Towards a hybrid methodology for domain ontology development

John, Santhosh

Award date:
2019

Awarding institution:
Coventry University

Towards a Hybrid Methodology for Domain Ontology Development

By

Santhosh John

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

May 2019

Certificate of Ethical Approval

Applicant:

Santhosh John

Project Title:

Towards A Hybrid Methodology for Domain Ontology Development

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Medium Risk

Date of approval:

19 September 2017

Project Reference Number:

P61284

Dedicated to almighty, beloved parents, my loving family and well-wishers

Abstract

The last few years have witnessed a paradigm shift in the World Wide Web, from a global information space of connected documents to the Semantic Web. The Semantic Web offers an effective knowledge representation with appropriate formalisms. Based on ontologies it has emerged as an appropriate engineering solution for the problems of developing systems to ensure the integration of data from different sources with a high level of interoperability, providing seamless services to web users. Though Ontology Engineering and Software Engineering are two complementary engineering branches, the maturity and the popularity level of the latter is too high compared to the former. It is evident from the literature that there exists a gap in the ontology-engineering process in terms of the availability of standardised development-methodologies. Unlike Software Engineering, the absence of a standardised methodology for developing ontologies, limits the growth of ontology engineering, by restricting the pace of large-scale ontology development.

The aim of this research is to anlyse the well-known and widely used existing ontology-development methodologies and to explore the potential of proposing a methodology for ontology development by extending the mature process-models and methodologies of Software Engineering. Building from this analysis, the research proposes a methodology for domain ontology development and applies the proposed methodology to a chosen domain as a proof of concept. What uniquely distinguishes the proposed methodology from the existing methodologies is the underpinning hybrid approach of linear and iterative software process-models.

This novel methodology classifies the core ontology development process into four stages and defines the development life cycle in terms of the hierarchy of its components. Specific workflows which encompass well-defined activities with a recommended list of techniques are the salient features of the proposed methodology. Furthermore, it has been applied for the prototype development of an educational domain ontology using Protégé as the development environment.

The methodology has been validated by a group of evaluators with a rich set of proven research and development experience by following a custom-built evaluation framework.

The proposed methodology can be considered as a fine choice for the future requirements of ontology developers. Moreover, it can be an ideal choice for software practitioners who wish

to extend their expertise to the ontology-development domain, to assist in large-scale ontology development accelerating the realisation of a semantic web vision.

Acknowledgements

I owe a lot of gratitude to people without their support; this thesis would not have been complete.

I express my deep and sincere gratitude to my Director of Studies **Dr Nazaraf Shah**, the members of supervisory panel **Dr Craig Stewart** and **Dr Arun N.S** for their outstanding guidance and encouragement throughout all the stages of this work. Throughout this work, Dr Shah provided valuable advice, great ideas, and constructive feedback. His confidence in my capabilities has sustained me throughout the period of research

I am extending my sincere thanks to **Dr Abdullah Saif Ahmed Al Sabahy**, Chairman of Middle East College, Oman, for his continual encouragement. My heartfelt gratitude goes to **Mr. Lefeer Muhamed Marakkarackayil**, Managing Director of Middle East College, who was instrumental in giving me the opportunity and motivation needed at the right time. Sincere thanks to **Mr. Ali Al Uraimi,** Deputy Managing Director of Middle East College for his continual support and motivation.

I am extremely grateful to **Dr Kiran G.R**, Dean of Middle East College for his continual support, brotherly care and motivation, without which this research would not have seen the light of the day. I am thankful to **Mr. Ashwin J.K**, the registrar of Middle East College for his motivation and friendly support.

I am obliged to my colleagues and friends, *Mr. Gladwin George, Ms. Preethy Kurian, Mr. Vikas Rao, Mr. Mustafa Siddique* and *Dr Manju Jose* who provided invaluable support during the proofreading, compilation and formatting of this thesis.

I thank the fellow researchers, *Thoufeeq Ahmed, Priya Mathew, Aliya Al Farsi and Khoula Al Harthy*. The friendship I have with them is what kept me going through tough times.

I am deeply indebted to my wife *Silvia Santhosh* for her friendship, trust, encouragement, and whole-hearted support. Special thanks to my loving children, *Saina Santhosh* and *Geogin Santhosh*, my all-time source of inspiration and energy.

Special gratitude is due to my hometown, ***Sooranad North, Kerala*** for always keeping me inspired.

List of Publications

1. Santhosh John, "**Development of an Educational Ontology for Java Programming (JLEO) with a Hybrid Methodology Derived from Conventional Software Engineering Process Models**," International Journal of Information and Education Technology vol. 4, no. 4, pp. 308-312, 2014. http://www.ijiet.org/show-47-465-1.html

2. Santhosh John, Nazaraf Shah, and Leonid Smalov, "**Incremental and Iterative Agile Methodology (IIAM): Hybrid Approach for Ontology Design towards Semantic Web Based Educational Systems Development**," International Journal of Knowledge Engineering vol. 2, no. 1, pp. 13-19, 2016. www.ijke.org/list-41-1.html

3. Santhosh John, Nazaraf Shah, and Mary Shanthi Rani ***"Proposal of an Hybrid Methodology for Ontology Development By Extending The Process Models Of Software Engineering"*** in an International Journal of Information Technology Convergence and Services vol.6 no.1, pp 37-44,2016. http://aircconline.com/ijitcs/V6N1/6116ijitcs04.pdf

4. Santhosh John, Nazaraf Shah, Craig Stewart "***Software Centric Innovative Methodology for Ontology development (SCIM)***" Proceedings of the 9th International ACM Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2017) – vol.2: KEOD in Madeira, Portugal Nov 1-3, 2017.pp139-146.ISBN:978-989-758-272-1,SCITEPRESS. https://www.scitepress.org/papers/2017/64829/64829.pdf

5. Santhosh John, Nazaraf Shah, Craig Stewart "***Towards a Software Centric Approach for Ontology Development: Novel Methodology and its Application***" in the proceedings of 15th IEEE International Conference on E Business Engineering – (ICEBE-2018), Xi'an, China, October 12-14, 2018.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
API	Application Program Interface
BRD	Business Requirements Documentation
CQ	Competency Questions
DE	Domain Expert
DOG	Domain Ontology Graph
DS	Design Science
EXPLODE	Extreme Programming for Lightweight Ontology Development
FIPA	Foundation for Intelligent Physical Agents
GQM	Goal-Question-Metric
GT	Glossary of Terms
ICT	Information and Communication Technology
IP	Internet programming
IS	Information System
ITP	Introduction to Programming
JAD	Joint Application Development
JLEO	Java Learning Educational Ontology
JOD	Joint Ontology Development
KE	Knowledge Engineer
KIF	Knowledge Interchange Format
LBS	Location Based Services
MOF	Meta Object Facility
ODE	Ontology Development Environment

ODLC	Ontology Development Life Cycle
ODT	Ontology Development Team
OE	Ontology Engineering
OIL	Ontology Interchange Language
OML	Ontology Markup Language
OOP	Object Oriented Programming
ORSD	Ontology Requirement Specification Document
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RUP	Rational Unified Process
SCIM	Software Centric Innovative Methodology
SDLC	Software Development Life Cycle
SE	Software Engineering
SHOE	Simple HTML Ontology Extension
TOVE	TOronto Virtual Enterprise
UML	Unified Modelling Language
XML	Extended Markup Language
XP.K	eXtreme Programming of Knowledge-based systems

1. Introduction

1.1 Chapter Overview

This chapter provides a detailed introduction to the research context, research problem and research objectives. Section 1.2 gives a brief introduction to the research context. Section 1.3 describes the research background in terms of the underpinning concepts of the research domain. Section 1.4 covers the research problem, research objectives, research questions and the scope of the research. Section 1.5 provides the outline of the thesis structure followed by the chapter summary.

1.2 Introduction to the Research Context

Ontology lies at the heart of W3C's Semantic web vision, which was: *the concept that allows entities to be shared and reused across application and enterprise domains*. It aims to provide a web of linked data and refers to a set of mechanisms for connecting structured data on the web. The adoption of best practice in the interrelationship of entities has led to the extension of the current web to a semantic-based web, with a global-information space from diverse domains, such as people, companies, books, scientific publications, films, music, television & radio programs, proteins, drugs & clinical trials, online communities, statistical and scientific data, and more (Bizer, Heath and Berners-Lee 2009).

The emergence of aspects such as linked data and semantic-web vision, aim to integrate and convert the vast amount of information available on the Internet into a machine-understandable network. This enables knowledge sharing and reusability across domains by making use of knowledge representation. Ontologies are considered to be the corner stone of the emerging semantic web vision. In terms of knowledge representation, a knowledge base starts from where ontology ends, ensuring the existence of a common vocabulary to share information in a particular domain. This includes a machine-interpretable glossary of domain concepts and the semantic structural-relations between them. Ontology in its technical perspective represents an artefact to achieve a computational objective which enables the modelling of a real or imaginary knowledge domain (Weller 2010). Ontologies have been developed for various domains in the past few years to progress towards the development of the semantic web. Though a lot of developments have happened in ontology domain, there is still a lack of a standardised methodology for ontology development. Researches in this direction have pointed out that the availability of an effective ontology-development

methodology could bring the semantic-web vision much closer for the user community. This research proposes an innovative methodology for the design and development of domain ontology.

1.3 Research Background

1.3.1 Ontology

The concept of ontology in its early stages first appeared in the domain of philosophy. Ontology refers to the systematic study of things that in general exist and how they relate to each other (Harper Collins 2005). It seems that the information systems in the domain of philosophy have borrowed the aspect of ontology from the domain of philosophy and interpreted it to be commensurate with it (ZWiga 2001). Furthermore, this aspect has been introduced to different contexts inclusive of knowledge engineering (Chandrasekaran, Josephson and Richard 1999). As well as being philosophical, in knowledge engineering, ontology is also referred to as a formal explicit specification of a conceptualisation (Gruber 1993). Moreover, ontologies are referred to as a formal and structured representation of a set of domain-specific terms (concepts) and the structural relationships among them.

Mizoguchi (Mizoguchi 1998) argued that an ‘‘Ontology provides a common vocabulary, and a making explicit of what has been often been left as implicit’’. As stated in the above cited resource, the linked data, systematisation and standardisation of knowledge, constitutes the backbone of knowledge representation within a knowledge-based system. In spite of the variations in the definitions of ontology, most of the literature has appraised the capability of ontology for its explicit clarification of domain concepts. It enables a shared understanding of domains among people and machine, by bringing about concepts from the terminologies of the domain, derived from a hierarchy of concepts. Figure 1 shows the role of ontologies in both semantic and pragmatic forms. This model was proposed during the 2007 Maryland ontology summit that was held in the United States.

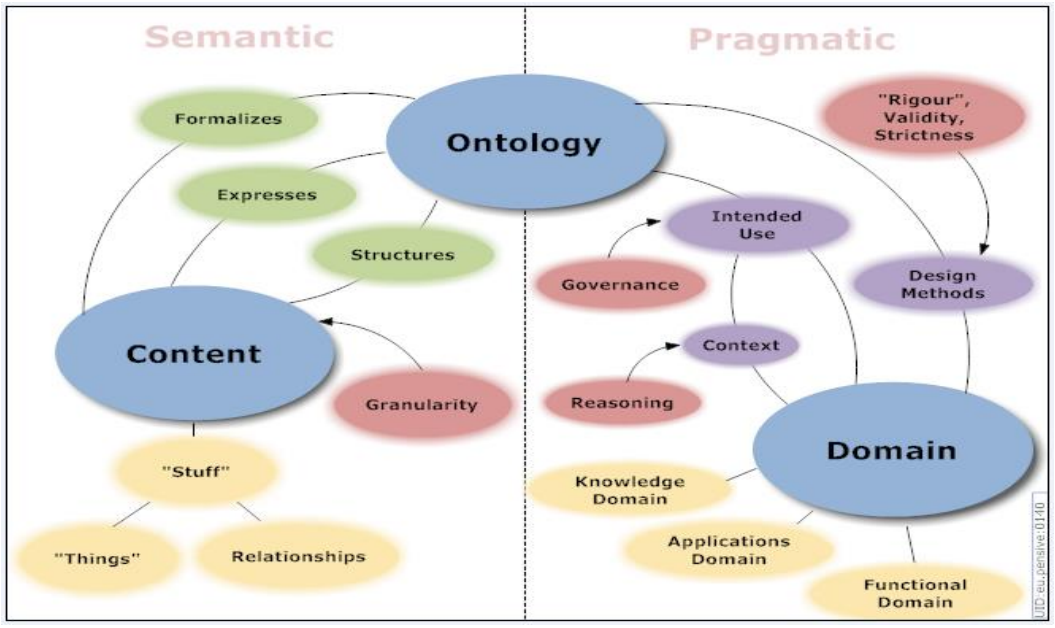


Figure 1: Ontology Dimension Map (Keet 2009)

1.3.2 Ontology Engineering

Knowledge engineering is a specific field in the artificial-intelligence domain which exclusively deals with development of knowledge-based systems. The role of ontology in knowledge engineering is vital for knowledge representation and knowledge management. Conceptually, a knowledge base starts from the point where ontology ends. Many ontology-based knowledge engineering/representation initiatives form part of the progress of the semantic web vision of W3C. Artificial Intelligence (AI) based expert systems compared to all other AI systems, have proven the significance of a knowledge-based system for solving real-world problems. Expert systems adopted a rule-based approach and knowledge engineering mainly utilised the power of a rule-based approach during the early stages of its growth. The literature has revealed that practitioners noticed the difficulty in the maintenance of a rule-based approach for the sharing and reuse of knowledge. As a result of the efforts to overcome the highlighted difficulties, knowledge engineering has started to evolve from rule-based to knowledge modelling and then to knowledge-based approaches. Knowledge-based systems have been a great advantage to the knowledge-engineering domain, however, obstacles still exist in the implementation of this technology in order to realise its full potential. Two major challenges in this direction are the lack of a mechanism to state the domain assumptions explicitly, and the absence of a meta-knowledge base.

Ontology engineering, a branch of knowledge engineering, mainly deals with the formal principles to build ontologies. This includes processes such as: development, management, analysis and reuse of ontologies. Methods, methodologies and a diverse set of tools are used for creating, editing and visualising ontologies. Ontology offers an opportunity to explicitly specify the domain assumptions. Therefore, ontology engineering provides a means to solve the inter-operability problems brought about by semantic obstacles, i.e. the obstacles related to the definitions of business terms and software classes (Pouchard, Ivezic and Schlenoff 2000). Guarino, Staab and Oberle (2009) stated that, "the process of building engineering ontologies to use in information systems remains an arcane art-form, which must become a rigorous engineering discipline". In the literature on ontology engineering, knowledge engineering is considered as its predecessor, though there exists a narrow line between knowledge-bases and ontologies.

1.3.3 Ontology Classifications

It has been observed from the literature that several classifications of ontology have been proposed (Lassila and McGuinness 2001,Gómez-Pérez, González and Lama 2004). Different strategies have been followed for ontology classification. For instance, one approach classified ontologies into representation and content ontologies. Representation ontologies focused on providing a framework, whereas content ontologies represent conceptualisation. The approach of classification more suitable to this research, has been based on: the expressivity of ontologies, the formality of the languages used, and the scope of the objects described by the ontology (Roussey et al. 2011). Section 1.3.3.1 discusses the classification based on language expressivity and formality, and Section 1.3.3.2 discusses the classification based on the scope of the objects described by the ontology.

1.3.3.1 Ontology classification based on language expressivity and formality

Based on the expressivity of ontology, different kinds of ontology components have been defined (e.g. concepts, properties, instances, axioms, etc.). Concepts, instances and properties are referenced by one or more symbols. Symbols are terms that humans, by reading them, can easily understand. Ontology components are connected through structural semantic-relationships. Four kinds of ontologies are described below,

Information Ontologies

Information ontologies focus on concepts, relationships and instances. They are composed of diagrams/sketches for clarity and are meant for humans. Information ontologies can be a useful tool during the inception phase of information-systems development. Visual languages such as Mind-Map are used to describe information ontologies as they can easily be understood by humans. A Mind-Map plug-in called Mind2Onto, acts as the ontology editor called OntoEdit (Sure, Angele and Staab 2003). Mind-Map is one of the effective mechanisms for visualising, generating structure and classifying ideas.

Linguistic and Terminological Ontologies

Linguistic ontologies are good for: concept clarification and knowledge sharing,classification systems, taxonomies and thesauri, data exchange, and data models. Terminological Ontologies mainly focus on concepts and the structural relationships between them. The main usage of this kind of ontology is to present and define the vocabulary used and to make an agreement between user communities. This agreement defines the term to be used to represent a concept with minimum ambiguity. This process is known as vocabulary normalisation which assists in the selection of the preferred term when a concept could be described by two synonym terms. Two languages that have the capability to express terminological ontologies are Simple Knowledge Organization Systems (SKOS) and the Resource Description Framework (RDF). For example, Urbamet (Guyot, Falquet and Teller 2010), which was a thesaurus developed and maintained by the French Centre for Urban Documentation.

Software Ontologies

Software ontologies are implementation-driven ontologies offering a conceptual schema and are focused on both the description and manipulation of data with the aim of absolute data consistency. In software ontologies, data is stored in the object properties (i.e. instance) so that it can be processed by methods. These kinds of ontologies are defined with conceptual-modelling languages used in software engineering, such as Unified Modelling Language (UML). The semantics of UML are mainly composed of informal descriptions in English (Donald Bell 2003). In the literature it was noted that UML alone was not sufficient to represent all complex reasoning processes (Cranefield 2001) like computation of the logical correctness of a formal ontology and deduction of new knowledge etc. The Meta Object Facility (MOF) model, designed by the Object Management Group (OMG), and the Meta Object Facility (MOF) described in the work of Columb et al. (2006) support various ontology-representation languages such as RDF and the Web Ontology Language (OWL).

MOF tools use metamodels to generate code for managing models and metadata. Therefore, as observed by John (2010), in March 2003, the OMG issued a request-for-proposal for an Ontology Definition Metamodel (ODM) which specified the requirement for: a specification of a MOF 2.0 (MOF2) compliant metamodel; a UML 2.0 profile; and any additional information needed to support the development of software ontologies, using UML modelling tools.

