such as network management [45] and aerospace [40] [24].

Ontologies have been used for Requirements Analysis [35]. There, the incompleteness and inconsistency in a Requirements Specification was determined by using ontologies. The quality of a specification was measured along with predictions made about requirement changes.

Ontology-based reasoning method for RE has also been introduced [21]. Here, requirements were mapped to functions in domain ontology. Then reasoning was applied to check for errors and other potential requirements. Ontology-driven guidance has been used for RE [24]. Evaluation was done based on a domain ontology and a set of requirements. Further progress has been made in manipulating ontologies by combining them. Combinations make an effective use of knowledge encapsulated in different ontologies [71]. A methodology has been established to perform combinations for RE [71]. Ontology-based RE for software customization, in the context of SPLs, has been performed using an interactive approach [75] [74].

Ontologies have been developed for various Requirements Engineering processes using a university course registration web application system as a case study [62]. There, a model called OntoPersonalURM, which uses a multi-step iterative ontology development process, was created for Requirements Engineers. Ontology-based relation mining has been used for Cloud software requirements [34]. Ontologies have also been used for Requirements Specification verification and validation [17]. Similarly, an ontology of requirements has been used in transforming informal requirements into a formal specification [44].

Table 2.1 summarizes the research covered in this section.

Area	Ontology used for
Requirements Engineering	• Use of ontologies in RE began in early 1980s [20] • Network management [45] • Aerospace [40] [24] • University course registration web application [62] • Cloud software [34]
Requirements Elicitation	• Reasoning method [21] • Evaluation [24] • Software customization using an interactive approach for Software Product Lines [75] [73] [74] • Combining ontologies [71]
Requirements Analysis	• Quality of a Requirements Specification and requirement changes [35] • Domain knowledge and semantics [53]
Requirements Specification	• Verification and Validation [17] • Transforming informal requirements into a formal Requirements Specification [44]

Table 2.1: Summary of Ontologies in RE

2.4 Interactive Requirements Elicitation

Recently, significant progress has been made towards interactive RE using ontologies. An interactive machine-guided elicitation of requirements has been developed for the customization of a SPL for SOA based software [74]. An ontology-based requirements model has been developed [73], as shown in Figure 2.4. Three main concepts have been identified in the model - *Requirement*, *Function* and *Quality*. Other concepts have been included as extensions - *Softgoal*, *Rank* and *OtherInfo*. Seven relationships have been developed - *Generalize*, *Decompose*, *Rely*, *Contradict*, *Associate*, *HasRank*, and *Invalid*. A group of ontology rules has also been developed for RE and ontology instantiation to retrieve implicit knowledge of a product line [74]. A nine-step process has been outlined for instantiating a domain model of a service-oriented architecture of a family of software products.

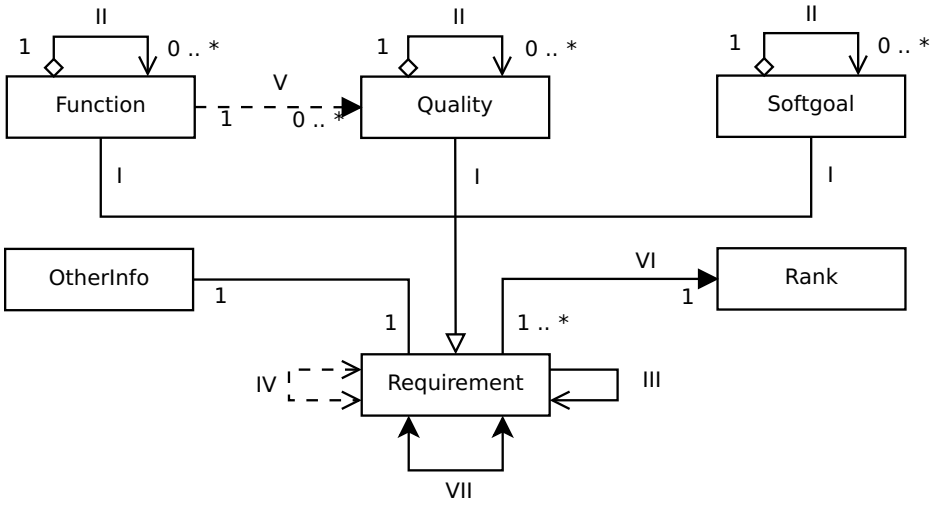


Figure 2.4: Ontology-based Requirement Model [73]

Here, RE is performed using a dialogue-based system. Ontologies are utilized for dialogue management. An ontology model [70] is used to manage dialogue interaction independently of domains [74]. In a related work, similar technique is applied to create customized software using conversational agents based on natural language interaction [73]. The algorithm for the RE process is shown in Figure 2.5 [73]. Complete

details of the evaluation process, along with additional algorithms cited in Figure 2.5, are described in that work [75].

```
1.  FOR each requirement R to be evaluated
2.      IF R is essential to the system THEN
3.          CALL PerformRequirementSelecting with R
4.      ELSE IF R is pre-selected THEN
5.          CALL PerformRequirementSelecting with R
6.      ELSE IF R is pre-dropped THEN
7.          CALL PerformRequirementDropping with R
8.      ELSE
9.          CALL evaluateRequirement with R
10.         IF R is to be selected THEN
11.             CALL PerformRequirementSelecting with R
12.         ELSE
13.             CALL PerformRequirementDropping with R
14.         END IF
15.     END IF
16. END FOR

17. PerformRequirementSelection with R
18.         CALL selectRequirement with R
19.         CALL preSelectRequirement with the requirements R relies on
20.         CALL preDropRequirement with the requirements R contradicts
21.         CALL preEvaluateRequirement with the requirements R decomposes into
22.         IF R is a function THEN
23.             CALL preEvaluateRequirement with the qualities R is associated with
24.         END IF

25. PerformRequirementDropping with R
26.         CALL dropRequirement with R
27.         CALL preDropRequirement with the requirements that relies on R
```

Figure 2.5: Pseudo code for requirement evaluation process [73]

After the final set of requirements have been obtained, their service descriptions are converted into an OWL-S ontology. Figure 2.6 shows the overview of the entire system [73].

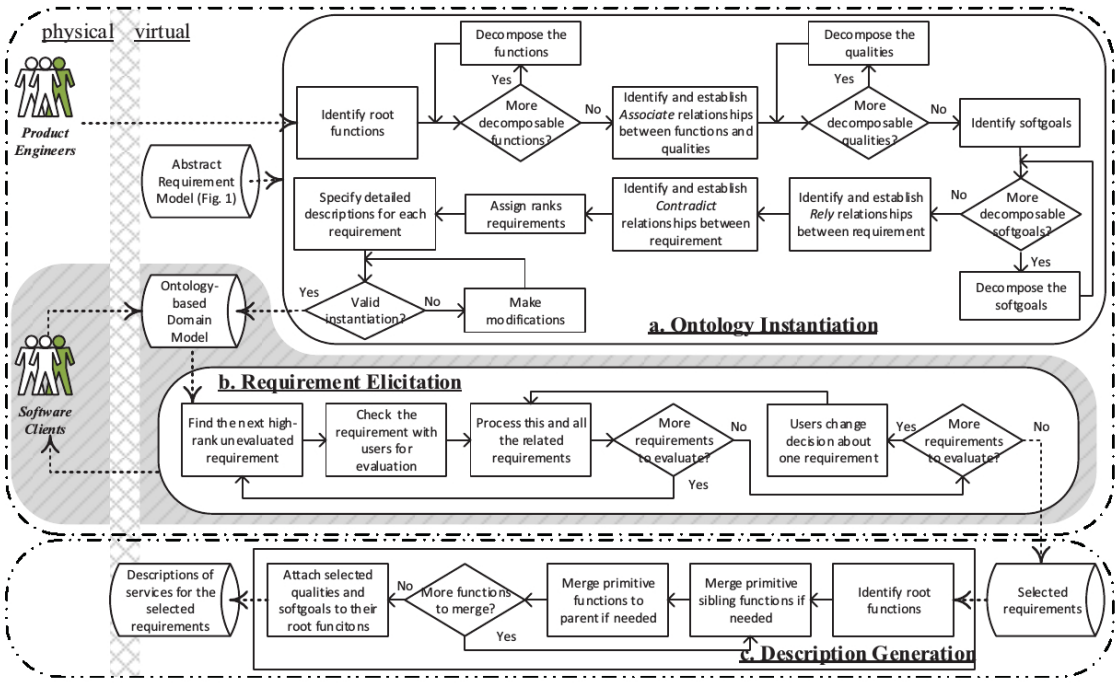


Figure 2.6: Interactive Requirements Elicitation system [73]

2.5 Ontology and Operations

2.5.1 Overview

This section covers the definition of Ontology as well as the various operations performed on them.

2.5.2 What is an Ontology?

As mentioned in Section 2.3.3, an Ontology is "a formal, explicit specification of a shared conceptualization" [65]. Ontologies enable knowledge sharing and reuse in a specific format. They have the advantage of being a formal and machine manipulable model of a domain of interest. Ontologies present a shared vocabulary in representing domain knowledge which allows reasoning to be performed.

Figure 2.7 shows an example of an ontology as a semantic network. Here, the ontology is modeled as Concepts and Relationships. Concepts can be abstract that represent intentions, beliefs, feelings etc., or they can be specific such as people, computers, tables, etc [30]. Relationships represent a type of association between Concepts of a domain [30]. In Figure 2.7, the Concepts are shown as ovals and the arrows designate the Relationships between the Concepts.

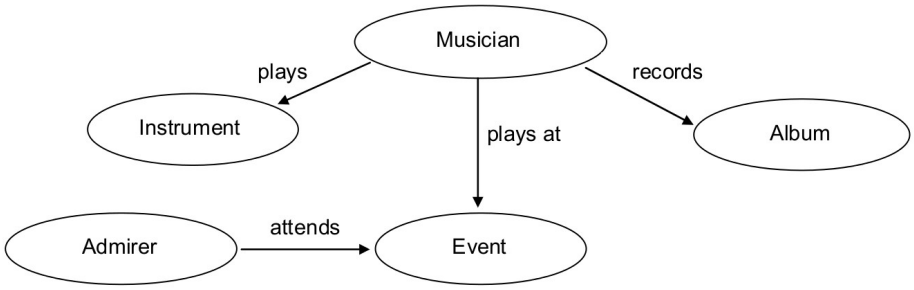


Figure 2.7: Ontology as a Semantic Network [27]

The same ontology is presented as a UML model in Figure 2.8. The boxes represent the Concepts and the lines between them represent Relationships. An excerpt of the

ontology represented in Web Ontology Language (OWL) format is shown in Figure 2.9. OWL is an ontology language for the Semantic Web with formally defined meaning.

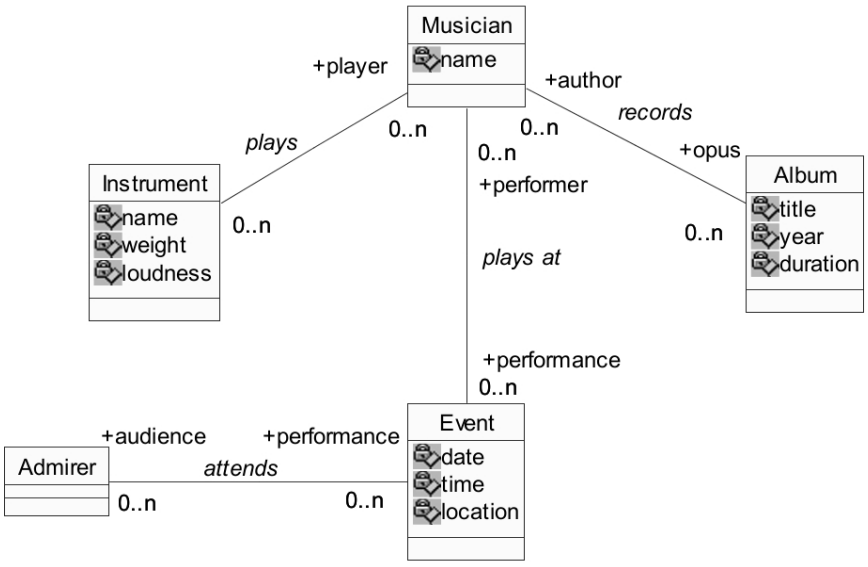


Figure 2.8: Ontology as a UML model [27]

```
<owl:Class rdf:ID="Event"/>
<owl:Class rdf:ID="Album"/>
<owl:Class rdf:ID="Instrument"/>
<owl:Class rdf:ID="Musician"/>
<owl:Class rdf:ID="Admirer"/>
<owl:ObjectProperty rdf:ID="author">
        <owl:inverseOf>
                <owl:ObjectProperty rdf:ID="opus"/>
        </owl:inverseOf>
        <rdfs:domain rdf:resource="#Album"/>
        <rdfs:range rdf:resource="#Musician"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="player">
        <rdfs:range rdf:resource="#Musician"/>
        <rdfs:domain rdf:resource="#Instrument"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="loudness">
        <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Instrument"/>
</owl:ObjectProperty>
…
```

Figure 2.9: Ontology represented in OWL (excerpt) [27]

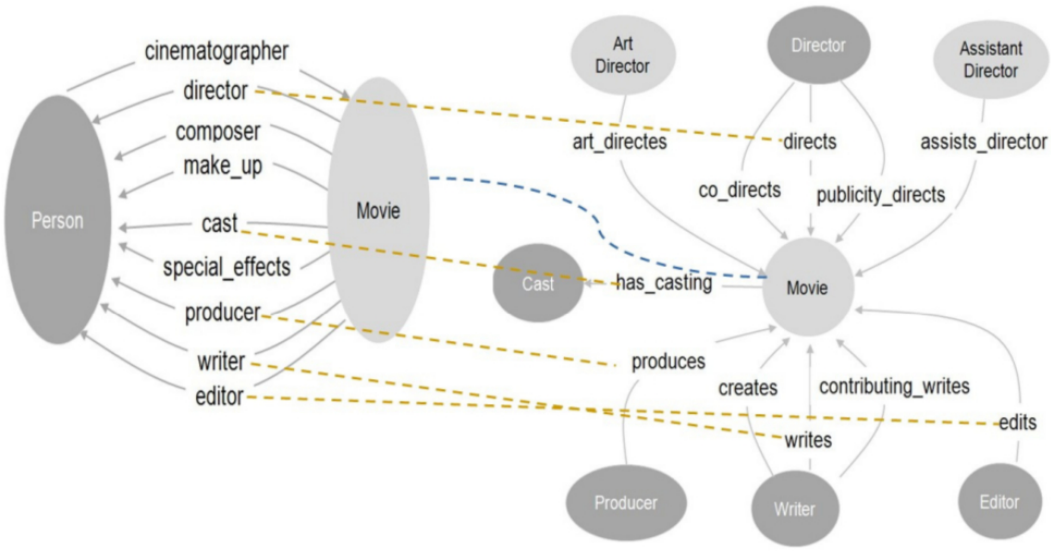
2.5.3 Operations on Ontologies

Over the previous decades of research, various operations on Ontologies have been identified [23]. They are: Matching, Alignment, Mapping, Integration and Merging.

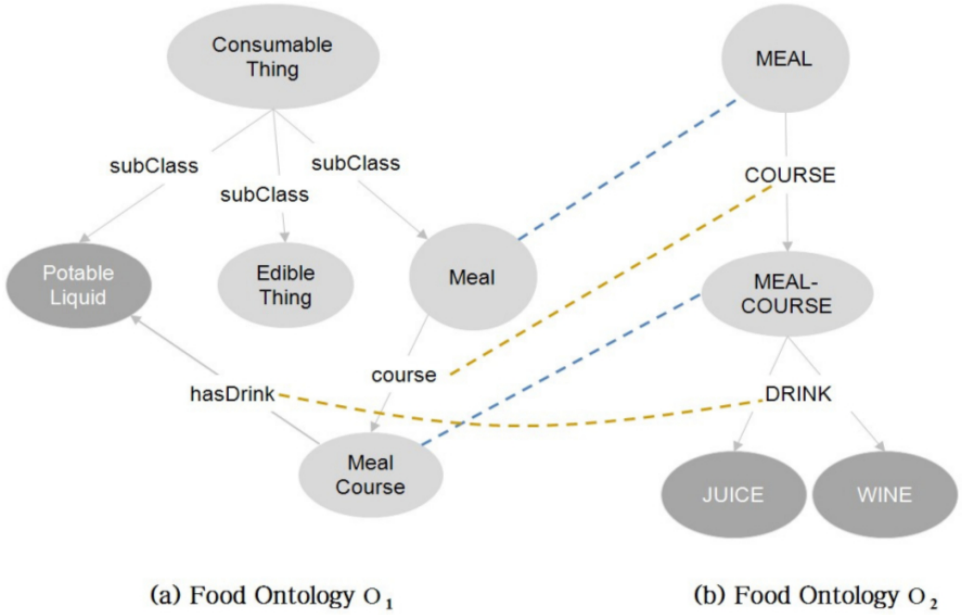
2.5.3.1 Matching

Matching is the process of finding relationships or correspondences between entities of different ontologies [23]. This area of research is becoming increasingly important for knowledge bases and the Semantic Web. Matching can be performed on Concepts, Attributes, and Relations of ontologies.

Figure 2.10 gives an example of Ontology Matching [42]. Figure 2.10a shows how concepts in the domain of the Motion Picture industry are represented in two different ontologies, O_1 and O_2. Relations within the two ontologies are also shown as arrows. The dotted lines represent the output of the Matching process. Similarly, in Figure 2.10b, ontologies O_1 and O_2 contain knowledge of the Food domain. Concepts and relations are matched between them and shown as dotted lines.



(a) Movie ontologies O_1, O_2 and matching results. Dotted lines mean a matching



(b) Food ontologies O_1, O_2 and matching results. Dotted lines mean a matching

Figure 2.10: Ontology Matching [42]

2.5.3.2 Alignment

Ontology Alignment is the process of bringing ontologies into agreement through the automatic discovery of mappings between related concepts [31]. It is a set of correspondences between two or more ontologies. The underlying principle in Alignment is that 'ontologies can approximate other ontologies and that ontologies to be matched are approximation of a common ideal ontology' [23].

An example of Alignment is given in Figure 2.11. The excerpt shown in this figure is the Alignment of two ontologies: the one on the left side is a fragment of the *Forest Fire Sensor* ontology and the one on the right side is a fragment of the *Fire Trucks Sensor* ontology. The dashed lines denote the Alignment obtained after applying an ontology alignment algorithm [25].

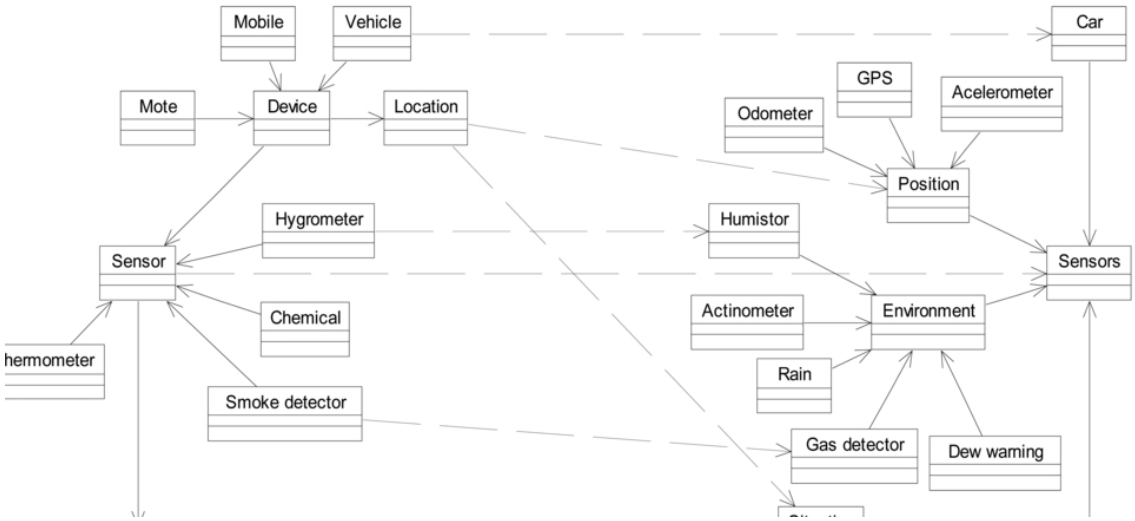
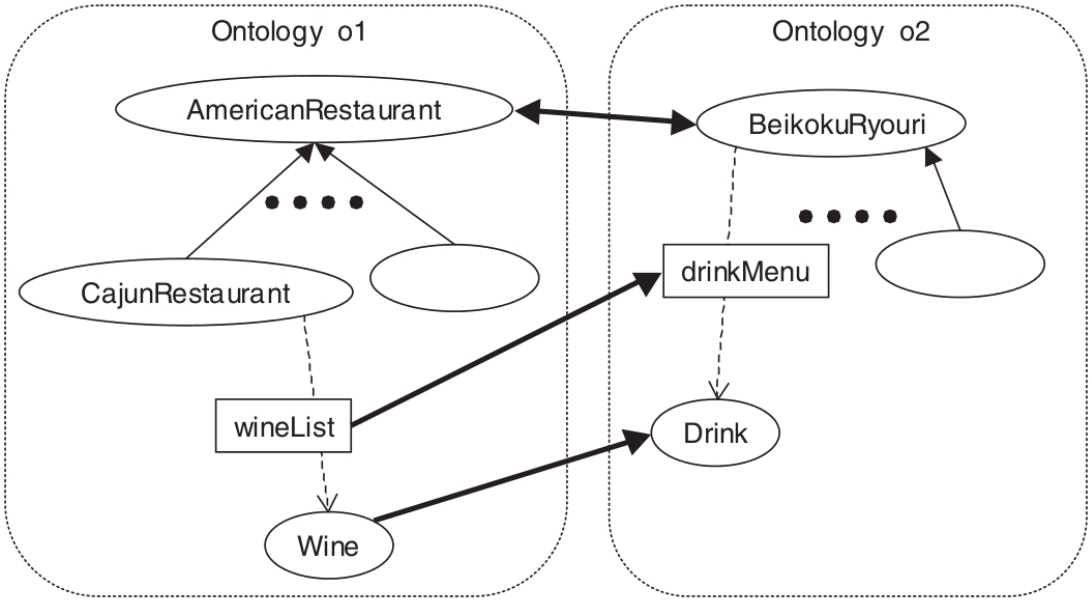


Figure 2.11: Partial view of an Ontology Alignment [25]

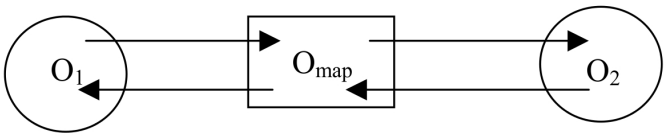
2.5.3.3 Mapping

Mapping is the oriented, or directed, version of the alignment which maps entities of one ontology to at most one entity of another ontology [23]. This can viewed as a collection of mapping rules oriented in a particular direction - from one ontology to another.

Figure 2.12 shows an example of Mapping. Both ontologies, *o1* and *o2*, represent knowledge in the Restaurant domain. Ontology *o1* encodes that knowledge in the context of American restaurants, whereas ontology *o2* does this in a Japanese context. The bold arrows represent the map generated between the two ontologies. Figure 2.12b presents an abstract view of the Mapping.



(a) Ontologies o1 and o2 with their mapping as bold arrows [14]



(b) 'Approximate ontology translation' for the ontology mapping [38]

Figure 2.12: Ontology Mapping

2.5.3.4 Integration

Integration is the inclusion in one ontology of another ontology [23]. The integrated ontology contains the knowledge of the original ontologies. Integration is different from Merging as one of the ontologies is modified whereas Merging creates a new ontology.

An example of Integration is given in Figure 2.13. *A* and *B*, are the initial ontologies. Integration results in *B* being 'absorbed' into *A*.

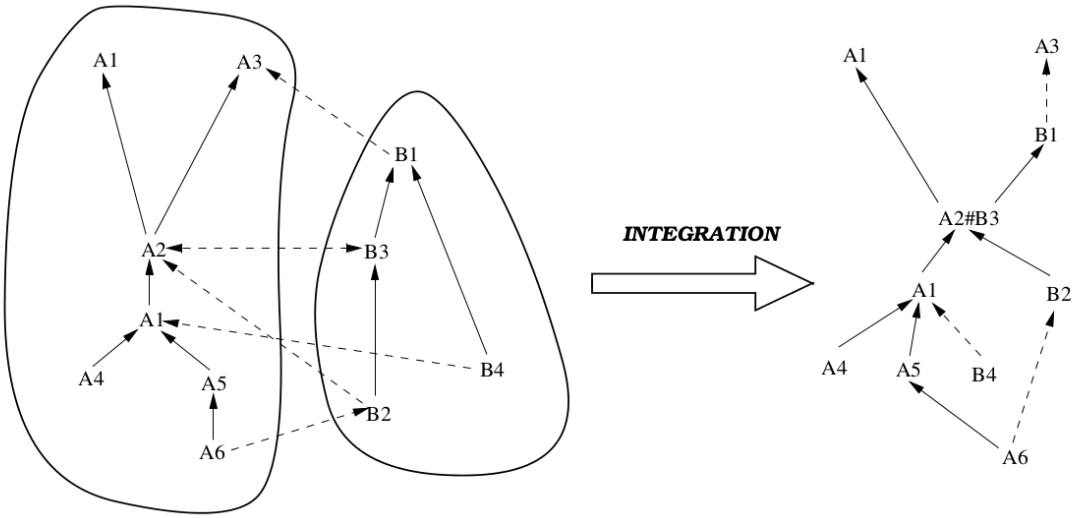


Figure 2.13: Ontology Integration [50]

2.5.3.5 Merging

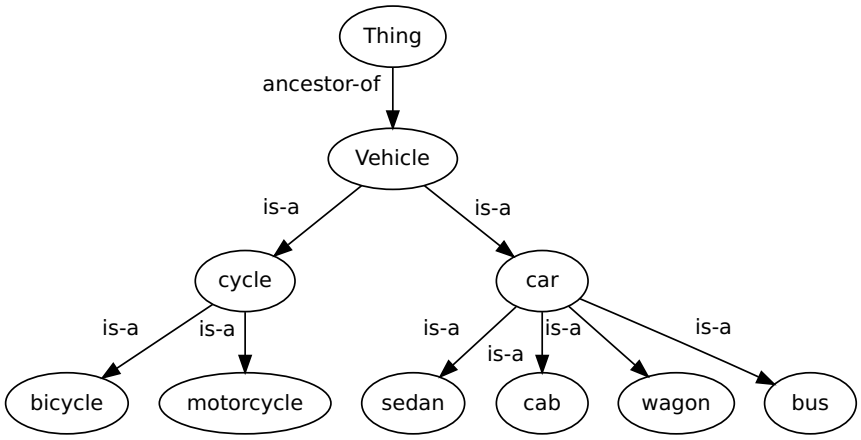
Merging is the creation of a new ontology from two, possibly overlapping, source ontologies [23]. The initial ontologies are not modified, with the new ontology incorporating the knowledge of both the ontologies.

Figure 2.14 shows an example of Merging. *Sample ontology 1* and *Sample ontology 2* consist of information about the domain of Cars. A third ontology generated after the Merging, as shown in Figure 2.14c, contains the knowledge of both *Sample ontology 1* and *Sample ontology 2* as a single ontology.

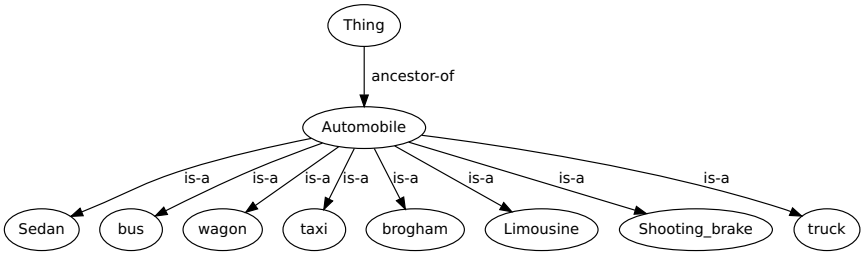
The ideas behind Ontology Merging can be traced back to the beginning of 1980s [18]. The SMART algorithm was an early semi-automatic approach to Ontology Merging and Alignment [54] . The PROMPT algorithm was an improvement of SMART and during its development various Ontology Merging operations were identified [55].

Mathematical frameworks have been applied to Ontology Merging. Merging has been done using Formal Concept Analysis (FCA-MERGE) [66]. Also, Category theory [33] has been applied towards merging and Simple PushOut (SPO) in algebraic graph transformation [48] has been used to merge ontologies. Description Logic (DL) based merging of Concrete and Fuzzy ontologies has also been accomplished [41].

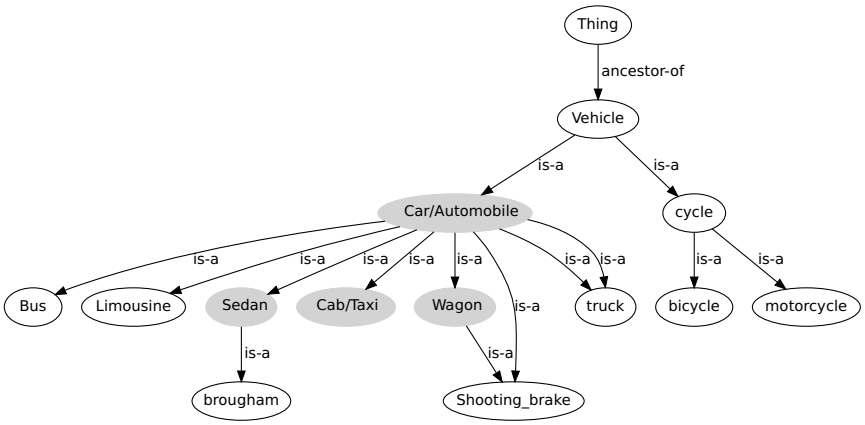
An ontology integration process has been proposed in which two ontologies are merged by generating an ontology intersection containing the maximum number of entities contained in the input ontologies and their corresponding non-contradictory axioms [69]. CODE [26] is a fully automated system that aims at preserving the source ontology knowledge. It uses natural language processing in combination with a semantic matching approach, along with scenario-based rules to make sure the merging process is accurate. While being very comprehensive, CODE is a holistic process - taking into account all aspects of the source ontologies including Class, Property and Instance. While this is powerful, it is not useful for a more lightweight approach where only the Classes of given ontologies need to be analyzed. It is a quite involved and cumbersome process - going through multiple stages to acquire a merge, and would be difficult to adapt to a nimble setting where quick operations are required.



(a) *Sample Ontology 1* about Car



(b) *Sample Ontology 2* about Car



(c) Merged Ontology

Figure 2.14: Ontology Merging [39]

Recently, Cloud-based ontology matching has been provided as a Service for integration and interoperability resolution primarily focused on biomedical systems [16]. A novel approach, but as the system has been built for a distributed architecture of a cloud, it would be difficult to extract and incorporate the technique for a more restrictive environment, such as a traditional, localized desktop system.

Also, ATOM base algorithm has been proposed that takes two ontologies and merges them using an equivalence mapping [61]. A very clear and consistent terminology is presented for the ATOM algorithm and lays down the foundations for developing similar algorithms. Equivalence mapping is clearly defined and applied. A major drawback of the algorithm is that it is limited to an IS-A relationship; it does not take into account other possible relationships.

Table 2.2 presents the various operations and approaches described in the earlier sections. This table compares the various works explored earlier on the basis of whether user intervention is required, the type of relationship that is being used in the work (if explicitly stated) and in the last column of the table, if the work is based on a Design-Time or Run-Time approach.

The following chapter provides further discussion.

Work	Automated	Relationship	Place
[Noy and Musen, 1999] (SMART) [54]	✗	-	Design-Time
[Noy and Musen, 2000] (PROMPT) [55]	✗	-	Design-Time
[Stumme and Maedche, 2001] (FCA-MERGE) [66]	✗	-	Design-Time
[Hitzler et al., 2005] (Category theory) [33]	✗	-	-
[Raunich and Rahm, 2011] (ATOM) [61]	✓	IS-A	Design-Time
[Fudholi et al., 2014] (CODE) [26]	✓	-	Design-Time
[Mahfoudh et al., 2014] (Algebraic SPO) [48]	✓	IS-A	Design-Time
[Wu, 2014] (CODE) [26]	✗	-	Design-Time
[Amin et al., 2015] (Cloud-based) [16]	✗	-	Design-Time
[Kumar and Harding, 2015] (Description Logic) [41]	✗	IS-A TYPE-OF PART-OF	Design-Time

Table 2.2: Comparison of Ontology Operations

Chapter 3

A New Method of Ontology Combination

3.1 Overview

This chapter covers the problem statement of this thesis and the proposed methodology as the solution. The specific details of the methodology are developed thoroughly. The corresponding Combine algorithm is covered comprehensively in the later sections.

3.2 Problem Statement

The interactive approach for RE, outlined in Section 2.4 above, uses a single ontology for modeling the domain requirements [73]. This can be further enhanced by acquiring knowledge from other domains. Multiple ontologies can be brought together for this purpose, enabling the approach to acquire additional knowledge from diverse domains.

Various operations on ontologies were covered in Section 2.5.3. Among these approaches, mathematical frameworks [66] [33] [48] require formulating the ontologies into mathematical structures such as lattices and fuzzy structures. This requires an extra 'overhead' operation, which is expensive in terms of the additional time that is

required. While bringing mathematical precision to the merging process, the present mathematical approaches lose flexibility and would fall short in performance during a dynamic use of such approaches.

All semi-automatic approaches [54] [55] [66] [33] [69] [16] [41] require human intervention at important stages of the merging process. Dynamic combination of ontologies, needed during run-time, should not require any human intervention. Any technique requiring user decisions during the merge process would defeat the purpose of interactive RE as the focus of the user needs to be on gathering requirements rather than merging ontologies. Also, in order to merge ontologies in this manner, the user will need to have an in-depth knowledge of the merging process. For the purpose of a software customization system, this should not be required and this, ideally, should be transparent to the user. The user should not be required to know about ontologies or of the process of merging ontologies; this should be taken care of in the background of the interactive system without involving the user.

Furthermore, all of them are design-time operations. As such, they cannot be applied to an interactive mode of knowledge extraction.

This thesis proposes a dynamic run-time operation of Ontology Combinations which, by overcoming these limitations, can enhance interactive RE immensely.

3.3 Ontology Combination

As outlined in Section 2.5, existing approaches are mainly focused on ‘deep’ merges of ontologies. Classes, relations, etc (some or all attributes of ontologies) are sought to be merged. RE does not need this as Requirements Artefacts are usually discrete items brought together to form a new system. An artefact is usually defined as a specification of a physical piece of information that is used or produced by a software development process [64]. RE needs ontology *combinations* so that new Requirements Specifications can be generated quickly from different ontologies. Reasoning should be relatively quick and ideally should have a minimal overhead in generating a combined ontology as the

entire structure of an ontology does not need to be merged.

Ontology Combination is similar to Ontology Merging but not the same operation. It is different as ontologies being combined together might not share any ideas except for the need of creating a joined ontology that might serve an entirely different purpose from that of the original two ontologies.

3.4 Example

A general example can be used to demonstrate Ontology Combinations. Figure 3.1 shows a *Pizza* ontology. It has, among other concepts, a *Food* concept. This concept is further extended to concepts like *Pizza*, *IceCream*, *PizzaTopping* etc. Now, another ontology can also contain information about food items such as the *Food* ontology shown in Figure 3.2. This ontology contains a concept *EdibleThing*. *EdibleThing* is refined into different types of consumable items such as *Dessert*, *SweetFruit* etc. If both these ontologies are combined, then a reasoning system based on the *Pizza* ontology can take advantage of the knowledge available in the *Food* ontology. A good instance of this would be in creating new and unexpected pizza topping combinations for ordering Pizzas like a topping of *SweetFruit* on a *SpicyPizza*. Figure 3.3 shows one possible pair of nodes to achieve this desired combination.