Formal Ontologies

Formal ontologies require unambiguous semantics for the language used to define the concepts, clear distinctions between concepts, and concrete rules, to define concepts and relationships. The meaning of the concept is clearly guaranteed by formal semantics (Borgo 2004). The presence of a logical definition is the highlight of this type of ontology and it is only available in formal ontologies. The logical definition of a concept is composed of one or more axioms, which are a combination of concepts and structural relationships. A knowledge base contains more expressive components than a conceptual schema. As well as being used for the storage and retrieval of data, reasoning was the purpose of formal ontologies, for example, the formal ontology for the Korean Architectural Domain, known as CoBra, was defined to facilitate the pervasive computing environment (Chen, Finin and Joshi 2003).

1.3.3.2 Ontology classification based on the scope of Ontology

The scope of ontology is another aspect of ontology classification. For example, the scope of a local ontology is smaller compared than the scope of a domain ontology. Domain ontologies are more specific than core-reference ontologies, which contain the fundamental concept of a domain. Foundational ontologies are meta-ontologies that describe abstract-level concepts used to define other ontologies. Figure 2 shows the classification of ontologies based on the scope.

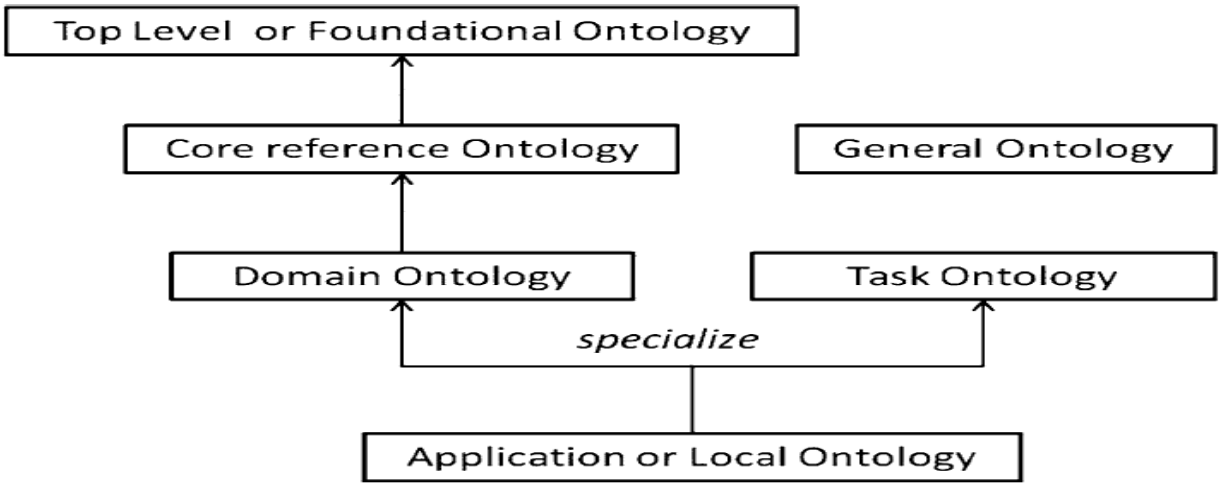


Figure 2: Ontology classification based on domain scope

Application and Domain Ontologies

Application ontologies are specialised domain ontologies engineered for a specific use or application, with minimal knowledge-sharing. These types of ontologies ideally represent the single viewpoint of a user. Mostly, it is present in its occurrences as a combination of both domain and task ontologies for the fulfilment of a specific application. These are a kind of closed ontology, where, from a critical-review perspective, its existence is questioned by the availability of domain ontologies.

Core-reference Ontology

A core-reference ontology is an ontology standard used by a different group of users. This ontology is associated with a domain, but it integrates different viewpoints related to a specific group of users. It is an integration of domain ontologies and is often built to identify the central concepts and relations of a domain.

General Ontology

General ontologies are not dedicated to a specific domain or fields, but they model the general knowledge of a huge area and are really closer to a knowledge base, e.g. OpenCyc Ontology, which is a knowledge base and common-sense reasoning engine. This ontology contains a number of terms, together with assertions which relate the terms to each other, forming a general ontology.

Top level Ontologies

Foundational/Top level ontologies are generic in nature and they are applicable to various domains. They are used to define ontology notions such as: objects, relations, events, processes, etc. Foundational ontologies can be compared with a metamodel of a conceptual

schema (Fonseca, Câmara and Davis Jr. 2003). Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), and Basic Formal Ontology (BFO) in the paper by Gangem et al. (2002), are the leading top-level ontologies which propose a different logical theory for representation of the 'world' assumption.

1.3.4 Ontology and Semantic Web

The semantic web vision of W3C is considered as an extension of the current Web. As per the aim and design goals of the semantic web, machines can understand the semantics of web contents, which enhance information discovery and search capability. Hence, it allows information retrieval in an intelligent manner (Lukasiewicz and Straccia 2008). Ontologies offer the capability to add machine-understandable metadata to web resources. Specifically, ontologies assist in the definition of concepts for web resources identified using the Uniform Resource Identifier (URI), which use software agents to process information, to achieve the expected semantic interoperability. Attempts of semantic-web vision implementations, accommodate objects of ontologies as a triple model of subject, predicate and object, known as RDF databases (Ding et al. 2005). The semantic web architecture includes layers, as shown in the popular semantic-web layer cake in Figure 3, and illustrates how the relevant technologies are arranged on the semantic web. It can be seen that the ontologies are placed at the heart of the layer cake. Practically, ontology defines the concepts which allow queries to be exchanged between software agents. (Cai, Eske and Wang 2003).

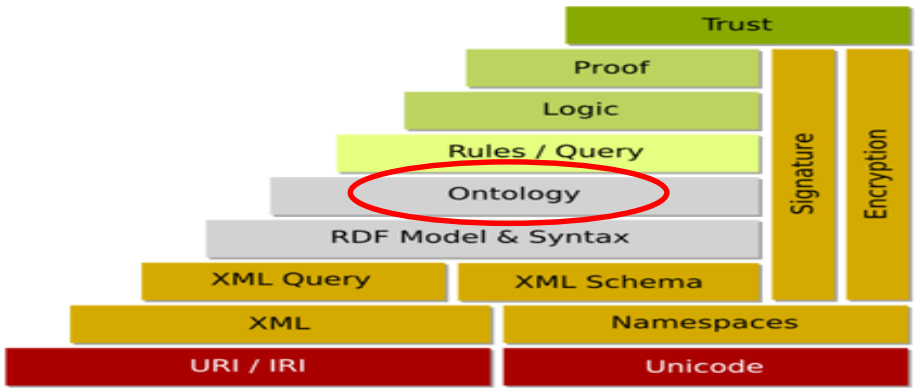


Figure 3: Semantic web layer cake (Burieson 2007)

Ontology offers an explicit common vocabulary for a specific domain, carried out by describing the concepts and classes of domain concerned, the encoding of the domain vocabulary being provided by an ontology language. Knowledge engineers and artificial-

intelligence support are provided to model some world domains including labelled concepts, attributes and relationships, sorted into specialisation hierarchies (Islam and Sheik 2015).

The semantic web offers the capability to facilitate knowledge management, to promote semantic interoperability between systems, to improve representation, to allow sharing and to provide considerable reuse of information to support decision-making. The semantic Web technologies have become the preferred and crucial alternative for managing the complexity of interoperable information-sharing for applications. A semantic web is a concept for information processing over linked relational-data (Berners-Lee 2006). It is expected that ontologies will play a significant role in various application domains on the Semantic Web due to the increased number of industrial projects. These projects have been chosen to formalise application knowledge using ontologies and semantic web representation languages such as RDFS, OWL, or WSML. Ding et al (2005) classified semantic-web ontologies into four core categories: a) Meta ontologies (containing small vocabulary, axioms and languages for ontology representation such as OWL, RDF and RDF Schema); b) Upper ontologies (giving a higher level structural representation of the things); c) Domain ontologies (providing an explicit vocabulary for concerned domains); and d) Simple/specialised ontologies (which emphasise commonly used concepts of a domain that can have a generic nature and can be used for knowledge sharing).

1.4 Motivation and problem

This section describes the current problems and open issues in the field of ontology engineering which motivated this research and lead to the formulation of the research problem, research objectives and research questions.

1.4.1 Research problem

Ontology engineering is an emerging field in computer science, which deals with the methods, methodologies and tools for building and managing ontologies. This branch of engineering aims at making explicit knowledge contained within software applications, enterprises and business procedures, for a particular domain. Though ontology engineering and software engineering are complementary, the level of maturity of software engineering is higher than ontology engineering. The rigorous development process for ontology building requires the use of methodologies and platforms that are equivalent to software development. A methodology with fewer learning curves for software engineers can make ontology

development appropriate for business users. This research tries to address the research gap by attempting to combine the mature software-engineering process models with the methodologies of ontology engineering. The outcomes of this research will be a solution to the problem on how to bridge the gap, with a reduced learning curve, between the complementary technologies of software and ontology engineering. Therefore, the problem that this research aims to solve is:

How to develop a methodology for ontology development which bridges the gap between the complementary engineering branches-Software Engineering (SE) and Ontology Engineering (OE)?

1.4.2 Research Objectives

The intent of this research is to evaluate the existing ontology-development methodologies and to propose a new methodology which bridges the gap between software engineering and ontology engineering. The following are the research objectives that guide the development of this research.

1. To analyse the existing Ontology Development Methodologies.
2. To propose a Software-Centric methodology for Domain Ontology Development
3. To define an Ontology-Development Life Cycle (ODLC) for the proposed methodology.
4. By using appropriate tools apply the proposed methodology to design domain ontology.
5. To measure the effectiveness of the methodology proposed by validating against existing methodologies.
6. To disseminate the research findings to high quality international conferences and high-impact journals.

1.4.3 Research Questions

The following are the research questions that the researcher had formulated with respect to the research problem mentioned above.

1. What are the limitations of existing ontology development methodologies?
2. What is the scope for developing a novel methodology for domain-ontology development by extending software-engineering process models and methodologies?
3. Can an Ontology Development Life Cycle (ODLC) be defined for the proposed methodology?
4. Can the proposed methodology be applied to develop an ontology for a specific case?
5. How can the proposed methodology for ontology development be reliably evaluated?

1.4.4 Scope of the research

This research considers the proposal of a Software-Centric Innovative Methodology (SCIM) for ontology development. This will comprise the components, the development life cycle, the application of the ontology to a specific domain and the validation of the proposed methodology by comparison with the leading existing ontology-development methodologies.

1.5 Structure of the thesis

This thesis has been structured in the following way for the convenience of readers.
Chapter 2 reviews the literature relevant to the subject of investigation. This includes the following main areas:

- Ontology-Representation languages and tools;
- Ontology-development methodologies;
- Ontology-development platforms;
- Domain-Ontology development; and
- Educational Ontologies

Chapter 3 describes the research methodology and research design used for this research, giving details of the customised design-science approach which has been discussed in detail with the necessary illustrations.

Chapter 4 introduces the proposed SCIM for ontology development with the background, components and Ontology Development Life Cycle (ODLC). A detailed description of the

hierarchy of components and an illustration of the final framework of SCIM has been incorporated in this chapter.

Chapter 5 describes the application of SCIM to a specific domain with the appropriate evidence. A prototype Java Learning Educational Ontology (JLEO) has been developed as the proof of concept. An appropriate ontology editor has been used for the modelling and necessary screenshots of the JLEO and has been incorporated in this chapter.

Chapter 6 covers the evaluation of SCIM against existing methodologies. An illustration of the evaluation framework used has been incorporated in this chapter. The details and results of the evaluation techniques applied have been described in this chapter.

Chapter 7 describes the conclusions, contributions, future directions, and limitations of this research, together with the reflections of the author.

1.6 Chapter summary

This chapter, as background for this research, gives a brief introduction to ontologies, ontology engineering and the use of ontologies for the semantic web. This is followed by an explicit definition of the problem, research questions and objectives.

2. Literature review

For the success of any category of research, one of the integral parts is the review of existing literature. In the case of exploratory research, the literature review plays a pivotal role in providing solutions which add value to existing theories/solutions which can only be proposed after a thorough review of the literature published in reliable resources.

2.1 Chapter Overview

The literature review offers an opportunity to the researcher to collect and critically analyse existing publications and to proposing new and innovative ways for addressing a specified research problem. Though the organisation of a literature review depends on its purpose and nature, it provides a thorough exploration of previous and current work. Since this research is exploratory in nature, review of the appropriate literatures will ensure that the research is properly justified. This research is concerned with the proposal of SCIM and its application to a specific case as a proof of concept. SCIM is a methodology with a defined ontology development life cycle (ODLC) and follows a hybrid model of proven software engineering process models. In this chapter, the literature relevant to the research domain has been reviewed and is divided into the following subsections.

Section 2.2 gives an historical overview of the ontology representation for tools and tool-dependent languages, used from the 80's and 90's up to the present day. Section 2.3 provides a critical review of the strengths and weaknesses of existing ontology-development methodologies. Book chapters, publications from the proceedings of highly ranked domain specific conferences and journals have been used for the review. Section 2.4 describes three popular ontology-development environments with the intention of finding the most appropriate choice for the proposed prototype-ontology development. Section 2.5 analyses the underpinning elements of four selected cases of domain ontology development, detailing the methodology followed and the ontology development process. The cases have been chosen from four different domains which review the components from an application perspective. The last section concludes the literature review with a chapter summary and links to the subsequent chapters of the thesis.

2.2 Ontology Representation Languages and Tools

The subsections below examine the scope, merits and limitations of the various ontology representation languages and tools, from the early ontology languages to recent ontology-representation formalisms.

2.2.1 Early Ontology Languages

From the beginning of the 1980's, various ontology representation languages have been introduced into the field of ontology engineering which have been accepted by practitioners. The literature revealed that in the 1980's, ontology languages were tool-dependent. The popular ontology-development tools of that era such as CLASSIC, KL-ONE and LOOM had their own native languages for knowledge representation. Though they supported ontology representation within the context of their native development tools, they did not claim to be a generic ontology development approach. In early 90's, Ontolingua (Gruber 1994), was proposed and developed by Knowledge Systems Lab (KSL) of Stanford University for ontology representation and the sharing of ontology. It was able to create, manage and exchange ontologies and had the capability to use and support a variety of ontology-representation languages compared to its predecessors (Bruijn 2003). It encompassed a frame-like representation and provided facilities for translation to the Knowledge Interchange Format (KIF and was used to translate from/to description logic languages such as Loom and Epikit (Ibrahim and Ataelfadiel 2017). Even though Ontolingua did not have inference functionality, it provided a set of ontology development functions and a library of modular and reusable ontologies (Gruber 1994) and since its introduction has been a key language for ontology representation. The issue of interoperability was one of the major problems faced by this language, but this has now been solved with the arrival of XML based languages (Brunnlieb and Holzer 2016).

2.2.2 Simple HTML Ontology Extension (SHOE)

Ontology Mark-up Language (OML) has been developed by the University of Maryland as an extension to HTML. SHOE makes it possible for agents to gather meaningful information about web pages and documents and for improvising search mechanisms (Luke et al. 1997). This has been achieved by incorporating machine-readable semantic knowledge in either HTML or other web documents. Later SHOE syntax was combined with XML by a three-phase process. These phases are: *a) Define ontology; b) Annotate HTML pages with*

ontological information and, c) Semantically retrieve information by searching all existing pages.

2.2.3 Ontology Markup Language (OML)

OML was considered as an XML serialization of SHOE (Kent 1999). Therefore, OML and SHOE share many features (Gómez-Pérez and Corcho 2002). OML exists in four different levels such as OML Core, Simple OML, Abbreviated OML and Standard OML. OML Core is related to the logical aspects of the language and is included in the rest of the layers. Simple OML maps directly to RDF(S), whereas, abbreviated OML encompasses conceptual graph features. Out of all the four levels of OML, standard OML is the most expressive version, but, unlike its predecessors, there are no other native tools available for the authoring of OML ontologies other than the general XML editing tools. Although OML had advantages over SHOE, it could not be considered as a W3C standard.

2.2.4 Ontology Interchange Language (OIL)

This Ontology representation language was introduced for the development of the ***Onto Knowledge*** project. OIL was considered as the first W3C standard-based ontology-representation language (Cover 2002). Semantic interoperability was permitted by OIL and offered a web-oriented ontology representation and an inference layer. It is a basic modelling approach, being used in web-oriented ontology development such as classes/concepts, hierarchy of concepts, relationships etc. and supports description logic, reasoning and formal semantics. Description logic (DL) provides the interchange that describes language concepts and restricted roles that form taxonomy classifications of knowledge. XML has been used to express OIL Syntaxes and ASCII used for presentations. (Gómez-Pérez and Corcho 2002).

2.2.5 DARP Agent Markup Language (DAML) +OIL

The variation, DAML+OIL language is designed and developed by a joint committee from the US and European Union in for DAML, a DARPA project for allowing semantic interoperability in XML (Gómez-Pérez and Corcho 2002). Therefore, DAML+OIL and OIL, both being built on DRFs, share the same objectives. The tools used for authoring DAML+OIL ontologies are OILEd, WebODE, OntoEdit and Protégé.

2.2.6 Resource Description Framework (RDF) and RDF Schema

RDF was W3C's first formal representation language and was developed exclusively for describing the resources from the web. It was used to offer a unique triple model and a graph with nodes and binary relationships (Brickley and Guha 2001), the graph model being a semantic network model (Ding et al. 2005). RDF was used for annotating web resources with machine understandable metadata. It was also used as a formalism to express knowledge in a limited way (Bruijn 2003) using XML syntax for definitions, and uses *Subject*, *Predicate* and *Object* to represent statements. These concepts are described in the following paragraphs.

- ***A Subject****:* can be any physical/logical thing that can be identified using a Uniform Resource Identifier (URI). The list includes webpages and individual XML elements.
- ***Predicate:*** a named resource and can be used as a property of a thing such as age or name.
- ***Object:*** a combination of resource, property and property value. (Cai, Eske and Wang 2003). Despite RDF having ontology expressive power, it did not provide mechanisms for defining the relationships between properties and resources.

Figure 4 shows the triple model and graph of a simple RDF document.

```
<? xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://www.openhealth.org/RDF/rdfExtractify.xsl">
<dc:author>Santhosh John</dc:author>
<dc:title>Researcher</dc:title>
</rdf:Description>
</rdf:RDF>
```

Subject	Predicate	Object
http://www.openhealth.org/RDF/rdfExtractify.xsl	http://purl.org/dc/elements/1.1/author	"Santhosh John"
http://www.openhealth.org/RDF/rdfExtractify.xsl	http://purl.org/dc/elements/1.1/title	"Researcher"

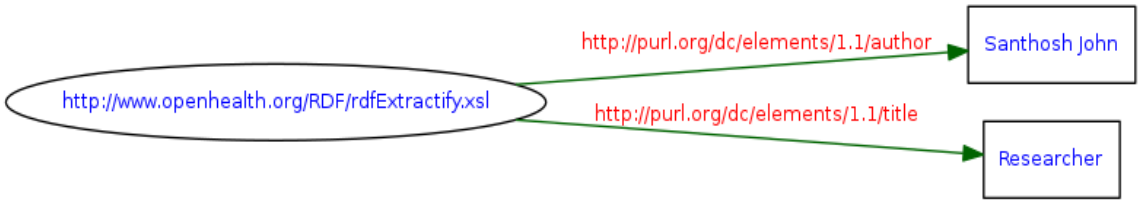


Figure 4: Simple RDF document, triple model and graph

An RDF Schema (RDFS) is the vocabulary used to effectively support the classification and definition of RDF. It is greatly influenced by the aspects of Object-Oriented Modelling and frame systems and arranges knowledge conceptually with the assistance of ontology building-blocks such as concept, facet, and slot. RDFS attached with an inherited axiom, allows the representation of a domain at various abstraction levels with a strong emphasis on concepts/entities. It offers constructs that enforce a minimal concept of dependent relations (Bruijn 2003), and is widely used as a representation format in tools such as Amaya, Protégé, and Mozila Siril (Gómez-Pérez and Corcho 2002).

2.2.7 Web Ontology Language (OWL)

OWL is the latest W3C standard to develop ontologies for any applications which share the need for information to be machine-readable and understandable (Group 2012). It has been built on top of its predecessors DAML+ and OIL, and designed like other standards such as XML, RDF and DAML. There are three variations of OWL available and all of them support solutions for issues relating to the expressiveness and complexity of ontology constructs. **OWL-Lite,** is a variation of OWL used for building a basic structural model which incorporates classes/subclass relationships, properties, and constraints. **OWL-DL,** another variation of OWL, focused on formal semantics which appear commonly in modern ontology development. This variation is more associated with description logics that offer add-on grammar constructs such as conjunction, disjunction, and negation, suitable for knowledge

capture. This version of OWL makes use of all OWL ontology constructs in a limited manner. **OWL-Full,** yet another variation of OWL is the version with the highest level of expressiveness, with no usage restrictions. Table 1 gives a quick comparison am for various XML-based ontology languages.

Language	W3C Support	Expressive Power	Editors	Other Supporting Tools / Languages
SHOE	×	Low	RFEdit	PIQ SHOE Search
RDF/RDFS	√	Medium	Protégée, NeOnToolkit, DOE, pOWL, Swoop, TopBraid	Jena SPARQL, RD-QL,
OIL	√	Low	OilEd	Fact
DAML + OIL	√	Low	OntoEdit Ontolingua, WebODE, IsaViz	DQL, Pellet, Racer
OWL	√	Strong	Protégé, DOE, Altova Semantic Works, WebODE, Swoop, Morla	Jena, OWL-QL, Fact++, Pellet, RacerPro

Table 1: Comparison of Ontology Languages (Islam and Sheik 2015)

2.3 Ontology Development Methodologies

A methodology is a "comprehensive set of organised methods and procedures for creating a general-system theory of how a class of thought-intensive tasks are to be performed" (IEEE Standard Glossary of Software Engineering 1990). Thus, a methodology in ontology engineering is composed of methods, techniques, processes and activities (Roussey et al. 2011) and may follow several approaches to develop ontology. Many methodologies have been introduced by various practitioners for ontology development in the past. However, due to many reasons, none of them could be considered to be standardised. One of the main reasons a standardised methodology has not been developed for ontology engineering is that most are project specific in nature. Little priority seems to have been given to persuade other domain ontology developers to standardise an existing methodology. It has been observed from the literature, that most of the methodologies proposed in ontology engineering, were lacking

details of the techniques and activities employed, with the appropriate mapping of the underpinning philosophy and approaches (Loppez 1999, Natalya F. Noy 2001).

It has also been observed from the related literature that most of the existing methodologies have emerged from the experience gained after developing ontologies for many different domains. The existing ontology development methodologies can be categorised into *forward engineering* and *backward engineering*. Forward engineering includes: analysing knowledge resources, extracting the concepts, and using domain experts to establish the relations between the concepts. Examples of forward engineered methodologies are:

- Enterprise Methodology – developed by Uschold and King (1995);
- TOronto Virtual Enterprise (TOVE) developed by the Enterprise Integration Laboratory – EIL (2002);
- METHONTOLOGY (Lopez, Gomez-perez and Sierrra 1999); and
- On-To-Knowledge (Staab 2001) are examples of available forward engineering methodologies.

Most of these methodologies define high-level ontology development phases (Noy and McGuinness 2001).

Backward engineering ontology-development methodologies include semi-automatic processes which extract noun/verbs from partially structured or structured text resources, using natural language processing (NLP) techniques, and construct ontologies which use machine-learning algorithms (Staab, Schnurr and Sure 2001). It has been observed that backward engineering is more suited to the development of knowledge-domain ontologies. However, the resources of the domain are rare and the extraction of noun terms from them is extremely difficult. Therefore, backward methodologies were only used for ontology development on very limited and specific projects.

The development of ontologies is more complex compared to any other kind of typical software project owing to factors which included: the necessity for sophisticated tool support and heterogeneous platforms, dynamic changes in business needs, a lack of performance engineering and low failure tolerance. The existing ontology-development methodologies and tools provide only an ad-hoc approach with limited functionality and performance. To address this gap, researchers of the ontology-engineering domain, devised a considerable number of innovative ideas and proposals for methodologies and tools. This process of developing ontologies has created a network of reusable ontologies which are available in online repositories (Sua´rez-Figueroa, Go´mez-Pe´rez, and Ferna´ndez-Lo´pez 2012).

With their Enterprise Ontology, Ushold and King (1995) were the first to propose their own methodology, instead of making use of the available guidelines. Though this methodology claimed to be the first methodology for ontology development, it failed to describe the techniques and activities in a precise manner. A better approach, for ontology construction, based on a logical model of knowledge followed, designed by Gruninger and Fox (1995). Prior to creating the model, an informal specification description was made which was then formalised using an ontological approach. Activities and techniques used in this methodology lacked sufficient detail and remained abstract (Lopez 1999). Unlike the predecessors, the METHONTOLOGY methodology (Fernández-López, Gómez-Pérez and Juristo 1997) was introduced by the Foundation for Intelligent Physical Agents (FIPA), employing from the beginning, activities and techniques for ontology engineering, to build domain ontologies (Lopez 1999). It categorised ontology-development activities as; specification, conceptualisation, formalisation, integration and implementation. Besides these activities, knowledge acquisition, evaluation, integration and documentation were also performed in parallel. The engineering behind METHONTOLOGY was the intermediate representation of elements for different models such as: the specification model (a semi-formal specification using a set of intermediate representations); the conceptual model; and a formalised model (e.g., Description Logic Ontology UML Profile) which was implemented in an ontology implementation language (i.e. Web Ontology Language). Although tested by developing ontologies for various domains, the reusability of ontologies was not properly addressed by the METHONTOLOGY methodology.

The importance of reusability in ontology engineering could not be denied as ontology development was a complicated and a time-consuming task. To address the problem of reusability, the Ontolingua server (Farquhar, Fikes and Rice 1997) was introduced which offered a library of pre-defined ontologies for the user who could reuse/redesign these ontologies and then extend the library by adding them to the server. Though the Ontolingua server provided reusability, it was noted that it didn't provide the necessary details for mapping that would enable the conversion from one ontology to another, Methodologies such as CYC and SENSUS introduced the notion of reusability, but focussed on the natural language domain (Corcho, Fernandez-Lopez and Gomez-Perez 2003. Even though both the methodologies had phases, they fell short of recommending a life cycle and lacked details for the pre-development and post-development processes.

Two other ontology development methodologies such as Common KAD and KACTUS focused on knowledge management solutions (Jones, Bench-Capon and Visser 1998). Common KAD was a widely used methodology for developing ontology-oriented knowledge base systems. KACTUS methodology, a follow up project of Common KAD focussed on ontology development which adopted an engineering approach emphasising modular design, redesign and reusability (Schreiber, Wielinga and Jansweijer 1995). This methodology gave more weight to the reusability of ontologies developed for the use of different applications in a domain, such as electrical networks, oil production platforms and the construction design of large ships (Szturcova and Rapant 2013). However, it didn't offer any support for collaborative ontology development and life cycle management.

Another methodology specifically proposed for handling enterprise solutions was the On-To-Knowledge methodology (Staab and Studer 2003), which was influenced by its predecessors KEM, Common KAD and METHONTOLOGY. It followed five steps for ontology development, which were: feasibility study, ontology kick off, refinement, application and evaluation. Although the methodology supported iterative development and lifecycle management, it didn't support reusability and collaborative ontology construction.

De Nicola, Missikoff and Navigli (2008) proposed a software engineering approach for Ontology building (UPON), combining stages and phases from software engineering and ontology engineering. This methodology focuses on the exploitation of the Unified Process (UP) and UML. It was a novel approach that recommended an iterative lifecycle for large scale ontology development, by combining the features of UP and UML. One of the weaknesses of the methodology though, was that it did not target the development of domain ontologies. It focused on ontologies that served specific consumers and automated systems and failed to provide comprehensive details for collaborative ontology-construction (Iqbal et al. 2013). However, the absence of agile techniques and the resulting complexity made it unsuitable for large-scale generic domain-ontology development.

Knublauch (2002) proposed the "eXtreme Programming of Knowledge-based systems" (XP.K) methodology, which was a lightweight agile methodology for the development of knowledge-base systems extended from Extreme Programming (XP). It followed the concept of XP but extended the value of communication to the community. Besides the practices of XP, additional practices were used for ontology development in XP.K. These consisted of: the use of an on-site domain expert; joint ontology design and pair modelling; the round-trip technique; engineering; modelling standards; shared symbols;

grounding; testing and constraint checking; use of a simple knowledge model, refactoring and design patterns; and planning games.

Extreme Programming for Lightweight Ontology Development (EXPLODE) was a methodology proposed by Hristozova (Hristozova and Sterling n.d.) for the agile development of lightweight ontologies. This methodology was extended from XP and had a set of rules and practices to be used when creating ontologies (Beck 2000). Practices such as collective ownership, small release, continuous integration, planning, metaphor, simple design, testing and refactoring practices were adopted for that methodology. Rapid OWL methodology (Soren and Herre 2006) promoted collaborative knowledge engineering, and was inspired by XP.K but is different because it focussed on the development of generic knowledge bases (Nicola and Missikoff 2016). This methodology encouraged joint ontology development, where domain experts became part time ontology engineers. Detailed comparison among the mentioned methodologies has been presented in Chapter 4 where the proposed methodology is introduced.

2.4 Ontology Development Platforms

Ontology-development platforms are time-consuming because designers/developers tend to pay more attention to abstract level aspects of ontology development instead of formal language syntax. This section describes briefly some of the popular ontology development environments.

2.4.1 SWOOP

SWOOP was built as a web ontology browser (Cardoso and Escórcio 2007). It is a free and open source web-based environment for the basic management of OWL ontologies. It offered a user-friendly environment for all the core stages of ontology design and development, such as browsing, editing, debugging and publishing interfaces (Slimani 2015). Figure 5 shows a screen shot of a SWOOP browser. The salient features of Swoop included the availability of a facility for the linking and importing of external ontologies. However, although it supported full importing, it didn't allow partial importing of external ontologies (Kalyanpur et al. 2005).

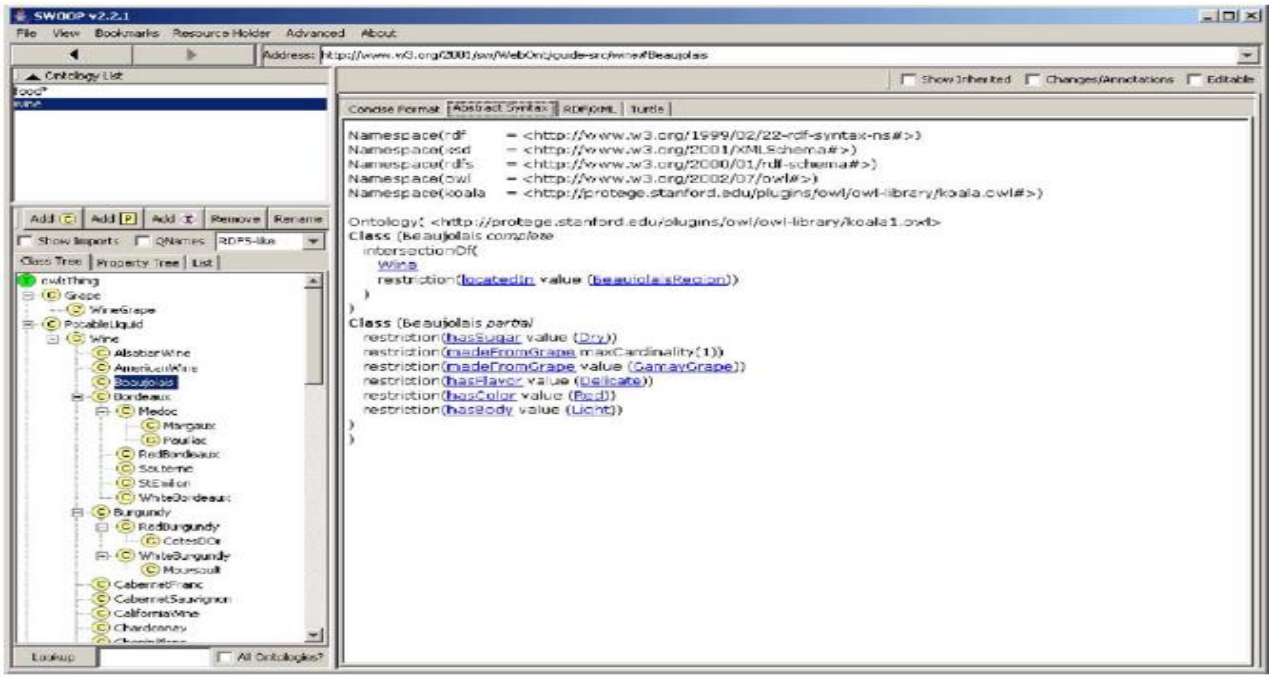


Figure 5: Screenshot of SWOOP browser

For domains with a large volume of data, ontology building was both difficult and time consuming. The other tools that are available to construct ontologies for domains with a huge set of concepts are TextToOnto, the ASIUM, OntoLT, OntoLearn and the Mo'k Workbench (Sivakumar and Swaminathan 2010). Most of these tools were exclusively built to serve the need of specific ontology models, so are not necessarily the best choice for developing all types of ontologies (Cimiano and VÄolker 2005).

2.4.2 ONTOGEN

ONTOGEN is a semi-automatic ontology editor with a user-friendly GUI (Fortuna, Grobelink and Maldenic 2006). It integrated both machine-learning and text-mining algorithms to overcome the complexity of ontology development, saving a considerable amount of time for ontology developers. The salient features of ONTOGEN supported both supervised and unsupervised methods for concept identification and naming, together with ontology and concept visualisation. By extending these features to domain experts, even those with limited skills in ontology engineering, this platform can save time for the development of ontologies.

2.4.3 PROTÉGÉ

PROTÉGÉ, an open source ontology-editing tool developed by Stanford University, is one of the well-used ontology editors with the capability of expandable platform-independent ontology building (Cardoso and Escórcio 2007). Different third-party plug-ins and Application Programming Interfaces (APIs) can be easily integrated with Protégé, which is a user-friendly graphical user-interface for the creation of knowledge bases. It supports a collaborative means for ontology development with Web Protégé, which is a powerful collaborative ontology-development environment for the Web (Alatrish 2013). In addition to the ability for implementing a set of knowledge modelling structures to support ontology creation, it also provides ontology management and visualization with the aid of add-ons like OWLViz. Customization is available to extend its support for knowledge model development and data entry (Alatrish 2013). Figure 6 shows a screenshot of the Protégé Ontology Editor.