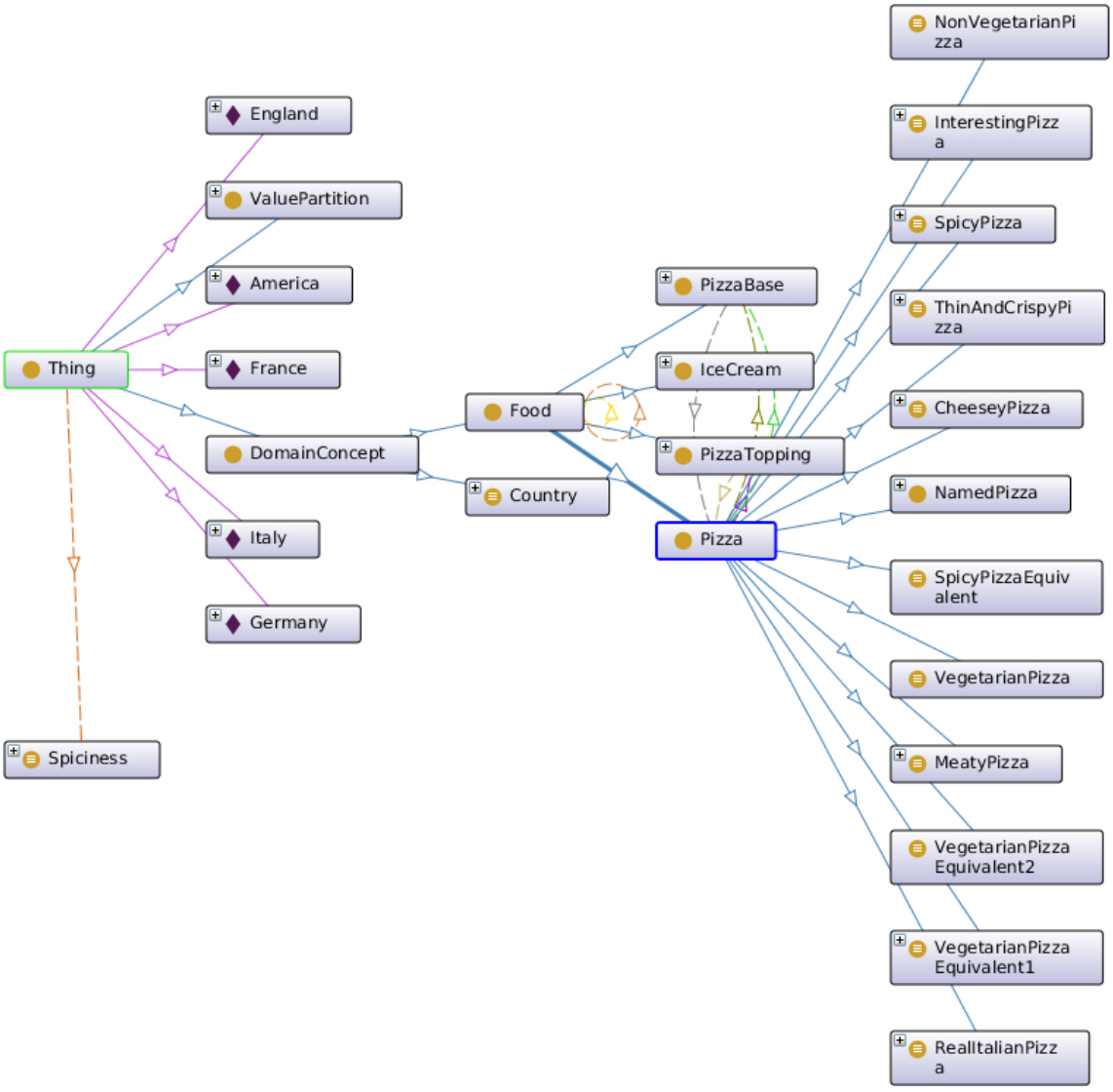


Figure 3.1: *Pizza* ontology [10]

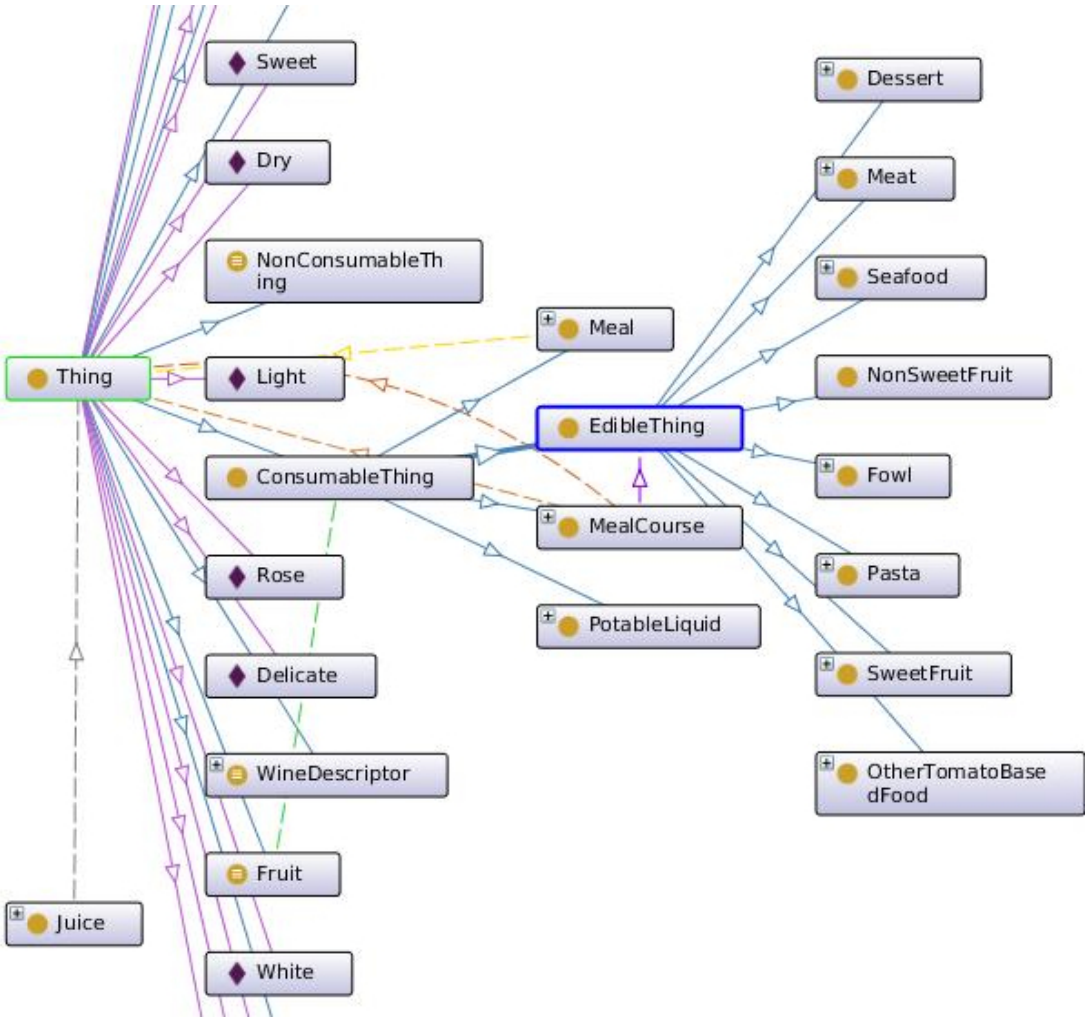


Figure 3.2: *Food* ontology [4]

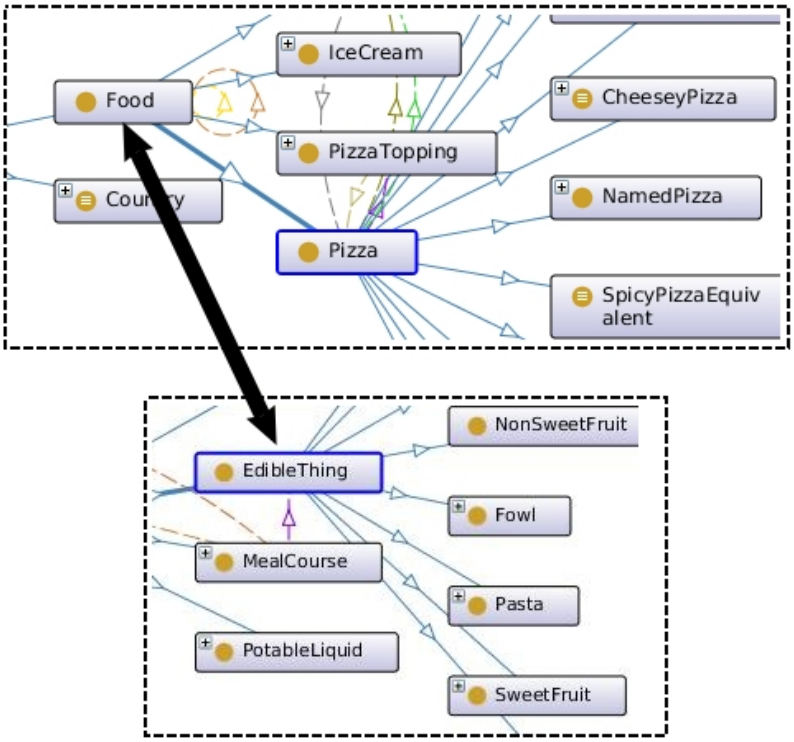


Figure 3.3: Excerpt of the Combined Ontology

3.5 Proposed Methodology

The methodology of performing an Ontology Combination is detailed in this section. Terminology used is presented in Section 3.5.1 Definitions. Then the steps are delineated. The summary is shown in Figure 3.4.

3.5.1 Definitions

Many similar notations for Ontologies exist in research literature. For this discussion, an ontology is defined as a tuple:

$$O_i = (C_i, R_i, I_i, A_i)$$

where

C_i is the set of concepts,

R_i is the set of relationships between the concepts,

I_i is the set of instances,

A_i is the set of axioms.

K_{RE} is a RE Knowledge Base holding ontologies and their instances specialized towards the acquisition of requirements. R_i and A_i are specific to O_i and I_i is part of K_{RE} and therefore will not be considered here.

Instances of ontologies will be assumed to honor the ontological evaluations after the combination. It is assumed that all the ontologies are consistent before the beginning of the combination process. Only concepts, C_i, are needed for the combination and will be analyzed here.

For the sake of brevity and simplicity, a combination of only two ontologies will be delineated here. Starting with primary ontology $O_p = (C_p, R_p, I_p, A_p)$, a secondary ontology $O_s = (C_s, R_s, I_s, A_s)$, is selected from K_{RE}. The steps taken to combine them are explained in the following sections.

3.5.2 Step 1: Generate Correspondences

A set of *concept correspondences* is defined as the set of (match) mapping between two ontologies [61]. Given two concepts, $p \in C_p$ and $s \in C_s$, a concept correspondence t, is defined as an ordered pair (p,s) of a primary ontology concept p and a secondary ontology concept s [61]. Each t is characterized by a type selected from *equivalence*, *is-a* and *inverse-isa* [61]. An *equivalence correspondence* is defined as a correspondence where p and s represent the same concept; an *is-a correspondence* is defined as a correspondence where p is a subclass of s and an *inverse-isa correspondence* is defined as a correspondence where s is a subclass of p [61]. An *is-a correspondence* is an oriented correspondence from a source concept to a target concept and expresses an *is-a* relationship between them [61]. An *inverse-isa correspondence* is similarly defined as the source concept being a 'superclass' of the target concept [61]. Here, a set of concept correspondences, T, will be used to identify the concepts in primary and secondary ontologies. T will be used to generate the links between them in *Step 2*.

3.5.3 Step 2: Generate Relationships

On the basis of the type of correspondences, relationships can be generated for the links that tie the ontologies together.

3.5.4 Step 3: Check consistency of combined ontology, O_c

The combined ontology, O_c, obtained after *Step 2*, will then be checked for consistency using a suitable reasoner.

3.5.5 Step 4: Validation of O_c

A simple reasoning test can be performed to ensure that the link produced is valid and produces sensible results.

Ontology-based requirements elicitation can be then be carried out [75] using the combined ontology O_c. After this, if a suitable set of requirements has not yet been obtained, this process can be iterated over again. The above steps can be iterated over as many times as needed until a satisfactory set of requirements is gathered.

As the use of this methodology matures, existing ontologies can be modified and newer ones can be added to K_{RE}. Over time, such a methodology would yield a mature collection of ontologies which would help in refining requirements even further, leading to a less ambiguous and a detailed set of Requirements Deliverables.

The next section, covering the Combine algorithm, gives the details of the algorithms involved in this methodology.

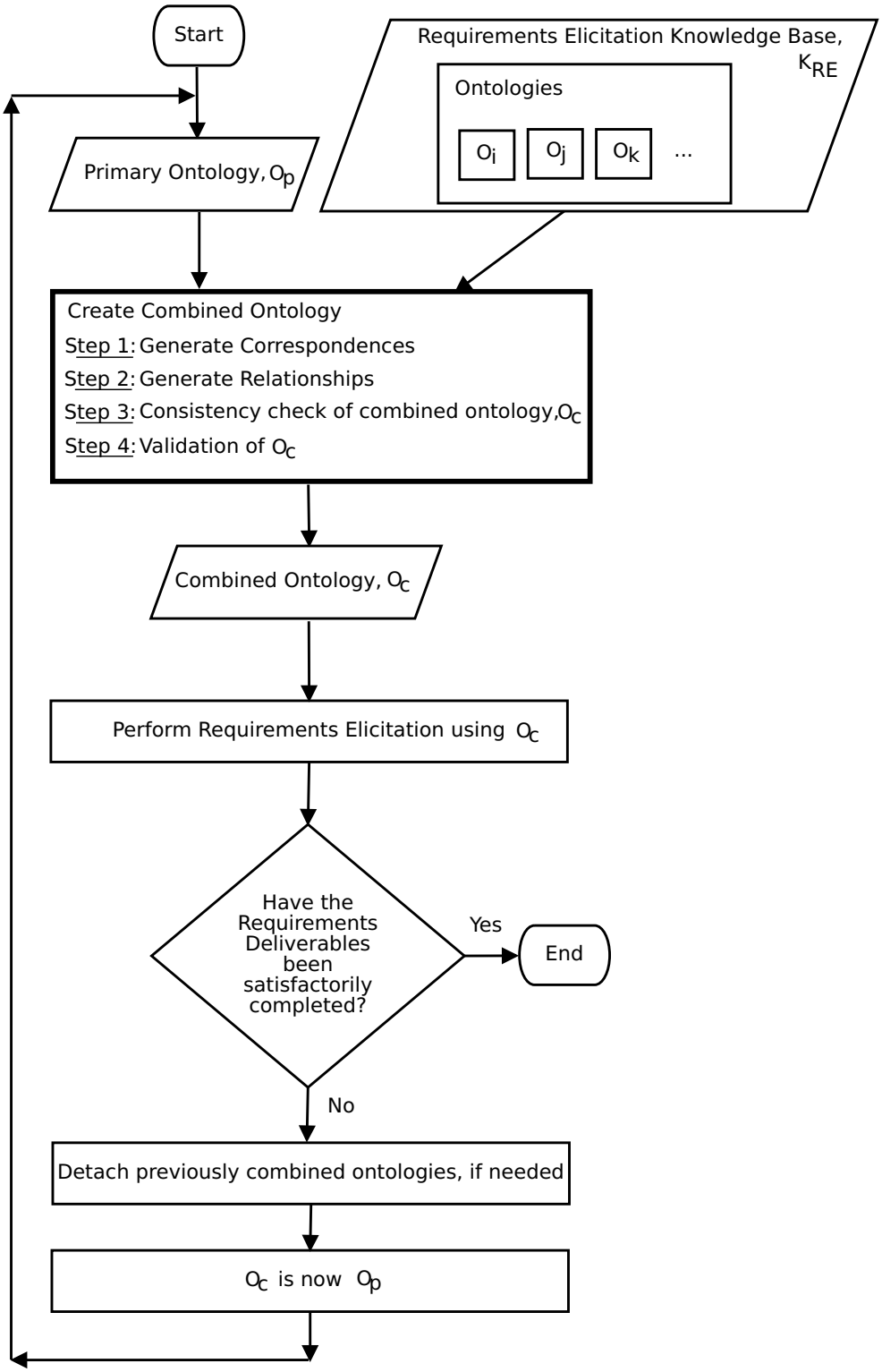


Figure 3.4: Methodology

3.6 Design of Algorithms

3.6.1 Overview

The Combine algorithm is called during the process of RE for a SPL, represented by the ontology, O_{SPL}. Another ontology, O_i, from the RE Knowledge Base, K_{RE}, is given as input to the algorithm to perform the combination.

The algorithm uses the following:

- WordNet [13] - is a lexical database for the English language, often described as a combination of a dictionary and thesaurus.
- Java WordNet Library (JWNL) [7] - is a free and open-source Java API for accessing WordNet.
- Apache Lucene [2] - is a free and open-source information retrieval Java library.
- SimMetrics [12] - is a free and open-source Java library of similarity and distance metrics for strings.

The Combine algorithm is composed of smaller algorithms - SelectLink, GetCorrespondences, GetRelationship, FindRelationshipJWNL and GetHighestCM - all of which are described in this section. The SelectLink algorithm is called initially with string, $strSPLLeafNode$ of the leaf node, V_l in O_{SPL} and the ontology to be combined, O_i. The algorithm then calls the GetCorrespondences algorithm to get *Correspondences*, if they exist, between $strLeafNode$ and any node in O_i. The GetCorrespondences algorithm in turn, calls the GetRelationship algorithm which tries to find the relationships (IS-A, TYPE-OF, and PART-OF). It does this through the use of the FindRelationshipJWNL algorithm, which uses the JWNL API for WordNet. The GetHighestCM algorithm is used to determine the *Correspondence* with the highest *Confidence Measure* in a given set of *Correspondences*. The SelectLink algorithm returns a *Correspondence* which is used to link O_{SPL} and O_i together. This enables the two ontologies to be linked together dynamically, resulting in a combined ontology. The combined ontology resides in memory. The algo-

rithm can be called as many times as needed to combine other ontologies in K_{RE} with the main O_{SPL} ontology.

3.6.2 SelectLink **algorithm**

The SelectLink algorithm is shown below:

Algorithm 1 SelectLink

Input: $strSPLLeafNode$ is string for the concept of the leaf node V_l in O_{SPL}
Input: O_i is ontology that is to be combined
Output: $cHighest$ is a *Correspondence* that will link O_{SPL} and O_i together

1: $c \leftarrow \emptyset$ ▷ set of *Correspondences*

2: **for each** $node \in O_i$ **do**
3: $c \leftarrow c \cup$ GetCorrespondences($strSPLLeafNode$, $node.label$)
4: **end for**

5: $cHighest \leftarrow$ GetHighestCM(c)

6: return $cHighest$

The input for the SelectLink algorithm is the string for the concept of the leaf node V_l in O_{SPL}, $strSPLLeafNode$, and the ontology that is to be combined, O_i. It returns as output, a *Correspondence* c, which contains the node in O_i and the relationship that will link O_{SPL} and O_i together.

The loop in Line 2 to Line 4 iterates over all the nodes in O_i to find out if there is a correspondence between the node and $strSPLLeafNode$. This is done by calling the GetCorrespondences algorithm (Section 3.6.3). All correspondences are collected into the set of *Correspondences*, c. The *Correspondence* with the highest *Confidence Measure* is selected by calling the GetHighestCM algorithm in Line 5. *Confidence Measure* is described in GetHighestCM algorithm section (Section 3.6.6). This *Correspondence* is then returned in Line 6 as the link between O_{SPL} and O_i.