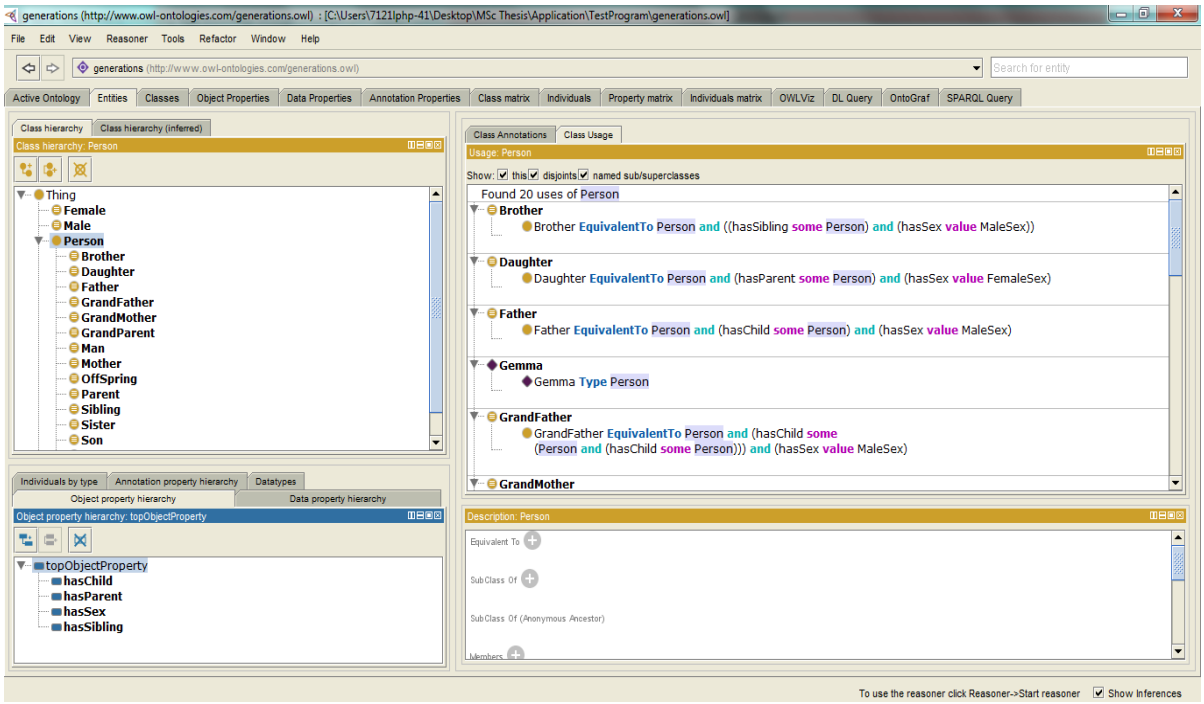


Figure 6: Screenshot of Protégé Ontology Editor

2.5 Domain Ontology Development-cases

Many ontology development attempts have been made by practitioners for different domains in the past. Domain ontology is developed with a specific domain in mind, preferably built using an existing top-level ontology, for the mapping and integration of different domain

ontologies. In this section, domain ontologies designed and developed for four different domains together with the underpinning methodologies, development processes, and tools and techniques used have been reviewed. The literatures for more than twenty domain ontologies have been considered and four have been chosen according to the requirements of this research.

2.5.1 E-Government Domain Ontology Development

An E-Government ontology development attempt was made by combining ontology-development methodologies and semantic-web platforms (Vincent, Dombeu and Huisan 2011). The ontology-based approach was suggested for the domain of E-government to address one of the primary challenges, which was the development of systems with easy integration and interoperability providing seamless service delivery to the public. According to the literature, semantic-web technologies with ontologies were a promising solution for the core engineering problems of E-Government. Uschold and King used a forward-engineering methodology which made use of two state-of-the-art semantic web platforms, which were; Protégé (as the ontology editor) and the Java Jena ontology API (as the high-level language for the implementation). These domain ontologies were primarily represented in human-understandable versions that can be processed by machines, and used by E-Government ontology-developers for ontology editing and implementing and they used semantic web machine-processable syntaxes, such as XML, RDF, and OWL. With the support of the latest software engineering techniques and agile methodologies, designers went for an optimal solution to the key problem in e-government which was the integration and interoperability of services.

Ontology domain

The government-service domain considered for the development of the E-government ontology was the monitoring and tracking of development projects in developing countries. Specifically, in developing countries as well as in Sub Saharan Africa (SSA), where almost every government department was involved in some way with the implementation of a programme aimed at improving the welfare of society. These programmes were commonly called development projects and included: infrastructure development, water supply, sanitation, education, rural development, health care, and ICT infrastructure development. Thus, applications that could interface with all the activities involved could bring tremendous advantages in E-Governance, for the implementation of development projects in a SSA

country. The title of the ontology was the Domain Ontology of Development Projects Monitoring (OntoDPM).

Ontology development process

The underpinning methodology prescribed the guidelines for the specification, conceptualisation, formalisation and implementation of the ontology (Calero, Ruiz and Piattini 2006). The specification phase defined the feasibility of the ontology such as the aim and role of the intended ontology as well as the detail of its intended users. During the conceptualisation phase, a conceptual model was built. In its simplest form, the conceptual model was represented graphically, where the vertices were the core concepts/entities of the domain, and the edges were lines interconnecting the pairs of vertices and representing the relationships between the concepts of the domain. Figure 7 shows the conceptual model of the OntoDPM ontology.

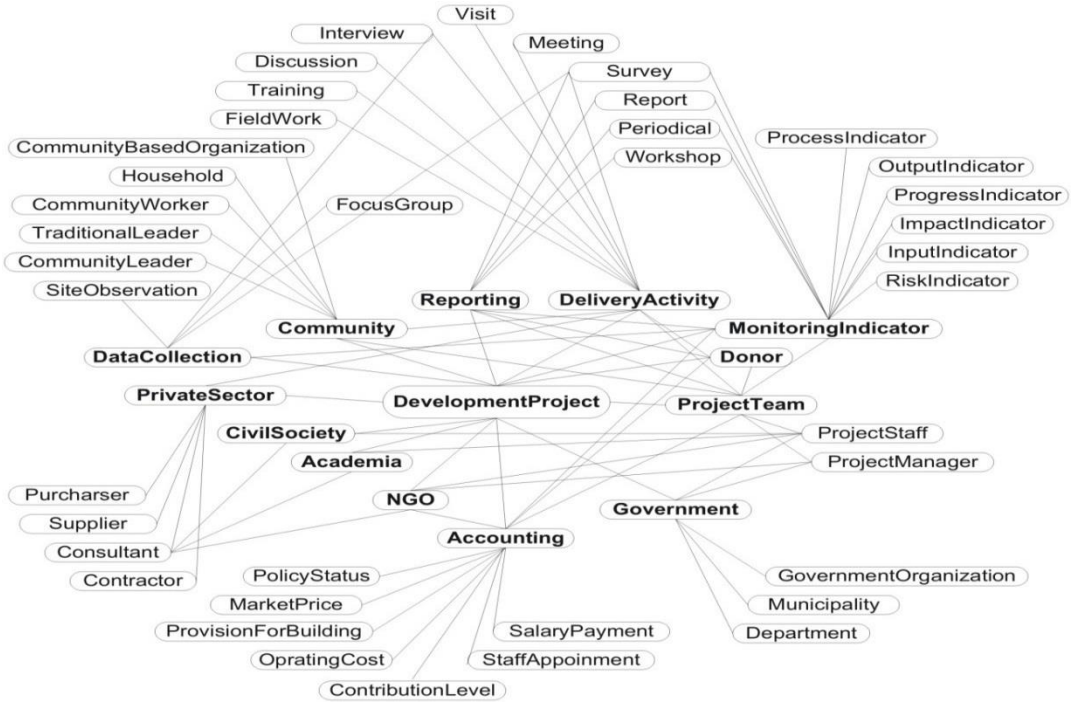


Figure 7: Conceptual model of the OntoDPM Ontology (Vincent, Dombeu and Huisan 2011)

During the formalisation phase of ontology development, the completed conceptual model was transformed into a semi-formal representation with the supporting formalism of UML (Ceccaroni and Kendall 2003). One of the static diagrams of a UML-class diagram was used as the choice for the semi-formal representation. During the implementation phase, the semi-formal version of the ontology was formally represented in Web Ontology Language (OWL). Protégé was used as the semantic-web platform for ontology editing as well as for generating

the OWL codes of the OntoDPM ontology. On the analysis of the generated OWL codes it was noticed that the mapping of the UML semi-formal model was produced in the usual OWL form. Furthermore, the RDF version of the OntoDPM was created with the Java Jena Ontology API, and the union method offered by the Jena Model Interface was used to integrate different branches of the large RDF graph.

In the review, it was found that there was a research gap in the literature, in that the OntoDPM did not comment on the storage and querying of the RDF ontology. In addition, the full capability of the Jena API parsing mechanism to bridge the gap between Protégé and Jena API needed to be researched in more detail. This could offer good possibilities for the repeatability of the research within the e-government development community and strengthen the adoption of semantic technologies for e-government-application development.

2.5.2 Philosophy Ontology Based on Philosophical texts

An ontology for the domain of philosophy was proposed and developed (Min-Kim et al. 2007) as an explicit formal specification of concepts and semantic relationships. The philosophy ontology was text-based with three major steps which included: planning, conceptualisation and implementation and fourteen sub steps were accommodated within the major steps. A web-based management system which includes a semi-automatic translator, takes the output of the conceptualisation step as input for the creation of the topic maps for the ontology-management system.

Ontology design and aspects

A three-layered architecture was designed for the development of the philosophy ontology. The layers were the Philosophy Reference Ontology (PRO), Philosophy Domain Ontology (PDO) and Philosophy Text Ontology (PTO). Figure 8 shows the detail of the three-layer architecture. The philosophy ontology conceptualises both the philosophical knowledge and textual information and presents both of these in the PDO and PTO respectively. PRO is the upper-level ontology offering an abstract level schema and templates to ensure the consistency of the layers underneath. For the creation of XML Topic Maps (XTM), a semi-automatic translator was developed by the team. The philosophy ontology firstly conceptualises and externalises the knowledge of the text content to catch the main concepts without reading the texts. Secondly it offers guidelines for developing text-based ontologies in other learning

domains such as language and art and can be used as a knowledge map of a digital library or a knowledge portal.

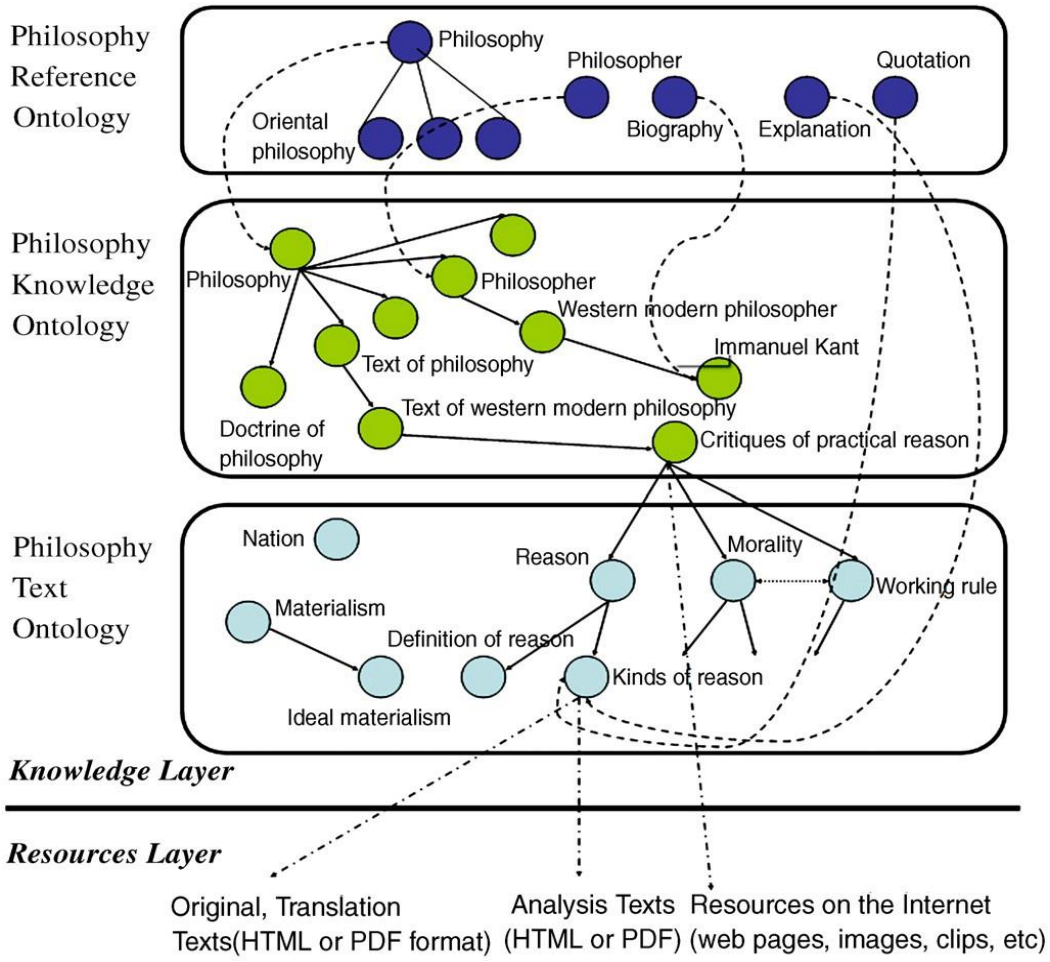


Figure 8: Three-layered architecture of philosophy ontology (Min-Kim et al. 2007)

Ontology development process

The development process of the philosophy ontology was similar to the software-development process. The core phases of software development such as planning, modelling, implementing, testing and maintaining were followed for the philosophy ontology development as a top-layer process. Figure 9 shows the development steps for the philosophy ontology. The characteristics of the philosophy ontology are the externalisation, formalisation and specification of knowledge which exists within the contents of the text. During the conceptualisation process, the core concepts are specialised with more specific concepts and the ontology designers formalise and organise the acquired knowledge. During the implementation, the conceptual model of the ontology is transferred to a machine-readable

model with the support of the topic maps. At the end, the topic-management system interprets the topic-map documents and constructs the philosophy ontology.

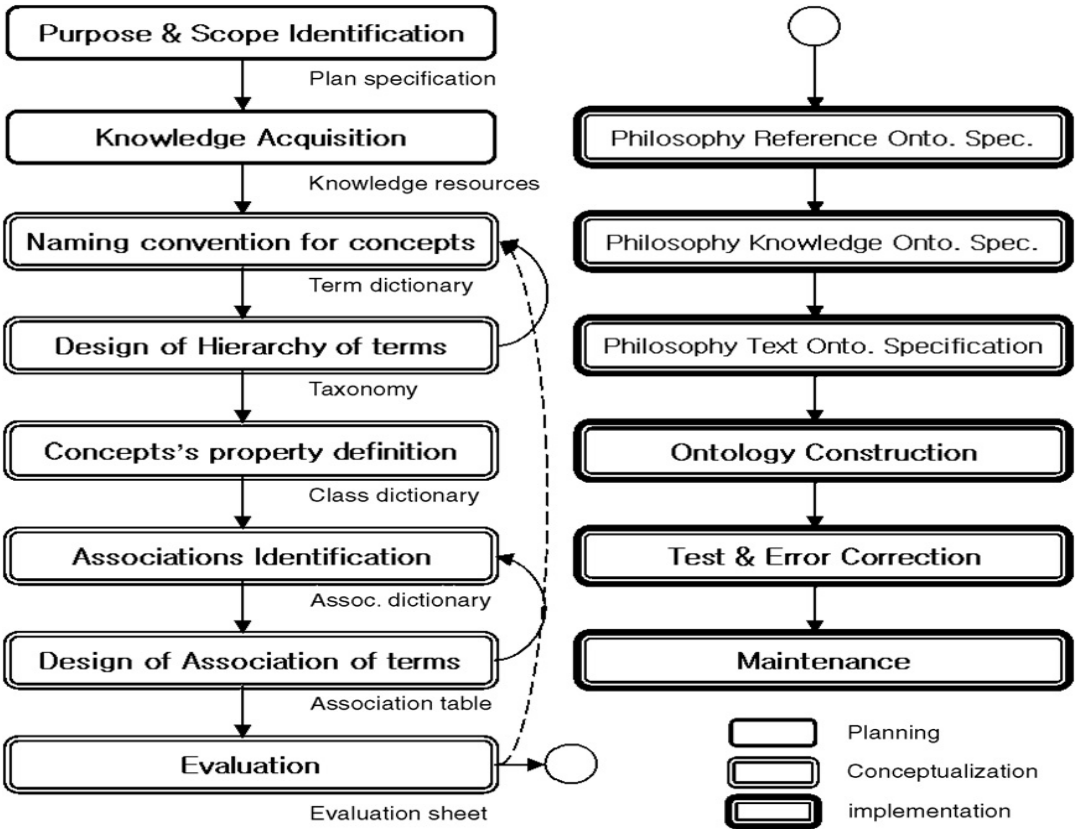


Figure 9: Philosophy ontology development process (Min-Kim et al. 2007)

In conclusion, while the attempt of the authors to develop a philosophy ontology based on a newly-derived ontology-engineering methodology is appreciated, it falls short of being an ideal solution. This is because it could have been a more effective initiative to bridge the gap between software engineering and ontology engineering, if some of the well-proven process models and tools had been used. Though the layered architecture proposed for the philosophy ontology had ample scope for an additional layer to capture tacit knowledge, the conceptualisation process relied on a semi-automatic translator for creating XTM documents, so there could be a possibility for manual errors when the properties are being filled-in by the domain experts. It would have been much better a more formal mechanism, such as RDF/S or OWL was used instead of topic maps.

2.5.3 Ontology for the Travel Industry

This domain ontology was designed and developed for the Thailand travel industry (Khruahong, Kong and Hoang 2015). The purpose of the ontology was to incorporate the unique deciding factors for the visitors' journey which were the constraints of the law, the festivals and the culture, which tended to be missed out by normal search engines. The ontology was designed using the semantic web to find local Thai events and their constraints, with an associated impact factor. The ontology was designed using both a Domain Ontology Graph (DOG) and Location Based Services (LBS). The ontology was applied to intelligent searching to make appropriate decisions for tourists and could well be used in future semantic tourism-applications.

Ontology design and aspects

The review of the literature revealed that compared to other domain ontology design approaches, DOG produces a high classification accuracy in a well-constructed ontology (Khruahong, Kong and Hoang 2015). The DOG approach was followed in the tourism ontology as it demanded both manual and automatic processing for the static and dynamic local-tourism information. This was in sharp contrast to the traditional relational-database approach which failed to update semantically the dynamic content for the special interests of tourists. Moreover, the ontology-based approach was able to process a large amount of data in order to make the appropriate travel recommendations to tourists and could also assist in the retrieval of real-time travel information.

A framework called 'knowledge seeker' was adopted for generating the DOG as well as to manage the document classification. Four components were defined in the knowledge seeker framework for: ontology modelling (of the ontology structure); representing the ontology data in the knowledge-seeker system, ontology learning (the learning algorithm), and to define the method of conceptualising a domain of knowledge, Ontology generation uses a text corpus to generate the ontology in a graphical form and the ontology query to support user operation. The ontology querying module was a crucial module that used a knowledge-seeker system to develop intelligent applications for text-classification and text-searching. Figure 10 shows the travel-ontology design architecture.