3.6.3 GetCorrespondences **algorithm**

The GetCorrespondences algorithm is shown below:

Algorithm 2 GetCorrespondences

Input: $strSPL$ is string for the concept of the leaf node V_l in O_{SPL}
Input: $strI$ is the string for the concept from O_i
Output: c is a *Correspondence*

```
1: if strSPL = strI then                                          ▷ same string
2:     c.Relationship ← IS-A
3:     c.CM = 0
4:     return c
5: end if

6: tokensSPL ← ∅
7: tokensI ← ∅
8: cw ← ∅

9: strSPL ← Lucene.StopWordsFilter(strSPL)
10: tokensSPL ← Lucene.Tokenize(strSPL)

11: strI ← Lucene.StopWordsFilter(strI)
12: tokensI ← Lucene.Tokenize(strI)

13: for each wSPL ∈ tokensSPL do                  ▷ first pass - try to find Anchor Word
14:     for each wI ∈ tokensI do
15:         if 0.0 ≤ SimMetrics.JaroWinkler(wSPL, wI) ≤ 0.1 then
16:             wAnchorWord ← shortestOf(wSPL, wI)
17:             if sizeof(tokensSPL) = sizeof(tokensI) = 1 then       ▷ only one word
18:                 c.Relationship ← IS-A
19:                 return c
20:             end if
21:         end if
22:     end for
23: end for

24: for each wSPL ∈ tokensSPL do                                  ▷ second pass
25:     for each wI ∈ tokensI do
26:         cw ← cw ∪ GetRelationship(wSPL, wI)
27:     end for
28: end for

29: cwHighest ← GetHighestCM(cw)

30: if (cw.size() ≤ 1) and (cwHighest.RelationshipFound = true) then
31:     c = cwHighest
32: else
```

```
33:     if wAnchorWord then
34:         if strSPL.length > strI.length then                ▷ more 'generic' concept
35:             c.Relationship ← IS-A
36:             c.CM = 0
37:         else if word before wAnchorWord in strSPL = word before wAnchorWord in strI =
    VERB) then
38:             c.Relationship ← PART-OF
39:             c.CM = 0
40:         end if
41:     else                                                   ▷ No relationship has been found
42:         c.Relationship = NULL
43:     end if
44: end if

45: return c
```

The GetCorrespondences algorithm tries to determine the correspondence between two concepts, $strSPL$ from O_{SPL} and $strI$ from O_i, through two levels of matches:

- String Level Match is done on the entire two strings to figure out the relationship between $strSPL$ and $strI$. Line 1 to Line 5 and Line 13 to Line 23 are the String Level Matching parts in the algorithm.
- Word Level Match is done on individual words of the strings to figure the relationship between them. Line 24 to Line 28 are the Word Level Matching parts in the algorithm.

The input to the GetCorrespondences algorithm are $strSPL$ which is the string for the concept of the leaf node V_l in O_{SPL} and $strI$ which is the string for the concept from O_i. The output of the algorithm is a *Correspondence* which contains the relationship between the two input strings and the *Confidence Measure* for that relationship.

A check is done in Line 1 to see if the two strings are equal and if they are, then the relationship is considered of type IS-A and the algorithm returns this as a *Correspondence* with *Confidence Measure* of 0. From Line 9 to Line 12, the Apache Lucene library is used to filter for stop words and to tokenize the input strings. The first pass through the two strings, from Line 13 to Line 23, tries to find an *AnchorWord*. An *AnchorWord* is used to determine a 'core' concept between the two strings. This is determined by

utilizing the Jaro-Winkler distance implemented in the SimMetrics library. If the words are similar and there is only one word in both the strings, then the relationship is of type IS-A and the correspondence has been found.

If the relationship has not been found yet, then a second pass is made through the string tokens, from Line 24 to Line 28. Each word in $tokensSPL$ is compared with each word in $tokensI$ to determine the relationship between them by calling the GetRelationship algorithm. All the *Correspondences* obtained are put into the set of *Correspondences*, cw. The *Correspondence* with the highest *Confidence Measure* is selected by calling the GetHighestCM algorithm in Line 29.

If no relationship has yet been found, then a relationship is sought based on the $wAnchorWord$ using the logic between Line 32 to Line 44. If $strSPL$ is longer than $strI$, then $strI$ represents a more 'general' concept and thus the relationship is of type IS-A. On the other hand, if the words before the $wAnchorWord$ - both in $strSPL$ and $strI$ are 'verbs' (for example, 'Select book name' and 'Find book author' - here 'Select' and 'Find' are verbs), then the relationship is assumed to be of type PART-OF.

If no relationship has been found yet and there was no $wAnchorWord$, then no relationship has been found and the algorithm returns a $NULL$ relationship.

3.6.4 GetRelationship **algorithm**

The GetRelationship algorithm is shown below:

Algorithm 3 GetRelationship

Input: $word1$ and $word2$ are strings
Output: cw is a correspondence between the two words

```
1:  word1POS ← ∅
2:  word2POS ← ∅
3:  cwAll ← ∅

4:  JWNL.Initialize()

5:  word1POS ← JWNL.GetPOS()          ▷ get all Parts-of-Speech for word1
6:  word2POS ← JWNL.GetPOS()          ▷ get all Parts-of-Speech for word2

7:  for each p1 ∈ word1POS do
8:      for each p2 ∈ word2POS do
9:          if p1 = p2 then
10:             r ← FindRelationshipJWNL(SYNONYM)
11:             if r ≠ NULL then
12:                 cw.Relationship ← IS-A
13:                 cw.ConfidenceMeasure ← r.Depth
14:                 cwAll ← cwAll ∪ cw
15:             end if

16:             r ← FindRelationshipJWNL(HYPERNYM)
17:             if r ≠ NULL then
18:                 cw.Relationship ← TYPE-OF
19:                 cw.ConfidenceMeasure ← r.Depth
20:                 cwAll ← cwAll ∪ cw
21:             end if

22:             r ← FindRelationshipJWNL(HYPONYM)
23:             if r ≠ NULL then
24:                 cw.Relationship ← TYPE-OF
25:                 cw.ConfidenceMeasure ← r.Depth
26:                 cwAll ← cwAll ∪ cw
27:             end if

28:             if p1 = p2 = VERB then
29:                 r ← FindRelationshipJWNL(TROPONYM)
30:                 if r ≠ NULL then
31:                     cw.Relationship ← TYPE-OF
32:                     cw.ConfidenceMeasure ← r.Depth
33:                     cwAll ← cwAll ∪ cw
34:                 end if
```

```
35:             end if

36:             r ← FindRelationshipJWNL(HOLONYM)
37:             if r ≠ NULL then
38:                 cw.Relationship ← PART-OF
39:                 cw.ConfidenceMeasure ← r.Depth
40:                 cwAll ← cwAll ∪ cw
41:             end if

42:             r ← FindRelationshipJWNL(MERONYM)
43:             if r ≠ NULL then
44:                 cw.Relationship ← PART-OF
45:                 cw.ConfidenceMeasure ← r.Depth
46:                 cwAll ← cwAll ∪ cw
47:             end if
48:         end if
49:     end for
50: end for

51: cwHighest ← GetHighestCM(cwAll)

52: return cwHighest
```

The GetRelationship algorithm takes as input two strings, $word1$ and $word2$ and returns as output a *Correspondence Word*, cw which contains the relationship between the two words and the corresponding *Confidence Measure*.

The JWNL library is initialized in Line 4. In Line 5 and Line 6, all the possible Parts-Of-Speech values for the given words are found, as it is possible for a word to be a NOUN or a VERB depending on the context. For example, the word 'act' can be used as a NOUN as in "Act II of Hamlet" and as a VERB - "acting in a movie". In the following part of the algorithm, from Line 7 to Line 50, all the Parts-Of-Speech found for each of the word is iterated through and if a match is found for the Part-Of-Speech, then the relationship and the *Confidence Measure* for that relationship is determined by calling the FindRelationshipJWNL algorithm with different WordNet Pointer Types such as SYNONYM, MERONYM, etc.

Since there can be multiple Parts-Of-Speech for the two words, the *Correspondence* with the highest *Confidence Measure* is obtained by calling the GetHighestCM algorithm in Line 51, which is then subsequently returned by the algorithm.

The following table summarizes the various WordNet Pointer and Relationship types:

WordNet Pointer Type	Relationship Type
SYNONYM (SIMILAR)	*is-a*
HYPONYM, HYPERNYM, TROPONYM	*type-of*
HOLONYM, MERONYM	*part-of*

Table 3.1: WordNet and Relationships

3.6.5 FindRelationshipJWNL **algorithm**

The FindRelationshipJWNL algorithm is shown below:

Algorithm 4 FindRelationshipJWNL

Input: $start$ and end are JWNL.IndexWords
Input: $type$ is the type of relationship being inquired about
Output: r is a JWNL.Relationship

```
1:  JWNL.Synset[] startSenses = start.getSenses()
2:  JWNL.Synset[] endSenses = end.getSenses()
3:  JWNL.Relationship r ← NULL

4:  for each s1 ∈ startSenses do        ▷ Check all against each other to find a relationship
5:      for each s2 ∈ endSenses do
6:          RelationshipList = JWNL.RelationshipFinder(startSenses[i], endSenses[j], type)
7:          if RelationshipList ≠ NULL then
8:              r ← RelationshipList.get(0)
9:              return r
10:         end if
11:     end for
12: end for

13: return NULL                         ▷ Relationship type not found for start and end
```

The FindRelationshipJWNL algorithm takes as input two JWNL.IndexWords, $start$ and end. It also needs the type of relationship that needs to be figured out (as shown in the table in Section 3.6.4), designated as $type$. If the requested relationship is found, the algorithm returns as output a *JWNL.Relationship*, r which contains the relationship between the two words and the corresponding *Confidence Measure*. Otherwise, it returns a $NULL$ value.

The WordNet senses for the two input words are retrieved in Line 1 and Line 2. Between Line 4 and Line 12, the senses for each word are iterated over to find a match for $type$. If a match is found, then the algorithm returns that relationship. Otherwise, a $NULL$ value is returned.

3.6.6 GetHighestCM **algorithm**

The GetHighestCM algorithm is shown below:

Algorithm 5 GetHighestCM

Input: c is set of *Correspondences*
Output: $cHighest$ is a *Correspondence* that has the highest *Confidence Measure*

```
1:  cHighest.RelationshipFound ← false
2:  if c.size() ≠ 0 then
3:      cHighest ← c.get(0)
4:      for each c1 ∈ c do
5:          if c1.RelationshipFound = true then
6:              if cHighest.ConfidenceMeasure ≥ c1.ConfidenceMeasure then
7:                  cHighest ← c1
8:              end if
9:          end if
10:     end for
11: end if

12: if (c.size() > 1) and (cHighest.CM > CM_THRESHOLD) then
13:     cHighest.Relationship ← NULL
14: end if

15: return cHighest
```

If more than one word match/relationship has been found (Line 2), then the *Confidence Measure (CM)* is used to resolve the relationship in the loop from Line 4 to Line 10. This measure represents the depth between the two words/concepts and comes directly from the JWNL library. The closer the *Confidence Measure* is to 0, the closer it is assumed to the 'real' relationship. A *Confidence Measure* of 0 represents a direct match. If two (or more) relationships have the same *Confidence Measure*, the last one is selected as the relationship between the two words. Similarly, if there are multiple word correspondences between two strings, the *Confidence Measure* is used to determine the eventual relationship between the two strings.

In Line 12 if the *Confidence Measure* found is greater then the *CM_THRESHOLD*, then it is determined that no relationship has been found. WordNet provides senses between two words that can very deep, and as such can provide very obscure relationships, which perhaps makes sense at some literary level but may not be useful in the normal

usage of the language. *CM_THRESHOLD* is thus used to ensure that no arcane or vague relationships are provided as output by the algorithm.

3.6.7 Time Complexity

Note: The discussion below follows the convention where the symbol V is used as a shorthand to denote $|V|$ (the number of vertices in a graph) in the context of asymptotic notation for Graph Algorithms [19].

For the Combine algorithm,

V_i is the set of nodes in the ontology O_i,

m is the number of words in $tokensSPL$,

n is the number of words in $tokensI$

The Combine algorithm is composed of the following algorithms: SelectLink , GetCorrespondences , GetRelationship , FindRelationshipJWNL and GetHighestCM algorithms. Since WordNet is a finite set of words and their senses, interaction with the WordNet database in GetRelationship and FindRelationshipJWNL algorithms is assumed to be of constant time for the purpose of determining the time complexity of the Combine algorithm.

The GetHighestCM algorithm contains the for loop in Line 4 which iterates over a given set of all the *Correspondences* to find out the highest *Confidence Measure*. This algorithm is primarily called from two places - in the SelectLink algorithm to determine the *Correspondence* to link O_{SPL} and O_i together and in the GetCorrespondences algorithm to obtain the correspondences between $strSPL$ and $strI$. When GetHighestCM is called in the SelectLink algorithm, the set of *Correspondences* can hold, at maximum, a *Correspondence* for each node in O_i, which is V_i. When the algorithm is called from the GetCorrespondences algorithm, the set of *Correspondences* can hold, at maximum, mn *Correspondences*.

SelectLink and GetCorrespondences are the main algorithms which have a direct im-

pact on the time complexity of the whole Combine algorithm. The SelectLink algorithm goes through all the nodes in O_i to determine if any *Correspondence* exists between the leaf node in O_{SPL} and O_i. Therefore, it will always loop for V_i iterations. The critical part of the Combine algorithm is the GetCorrespondences algorithm. It is where the decision is made for the relationship between the two nodes using the two strings, one from O_{SPL} and the other from O_i. The GetCorrespondences algorithm tries find a *Correspondence* between $strSPL$ and $strI$. The loops - between Line 13 and Line 23, Line 24 and Line 28 - go over the tokens generated for $strSPL$ and $strI$, m and n times. For GetCorrespondences algorithm, $(mn+mn) = 2mn$ which in turn, implies mn operations.

From the above analysis, it can be seen that for the average case, the time complexity of the Combine algorithm is $O(mnV_i)$. In the worst case, the length of both $strPL$ and $strI$ can be the same, yielding a value of $O(n^2V_i)$. In the best case scenario, $strSPL$ or $strI$ or both, can have just one word, which would then yield a time-complexity of $O(kV_i)$, assuming $k = mn$. The various cases are summarized below:

Case	Time Complexity
Average Case	$\boldsymbol{O(mnV_i)}$
Worst Case	$\boldsymbol{O(n^2V_i)}$
Best Case	$\boldsymbol{O(kV_i)}$ $k = mn$, where $m = 1$ or $n = 1$

Table 3.2: Time Complexity of Combine algorithm

Chapter 4

Experiments

4.1 Overview

This chapter covers the details of the implementation of the proposed approach. It also covers the details of the experiments undertaken after the implementation. The last section lists the contributions of this study.

4.2 Software

The following is a comprehensive list of all the software, libraries and APIs used for implementing the solution:

- Java [6] - the programming language
- Eclipse [3] - IDE for Java
- OWL API [8] - Java API for creating, manipulating and serializing OWL ontologies
- Protégé [11] - Ontology editor
- Pellet [9] - OWL DL reasoner
- WordNet [13] - lexical database for the English language

- Java WordNet Library (JWNL) [7] - Java API for accessing WordNet
- Apache Lucene [2] - Java library used for information retrieval
- SimMetrics [12] - Java library for similarity and distance metrics for strings
- Apache Ant [1] - Java library for automating software build processes

The Combine algorithm, as discussed in Chapter 3, was implemented within the existing interactive RE system [75] [73].

4.3 Interface

Figure 4.1 shows the existing interface of the dialogue system [73].

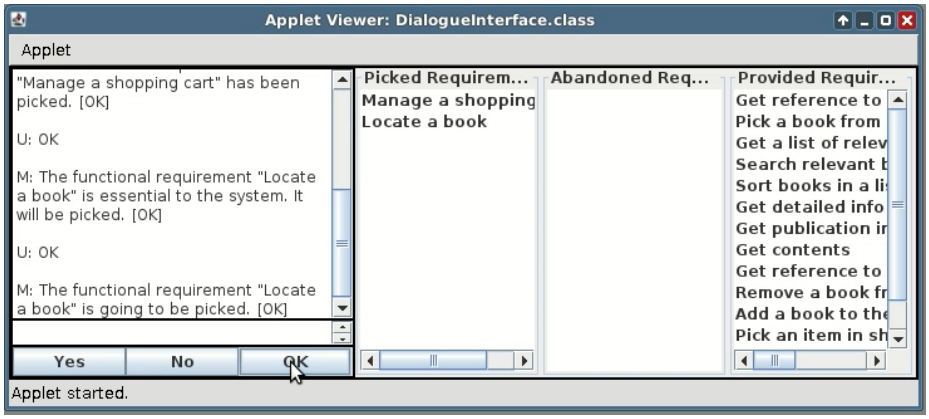


Figure 4.1: Existing interface of Interactive Requirements Elicitation system

The 'Ontology Combination Viewer' (OC Viewer), shown in Figure 4.2, was added to the interface to display the status of the various ontological combinations happening in the background during the session. The output is color-coded to help convey the information quickly:

- **Green** - means the ontology combination was successful
- **Red** - means the ontology combination was unsuccessful
- **Blue** - means that a leaf node was encountered

- **Magenta** - means that a mobile platform ontology was successfully combined

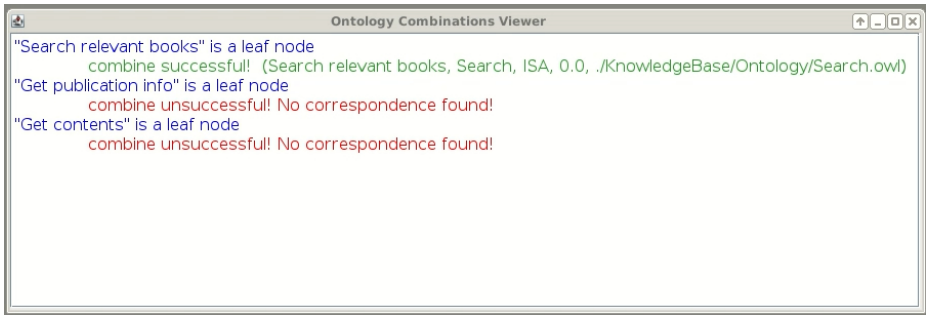


Figure 4.2: Ontology Combination Viewer (OC Viewer)

Figure 4.3 shows the initial state of the system. The 'Ontology Combination Viewer' (OC Viewer) is shown on the bottom right side. The OC Viewer displays the output of the Combine algorithm during the entire session.

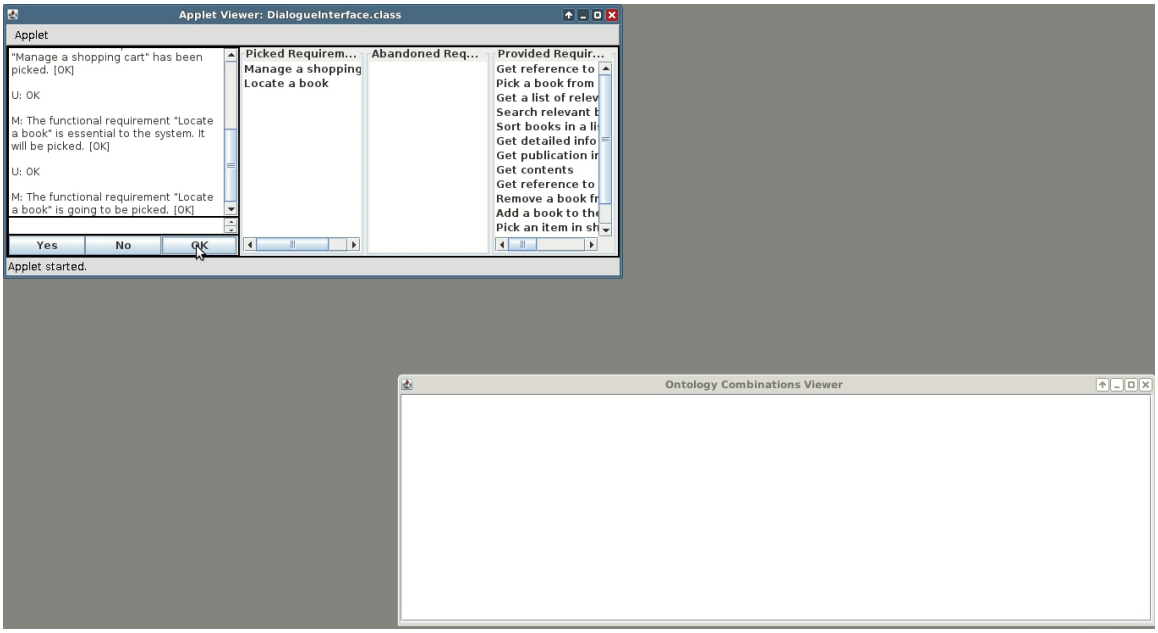


Figure 4.3: Initial state of Interactive Requirements Elicitation system

Figure 4.4 shows the state of the system after some requirements have been selected, with the OC Viewer displaying the corresponding messages on the status of ontology combinations.