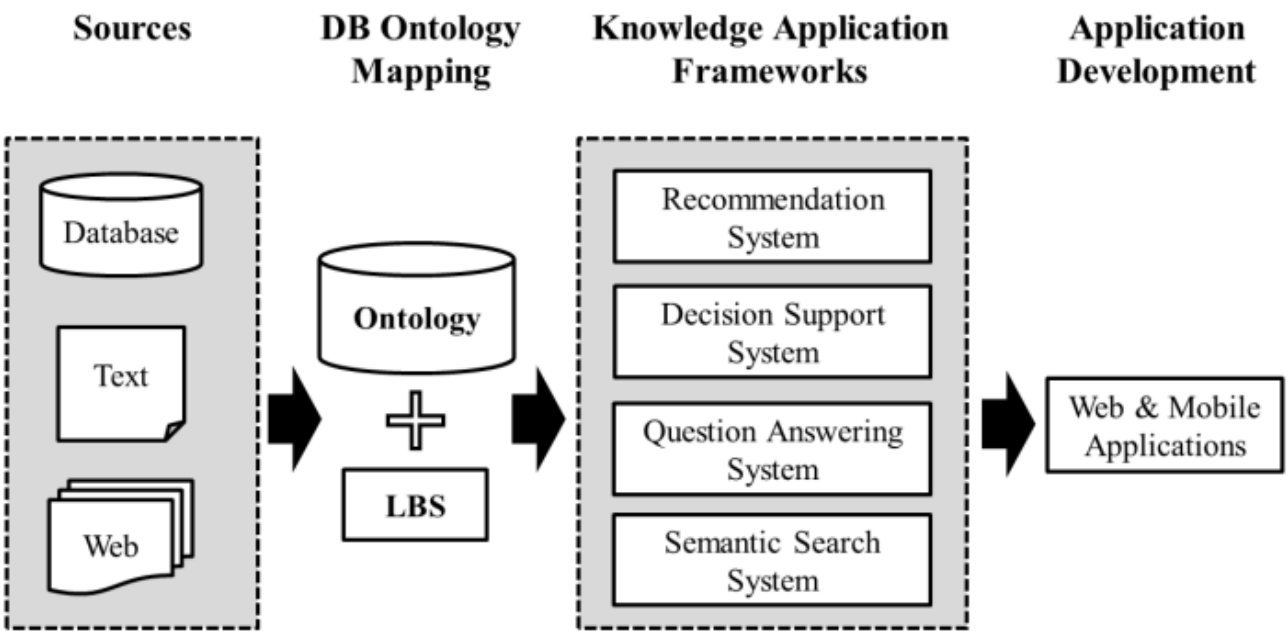


Figure 10: Travel ontology design architecture (Khruahong, Kong and Hoang 2015)

Ontology development process

Two types of travel ontology were developed which were the travel domain ontology and the travel-task ontology. The travel-domain ontology contained the static information which changed slowly over time, such as information about attractive places, hotels, restaurant information etc. In contrast, the travel-task ontology included the dynamic information that changed more quickly over time such as weather reports and traffic reports which could change daily. In our design, both static and dynamic information are adapted to use general information and particular information for the local context, but, the ontology developed for this research had two main nodes which were the general-information class and specific-information class nodes. The general information class has been shown in Figure 11. OWL was used for the formal implementation of the travel ontology proposed.

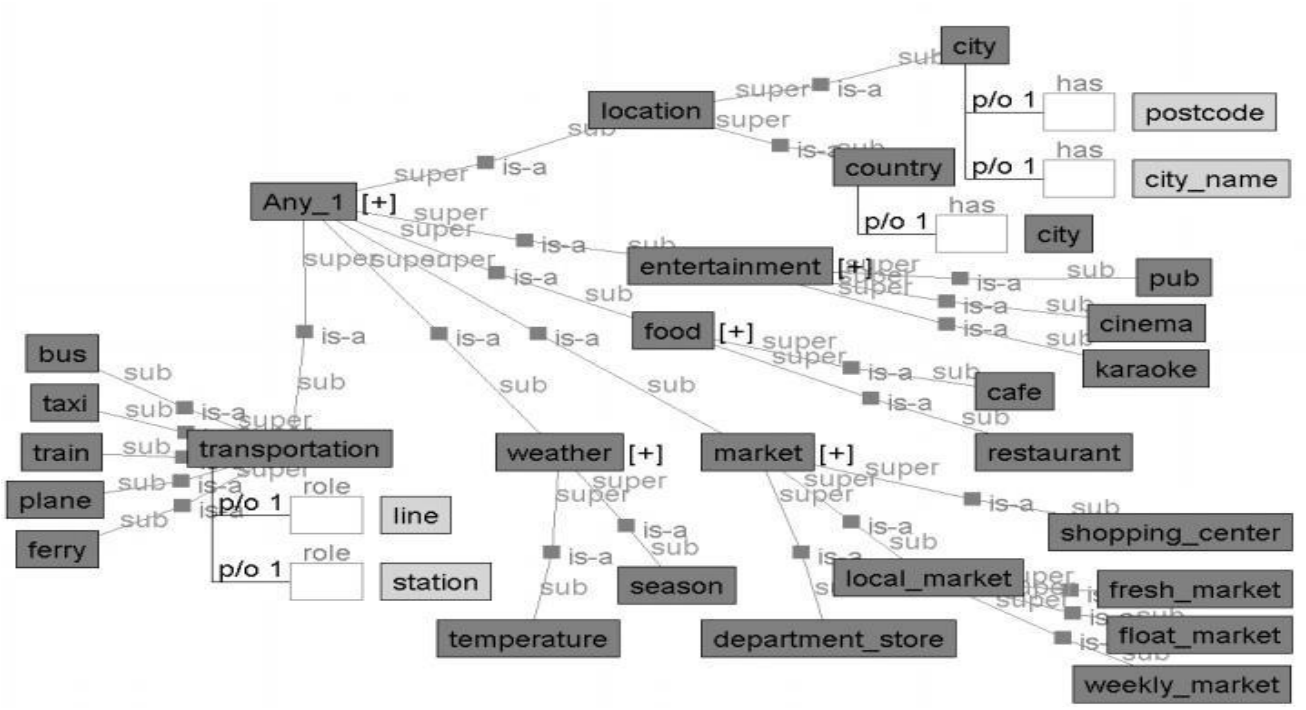


Figure 11: General information class of Travel ontology (Khruahong, Kong and Hoang 2015)

As a positive critique, the efforts made by the designers to integrate ontology and Location Based Services (LBS) are really appreciated. The domain chosen was a worthy one for the ontology design as it demanded the structural representation of concepts and the relationships between them. However, a standard methodology could have been used for the design and development of the domain ontology. Considering the scope for deriving semantic web applications and mobile apps based on the ontology developed, a software- centric methodology could have been an ideal candidate for ontology development.

2.5.4 Educational Ontologies

Many conceptual and practical attempts for the development of educational ontologies have been proposed by researchers and ontology practitioners across the globe (Meenachi and Baba 2012). These include the university ontology, “Topic Maps for E-learning” (TM4L) proposed in the work of Malik, Prakash and Rizvi (2010); the on-line “Ontologies for the Use of Digital Learning Resources and Semantic Annotations” (OURAL); the virtual-lab ontology (Meenachi and Baba 2012), “Cultural Artefacts in Education” (CAE). The literature revealed that various ontology-development attempts for learning computer programming languages have been designed. Two examples of ontologies for this domain are described below.

Case 1: The development of an educational ontology for C-programming (Sosnovsky and Gavrilova 2006) was proposed by Tatiana Gavrilova, which followed a five-step algorithm for visual-ontology design. The knowledge base developed using the C programming language was not just a taxonomy of the language, but an application ontology that accumulated the tacit knowledge of practitioners, delivering courses on the language. It was noted that the ontology assisted in the disintegration of both operational and domain knowledge. Hence, while the order of teaching topics varied from teacher to teacher, the taxonomical structure of concepts remained unaltered. Table 2 shows the five steps of the algorithm used in the development of the C Programming ontology.

Step	Description
Glossary development	*This step focuses on identifying the relevant domain concepts. This is the core step as the domain concepts are eventually considered as the primary classes of the ontology. The fundamental aim of the glossary development is the finalization and representation of core classes derived from domain specific concepts.*
Laddering	*This step follows Glossary development after the finalization of classes/concepts of the domain concerned. It defines the taxonomy for the concepts based on semantic structural relationships. The type of ontology classification will be revealed at this step as it is vital to the remaining stages of the ontology development.*
Disintegration	*The main purpose of the disintegration step is the detailed split up of the higher level hierarchy defined in the laddering step wherever required. A top-down approach can be followed from the root level for disintegration.*
Categorization	*During the categorization step, leaf-level classes are exposed in a structured manner. Generalization is applied through a bottom-up approach for modelling concepts. This is done by correlating like-concepts to generate meta-concepts from leaves of the abovementioned hierarchy.*
Refinement	*The last step is dedicated to the optimization of the graphical structure by eliminating the contradictions, irrationality and synonymy of domain concepts.*

Table 2: Five steps algorithm followed in the C programming ontology

The algorithm stated above has been established in the upper four levels of the ontology developed in this research. The partonomy relationship was mainly used in the hierarchy of the concepts. The uppermost level is the C programming node and the lower levels represent

the meta-concepts which combine the other entities. Figure 12 shows the top levels of the C-Programming Ontology.

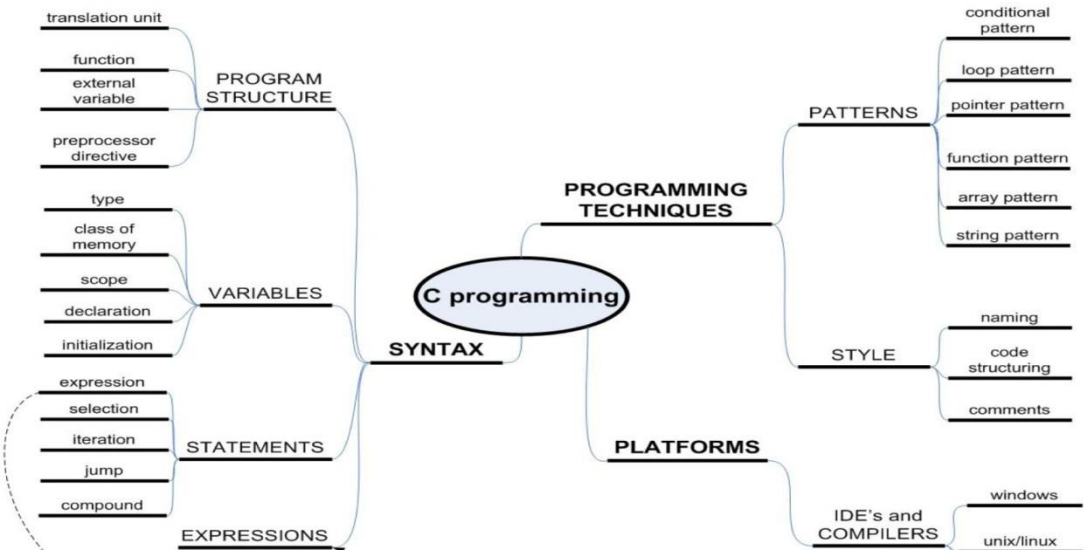


Figure 12: The top levels of C Programming Ontology

Later an ontology for teaching Java programming was developed based on the same five-step algorithm (Ganapathi, Lourdsamy and Rajaram 2011) which is described in the paragraph following.

Case 2: Lee and Wang presented a framework as a Java Learning Object Ontology (JLOO) which was (Lee, Ye and Wang 2005) used as a guideline for the development and organisation of learning objects, in introductory Java courses, as an adaptive learning system. The classification in JLOO was based on the computing curricula CC2001 of the ACM and IEEE/CS.

It was observed that different kinds of learning tracks have been followed by many higher educational institutions to teach java programming, as no pre-defined teaching streams have been specified. Hence a java learning ontology can act as a guideline for semantic connectivity in java learning, regardless of the streams. The methodology followed for the development of JLOO is a "purpose–oriented model". The different steps which followed in a sequential manner under the umbrella of the methodology are given in Table 3.

Step	Description
Define the Scope	*This stage explicitly states the domain, finalizing the purpose and scope of the ontology. In a nutshell, domain feasibility is to be carried out during this stage.*
Reusability of existing ontologies	*This stage looks into the possibility of reusing existing ontology, if there are any, for the domain concerned. Web based Ontology repositories like Ontolingua and DAML ontology library can be referred to at this stage*
Concept Identification	*A kind of glossary development is conducted at this stage which includes listing out all the key terms of the domain concerned in line with the purpose of ontology. The concepts identified at this stage are eventually the pillars of the ontology.*
Finalization of classes and taxonomy of classes	*The purpose of this step is the construction of the concept/class hierarchy of the domain concerned. The hierarchical taxonomy of classes will be created with reference to the sub class and instance concepts.*
Outline slots	*At this stage, slots will be added to the classes identified in tune with the purpose of the targeted ontology.*
Instances creation	*Relevant instances/objects will be created at this stage. The knowledge base will be derived from the collection of instances.*

Table 3: Sequential steps followed in JLOO

2.6 Chapter Summary

The literature revealed that there is not one correct methodology for developing an ontology, as there is more than one way to model a domain (Gasevic, Djuric and Devedzic 2006). This restricts large-scale ontology development to a considerable extent, making the process very time consuming. The cases reviewed highlight the requirement for a software-centric methodology for ontology development. This is required for a number of reasons which include: the need for faster development, easy integration; reducing the learning curve of software practitioners, and the requirement for large-scale ontologies. However, obvious overlaps between both fields are apparent and many researchers now acknowledge the merit of a hybrid approach for systems development, combining semantic web technologies and formalisms, as well as languages like UML (Tetlow et al. 2006).

This chapter has described the various components of ontology development such as the languages, methodologies, and Integrated Development Environments (IDES), based on a constructive and critical review of the appropriate literature. As this research is attempting to bridge the gap between software engineering and ontology engineering by proposing a novel software-centric methodology for ontology development (SCIM), the relevance of the research has been justified by the literature review and is further described in Chapter 4, where the conceptual framework of SCIM is introduced.

3. Research Methodology

A research methodology emphasises the methods and mechanisms followed by a researcher to systematically complete the research process. This includes the selection of the appropriate research methods and the development of a research design. The subsections that follow describe the research methodology selected for this research.

3.1 Chapter overview

The chapter has been organised by providing a generic write up on various aspects of the methodology followed by the specific research design and approaches used in this research. A brief introduction to the research process is presented in the next section. The detailed research design with its components has been described in section 3.3. Different types of proven research methodology approaches and their characteristics, together with a discussion of their viability are described in section 3.4. A detailed description of the specific approach followed by this research has been provided in section 3.5 together with the necessary illustrations. A brief description of the prototype development proposed as a proof of concept are described in Section 3.6, which includes an overview of the chosen domain and case environment. The research methods and case tool planned for the application has been explained in section 3.7 and 3.8 respectively. This has been followed by the chapter summary.

3.2 Introduction to Research Process

The aim of any research process is the discovery of an appropriate solution to the research problem through the systematic application of scientific procedures regardless of the nature of the research. To achieve this aim, the researcher needs to have a defined research process which encompasses the steps to be carried out in their order of execution. An overview of the typical research process is illustrated in Figure 13 with the sequence of the steps involved.

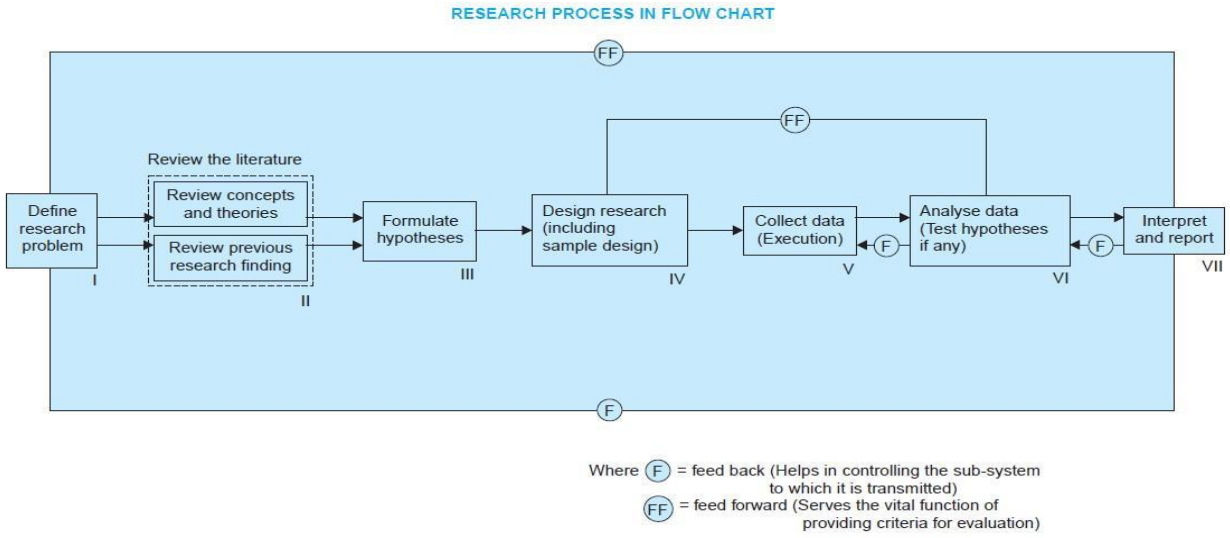


Figure 13: Research process with sequence of steps (Kothari and Grag 2014)

In order to find an ultimate solution to a research problem, researchers traditionally relied on quality research questions and related objectives which to address the specific research problem. Two key activities in the research process were the *formulation of research design* and *the identification of the research methodology approach to be used*. These activities were connected with the research methods, data collection, data analysis, and the proof of concept development etc.

The terms research methodology and research methods are highly interconnected, however, there are certain distinctions between them. Research methods are the techniques that are used for conducting research (Kothari and Grag 2014) whereas research methodology is a systematic way to solve a research problem. A typical research methodology includes: the research model, theoretical framework and research methods, etc. In a research process, the researcher not only needs to understand both the methods and techniques and how to apply them, but also the rationale behind the techniques used in relation to the context of the research. The research methodology identifies the methods and strategies used for data collection and analysis which should contribute significantly towards the achievement of the research objectives. Therefore, it is mandatory for the researcher to design/adopt a research methodology that is most suited for investigating the particular research problem.

3.3 Research Design

One of the popular publications on research states that "A research design is the arrangement of conditions for collection and analysis of data in a manner that aims to combine relevance to the research purpose with economy in procedure" (Selltiz 1962). However, research design needs to be customised, as it defines the research type, research problem, research gap, research questions, hypotheses and research methods. In fact, the research design can be considered to be a conceptual framework to be followed by the researcher for carrying out the research. Regardless of the research type, a good research design should always be flexible, appropriate and efficient. For this research, which is exploratory in nature, a flexible research design with good scope for customisation is mandatory, for finding the optimal solution to the research problem.

This research was initiated to bridge the gap between software engineering (SE) and ontology engineering (OE), two complementary branches of engineering. Therefore, the research area was well suited for exploratory research. The generic sequence of iterative steps followed in this research is given in Figure 14. These steps have been organised in the following sequence.

1. Research idea: The problem addressed by this exploratory research has been initiated to address the research gap that exists between the complementary engineering branches of SE and OE.
2. Literature review: The recent publications relating to the domain of ontology engineering and other related areas, and the case studies of domain ontology implementation, have been extensively reviewed.
3. Formulation of the research problem: From the case studies and the literature on existing ontology development methodologies, a matrix has been defined to make an exclusive comparison among the existing methodologies, based on a set of carefully defined parameters. The outcome of the comparison has been used to formulate the research problem in its final form**.**
4. Design of a solution: A conceptual framework of the solution to the research problem, in the form of a SCIM has been developed.
5. Proof of concept: To apply the solution to a particular case to prove the concept and demonstrate it working in practice.

6. Validation: Evaluation by field experts to validate the proposed solution.
7. Interpretation: Present/publish the research outcomes at international conferences/journals,

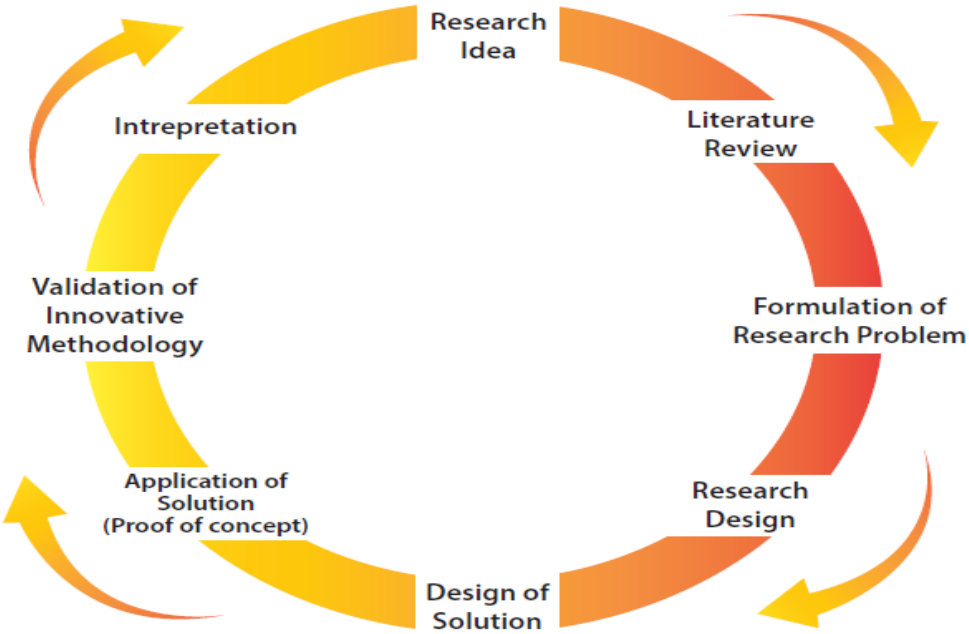


Figure 14: Steps followed in the research (Get Research Design Assignment Help Now n.d.)

The steps mentioned in Figure 14 have been performed in this research with the necessary level of customisation. Considering the nature of this research, a few of the steps have been performed both in a parallel and iterative mode. An outline of the customised steps with their inter-connectivity has been illustrated in Figure 15. Since the contribution to knowledge of this research is a novel methodology, more weight has been given to secondary-research methods. However, primary research techniques were proposed for the proof of concept/prototype ontology development, where the outcome of the research was applied.

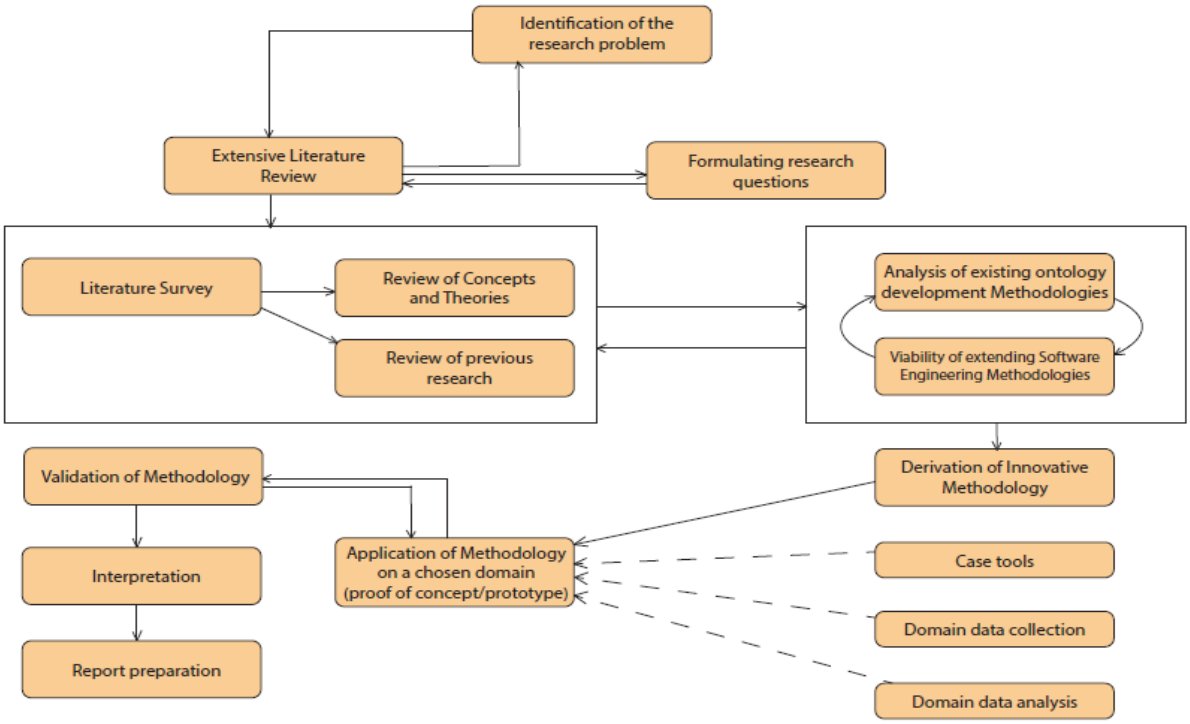


Figure 15: Illustration of the customised steps of research design

The type of this research is exploratory as it is formulating a research problem for more detailed investigation from an operational point of view. In this research, more emphasis has been given on the need to fill the research gaps in the literature by exploring new ideas and insights based on existing theories. Therefore, this research follows a flexible research design which allows the transformation, from a broadly defined research problem, to a more precise form, based on extensive survey of the related literatures. In addition to the examination of existing theories and facts, this research attempts to apply the concepts and theories developed in other research contexts. The abstract nature of the design of this research is represented in Table 4.

Research Type	*Exploratory or Formulative*
Research Design	**Flexible Design:** The design offers good scope to consider different aspects/views/theories of the research problem at various stages.
Research Problem	Formulated from a research area where a significant research gap exists. The research problem is further tuned based on the extensive review of literatures and studies conducted on the domain. ***How to develop a methodology for ontology development which bridges the gap between the complementary engineering branches- Software Engineering (SE) and Ontology Engineering (OE)?***
Research Objectives	• To analyse the Existing Ontology Development Methodologies • To propose a Software Centric methodology for Domain Ontology Development • To define an Ontology Development Life Cycle (ODLC) for the proposed methodology. • By using appropriate tools apply the proposed methodology to design a domain ontology • To measure the effectiveness of the methodology proposed by validating against existing methodologies • To disseminate the research findings to high quality international conferences and high-impact journals

Research Questions	Derivation of the research questions explored with the help of flexible research design to achieve the research objectives. ● *What are the limitations of existing ontology development methodologies?* ● *What is the scope of developing a novel methodology for domain ontology development by extending software engineering process models and methodologies?* ● *Can an Ontology Development Life Cycle (ODLC) be defined for the proposed methodology?* ● *Can the proposed methodology be applied to develop an ontology for a specific case?* ● *How can the proposed methodology for ontology development be reliably evaluated?*
Literature Review	Review of literatures stating the concept, theories and previous research. ● *Ontology Representation languages* ● *Ontology development methodologies* ● *Ontology development platforms* ● *Domain Ontology developments* ● *Case studies* o *E-government ontology* o *Philosophy ontology* o *Tourism ontology* o *Educational ontologies*
Design of solution	Solution design in-line with the research questions and research problem. ● *Software process models* ● *Leveraging software engineering process models to ontology engineering* ● *Derivation of hybrid methodology* ● *Formulation of engineering and philosophy* ● *Identification of stages, workflows, activity and techniques* ● *Design of the final framework*
Application of the Solution • Sampling Design: *Purposive Sampling* • Statistical Design: *No pre-planned design*	***Proof of concept (Prototype development-JLEO)*** *Domain chosen: Educational domain (Basic Java Teaching and Learning)*

for analysis • Observational Design: *Mixed elicitation instruments* • Operational Design: *No fixed decisions about the operational procedures*.	*Case: Middle East College Computing curriculum (Three introductory Java courses)* *Research Methods: Document Analysis* *: JOD sessions* *:Competency Questions/Survey* *: Informal Interviews* *Formal Language: OWL* *Modelling Language: UML* *Tool set: Protégé*
Validation of solution (Evaluation)	*Validation of solution against existing methodologies (evaluation matrix with qualitative opinions from evaluators)-Customised GQM Approach*
Knowledge Contribution	*Novel Methodology for Ontology development*
Interpretation and Reporting	*Thesis preparation.*

Table 4: Abstract view of the Research Design

3.4 Research Methodology and Approaches

The research methodology process focuses on data collection for the purpose of solving the research problem. Therefore, the integral part of research methodology is the selection and usage of the appropriate research methods and techniques. A good research methodology is necessary while developing new ideas or insights as part of exploratory research. Research methods and their associated application techniques are used for conducting the primary and secondary-data collection. The major categories of research methods are Quantitative, Qualitative Design Science and Mixed methods. A brief discussion on each of these research methodologies and a comparison between them has been given below. Subsequently the discussion leads to the research methodology and approaches used in this research.

Quantitative methods: These methods aim to classify and count the features, create statistical models, test the hypothesis and describe the observations. "Quantitative methods emphasise objective measurements and the statistical, mathematical, or numerical analysis of data collected through polls, questionnaires and surveys, or by manipulating pre-existing statistical data using computational techniques" (Babbie and Earl, 2010).

The ultimate aim for conducting a quantitative research study is to determine the relationship of an independent variable to another dependent variable (or outcome variable) within a sample (or given population). Quantitative research methods are applied regularly for descriptive and experimental research, as the former is used to establish associations between research variables while the latter establishes causality. In quantitative-research methods, data is gathered through structured data-collection instruments such as questionnaires or computer

software. The results are based on sample size that represents the population. Based on the reliability, research can usually be replicated or repeated. In quantitative-research methods, objective answers are sought for a well-defined research problem. Data is in the form of numbers and statistics, often arranged in tables, charts, figures, or other non-textual forms.

Qualitative methods: The handbook of qualitative research (Denzin and Lincoln 2005) describes qualitative research as involving "… an interpretive naturalistic-approach to the world. This means that qualitative researchers study things in their natural settings, attempting to make sense or interpret phenomena in terms of the meanings people bring to them". Qualitative methods mainly find out the opinions of people and immeasurable elements by making use of appropriate data-gathering instruments. Qualitative methods seek to answer the research problem by making use of a set of pre-defined procedures. The primary goals of qualitative methods are exploration, description and interpretation with the aim of collecting evidence from new observations without the influence of pre-determined findings. Within the context of exploratory research, qualitative methods produce results within the scope of their application, but which are beyond the nearest boundaries of the research. Unlike quantitative methods, the researcher is the instrument for data collection in qualitative methods. The three most common techniques applied in qualitative methods are participant/process observation, in-depth interviews and focus groups. In the case of this research, qualitative elements have been applied by both existing methodology comparison and the evaluation of a methodology. The following techniques have been applied in this research.

Participant/process observation: This technique is appropriate for the collection of data on naturally-occurring behaviours in their regular contexts.

In-depth interviews: This is an optimal technique for collecting data based on the tacit knowledge of participants such as their personal background, perspectives, and experiences, and is an appropriate technique for research where sensitive topics are being explored. ***Focus groups:*** This is an effective data-collection technique where the focus of research is generating broad overviews on the issues of concern to groups or subgroups.

In addition to the common techniques listed above, good application of secondary-research methods is encouraged by qualitative research to make use of the proven results of previous researchers. This has been achieved through the review of relevant literature and the critical evaluation of the results published in reports, case studies and other publications. Unlike quantitative methods, qualitative methods assist the researcher to develop hypotheses for further exploration by exploratory research.

Mixed Method: This is a methodology which combines both qualitative and quantitative methods, and involves a collection of techniques to collect, analyse, and integrate quantitative and qualitative data into a single research method.

Design Science Approach (DS): According to the Design Science in Education (2004) report, this approach includes yet another set of analytical techniques and complementing perspectives for qualitative methods (i.e. positivist and interpretive) to carry out research in information systems (IS). It is well suited to exploratory research in IS as it involves the creation of new findings/knowledge by the use of novel innovative artefacts which improve certain aspects of the behaviour of information systems.

New algorithms, methodologies and languages are a few of the instances of such artefacts. Since research in engineering and computer science has led to the contribution of new knowledge and in most of the cases, the formation of a new theory, algorithm, methodology or information system, DS is the most recommended approach for exploratory researches (Gregor and Hevner 2013). By focussing on the design activity of innovative new knowledge, DS uses man-made phenomena to meet certain research goals. The underpinning philosophy of DS is not new to Information and Communication Technology (ICT), but what distinguishes DS from routine research is the production of new knowledge for an interested community (Wieringa 2014). Therefore, the intellectual risk is a factor being taken care of by this approach. DS is one of the best approaches for filling an existing research gap with new findings, by making use of the available knowledge. This motivates the investigator to extend the research to fill the knowledge gap as well as to face the challenges involved. A comparison between the three categories of methods described above is given in Table 5.

Aspect	Qualitative	Quantitative	Design Science
Focus	Reasons, opinions, and motivations (Quality)	Frequency, Magnitude (Quantity)	New Insights and theories
Philosophy	Constructivism, Interpretivism	Positivism	Constructivism, Interpretivism
Aim of Investigation	Understand, Describe and Discover	Predict, Control, Confirm and Test	Discovery of New Knowledge
Design characteristics	Flexible, evolving and emergent	Structured and Pre determined	Flexible, Iterative and dynamic
Data Collection	Researcher as an instrument	External Instruments, tests, surveys	Flexible instruments, mixed, secondary data
Role of hypotheses	Develop hypotheses	Test hypotheses.	Develop hypotheses.

Table 5: Comparison among Research Methods

To achieve the research objectives set out in the research design, leading towards the solution of the research problem, a customised research methodology with a mixture of approaches has been followed. The nature of the research being exploratory was the primary reason for the choice of DS for the methodology. Documented observations, a survey of a limited community and informal interview sessions were chosen as the research methods.

A combination of both inductive and deductive processes were used for secondary data collection. The inductive approach was applied by generalising the existing facts and theories to achieve a conceptual model of the proposed methodology. On the other hand, deduction was also applied because the facts and theories were first obtained from existing software-engineering methodologies.

3.5 Customised Design Science Approach

In this section, details of the steps followed in this research have been described in depth. The methods and techniques proposed for the steps have been elaborated. Figure 16 shows the customised design research methodology followed in this research. An outline of each process has been described in the subsections.