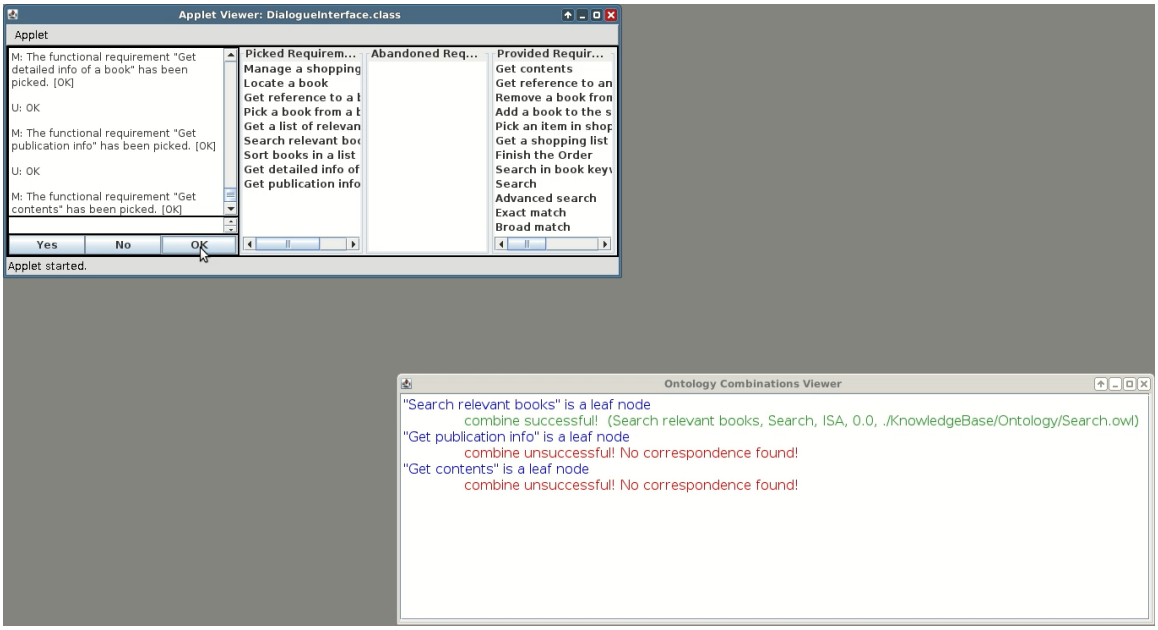


Figure 4.4: State of Interactive Requirements Elicitation system after selecting some requirements

4.4 Experiments

4.4.1 Scenario I - Single Ontology

As mentioned previously, the existing system [73] works with a single ontology. The Ontology Combination methodology seeks to improve this by letting the system harness other ontologies dynamically at run-time.

4.4.2 Scenario II - Multiple Ontologies

To illustrate the methodology, a scenario of requirements elicitation performed for creating an online bookstore system can be used. *BookStore* ontology, shown in Figure 4.5, can be assumed as the initial primary ontology O_{SPL} for the new online bookstore system [75]. This ontology represents the knowledge of an online bookstore system. The RE Knowledge Base, K_{RE}, contains three more ontologies, *Search*, *OrderSummary* and *ManagePaymentInfo* [75], shown in Figure 4.6, Figure 4.7 and Figure 4.8 respectively. *Search*, *OrderSummary* and *ManagePaymentInfo* are designated as O_i, O_j and O_k respectively.

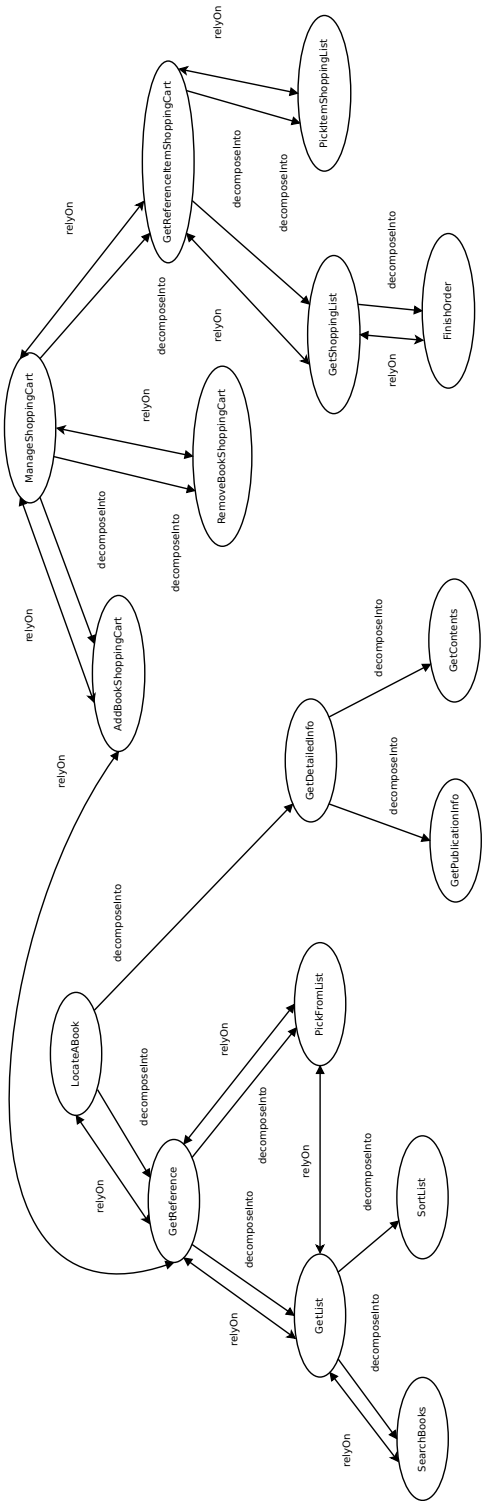


Figure 4.5: *BookStore* ontology, O_{SPL} in K_{RE}

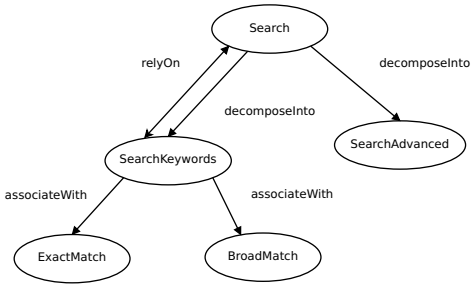


Figure 4.6: *Search* ontology, O_i in K_{RE}

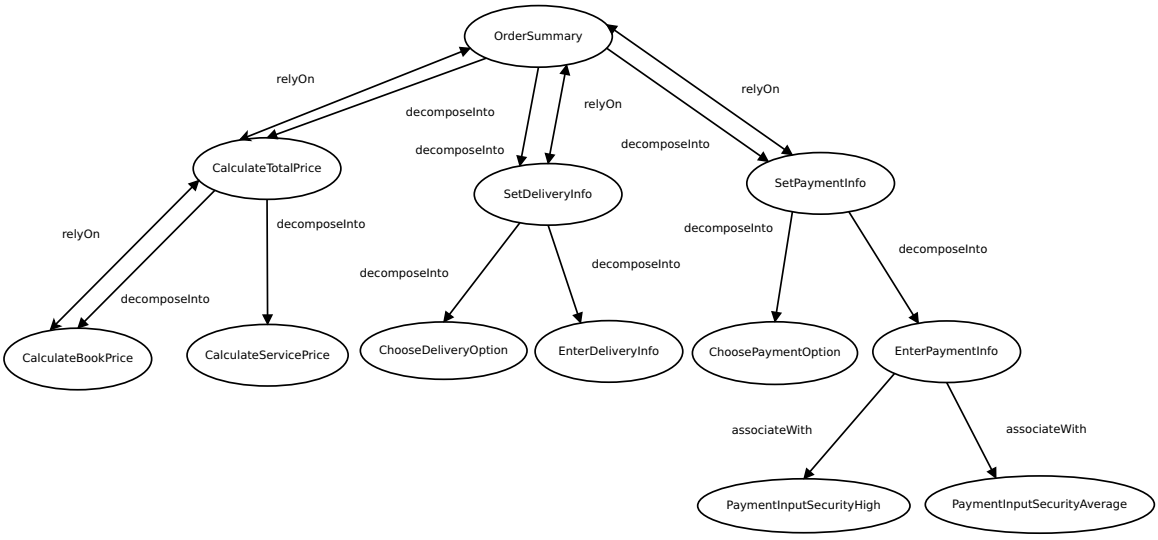


Figure 4.7: *OrderSummary* ontology, O_j in K_{RE}

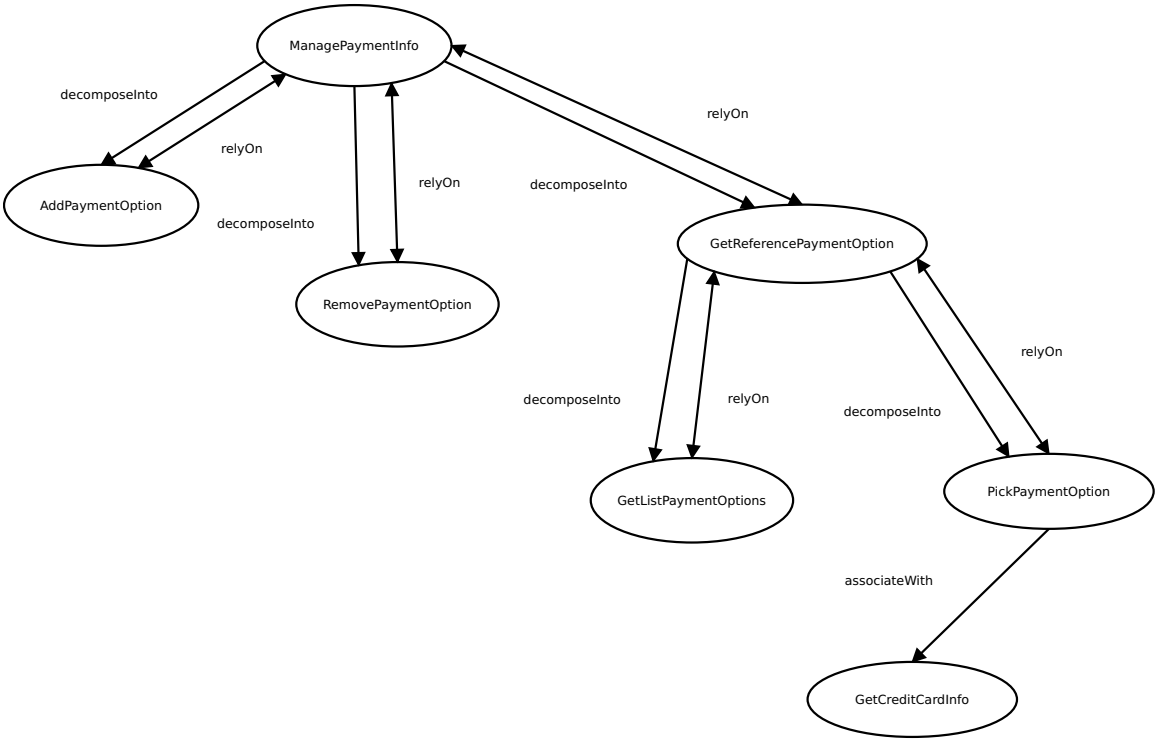


Figure 4.8: *ManagePaymentInfo* ontology, O_k in K_{RE}

The process begins with the primary ontology O_{SPL} being selected in the interactive RE system [73]. As the various requirements are selected, if they are a leaf node, i.e., they don't rely on or decompose into, other requirements, the Combine algorithm is called for that leaf node. For the *SearchBooks* leaf node of the O_{SPL}, O_i is selected by the Combine algorithm from K_{RE}. O_i is then combined with O_{SPL} with an IS-A relationship with the node *Search* in O_i to form a combined ontology. An excerpt of O_{SPL} after combination is shown in Figure 4.9.

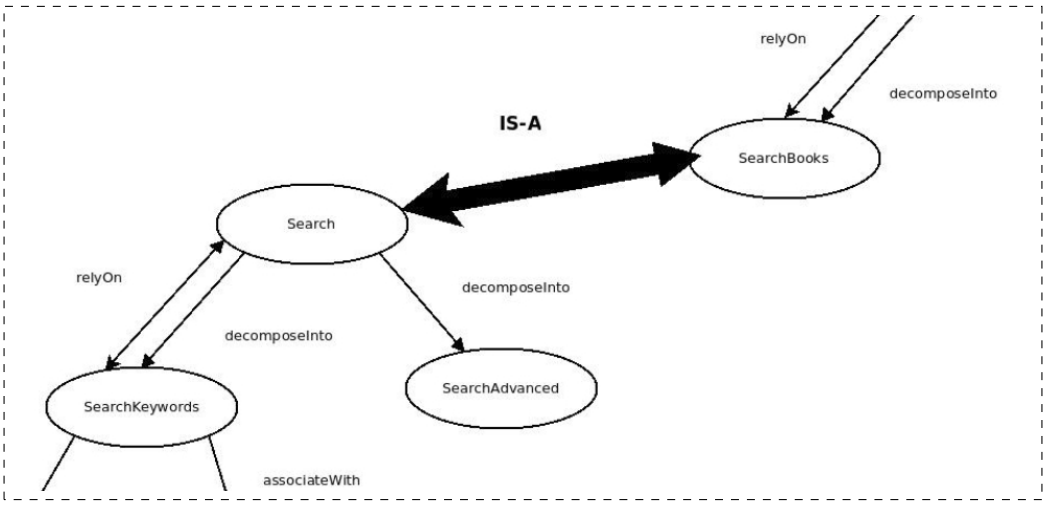


Figure 4.9: Excerpt of Combined Ontology - O_{SPL} and O_i

Figure 4.10 displays the output in the OC Viewer after the O_{SPL} and O_i have been successfully combined. When the "Search Relevant Books" (*SearchBooks*) leaf node is encountered, the OC Viewer displays the message in **Blue**. The successful combination of O_{SPL} and O_i is logged in **Green** and gives the name of the leaf node in O_{SPL} - "Search relevant books" (*SearchBooks*), the name of the node that was matched in O_i - "Search" (*Search*), the type of relationship found - "ISA", *Confidence Measure* of 0.0 and the name (along with the path) of the ontology file - "./KnowledgeBase/Ontology/Search.owl".

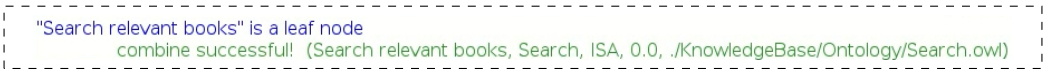


Figure 4.10: OC Viewer output after O_{SPL} and O_i combination

Ontology-based requirements elicitation is then continued on using the combined ontology. As elicitation moves on, once the *FinishOrder* leaf node is selected, the Combine algorithm is called again. This time O_j is selected from K_{RE}. It is combined with O_{SPL} with *OrderSummary* of O_j using the TYPE-OF relationship. An excerpt of the combined ontology is shown in Figure 4.11.

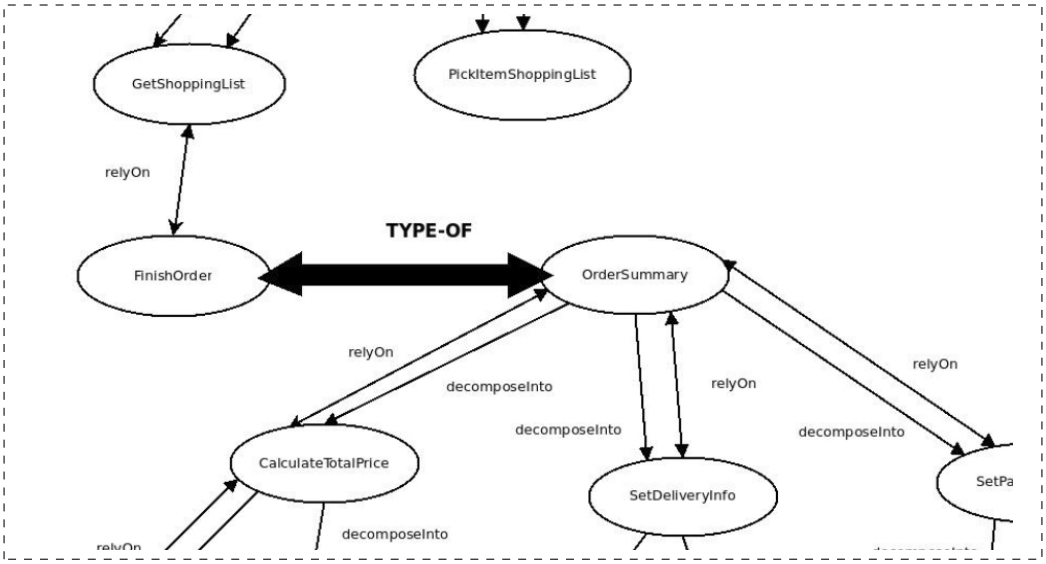


Figure 4.11: Excerpt of Combined Ontology - O_{SPL} and O_j

Figure 4.12 displays the output in the OC Viewer after the O_{SPL} and O_j have been successfully combined. When the "Finish the Order" (*FinishOrder*) leaf node is encountered, the OC Viewer displays the message in **Blue**. The successful combination of O_{SPL} and O_j is logged in **Green** and gives the name of the leaf node in O_{SPL} - "Finish the Order" (*FinishOrder*), the name of the node that was matched in O_j - "Get the Order Summary" (*OrderSummary*), the type of relationship found - "TYPEOF", *Confidence Measure* of 0.0 and the name (along with the path) of the ontology file - "./KnowledgeBase/Ontology/OrderSummary.owl".

Figure 4.12: OC Viewer output after O_{SPL} and O_j combination

After the leaf node of *ChoosePaymentOption* is selected, the Combine algorithm is called again, and this time the *ManagePaymentInfo* ontology, O_k, is selected for combination with O_{SPL} to help refine the *ChoosePaymentOption* concept. The node *ManagePaymentInfo* in O_k is linked to *ChoosePaymentOption* in O_{SPL} with a PART-OF relationship. An excerpt of the combined ontology is shown in Figure 4.13. The complete ontology, after the three iterations, is given in Figure 4.15.