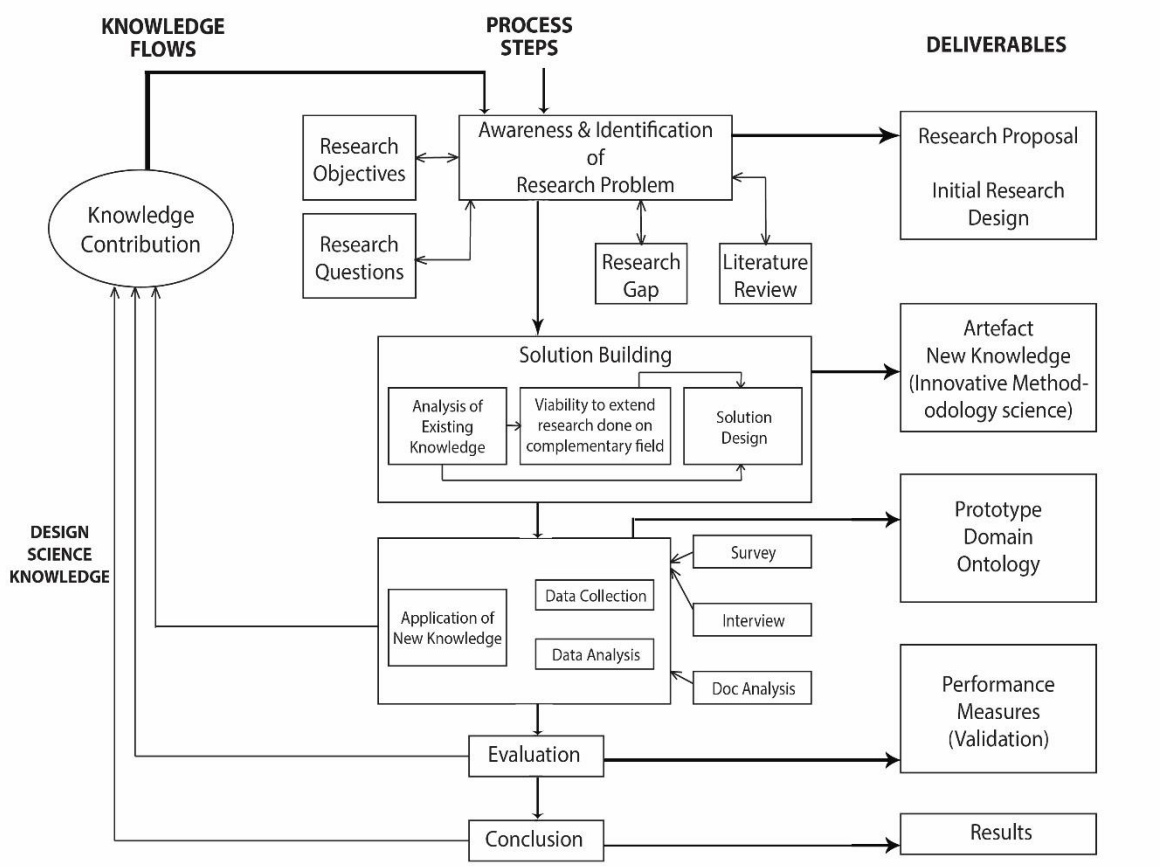


Figure 16: Illustration of the Customised Design Science approach

3.5.1 Awareness and Identification of Research Problem

The literature has revealed that reasonable attempts towards domain ontology development methodologies began in the second half of 1990's (Loppez 1999), but these initial attempts have mainly focused on specific problems and domains. Initiatives on generic ontology-development methodologies were minimal in those days, however, initiatives such as METHONTOLOGY and TOVE were attempted for the implementation of general-purpose ontologies. The trend towards the development of knowledge-based systems and semantic-web initiatives had significantly increased the volume of research on the related technologies. In the 2000s, the availability of ontology-based applications was introduced for various domains including e-governance, agriculture, education and tourism etc.

In fact, in the second half of 2000's, ontology engineering was introduced by the research community, mainly focusing on the methods, methodologies and tools for the development and maintenance of domain ontologies (Pére et al. 2004). In the late 2000's research communities started looking into the similarities between software engineering and ontology

engineering (John, Shah and Smalov 2016). It has been observed that although these two engineering branches were complementary, there existed significant research gaps between them. One of the gaps identified by this research was that a standardised methodology which also provided an integrated-tool for ontology development was not available. The first publication (John 2010) by this researcher proposed an extension to the well-proven software-engineering process models, for ontology engineering. Based on further tuning and updating, the first statement of the research problem for this research is stated below.

How to bridge the gap between software engineering and ontology engineering?

Awareness Revisited

As the customised design-science methodology offers the flexibility for spontaneous revisiting of any of its stages (especially in the early stages), the research problem, objectives and research questions have been revisited iteratively based on: discussions with domain experts, the use of an extensive literature review and milestone meetings. Figure 17 is an illustration of the activities followed for the awareness and identification of the research problem. After revisiting, the research problem was revised as stated below.

How to develop a methodology for ontology development which bridges the gap between two complementary engineering branches- Software Engineering (SE) and Ontology Engineering (OE)?

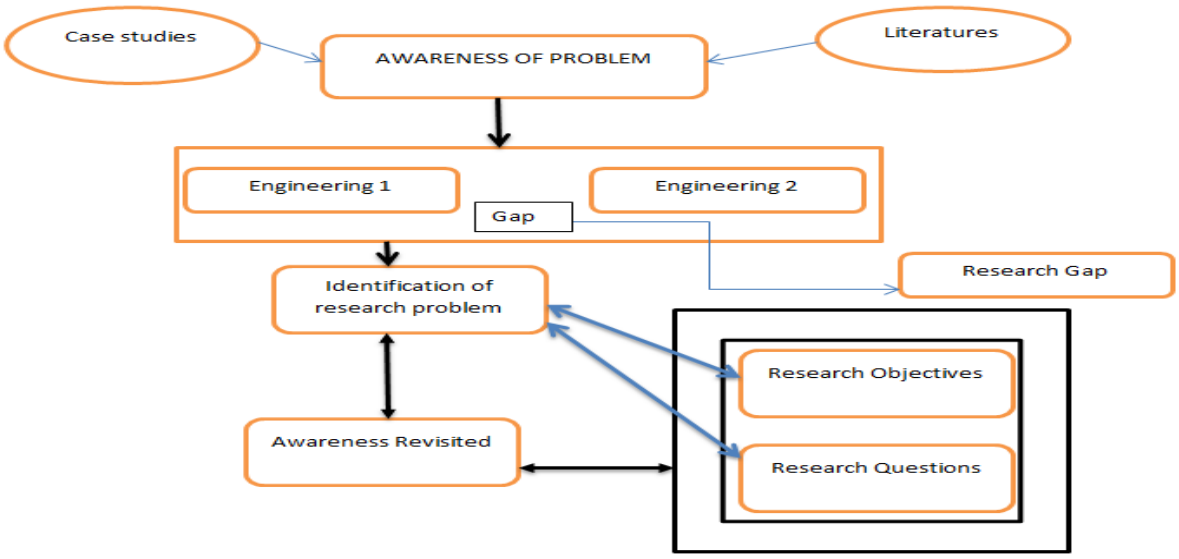


Figure 17: Illustration of the awareness and identification of research problem.

3.5.2 Design of Solution

The artefact of the ‘design of solution’ phase of the DS methodology is straightforward compared to other research approaches. The deliverable of the ‘awareness of problem’ and ‘research-problem identification’ stages will be further realised at this stage. The techniques and tools for the solution building will vary according to the requirement of the research problem. The solution implementation makes use of the state-of-the-art practices without considering novelty, however, the emphasis on novelty is mainly at the ‘solution design’ rather than the implementation stage.

The contribution of this research to existing knowledge is the conceptual design of a methodology for domain-ontology development. The design of a new prototype methodology and its application on a domain, making use of the available tools, is the main contribution of this stage. Aspects of similar works from the recent literature and case studies will be added to the proven practices of complementary engineering fields with the necessary customisation as the core of this new knowledge. In this research, existing ontology-development methodologies such as METHONTOLGY and UPON have been taken as similar works. Well-proven process models of software engineering (i.e. Linear Waterfall and Iterative RUP) together with the necessary customisation were selected for amalgamation into a hybrid model. An illustration of the activities followed at this stage shown in Figure 18. The conceptual framework for a new ontology-development methodology is represented by the ‘conceptual design of new knowledge’ box in the figure.

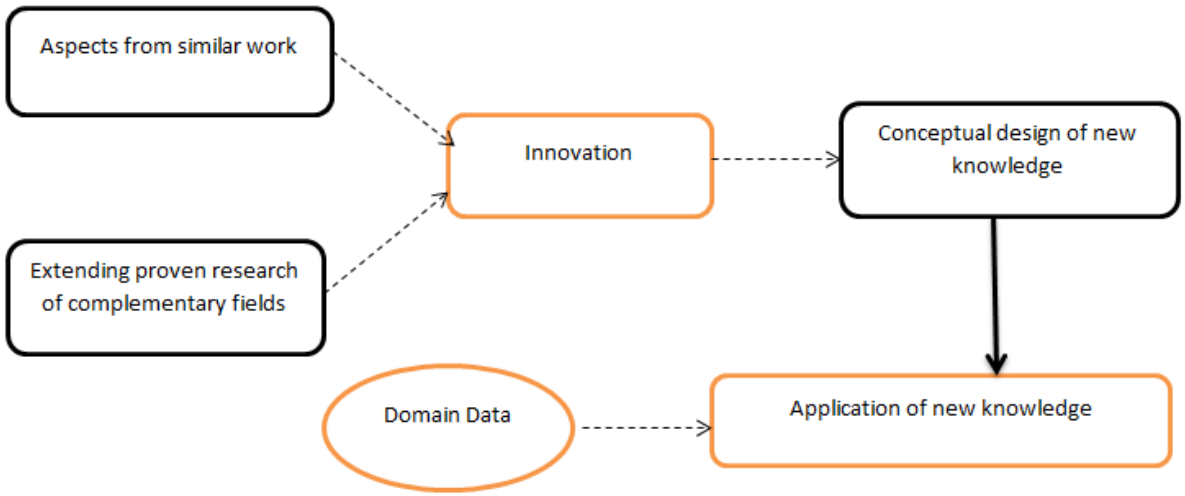


Figure 18: Illustration of the Design of Solution stage

Application of new Knowledge

In this research, the 'application of new knowledge' box in Figure 18 above was represented in this research by the proposed methodology being applied to the design and development of the prototype ontology, and an appropriate case has been chosen for the implementation. The prototype domain ontology was implemented as a proof of concept for the application of the methodology, and the hierarchy followed was **Stages>workflow>activities>techniques**. The guidelines to be considered/followed for the successful completion of the activities were proposed in the methodology. The conceptual framework of the proposed methodology was followed in the prototype implementation. For the development of the prototype ontology, appropriate elicitation techniques such as document analysis, survey and interviews were conducted. The most popular Ontology Development Environment (ODE) was used for the prototype design. A brief discussion on the key elements of prototype development was given in section 3.7 and 3.8.

3.5.3 Evaluation

Evaluation is an inevitable part of any research, however, the approaches may vary, subject to the category and nature of the research, although evaluation in its simplest form is the measure of the deviations from the norm. Both qualitative and quantitative artefacts of the research should be evaluated. Unlike the explanatory researches where evaluation either contradicts or confirms the hypothesis, occasions are rare in design science where the initial hypothesis is confirmed by the actual behaviour. Evaluation takes place continuously in the design-science approach, therefore, the evaluation stage produces an extra piece of information which is obtained in the solution development stage. Many instances of evaluation take place during the design of solution stage, resulting in decisions being taken which influence the design, For the evaluation of the conceptual framework, a pool of parameters needed to be identified and sequenced in order to conduct the evaluation.

In this research, an appropriate mechanism was chosen, supported by the literature, derived from the evaluation of engineering processes, which used a set of derived parameters to measure the efficiency and accuracy of the methodology. A set of experts with the appropriate experience in ontology engineering were chosen for the evaluation and a detailed description of the mechanism used is provided in Chapter 6 of this thesis.

Evaluation Mechanism

For the validation of the methodology and ontology, a mechanism called Customised Goal-Question-Metric (GQM) was chosen. Unlike the popular logic-based or feature-based mechanisms, the evaluation-based approach provided by the GQM mechanism was the most suitable for the validation. A systematic approach was followed for the validation of the methodology where ten experts from the field of ontology engineering were chosen as the target group for evaluation. A matrix, based on existing ontology- development methodologies was prepared, created from specially defined parameters. The parameters were designed so that the validated methodology could be termed 'standardised'. This was achieved by proposing credible solutions to overcome the limitations of existing methodologies. The software tool used for the prototype development had an integrated mechanism, called the 'reasoner' which automatically validated the logical consistency of the prototype ontology. Besides the integrated mechanism, a set of additional parameters were also added for the validation of the various processes. Therefore, the specific parameters used for the matrix preparation gave due consideration to both the engineering and philosophical aspects of the existing ontology development methodologies.

3.5.4 Conclusion phase of the DS approach

This phase of the Design Science focusses on the interpretation of the results of the research. In the case of this research, this has been done primarily as the compilation of the results of this thesis complemented by a number of relevant publications. Though scope for improvement exists in all research, the results could then be considered as acceptable. In addition to the results obtained, the contribution of the research to the body of the knowledge will also be highlighted in this phase. Therefore, a leftward arrow has been indicated in Figure 16 from the conclusion phase. However, the expectations from this phase may vary, depending on the contribution to existing knowledge, by factors such as the nature and depth of the new knowledge.

In this research, the conclusion phase will put more emphasis on the contribution of new knowledge, such as the underpinning philosophy and engineering aspects of the newly proposed methodology. Furthermore, a quick summary of the stages, workflows, activities and techniques to be followed by practitioners, for the adoption of new methodologies will be described here. The components of the life cycle for the proposed methodology will be restated in the conclusion phase. The novelty and uniqueness of this research will be

highlighted in the conclusion by citing the artefacts and techniques proposed by the new methodology. The limitations of this research and the future research directions will also be covered in the conclusion. As the last section of the conclusion phase, the reflections of the researcher will be included.

3.6 Prototype/Proof of Concept Development

As stated in the 'application of new knowledge' phase, the proposed methodology has been applied as a proof of concept to prove its suitability for ontology development. An appropriate domain has been chosen by the researcher to apply the proposed methodology. The selection of both has been justified by the researcher with support from the literature. Commonly used research methods have been applied for the domain data collection and the best available software tools have been used for the practical development. The following subsections describe the prototype/proof of concept in detail.

3.6.1 Domain Overview

The domain chosen for the application of the proposed methodology was taken from the educational sector where knowledge-sharing and reusability was important. Semantic-web based educational systems, using semantic-web technologies, inclusive of ontologies. support more personalised learning (Yarandi, Jahankhani and Tawil 2013). The proposed methodology, SCIM, was applied to the development of a prototype ontology called Java Learning Educational Ontology (JLEO) for learning introductory Java programming. One of the motivating factors behind building an ontology for Java programming was an attempt to organise the learning aspects of a widely-adopted industry language by unifying the different views on the domain. The availability of a Java-learning ontology ensured uniformity among the teaching staff regarding the domain concepts and the relationships between them. The proposed ontology ensured that the basic hierarchical semantic-structure was not violated, even though the order of the material could vary when presented by different teachers. Three different variations of introductory Java-programming, taken from modules of the undergraduate curriculum of a premier higher education institution, based in Oman, were selected as case studies for the design of the JLEO. The basic learning units of JLEO were defined based on the three chosen modules from the computing curriculum of Middle East College (MEC). The 'Introduction to Programming', 'Object Oriented Programming' and 'Internet Programming' were the chosen modules. The methodology components of the

Ontology Development Life Cycle (ODLC) defined in the SCIM were applied to the development of the JLEO. A more detailed explanation of the proposed methodology is given in Chapter 5 of this thesis.

3.6.2 Case Study Overview

MEC is one of the leading Higher Education Institutions in the Sultanate of Oman, located at Knowledge Oasis, Muscat, and works in collaboration with Coventry University in the United Kingdom. The college offers 17 undergraduate and 3 postgraduate programmes in different areas of engineering, technology and business. The computing department of MEC offers undergraduate programmes in various streams including computing and information technology, software technology and computer science. Java-programming based modules were spread across the curriculum of modules of the above-mentioned streams. Three introductory Java modules were chosen as the case study environment for the development of the JLEO.

3.7 Research Methods

The following research methods were planned for the data collection. The necessary ethical approval was given by the University to use the elicitation instruments among the target community. A brief description for each technique is described below.

- **Document Analysis**

Document analysis was one of the elicitation techniques used to gather the data during the requirement analysis stage. As part of document analysis, the approved module-related documents such as the MEC Module descriptor, module guide and module-review documents were analysed. The purpose of this exercise was to identify the core domain-specific concepts and the semantic relationships between them.

- **Questionnaire/Survey**

A survey was conducted among the module tutors of the three chosen modules by making use of a set of competency questions. The intended purpose was to gather the domain-specific concepts based on the tacit knowledge of the tutors. The questionnaire gave scope for the different heads of department to add new concepts to the existing concepts. The more domain concepts that were identified, the more will be added to the domain vocabulary and enable the ontology designer to design and implement a more comprehensive ontology. The survey

was conducted in accordance with the conditions stipulated in the consent form approved by the MEC.

- **Interviews**

Three informal interviews for gathering domain information were conducted with the module leaders of the three chosen modules. The intended purpose of the interviews was the collection of information related to the qualitative aspects of Java teaching and learning, including concept hierarchy and semantic structural relationships, etc.

3.8 CASE Tool considered

One of the popular free and open source platforms, the neutral ontology development editor (ODE), and known as Protégé, developed by Stanford University was chosen as the tool for the implementation of JLEO. The OWL codes were generated automatically based on the design of the ontology. The reasoner was one of the other interesting features that were behind the selection of Protégé. The plug-in, called OWLViz was added to Protégé for the visual display of the concept hierarchy. More details of the case tool are described in Chapter 5.

3.9 Chapter Summary

A detailed description of the research methodology followed by this research was presented in this chapter. Detailed coverage of the research process and the research design of this research have been incorporated. The popular design science approach was followed in this research as it was more suited to the nature of this exploratory research. The specific research design, with the activities stated in the chapter, has been followed. Besides the aspects of design science, a mixed mode of research method techniques has been used for the data collection. Since the research was of an exploratory type, more focus was given to the novelty of the approach adopted. The application of the research design will be described in subsequent chapters.

4. Proposed Methodology for Domain Ontology Development

This chapter provides a detailed description of the proposed solution to the research problem. Section 4.1 contains an introduction to the context, section 4.2 describes an overview of the software-engineering methodologies and their viability for ontology development, section 4.3 introduces the background of the proposed methodology and a detailed overview. A detailed description of the Ontology Development Life Cycle (ODLC) for SCIM has been provided in Section 4.4. Section 4.5 details the workflows and activities. A consolidated write up on the conceptual framework with an illustration is presented in Section 4.6.

4.1 Introduction to the context

The last decade has witnessed a paradigm shift in the world-wide web, from a global-information space of connected documents to the formal and shareable knowledge-based system which is the semantic web. Traditionally, data published on the web has been made available in specific formats compromising much of its structure and semantics. This has evolved into the structural model of the semantic web where data and documents have been linked and the relationships defined between them. Ontologies, in particular have fulfilled the requirement for knowledge representation and technologies based on these have emerged as appropriate engineering solutions. The solutions have then been applied to the problem of developing systems that assure the integration of data from different sources with high-end interoperability to provide seamless services to web users.