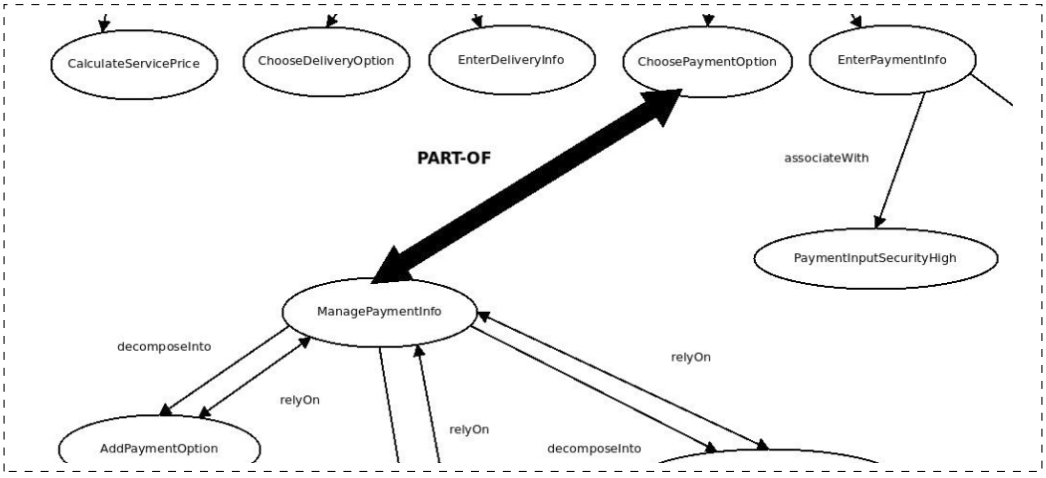


Figure 4.13: Excerpt of Combined Ontology - O_{SPL} and O_k

Figure 4.14 displays the output in the OC Viewer after the O_{SPL} and O_k have been successfully combined. When the "Choose Payment Option" (*ChoosePaymentOption*) leaf node is encountered, the OC Viewer displays the message in **Blue**. The successful combination of O_{SPL} and O_k is logged in **Green** and gives the name of the leaf node in O_{SPL} - "Choose Payment Option" (*ChoosePaymentOption*), the name of the node that was matched in O_k - "Manage Payment Information" (*ManagePaymentInfo*), the type of relationship found - "PARTOF", *Confidence Measure* of -1.0 and the name (along with the path) of the ontology file - "./KnowledgeBase/Ontology/ManagePaymentInfo.owl".

Figure 4.14: OC Viewer output after O_{SPL} and O_k combination

After ontology combinations, some relationships might need to be updated. The fol-

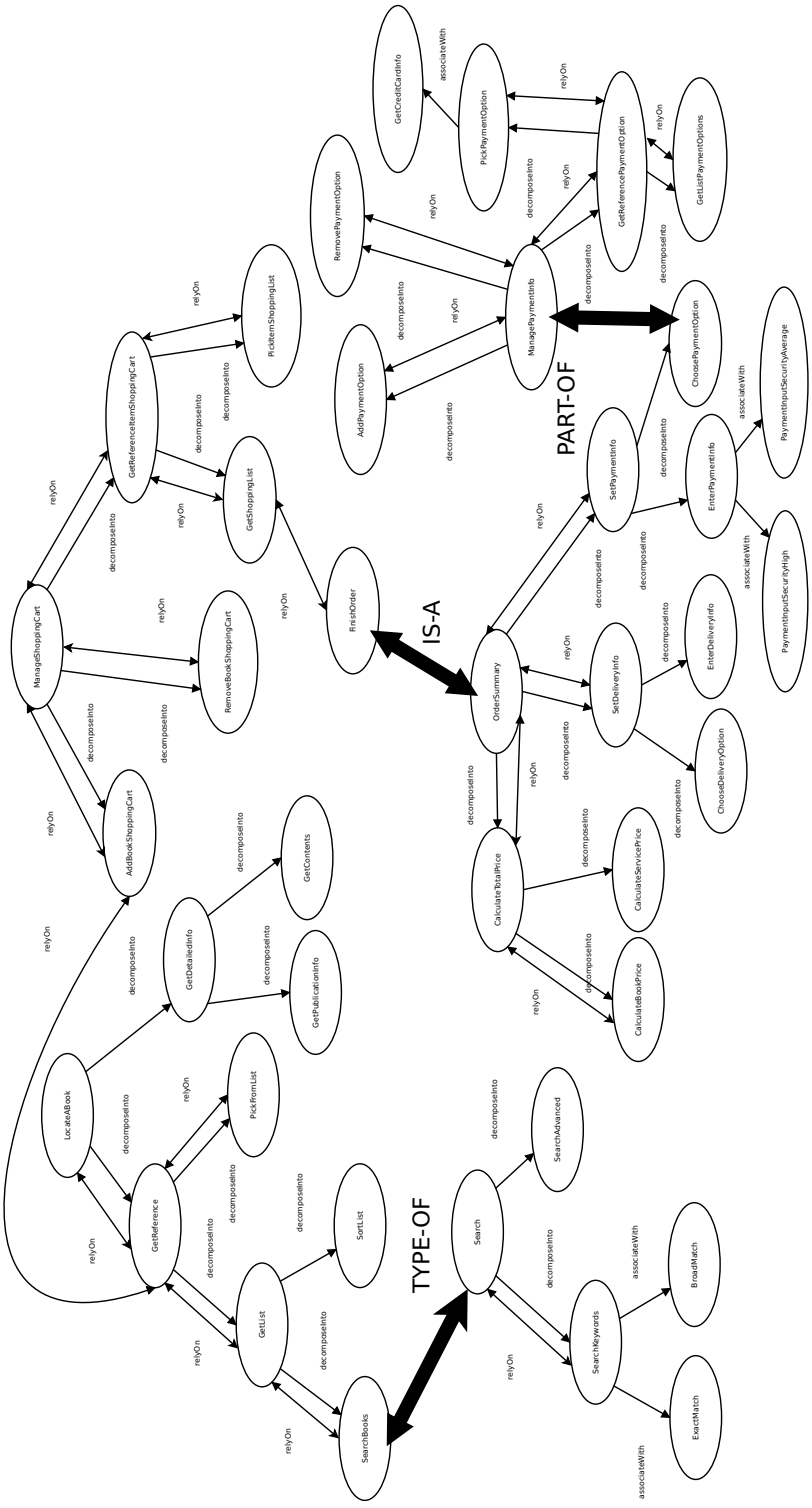


Figure 4.15: Complete Combined Ontology after three iterations

lowing section discusses this point in the context of combining ontologies for generating specifications for different platforms.

4.4.3 Scenario III - Ontology of Mobile SOA Functions

By including an ontology with platform-dependent functions, Ontology Combination enables the system to be extended to produce specifications for different operating systems. To illustrate this, the PlatformMobile ontology, O_{Mobile}, in Figure 4.16 is used. The O_{Mobile} ontology has been developed in particular to show how ontology combinations can allow for the dynamic inclusion of concepts that are platform-specific while gathering requirements interactively. It contains knowledge of platform-dependent concepts, such as *GetGPSCoordinates* and *GetMobileWallet*, which are specific to mobile (and similar) devices. When the O_{Mobile} ontology is combined during interactive RE, it allows for the dynamic selection of these platform-dependent requirements.

After a leaf node has been reached, then a search is carried out in K_{RE} for ontologies matching the node with the wildcard character (eg. *locate*). If a corresponding match for the filename is found, then the first matched file is selected for trying the combination (this is the default Ontology Combination approach). If no file has been found, then a second attempt is made with the mobile ontology to find a correspondence between the leaf node and any concepts inside the mobile ontology. This two-step search process ensures that the mobile ontology is tried at least once for combination. With O_{SPL}, the leaf node of *ChooseDeliveryOption* is matched with *DeliveryInfo* in O_{Mobile} using an IS-A relationship. An excerpt of the combined ontology is shown in Figure 4.17.

Figure 4.18 displays the output in the OC Viewer after the O_{SPL} and O_{Mobile} have been successfully combined. When the "Choose Delivery Option" (*ChooseDeliveryOption*) leaf node is encountered, the OC Viewer displays the message in **Blue**. The successful combination of O_{SPL} and O_{Mobile} is logged in **Magenta** and gives the name of the leaf node in O_{SPL} - "Choose Delivery Option" (*ChooseDeliveryOption*), the name of the node that was matched in O_{Mobile} - "Delivery Information" (*DeliveryInfo*), the type of relationship found - "ISA", *Confidence Measure* of 0.0 and the name (along with the path) of the ontology file - "./KnowledgeBase/OntologyMobile/PlatformMobile.owl".

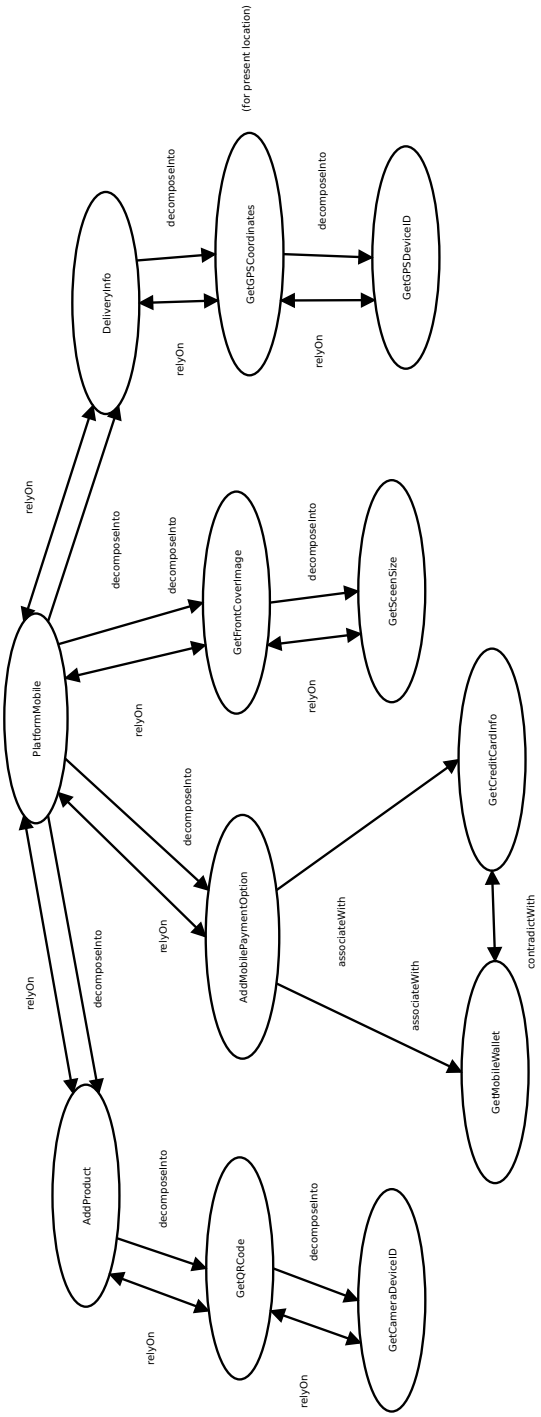


Figure 4.16: *PlatformMobile* ontology, O_{Mobile} in K_{RE}

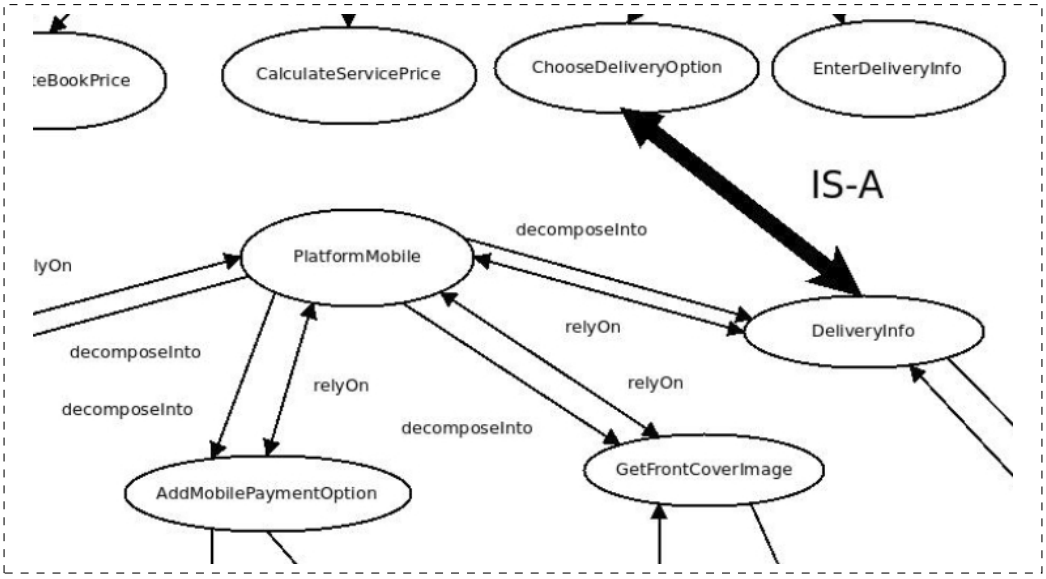


Figure 4.17: Excerpt of Combined Ontology - O_{SPL} and O_{Mobile}

```
"Choose Delivery Option" is a leaf node
        combine successful! with mobile platform ontology (Choose Delivery Option, Delivery Information,
ISA, 0.0, KnowledgeBase/OntologyMobile/PlatformMobile.owl)
```

Figure 4.18: OC Viewer output after O_{SPL} and O_{Mobile} combination

After the ontologies are combined if the same requirement exists in two ontologies, they are ‘merged’ together. Only one copy of a requirement is inside the combined ontology - the ‘oldest’ one. The ‘incoming’ requirement is ‘merged’ with the old one - the various properties (relyOn, decomposeInto, etc) are updated with the IRIs of the ’incoming’ requirement. The ‘neighboring’ requirements in the incoming ontology will automatically be linked to the ‘old’ requirement in the combined ontology. The requirement will then be evaluated just like any other regular requirement.

4.5 Case Study

This section provides a case study of an online book shopping software system. The book shopping system is customized through an interactive RE session. Ontology combinations occur during the interactive session, which are shown in the OC Viewer. Finally, an OWL-S file is generated as the output of the customization.

To customize the SOA based online book shopping system, RE is performed using the *BookStore* ontology, O_{SPL} [75]. The user needs the software system to retrieve information about books [75] and to manage shopping carts as well. So the software system should be able to retrieve details of books and provide shopping functionality such as payment and delivery information. To accomplish this, the interactive RE system is used to perform the customization. Listings 4.1 to 4.11 show the dialogue utterances between the user (U) and the system (M) during the interactive session.

In Listing 4.1, the user is informed that the essential requirement "Manage a shopping cart" has been selected. Similarly, the system informs the user that another essential requirement "Locate a book" has been picked. Following these selections, groups of functional requirements, which are part of a particular requirement, are pre-evaluated. Then, the system prompts the user for choosing various requirements. While selecting requirements, if a leaf node is encountered, then the system automatically combines an ontology from the RE Knowledge Base, K_{RE}, using the Combine algorithm. This is done for the following leaf nodes: "Search relevant books" in Listing 4.2, "Finish the Order"

in Listing 4.3, "Choose Delivery Option" in Listing 4.6 and "Choose Payment Option" in Listing 4.7. This is done in a manner that is transparent to the user. After ontology combinations, the new requirements are loaded into the session automatically. The user is offered to choose the new requirement just like any other regular requirement - the user does not know the difference between a requirement from the original ontology (O_{SPL}) and a requirement obtained from an ontology in K_{RE} (O_i, O_j, O_k, O_{Mobile}) which has been combined with the original ontology. The OC Viewer displays the information about ontology combinations in the background during the interactive session. After completion, the user is given the choice of changing his decisions, and if no further changes are needed, then the RE session is finished.

```
1  M: Do you like to customize online
       book shopping service? [YES]
2  U: YES
3  M: The functional requirement "
       Manage a shopping cart" is
       essential to the system. It will
       be picked. [OK]
4  U: OK
5  M: The functional requirement "
       Manage a shopping cart" has
       been picked. [OK]
6  U: OK
7  M: The functional requirement "
       Locate a book" is essential to
       the system. It will be picked. [
       OK]
8  U: OK
9  M: The functional requirement "
       Locate a book" is going to be
       picked. [OK]
10 U: OK
11 M: A group of functional
       requirements, which compose
       this requirement, are going to
       be pre-evaluated. [OK]
12 U: OK
13 M: The requirement "Get detailed
       info of a book" need to be
       evaluated. Would you like to
       choose the functional
       requirement "Get detailed info
       of a book"? [YES, NO]
14 U: YES
15 M: Are you sure you want to pick this
       requirement "Get detailed info
       of a book"? [YES]
16 U: YES
17 M: The group of functional
       requirements, which compose
       the requirement "Locate a book
       ", have been pre-evaluated. [
       OK]
18 U: OK
19 M: The functional requirement "
       Locate a book" has been picked.
        [OK]
20 U: OK
21 M: The functional requirement "Get
       reference to a book" has been
       picked. [OK]
22 U: OK
23 M: The functional requirement "Pick a
        book from a book list" has been
        picked. [OK]
24 U: OK
25 M: The functional requirement "Get a
       list of relevant books" is going
       to be picked. [OK]
26 U: OK
```

Listing 4.1: Dialogue Utterances (Part 1)

```
27 M: A group of functional
       requirements, which compose
       this requirement, are going to
       be pre-evaluated. [OK]
28 U: OK
29 M: The requirement "Sort books in a
       list" need to be evaluated.
       Would you like to choose the
       functional requirement "Sort
       books in a list"? [YES, NO]
30 U: YES
31 M: Are you sure you want to pick this
       requirement "Sort books in a list
       "? [YES]
32 U: YES
33 M: The group of functional
       requirements, which compose
       the requirement "Get a list of
       relevant books", have been pre
       -evaluated. [OK]
34 U: OK
35 M: The functional requirement "Get a
       list of relevant books" has been
       picked. [OK]
36 U: OK
37 M: The functional requirement "
       Search relevant books" has
       been picked. [OK]
38 U: OK
39 M: The functional requirement "Sort
       books in a list" has been picked.
        [OK]
40 U: OK
41 M: The functional requirement "Get
       detailed info of a book" is going
       to be picked. [OK]
42 U: OK
43 M: A group of functional
       requirements, which compose
       this requirement, are going to
       be pre-evaluated. [OK]
44 U: OK
45 M: The requirement "Get publication
       info" need to be evaluated.
       Would you like to choose the
       functional requirement "Get
       publication info"? [YES, NO]
46 U: YES
47 M: Are you sure you want to pick this
       requirement "Get publication
       info"? [YES]
48 U: YES
49 M: The requirement "Get contents"
       need to be evaluated. Would
       you like to choose the functional
        requirement "Get contents"? [
       YES, NO]
50 U: YES
```

Listing 4.2: Dialogue Utterances (Part 2)

```
51 M: Are you sure you want to pick this
       requirement "Get contents"? [
       YES]
52 U: YES
53 M: The group of functional
       requirements, which compose
       the requirement "Get detailed
       info of a book", have been pre-
       evaluated. [OK]
54 U: OK
55 M: The functional requirement "Get
       detailed info of a book" has
       been picked. [OK]
56 U: OK
57 M: The functional requirement "Get
       publication info" has been
       picked. [OK]
58 U: OK
59 M: The functional requirement "Get
       contents" has been picked. [OK]
60 U: OK
61 M: The functional requirement "Get
       reference to an item in the
       shopping cart" has been picked.
       [OK]
62 U: OK
63 M: The functional requirement "
       Remove a book from cart" has
       been picked. [OK]
64 U: OK
65 M: The functional requirement "Add a
        book to the shopping cart" has
       been picked. [OK]
66 U: OK
67 M: The functional requirement "Pick
       an item in shopping list" has
       been picked. [OK]
68 U: OK
69 M: The functional requirement "Get a
       shopping list" has been picked. [
       OK]
70 U: OK
71 M: The functional requirement "Finish
        the Order" has been picked. [
       OK]
72 U: OK
73 M: The requirement "Search in book
       keywords" need to be evaluated
       . Would you like to choose the
       functional requirement "Search
       in book keywords"? [YES, NO]
74 U: YES
75 M: Are you sure you want to pick this
       requirement "Search in book
       keywords"? [YES]
76 U: YES
```

Listing 4.3: Dialogue Utterances (Part 3)

```
77 M: The functional requirement "
       Search in book keywords" is
       going to be picked. [OK]
78 U: OK
79 M: A group of quality requirements,
       which are related with this
       requirement, are going to be pre
       -evaluated. [OK]
80 U: OK
81 M: The requirement "Exact match"
       need to be evaluated. Would
       you like to choose the quality
       requirement "Exact match"? [
       YES, NO]
82 U: YES
83 M: If you choose to pick the quality
       requirement "Exact match", 1
       requirement, which hasn't been
       evaluated, is contradicted with
       this requirement. It cannot be
       picked. Are you sure you want to
        pick this requirement "Exact
       match"? [YES]
84 U: YES
85 M: The group of qulity requirements
       which, are related with
       requirement "Search in book
       keywords", have been pre-
       evaluated. [OK]
86 U: OK
87 M: The functional requirement "
       Search in book keywords" has
       been picked. [OK]
88 U: OK
89 M: The requirement "Search" need to
        be evaluated. Would you like to
       choose the functional
       requirement "Search"? [YES, NO
       ]
90 U: YES
91 M: Are you sure you want to pick this
       requirement "Search"? [YES]
92 U: YES
93 M: The functional requirement "
       Search" is going to be picked. [
       OK]
94 U: OK
95 M: A group of functional
       requirements, which compose
       this requirement, are going to
       be pre-evaluated. [OK]
96 U: OK
97 M: The requirement "Advanced
       search" need to be evaluated.
       Would you like to choose the
       functional requirement "
       Advanced search"? [YES, NO]
98 U: YES
```

Listing 4.4: Dialogue Utterances (Part 4)

```
 99 M: Are you sure you want to pick this
        requirement "Advanced search
        "? [YES]
100 U: YES
101 M: The group of functional
        requirements, which compose
        the requirement "Search", have
        been pre-evaluated. [OK]
102 U: OK
103 M: The functional requirement "
        Search" has been picked. [OK]
104 U: OK
105 M: The functional requirement "
        Advanced search" has been
        picked. [OK]
106 U: OK
107 M: The quality requirement "Exact
        match" has been picked. [OK]
108 U: OK
109 M: The quality requirement "Broad
        match" has been abandoned. [
        OK]
110 U: OK
111 M: The requirement "Get the Order
        Summary" need to be evaluated
        . Would you like to choose the
        functional requirement "Get the
        Order Summary"? [YES, NO]
112 U: YES
113 M: If you choose to pick the
        functional requirement "Get the
        Order Summary", 4
        requirements, which haven't
        been evaluated, are prerequisite
         of this requirement. They will
        be picked. Are you sure you
        want to pick this requirement "
        Get the Order Summary"? [YES]
114 U: YES
115 M: The functional requirement "Get
        the Order Summary" has been
        picked. [OK]
116 U: OK
117 M: The functional requirement "
        Calculate Total Price" is going to
        be picked. [OK]
118 U: OK
119 M: A group of functional
        requirements, which compose
        this requirement, are going to
        be pre-evaluated. [OK]
120 U: OK
121 M: The requirement "Calculate
        Service Price" need to be
        evaluated. Would you like to
        choose the functional
        requirement "Calculate Service
        Price"? [YES, NO]
122 U: YES
```

Listing 4.5: Dialogue Utterances (Part 5)

```
123 M: Are you sure you want to pick this
        requirement "Calculate Service
        Price"? [YES]
124 U: YES
125 M: The group of functional
        requirements, which compose
        the requirement "Calculate Total
         Price", have been pre-
        evaluated. [OK]
126 U: OK
127 M: The functional requirement "
        Calculate Total Price" has been
        picked. [OK]
128 U: OK
129 M: The functional requirement "
        Calculate Book Price" has been
        picked. [OK]
130 U: OK
131 M: The functional requirement "
        Calculate Service Price" has
        been picked. [OK]
132 U: OK
133 M: The functional requirement "Set
        Delivery Information" is going to
         be picked. [OK]
134 U: OK
135 M: A group of functional
        requirements, which compose
        this requirement, are going to
        be pre-evaluated. [OK]
136 U: OK
137 M: The requirement "Enter Delivery
        Information" need to be
        evaluated. Would you like to
        choose the functional
        requirement "Enter Delivery
        Information"? [YES, NO]
138 U: YES
139 M: Are you sure you want to pick this
        requirement "Enter Delivery
        Information"? [YES]
140 U: YES
141 M: The requirement "Choose Delivery
         Option" need to be evaluated.
        Would you like to choose the
        functional requirement "Choose
        Delivery Option"? [YES, NO]
142 U: YES
143 M: Are you sure you want to pick this
        requirement "Choose Delivery
        Option"? [YES]
144 U: YES
```

Listing 4.6: Dialogue Utterances (Part 6)

```
145 M: The group of functional
        requirements, which compose
        the requirement "Set Delivery
        Information", have been pre-
        evaluated. [OK]
146 U: OK
147 M: The functional requirement "Set
        Delivery Information" has been
        picked. [OK]
148 U: OK
149 M: The functional requirement "
        Choose Delivery Option" has
        been picked. [OK]
150 U: OK
151 M: The functional requirement "Enter
        Delivery Information" has been
        picked. [OK]
152 U: OK
153 M: The functional requirement "Set
        Payment Information" is going
        to be picked. [OK]
154 U: OK
155 M: A group of functional
        requirements, which compose
        this requirement, are going to
        be pre-evaluated. [OK]
156 U: OK
157 M: The requirement "Choose
        Payment Option" need to be
        evaluated. Would you like to
        choose the functional
        requirement "Choose Payment
        Option"? [YES, NO]
158 U: YES
159 M: Are you sure you want to pick this
        requirement "Choose Payment
        Option"? [YES]
160 U: YES
161 M: The requirement "Enter Payment
        Information" need to be
        evaluated. Would you like to
        choose the functional
        requirement "Enter Payment
        Information"? [YES, NO]
162 U: YES
163 M: Are you sure you want to pick this
        requirement "Enter Payment
        Information"? [YES]
164 U: YES
165 M: The group of functional
        requirements, which compose
        the requirement "Set Payment
        Information", have been pre-
        evaluated. [OK]
166 U: OK
167 M: The functional requirement "Set
        Payment Information" has been
        picked. [OK]
168 U: OK
```

Listing 4.7: Dialogue Utterances (Part 7)

```
169 M: The functional requirement "
        Choose Payment Option" has
        been picked. [OK]
170 U: OK
171 M: The functional requirement "Enter
        Payment Information" is going
        to be picked. [OK]
172 U: OK
173 M: A group of quality requirements,
        which are related with this
        requirement, are going to be pre
        -evaluated. [OK]
174 U: OK
175 M: The requirement "Average
        security for payment input"
        need to be evaluated. Would
        you like to choose the quality
        requirement "Average security
        for payment input"? [YES, NO]
176 U: NO
177 M: Are you sure you want to abandon
        this requirement "Average
        security for payment input"? [
        YES]
178 U: YES
179 M: The requirement "High security for
        payment input" need to be
        evaluated. Would you like to
        choose the quality requirement
        "High security for payment input
        "? [YES, NO]
180 U: YES
181 M: Are you sure you want to pick this
        requirement "High security for
        payment input"? [YES]
182 U: YES
183 M: The group of qulity requirements
        which, are related with
        requirement "Enter Payment
        Information", have been pre-
        evaluated. [OK]
184 U: OK
185 M: The functional requirement "Enter
        Payment Information" has been
        picked. [OK]
186 U: OK
187 M: The quality requirement "Average
        security for payment input" has
        been abandoned. [OK]
188 U: OK
189 M: The quality requirement "High
        security for payment input" has
        been picked. [OK]
190 U: OK
```

Listing 4.8: Dialogue Utterances (Part 8)

```
191 M: The requirement "Platform Mobile" need to be evaluated. Would you like to choose the functional requirement "Platform Mobile"? [YES, NO]
192 U: YES
193 M: If you choose to pick the functional requirement "Platform Mobile", 9 requirements, which haven't been evaluated, are prerequisite of this requirement. They will be picked. Are you sure you want to pick this requirement "Platform Mobile"? [YES]
194 U: YES
195 M: The functional requirement "Platform Mobile" has been picked. [OK]
196 U: OK
197 M: The functional requirement "Add a mobile payment option" is going to be picked. [OK]
198 U: OK
199 M: A group of quality requirements, which are related with this requirement, are going to be pre-evaluated. [OK]
200 U: OK
201 M: The requirement "Get Credit Card Information" need to be evaluated. Would you like to choose the quality requirement "Get Credit Card Information"? [YES, NO]
202 U: YES
203 M: If you choose to pick the quality requirement "Get Credit Card Information", 1 requirement, which hasn't been evaluated, is contradicted with this requirement. It cannot be picked. 1 requirement, which hasn't been evaluated, is prerequisite of this requirement. It will be picked. Are you sure you want to pick this requirement "Get Credit Card Information"? [YES]
204 U: YES
205 M: The group of qulity requirements which, are related with requirement "Add a mobile payment option", have been pre-evaluated. [OK]
206 U: OK
```

Listing 4.9: Dialogue Utterances (Part 9)

```
207 M: The functional requirement "Add a mobile payment option" has been picked. [OK]
208 U: OK
209 M: The functional requirement "Delivery Information" has been picked. [OK]
210 U: OK
211 M: The functional requirement "Add product" has been picked. [OK]
212 U: OK
213 M: The functional requirement "Get Front Cover Image" has been picked. [OK]
214 U: OK
215 M: The functional requirement "Get Screen Size" has been picked. [OK]
216 U: OK
217 M: The functional requirement "Get GPS Coordinates" has been picked. [OK]
218 U: OK
219 M: The functional requirement "Get QR Code" has been picked. [OK]
220 U: OK
221 M: The functional requirement "Get Camera Device ID" has been picked. [OK]
222 U: OK
223 M: The functional requirement "Get GPS Device ID" has been picked. [OK]
224 U: OK
225 M: The quality requirement "Get Credit Card Information" has been picked. [OK]
226 U: OK
227 M: The quality requirement "Get Mobile Wallet" has been abandoned. [OK]
228 U: OK
229 M: The requirement "Manage Payment Information" need to be evaluated. Would you like to choose the functional requirement "Manage Payment Information"? [YES, NO]
230 U: YES
```

Listing 4.10: Dialogue Utterances (Part 10)

231	M: If you choose to pick the functional requirement "Manage Payment Information", 4 requirements, which haven’t been evaluated, are prerequisite of this requirement. They will be picked. Are you sure you want to pick this requirement " Manage Payment Information"? [YES]
232	U: YES
233	M: The functional requirement " Manage Payment Information" has been picked. [OK]
234	U: OK
235	M: The functional requirement "Get Reference to a Payment Option" has been picked. [OK]
236	U: OK
237	M: The functional requirement " Remove a Payment Option" has been picked. [OK]
238	U: OK
239	M: The functional requirement "Add a Payment Option" has been picked. [OK]
240	U: OK
241	M: The functional requirement "Pick a payment option" has been picked. [OK]
242	U: OK
243	M: The functional requirement "Get a list of payment options" has been picked. [OK]
244	U: OK
245	M: All requirements have been evaluated. Would you like to make any change to your selections (e.g. add requirements, remove requirements)? [ADD, REMOVE, NO]
246	U: NO
247	M: Requirement elicitation has been completed. Thank you very much for using. [OK]
248	U: OK

Listing 4.11: Dialogue Utterances (Part 11)

The final list of picked requirements is displayed in Listing 4.12. Listing 4.13 shows the final list of abandoned requirements.

```
 1 Manage a shopping cart
 2 Locate a book
 3 Get reference to a book
 4 Pick a book from a book list
 5 Get a list of relevant books
 6 Search relevant books
 7 Sort books in a list
 8 Get detailed info of a book
 9 Get publication info
10 Get contents
11 Get reference to an item in the
       shopping cart
12 Remove a book from cart
13 Add a book to the shopping cart
14 Pick an item in shopping list
15 Get a shopping list
16 Finish the Order
17 Search in book keywords
18 Search
19 Advanced search
20 Exact match
21 Get the Order Summary
22 Calculate Total Price
23 Calculate Book Price
24 Calculate Service Price
25 Set Delivery Information
26 Choose Delivery Option
27 Enter Delivery Information
28 Set Payment Information
29 Choose Payment Option
30 Enter Payment Information
31 High security for payment input
32 Platform Mobile
33 Add a mobile payment option
34 Delivery Information
35 Add product
36 Get Front Cover Image
37 Get Screen Size
38 Get GPS Coordinates
39 Get QR Code
40 Get Camera Device ID
41 Get GPS Device ID
42 Get Credit Card Information
43 Manage Payment Information
44 Get Reference to a Payment Option
45 Remove a Payment Option
46 Add a Payment Option
47 Pick a payment option
48 Get a list of payment options
```

Listing 4.12: Picked Requirements

```
1 Broad match
2 Average security for payment input
3 Get Mobile Wallet
```

Listing 4.13: Abandoned Requirements

Listing 4.14 shows the output of the OC Viewer. It lists all the ontology combinations that occurred during the session. Line 2 displays the output of the combination with *Search* ontology, O_i. The combination with *OrderSummary* ontology, O_j, is displayed in Line 12. Line 20 shows the output of the combination with *PlatformMobile* ontology, O_{Mobile}. The output of the combination with *ManagePaymentInfo* ontology, O_k, is shown in Line 24.

```
1  "Search relevant books" is a leaf node
2      combine successful! (Search relevant books, Search, ISA, 0.0, ./KnowledgeBase/
           Ontology/Search.owl)
3  "Get publication info" is a leaf node
4      combine unsuccessful! No correspondence found!
5  "Get contents" is a leaf node
6      combine unsuccessful! No correspondence found!
7  "Remove a book from cart" is a leaf node
8      combine unsuccessful! No correspondence found!
9  "Pick an item in shopping list" is a leaf node
10     combine unsuccessful! No correspondence found!
11 "Finish the Order" is a leaf node
12     combine successful! (Finish the Order, Get the Order Summary, TYPEOF, 0.0, ./
           KnowledgeBase/Ontology/OrderSummary.owl)
13 "Advanced search" is a leaf node
14     combine unsuccessful! No correspondence found!
15 "Calculate Book Price" is a leaf node
16     combine unsuccessful! No correspondence found!
17 "Calculate Service Price" is a leaf node
18     combine unsuccessful! No correspondence found!
19 "Choose Delivery Option" is a leaf node
20     combine successful! with mobile platform ontology (Choose Delivery Option,
           Delivery Information, ISA, 0.0, KnowledgeBase/OntologyMobile/
           PlatformMobile.owl)
21 "Enter Delivery Information" is a leaf node
22     combine unsuccessful! No correspondence found!
23 "Choose Payment Option" is a leaf node
24     combine successful! (Choose Payment Option, Manage Payment Information,
           PARTOF, -1.0, ./KnowledgeBase/Ontology/ManagePaymentInfo.owl)
25 "Get Screen Size" is a leaf node
26     combine unsuccessful! No correspondence found!
27 "Get Camera Device ID" is a leaf node
28     combine unsuccessful! No correspondence found!
29 "Get GPS Device ID" is a leaf node
30     combine unsuccessful! No correspondence found!
31 "Remove a Payment Option" is a leaf node
32     combine unsuccessful! No correspondence found!
33 "Add a Payment Option" is a leaf node
34     combine unsuccessful! No correspondence found!
35 "Get a list of payment options" is a leaf node
36     combine unsuccessful! No correspondence found!
```

Listing 4.14: Entire OC Viewer output

The interactive RE session output is generated using the requirement evaluation process [75] mentioned in Section 2.4. Through the use of ontology combinations, platform-dependent requirements are included and evaluated. The requirement evaluation algorithm merges the various requirements such as *AddMobilePaymentOption*, *DeliveryInfo*, *GetFrontCoverImage* into the *ManageShoppingCart* requirement, taking into account the inputs and outputs of all these requirements. The output OWL-S document is presented in Listing 4.15. The platform-dependent details are highlighted in gray. The documents, BookShoppingProcess.owl and BookShoppingQuality.owl, are imported by the profile document and define the instances of inputs, outputs and qualities [75]. By using this OWL-S description, services can be discovered by semantic capability matching. Then the services can be composed and executed based on the corresponding service composition information offered by the service providers [75].

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3                    xmlns:profile="http://www.daml.org/services/owl-s/1.2/Profile.owl
                         #"
4                    xmlns:owl="http://www.w3.org/2002/07/owl#">
5
6  <owl:Ontology rdf:about="">
7                    <owl:imports rdf:resource="http://www.daml.org/services/owl-s
                         /1.2/Profile.owl"/>
8                    <owl:imports rdf:resource="http://www.w3.org/1999/02/22-rdf-
                         syntax-ns"/>
9                    <owl:imports rdf:resource="ttp://www.w3.org/2002/07/owl"/>
10                   <owl:imports rdf:resource="http://www.semanticweb.org/ontologies
                         /InteractiveRE/BookShoppingProcess.owl"/>
11                   <owl:imports rdf:resource="http://www.semanticweb.org/ontologies
                         /InteractiveRE/BookShoppingQuality.owl"/>
12 </owl:Ontology>
13
14 <profile:Profile rdf:ID="Manage_a_shopping_cart">
15     <profile:textDescription>Manage a shopping cart</profile:textDescription>
16
17     <profile:has_process ref:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#ManageAShoppingCartProcess"/>
18
19     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#ListOfServices"/>
20     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#EnterNameInfo"/>
21     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#ReferenceToBookToRemove"/>
22     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#ListOfBooks"/>
23     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
           InteractiveRE/BookShoppingProcess.owl#ShoppingCartID"/>
24     <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
```

```
        InteractiveRE/BookShoppingProcess.owl#EnterAddressInfo"/>
25      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#MobileDeviceID"/>
26      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ReferenceToShoppingCart"/>
27
28      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#OrderServiceTotal"/>
29      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#OrderBookTotal"/>
30      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#PathToFrontCoverImage"/>
31      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#PaymentOption"/>
32      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ListOfServicePrice"/>
33      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ReferenceToPaymentInfo"/>
34      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ReferenceToSecurityInfo"/>
35      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#QRCode"/>
36      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ReferenceToABookInShoppingCart
            "/>
37      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ListOfItemsInShoppingCart"/>
38      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#BookIndexInShoppingCart"/>
39      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ShoppingCostTotal"/>
40      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#OrderTotal"/>
41      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ListOfPaymentOptions"/>
42      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#PaymentOptionDetails"/>
43      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#DeliveryInfo"/>
44      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#MobileDeviceProfile"/>
45      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#GPSCoordinates"/>
46      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#AddressDetails"/>
47      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ReferenceToPaymentOption"/>
48      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#GPSDeviceID"/>
49      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#MobileDevicePaymentOptions"/>
50      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#DeliveryOption"/>
51      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ProductID"/>
```

```
52      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ListOfBookPrices"/>
53      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ScreenSize"/>
54      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#CameraID"/>
55
56      <profile:serviceParameter rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingQuality.owl#PaymentInputSecurityAverage"/>
57      <profile:serviceParameter rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingQuality.owl#GetCreditCardInfo"/>
58  </profile:Profile>
59
60  <profile:Profile rdf:ID="Locate_a_book">
61      <profile:textDescription>Locate a book</profile:textDescription>
62
63      <profile:has_process ref:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#LocateABookProcess"/>
64
65      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#SortingOrder"/>
66      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#PhrasesFromUserInput"/>
67      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#BookIndexInTheList"/>
68      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#SearchFields"/>
69      <profile:hasInput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#PhrasesForSearchFields"/>
70
71      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#ListOfRelevantBooks"/>
72      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#BookPublicationInfo"/>
73      <profile:hasOutput rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingProcess.owl#BookContents"/>
74
75      <profile:serviceParameter rdf:resource="http://www.semanticweb.org/ontologies/
            InteractiveRE/BookShoppingQuality.owl#ExactMatch"/>
76  </profile:Profile>
77
78  </rdf:RDF>
```

Listing 4.15: OWL-S output file with platform-dependent details highlighted in gray

4.6 Contributions

4.6.1 Overview

This section covers the contributions of the study undertaken for this thesis.

4.6.2 Enhanced Interactive Requirements Elicitation

The Ontology Combination methodology [71] [72], as shown in this thesis, provides a seamless interactive experience for acquiring requirements from multiple ontologies. Ontology Combinations are invisible to the user as they happen in the background and without the user's knowledge. The dynamic nature of Ontology Combination, in the context of interactive RE, is shown in an abstract manner by Figure 4.19.

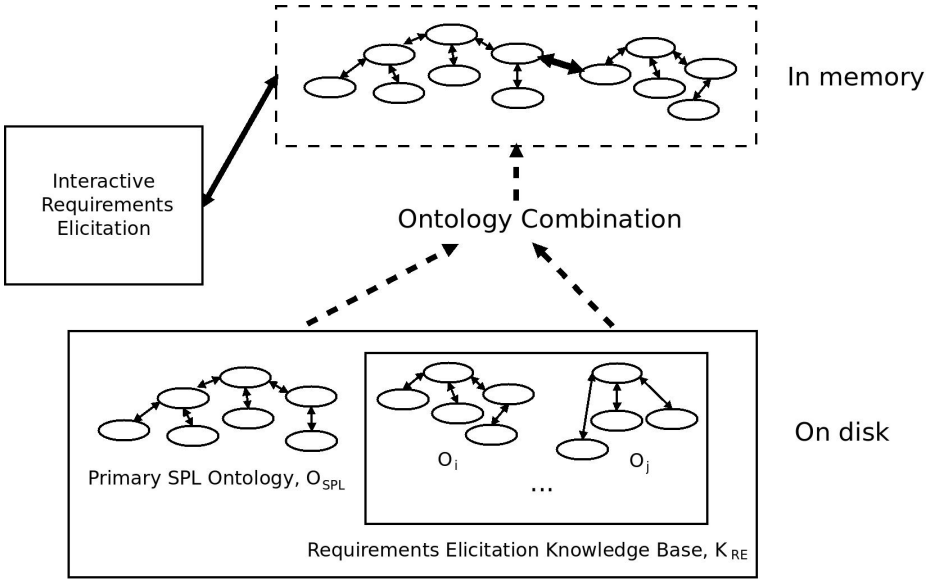


Figure 4.19: Ontology Combination - Abstract view

Ontologies that are being combined are assumed to be distinct, i.e., a concept to be matched only exists in the primary SPL ontology, O_{SPL} as an 'imprecise' concept, and ontologies, O_i, O_j, O_k, and O_{Mobile}, taken from the RE Knowledge base, K_{RE}, serve to refine that concept. This also illustrates the advantages of Ontology Combination over ontology merging. Instead of creating a merged ontology, which would require more

resources in terms of time and memory, combining ontologies is less resource-intensive. Iterating over the Ontology Combination process, requirements can be fine-tuned to the smallest level of detail that the ontologies provide in K_{RE}.

Experiments were conducted based on the number of ontologies being combined together during interactive requirements elicitation sessions. The desired outcome was logical combinations between ontologies being combined together in a responsive manner, transparent to the user. The requirements specification generated is 'richer' after the combination process as compared to the specifications generated before the combination occurred. In Section 4.4.2, there were sixteen requirements in the primary SPL ontology O_{SPL} - by combining O_{SPL} with O_i, O_j, O_k and O_{Mobile}, the number of requirements were tripled to forty-eight requirements. This is a significant increase over the initial ontology in the potential of acquiring requirements.

4.6.3 Extending Customization to Mobile Applications

The previous interactive RE system for SOA-based SPL produced specifications for traditional (desktop-based) applications [75]. This thesis has extended that approach to produce specifications for generating SOA applications for mobile operating systems. A use-case was given in Section 4.4.3 Scenario III - Ontology of Mobile SOA Functions. Contradictions due to platform-dependent features were resolved through the Ontology Combination process. As the ecosystem of mobile software and hardware changes rapidly, this will enable applications to be built for specific versions of particular mobile operating systems in a rapid manner.

Chapter 5

Conclusion and Future Directions

5.1 Overview

This chapter concludes the thesis and presents some potential directions of future research.

5.1.1 Conclusion

The methodology of Ontology Combination has been proposed in this thesis. The Combine algorithm brings ontologies together at run-time, dynamically enhancing the interactive RE process. The Scenarios presented in this thesis illustrate the effectiveness of the Ontology Combination methodology. Using this approach, interactive RE can also be used seamlessly for the purpose of customizing software for specific platforms, thereby helping to automate SPLs.

5.1.2 Future Directions

Run-time application of Ontology Combinations provides multiple future directions of further research. The Ontology Combination methodology presented in this work uses ontologies that are derived from the ontology requirement meta-model and performs combination on these ontologies. A future direction of research can look into combining ontologies that follow different designs, where the primary focus would be on trying to accommodate the overlap and contradictions that the set of axioms from different ontologies would entail.

Also, the Ontology Combination approach can be expanded by realizing a complete SPL framework for SOA based applications. This framework can potentially create applications targeting multiple platforms, with features developed and maintained to suit different hardware specifications. A comprehensive SPL framework can be extended to both software and hardware feature resolution and creation of SOA software that synthesizes the knowledge of both software and hardware features.

Bibliography

[1] (2016) Apache Ant. [Online]. Available: http://ant.apache.org/

[2] (2016) Apache Lucene. [Online]. Available: https://lucene.apache.org/

[3] (2016) Eclipse IDE. [Online]. Available: https://eclipse.org/

[4] (2016) Food ontology. [Online]. Available: http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf

[5] (2016) IEEE Rightslink. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/rights_link.html

[6] (2016) Java. [Online]. Available: https://www.java.com/

[7] (2016) Java WordNet Library (JWNL). [Online]. Available: https://sourceforge.net/projects/jwordnet/

[8] (2016) The OWL API. [Online]. Available: http://owlapi.sourceforge.net/

[9] (2016) Pellet reasoner. [Online]. Available: http://pellet.owldl.com/

[10] (2016) Pizza ontology. [Online]. Available: http://protege.stanford.edu/ontologies/pizza/pizza.owl

[11] (2016) Protege project. [Online]. Available: http://protege.stanford.edu

[12] (2016) SimMetrics. [Online]. Available: https://sourceforge.net/projects/simmetrics/

[13] (2016) WordNet. [Online]. Available: https://wordnet.princeton.edu/

[14] Akahani J.-i., Hiramatsu K., and Kogure K., "Coordinating heterogeneous information services based on approximate ontology translation," in *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*. Citeseer, 2002, pp. 10–14.

[15] Altintas N. I., Cetin S., and Dogru A. H., "Industrializing software development: The "Factory Automation" way," in *Trends in Enterprise Application Architecture*. Springer, 2007, pp. 54–68.

[16] Amin M. B., Ahmad M., Khan W. A., and Lee S., "Biomedical ontology matching as a service," in *Smart Homes and Health Telematics*. Springer, 2015, pp. 195–203.

[17] Avdeenko T. and Pustovalova N., "The ontology-based approach to support the completeness and consistency of the requirements specification," *SIBCON, 2015 IEEE International Siberian Conference on Control and Communications*, p. to appear, 2015.

[18] Batini C., Lenzerini M., and Navathe S. B., "A comparative analysis of methodologies for database schema integration," *ACM computing surveys (CSUR)*, vol. 18, no. 4, pp. 323–364, 1986.

[19] Cormen T. H., *Introduction to algorithms*. MIT press, 2009.

[20] Dobson G. and Sawyer P., "Revisiting ontology-based requirements engineering in the age of the semantic web," in *Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs*, 2006, pp. 27–29.

[21] Dzung D. V. and Ohnishi A., "Ontology-based reasoning in requirements elicitation," in *Software Engineering and Formal Methods, 2009 Seventh IEEE International Conference on*. IEEE, 2009, pp. 263–272.

[22] Erl T., "Service-oriented architecture: Concepts, technology, and design," 2005.

[23] Euzenat J. and Shvaiko P., *Ontology Matching*. Springer Science & Business Media, 2013.

[24] Farfeleder S., Moser T., Krall A., Stålhane T., Omoronyia I., and Zojer H., "Ontology-driven guidance for requirements elicitation," in *The Semanic Web: Research and Applications*. Springer, 2011, pp. 212–226.

[25] Fernandez S., Marsa-Maestre I., Velasco J. R., and Alarcos B., "Ontology alignment architecture for semantic sensor web integration," *Sensors*, vol. 13, no. 9, pp. 12 581–12 604, 2013.

[26] Fudholi D. H., Rahayu W., and Pardede E., "Code (common ontology development): A knowledge integration approach from multiple ontologies," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*. IEEE, 2014, pp. 751–758.

[27] Gaševic D., Djuric D., and Devedžic V., *Model driven engineering and ontology development*. Springer Science & Business Media, 2009.

[28] Goguen J. A. and Linde C., "Techniques for requirements elicitation." *RE*, vol. 93, pp. 152–164, 1993.

[29] Gomaa H. and Shin M. E., "Automated software product line engineering and product derivation," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*. IEEE, 2007, pp. 285a–285a.

[30] Gomez-Perez A., Fernández-López M., and Corcho O., *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[31] Granitzer M., Sabol V., Onn K. W., Lukose D., and Tochtermann K., "Ontology alignment - a survey with focus on visually supported semi-automatic techniques," *Future Internet*, vol. 2, no. 3, pp. 238–258, 2010.

[32] Helferich A., Herzwurm G., Jesse S., and Mikusz M., "Software product lines, service-oriented architecture and frameworks: worlds apart or ideal partners?" in *Trends in Enterprise Application Architecture*. Springer, 2007, pp. 187–201.

[33] Hitzler P., Krötzsch M., Ehrig M., and Sure Y., "What is ontology merging?" in *American Association for Artificial Intelligence*, 2005.

[34] Hu J., Jia S., and Wu K., "Semantic-based requirements content management for cloud software," *Mathematical Problems in Engineering*, vol. 501, p. 474157, 2015.

[35] Kaiya H. and Saeki M., "Ontology based requirements analysis: lightweight semantic processing approach," in *Quality Software, 2005.(QSIC 2005). Fifth International Conference on*. IEEE, 2005, pp. 223–230.

[36] Kang D. and Baik D.-K., "Bridging software product lines and service-oriented architectures for service identification using bpm and fm," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*. IEEE, 2010, pp. 755–759.

[37] Kang D., Song C.-y., and Baik D.-K., "A method of service identification for product line," in *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, vol. 2. IEEE, 2008, pp. 1040–1045.

[38] Keet C. M., "Aspects of ontology integration," *The PhD Proposal, School of Computing, Napier University, Scotland*, 2004.

[39] Kong H., Hwang M., and Kim P., "A new methodology for merging the heterogeneous domain ontologies based on the wordnet," in *Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on*. IEEE, 2005, pp. 6–pp.

[40] Kossmann M., Gillies A., Odeh M., and Watts S., "Ontology-driven requirements engineering with reference to the aerospace industry," in *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the*. IEEE, 2009, pp. 95–103.

[41] Kumar S. K. and Harding J. A., "Description logic–based knowledge merging for concrete-and fuzzy-domain ontologies," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, p. 0954405414564404, 2015.

[42] Kwak J. and Yong H.-S., "Ontology matching based on hypernym, hyponym, holonym, and meronym sets in word net," *International journal of Web & Semantic Technology (I West)*, vol. 1, no. 2, 2010.

[43] Lee J., Muthig D., and Naab M., "An approach for developing service oriented product lines," in *Software Product Line Conference, 2008. SPLC'08. 12th International*. IEEE, 2008, pp. 275–284.

[44] Li F.-L., Horkoff J., Borgida A., Guizzardi G., Liu L., and Mylopoulos J., "From stakeholder requirements to formal specifications through refinement," in *Requirements Engineering: Foundation for Software Quality*. Springer, 2015, pp. 164–180.

[45] Lin C.-Y. I. and Ho C.-S., "A generic ontology-based approach for requirement analysis and its application in network management software," *AI EDAM*, vol. 13, no. 01, pp. 37–61, 1999.

[46] Loucopoulos P. and Karakostas V., *System requirements engineering*. McGraw-Hill, Inc., 1995.

[47] Macaulay L., "Requirements for requirements engineering techniques," in *Requirements Engineering, 1996., Proceedings of the Second International Conference on*. IEEE, 1996, pp. 157–164.

[48] Mahfoudh M., Thiry L., Forestier G., and Hassenforder M., "Algebraic graph transformations for merging ontologies," in *Model and Data Engineering*. Springer, 2014, pp. 154–168.

[49] Medeiros F. M., de Almeida E. S., and de Lemos Meira S. R., “Towards an approach for service-oriented product line architectures,” in *Proceedings of the Workshop on Service-oriented Architectures and Software Product Lines*, 2009, pp. 1–7.

[50] Mena E., Kashyap V., Illarramendi A., and Sheth A. P., “Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information,” 1996.

[51] Metzger A. and Pohl K., “Software product line engineering and variability management: achievements and challenges,” in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 70–84.

[52] Mohabbati B., Asadi M., Gašević D., Hatala M., and Müller H. A., “Combining service-orientation and software product line engineering: A systematic mapping study,” *Information and Software Technology*, vol. 55, no. 11, pp. 1845–1859, 2013.

[53] Nguyen T. H., Grundy J., and Almorsy M., “Guitar: An ontology-based automated requirements analysis tool,” in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. IEEE, 2014, pp. 315–316.

[54] Noy N. F. and Musen M. A., “Smart: Automated support for ontology merging and alignment,” in *Proc. of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW’99), Banf, Canada*, 1999.

[55] ——, “Algorithm and tool for automated ontology merging and alignment,” in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*, 2000.

[56] Parra C. and Joya D., “Split: An automated approach for enterprise product line adoption through soa,” *Journal of Internet Services and Information Security (JISIS)*, vol. 5, no. 1, pp. 29–52, 2015.

[57] Parra C., Joya D., Giral L., and Infante A., "An soa approach for automating software product line adoption," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 2014, pp. 1231–1238.

[58] Picco G. P., Julien C., Murphy A. L., Musolesi M., and Roman G.-C., "Software engineering for mobility: reflecting on the past, peering into the future," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 13–28.

[59] Pohl K., Böckle G., and van der Linden F. J., *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

[60] Rabiser R., Grünbacher P., and Dhungana D., "Requirements for product derivation support: Results from a systematic literature review and an expert survey," *Information and Software Technology*, vol. 52, no. 3, pp. 324–346, 2010.

[61] Raunich S. and Rahm E., "Atom: Automatic target-driven ontology merging," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 1276–1279.

[62] Sim W. W. and Brouse P., "Developing ontologies and persona to support and enhance requirements engineering activities–a case study," *Procedia Computer Science*, vol. 44, pp. 275–284, 2015.

[63] Sommerville I. and Sawyer P., *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.

[64] Specification O. A., "Omg unified modeling language (omg uml), superstructure, v2. 1.2," *Object Management Group*, 2007.

[65] Studer R., Benjamins V. R., and Fensel D., "Knowledge engineering: principles and methods," *Data & knowledge engineering*, vol. 25, no. 1, pp. 161–197, 1998.

[66] Stumme G. and Maedche A., "Fca-merge: Bottom-up merging of ontologies," in *IJCAI*, vol. 1, 2001, pp. 225–230.

[67] Trujillo S., Kästner C., and Apel S., "Product lines that supply other product lines: A service-oriented approach," in *SPLC Workshop: Service-Oriented Architectures and Product Lines–What is the Connection*, 2007.

[68] Tsui F. F., Karam O., and Bernal B., *Essentials of software engineering*. Jones & Bartlett Publishers, 2014.

[69] Wu D., "Context knowledge base for ontology integration," Ph.D. dissertation, KTH Royal Institute of Technology, 2014.

[70] Yuan X. and Liu G., "A task ontology model for domain independent dialogue management," in *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2012 IEEE International Conference on*. IEEE, 2012, pp. 148–153.

[71] Yuan X. and Tripathi S., "Combining ontologies for requirements elicitation," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2015 IEEE International*. IEEE, 2015, pp. 1–5.

[72] ——, "An approach of dynamically combining ontologies for interactive requirements elicitation," in *Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on*. IEEE, 2016, (to appear).

[73] Yuan X. and Zhang X., "An interactive approach of online software customization via conversational web agents," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 327–334.

[74] ——, "An ontology-based requirement modeling for interactive software customization," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2015 IEEE International*. IEEE, 2015, pp. 1–10.

[75] Zhang X., “An interactive approach of ontology-based requirement elicitation for software customization,” Master’s thesis, University of Windsor, 2011, http://scholar.uwindsor.ca/etd/347.

Vita Auctoris

NAME	Shubhrendu Tripathi
PLACE OF BIRTH	New Delhi, India
YEAR OF BIRTH	1983
EDUCATION	Memorial High School, Houston, Texas, USA
	1998-1999
	B.Sc., University of Houston, Texas, USA
	1999-2002