Linked data, in its simplest form is a set of best practices for publishing and connecting structured data on the web. It refers to the data published on the web in such a way that it is machine-readable, its meaning being explicitly defined which can in turn be linked to external data sets (Christain B 2010). From the literature and practice, it is evident that this is aligned with the idea of a semantic-web vision, which has provided an opportunity to represent information on the web in such a way that it can be understood and manipulated by software agents and systems. It offers an environment which is more adaptable, personalised and intelligent (Ig Ibert Bittencourt 2009). Ontologies are used for various purposes such as natural language processing, information extraction, intelligent search engines, digital libraries, and business process modelling, etc. They are mainly used to establish explicit ontological agreements which serve as the basis for communication between either humans or software agents. Hence it is mandatory to reduce the language

ambiguity and differences in knowledge between parties involved to avoid confusions, errors and inefficiency (Carlos Blanco 2011). Ontology engineering is a branch of knowledge engineering mainly dealing with the formal principles to build an ontology. This includes processes such as the development, management, analysis and reuse of ontologies. These processes include the methods, methodologies and diverse set of tools used for creating, editing and visualising ontologies.

The development of a domain ontology exhibits both structural (Carlos Blanco 2011) and logical complexity comparable to the development of software systems. However, it is more complex and has a longer learning curve compared to any kind of software development. This is due to the need to provide diverse tool support and heterogeneous platforms. These things coupled with the need to adapt to the changing needs of the business and the problems of poor fault tolerance and a lack of performance engineering just accentuate the problem. As discussed in Section 2.4 of chapter 2, unlike software engineering, the absence of standardised methodologies for supporting development, restricts the availability of large-scale high-quality domain ontologies. Although these methodologies provide an engineering approach with an adequate level of detail, most of them lack sufficient detail on the techniques and activities employed (Iqbal et al. 2013).

4.1.1 Components of Domain Ontology

The literature on ontologies states clearly that the core components of domain ontologies are *concepts*, *relationships*, *instances*, *constants*, *attributes*, *formal axioms* and *rules* (Corcho et al. 2005).

Concepts are the core component of a domain ontology and are comprised of class, type and universal aliases. A concept usually represents a group of individuals having some common characteristics (Lord 2010), which are used to show the resemblances with classes in an object orientation. For example, in the teaching and learning domain, concepts are module, student and teacher etc. Concepts in ontologies were traditionally organised in taxonomies, but in this research, a *subclass-of* relationship has been used to establish the hierarchical relationship between the concepts.

Relationships in ontology mainly describe how individuals are related to each other. This can be represented directly between the individuals (e.g. “this java module has a tutor called Alex”) or between the concepts (e.g. “this module has a tutor who is also a person”). In the second case, the relation describes the relationship between all the individuals of the

concepts. In general, a relation in domain ontology represents an association between the concepts of the domain. The relationship which connects two concepts is called a binary relationship. For example, the relation *teaches* links *teacher* to *module* in the teaching and learning domain. Each binary can have its inverse relation that connects the concepts in the opposite direction. For example, *is taught* is the inverse of *teaches*. Though it depends on the formal languages, it is often possible to express various kinds of relationships between concepts such as *existentially quantified*, or *universally quantified***.**

Instances are also called individuals and are the building blocks of a domain ontology. They are the basic runtime unit of an ontology. Individuals can model both concrete and abstract objects. For example, an instance of the concept module is *network programming* (Corcho et al. 2005).

Constants are represented by literals or numeric values. For example, the number of access modifiers in Java is four and will not change, so is a constant.

Attributes describes the properties of both instances and concepts. Two major categories of attributes are *instance attributes* and *class attributes*. Similar to that of object-oriented design, an instance attribute describes an instance-specific value. These attributes are defined in a concept and are inherited by its sub-concepts and instances. For example, the code of a module is specific to each individual. On the other hand, class attributes describe class-specific values for the concept where they are defined. Class attributes are neither inherited by the subclasses or the instances. Ontology-development tools usually provide predefined domain-independent class attributes for all the concepts, such as concept documentation, synonyms, acronyms, etc. Besides these, other user-defined domain-dependent class attributes can usually be created.

Formal axioms are used to constrain values for concepts or instances. They are logical expressions that are always true and are normally used to specify constraints in an ontology. Properties of relationships are a kind of axiom. Generally, rules are used to conclude knowledge in an ontology, such as attribute values and relation instances etc.

4.1.2 Evaluation of Existing Methodologies

Many methodologies have been proposed and were followed by the experts in the past for domain-ontology development. A detailed description of these leading methodologies were presented in section 2.3 of chapter 2, where the pros and cons of ten leading methodologies were identified and described. However, for a comparison between those methodologies a

criterion has been followed. The criterion has been established based on a thorough review of the related literature and the observation of the needs and trends which have evolved in the field of ontology engineering in general and particularly for ontology-development methodologies. This section describes the parameters of the criterion and gives a consolidated comparison of ten chosen methodologies in a tabular form. The rationale behind this comparison was to identify the limitations of existing methodologies when compared against the specific parameters. The outcome of the comparison of the methodologies was used as a vital input for the design, and the matrix based on existing ontology development was used for the evaluation of SCIM. A detailed description of the evaluation is presented in Chapter 6 of this thesis.

The comparison criterion was populated based on a set of parameters which reflected the combination of both high-level and technical-level aspects of ontology development. Seven different parameters were identified for the stated purpose, which included the essential aspects of any ontology-development methodology. Moreover, the parameters were chosen with the intention of providing a quick understanding of the chosen methodologies. Parameters representing the higher level of an ontology-development methodology were: *mode of development, support for collaborative ontology development, support for re-usability, and support for interoperability. Parameters representing the technical aspects of the methodology were: the extent of application dependency, ontology life-cycle support* and the *coverage of employed methods and activities*. A comprehensive comparison of the chosen parameters for each of the selected methodologies is presented in Table 6. A description of each of the parameters is presented in the following paragraphs.

Mode of development states the broad category of a methodology which includes: *stage-based,* an *evolving prototype* and a *set of guidelines*. A stage-based mode was suitable in situations where the initial requirements were clear, whereas an evolving-prototype mode was best-suited to situations where the requirements were not clear. The third mode mainly emphasised guidelines rather than the details of the overall development. Each one of these modes had its own pros and cons, however, an ideal methodology should have at least one of these modes of development. These can assist practitioners to select the right methodology according to the availability of the requirements for the ontology.

Support for collaborative ontology-development states whether a methodology allows different members of the ontology development team (ODT) to work together on a single ontology in a shared manner. This can be done by proposing activities/techniques which

encompass mechanisms to encourage collaborative-ontology development at a minimal level. The maximum level in this regard can be a facility offered for ODT to work on a single ontology concurrently regardless of their location.

Support for reusability is one of the key parameters in ontology engineering as ontology development is a complicated and time-consuming task. Therefore, understanding the methods used on existing methodologies to support reusability is vital when developing a new methodology.

Support for interoperability states whether the methodology supports interoperability between different systems and shares the abstract-level knowledge structure.

E**xtent of application dependency** states whether the methodology is: developed on the basis of pre-planned knowledge (i.e. application dependent), semi-independent (i.e. using some existing ontology scenarios for the specification) or application independent (i.e. no assumptions made for the applications which uses the ontology).

Ontology life cycle support represents whether the methodology defines the stages of the ODLC. A well-defined methodology is one of the mandatory requirements for a standardised methodology.

Coverage of employed methods and activities states whether the methodology defines the activities and techniques within the ontology life cycle.

It has been observed from the analysis that the various parameters used for comparison in most of the existing ontology-development methodologies (ODMs) have limitations, which indicates the need for proposing a new methodology which can overcome them. For instance, it has been noted that most of the existing methodologies failed to propose a well-defined ODLC. Furthermore, it has also been noted that many existing methodologies lack an adequate coverage of the methods and activities employed. The notion of reusability/re-engineering is limited to only a few ODMs. Collaborative ontology development can be enhanced further by applying agile techniques which have been proven in other branches of engineering. Literature and practice have shown that the key factors behind the success of software engineering is the availability of standardised and well proven methodologies, a rich set of mature development platforms with integrated tool support and an Application Program Interface (API). Unlike software engineering, there is no single methodology for ontology development which can be described as ‘fully mature’. Hence, there are many correct ways to model a domain (Gaševic, Djuric and Devedžic 2009). The literature had pointed out the

fact that the process of ontology design and development would be simpler and less time consuming, if there is a methodology available having a close resemblance with the software development process due to the complementary nature of these two popular branches of engineering (De Nicola, Missikoff and Navigli 2009). For this reason, this researcher strongly believes that a standardised methodology with tool support for domain modelling can make a significant difference by bridging the gap between software engineering and ontology engineering.

Name of Methodology	Mode of development	Support for collaborative ontology development	Support for re usability	Support for interoper ability	Extent of Application dependency	Ontology Life Cycle support	Coverage of employed methods and activities
Ushold and King (KEM)	Stage based	No	No	No	Application dependent	No	Limited coverage available for purpose identification, ontology building and evaluation.
Gruninger and Fox (TOVE)	Stage based	No	Yes	No	Application Semi-independent	No	Limited coverage available for informal specification, and formulation of the competency question
METHONTOLOGY	Evolutionary prototype	No	Yes	No	Application independent	Yes	Sufficient coverage for specification, conceptualization, formalization, integration, implementation and maintenance
Ontoligua	Modular development	No	Yes	Yes	Application independent	No	Limited coverage on ontology development and integration.
Common KADs and KACTUS	Modular development	No	Yes	No	Application dependent	No	Limited coverage on ontology design and development
On-To-Knowledge	Evolutionary prototype	No	No	No	Application dependent	Yes	Limited coverage on ontology design and development
UPON	Evolutionary prototype	No	Yes	No	Application independent	Yes	Limited coverage on ontology design and development
XP.K	Evolutionary prototype	Yes	No	No	Application independent	No	Limited coverage on ontology development
EXPLODE	Evolutionary prototype	Yes	No	No	Application independent	Yes	Limited coverage on ontology development
RapidOWL	Evolutionary prototype	Yes	Yes	No	Application independent	No	Limited coverage on ontology development

Table 6: Comparison among well-known ontology development methodologies

4.2 Software Process models and their viability for Ontology Development

Software engineering is defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software" (IEEE Standard Glossary of Software Engineering Terminology-Description 1990). This definition itself implies that software engineering relies on the Software Development Life Cycle (SDLC). The SDLC is a well-defined, structured sequence of phases/stages to be followed to develop a software product (Software Development Life Cycle n.d.). In general, the typical SDLC phases comprise: analysis, design, implementation, testing and maintenance (Sommerville 2010). The SDLC follows a focused approach with phase-specific activities/techniques. The placing of activities in the phases and the expected deliverables against the phases offer scope for review at the end of each phase.

Since the very beginning of the era of software engineering, by following the SDLC, the systematic development process models for software projects was assured. Though each of the process models had strengths and weakness, all of them encompassed the key phases of software development. It has been observed from the related literature that well-proven software-engineering process models can be categorised into either linear, iterative, incremental or agile.

Linear process models complete various phases of software development in a sequential manner (Avison and Fitzgerald 2003). An instance of this approach is the powerful waterfall process model, which, due to the sequential arrangement of the SDLC phases, can help new developers to understand the big picture of software development. Therefore, this model assists the stakeholders to correctly define the business requirements documentation (BRD) and the software requirement specification (SRS). Waterfall is an ideal choice if the requirements are clear, however, if they are not, problems can be caused on projects because of scope creep, particularly if the project sponsors are indecisive. Inadequate communication with users, combined with a lack of their involvement during the software development, are other pitfalls. The rigid nature of this process model gives little opportunity to respond to a change in requirements.

Since the requirements for ontologies change as the ontology evolves, following a linear software process model doesn't guarantee the delivery of a mature ontology. Moreover, all the domain concepts, attributes and semantic structural relationships may not be identified in the beginning. The literature reveals that ontology development is an iterative process requiring phases to be re-visited when change is needed. However, although the rigidity of

the linear engineering process model can lead to quality problems when change occurs, nevertheless, it is one of the models that is worth considering when proposing an ontology-development methodology.

Iterative process models allow the development team to revisit the same phase/stage until the expected outcome is achieved (Avison and Fitzgerald 2003). Compared to the other categories, this type of process model is more effective for the management of risk. The capacity to accommodate changes during any phase of the SDLC gives adaptability and rapid turnaround. This kind of model works well for software projects which are large, complex or have high costs, because this adds to the risk of failure. For high risk projects special skills are required for risk management and additional mechanisms are needed for risk mitigation.

Although ontology development is iterative in nature, the adoption of a well proven methodology, with a mechanism such as Rational Unified Process (RUP) which gives good risk mitigation, can be an appropriate choice for ontology development. The components provided within RUP such as cycles, iterations, phases and workflows offer flexibility throughout the process. Due to the similarities between the SDLC and the ODLC, the phases of RUP can be followed for ontology development. However, workflows need to be customised according to the specific nature of the ontology, a possible extension being the inclusion of appropriate agile techniques to meet the needs of the domain.

Incremental process models are flexible in nature and offer a piece of working software in a quicker time frame when compared to other process models. Instances of this process model give maximum flexibility for accommodating a change in requirements with fewer cost implications. As the underpinning principle of this process model is '*plan a little, design a little, develop a little and test a little*', evaluation is integrated with small iterations in each phase. The mapping of milestones within each iteration is possible within an incremental model, hence, each iteration can produce a small piece of the deliverable for each phase of the SDLC. Although this offers flexibility, each phase of the iteration is rigid, as there is no overlap of phases. Moreover, problems may arise pertaining to the system architecture if all the requirements are not known at the beginning of the SDLC. The rigidity of the iterations is a stumbling block for ontology development, however, the integration of an evaluation stage in each iteration is a positive aspect which can be incorporated in standardised ODMs.

Unlike the categories mentioned above, the fast delivery of software from agile methodologies offers a project development style suited to the needs of the ever-changing software-development environment. An agile methodology follows an incremental model with a rapid cycle time with small incremental releases of the software throughout the SDLC. These kinds of methodologies incorporated activities/techniques like Joint Application Development (JAD) to boost collaborative effort for software development. This emphasised the people required and the interaction between them rather than the processes and the tools needed. One of the criticisms of agile methodologies is the lack of depth in the documentation for the design phase of the SDLC. The need for customer involvement can also be a problem, for example, if the customer representative is not clear on the final outcome, the project can easily go off track. Ideally agile methodologies are more suited to small to medium-sized projects.

For ontology development, the size and complexity of an agile software-methodology can prevent it being considered as an option for development. Although some lightweight ODMs have been proposed, the design of a standardised agile methodology for ontology development can prove to be a difficult task because of the demand for active user involvement and collaboration throughout the development life cycle. As ontology development is a time-consuming process, these demands on the user can prove to be a problem. However, certain agile techniques such as the integration of evaluation and user involvement, can be applied to the ODM throughout the ODLC with the necessary customisation.

To summarise, no single process model for software engineering can be applied to ontology development as such, because they were designed exclusively for software development. However, there is plenty of scope to consider the many features of software process-models when proposing a software-centric approach for methodology development. A hybrid built from the best features of many different software process-models together with the necessary customisation can be a good option as the underpinning philosophy of a new methodology. The next section introduces the proposed ODM where these ideas have been employed.

4.3 Proposed Methodology for Ontology Development

This section describes the background, overview of the processes, lifecycle and the hierarchy of components in the proposed ODM which are detailed in the following sub-sections.

4.3.1 Background of proposed methodology

The rigorous process of ontology development requires a methodology with well-defined stages and workflows as it is composed of critical activities and techniques. Some of the existing methodologies mainly prescribe the guidelines for the generic stages of ontology development such as specification, conceptualisation, formalisation and implementation (Calero, Ruiz and Piattini 2006). The specification stage primarily covers the aims and purpose of the ontology together with an indication of its intended users. The conceptualisation stage, at its simplest level, covers the vocabulary of the domain represented in a conceptual model. During the formalisation stage, the conceptual model is transformed to a semi-formal representation. The implementation stage creates a formal representation from the semi-formal version by applying formal representation-languages such as the Resource Description Framework (RDF), Web Ontology Language (OWL), and the support of available ontology-editing platforms.

A new methodology which is called SCIM has been designed in this research which applies proven process models of software engineering to ontology engineering. The need for this has been demonstrated in the literature and the rationale for it is the similarity between these two complementary branches of engineering. The four stages discussed have been considered as the foundation of the methodology as they are the main stages for ontology development in any domain. The other components of the proposed methodology are built on top of these stages.

A hybrid model combining the features of both linear and iterative models are the cornerstone of the proposed methodology. The stages, workflows, activities and techniques of the ontology development life cycle (ODLC) have been defined and a sequential approach has been applied to each stage. The iterative model has been applied to the activities within the workflow. The integrated evaluation-mechanism of the incremental model has been applied to ensure that each iteration of the development is evaluated. The best techniques from lightweight agile methodologies have been applied to support collaborated ontology development. The detail of the steps involved in the development of SCIM have been described in the respective sections of this chapter.

4.3.2 Overview of SCIM

SCIM classifies the core ontology development process into four stages. These stages are *Planning, Conceptualisation, Development followed by Implementation and Deployment.* The base of the stages has been triggered from the lifecycle of one of the existing methodologies called METHONTOLOGY which has been generally accepted by the ontology-development community. The engineering principle behind METHONTOLOGY is the intermediate representation of ontology-milestone deliverables. SCIM follows the same engineering principles with differences in the way in which the components are represented. More software-engineering techniques have been applied in the selection of model components and techniques. For example, the conceptual model is represented by the Unified Modelling Language (UML), and the formal representation languages adopted for the ontology implementation are RDF and OWL. Since both methodologies are following the same engineering principles, an attempt has been made to map the main stages of SCIM with METHONTOLOGY as shown in Figure 19. A detailed description of the activities and techniques of SCIM are described in the ODLC in Section 4.5.

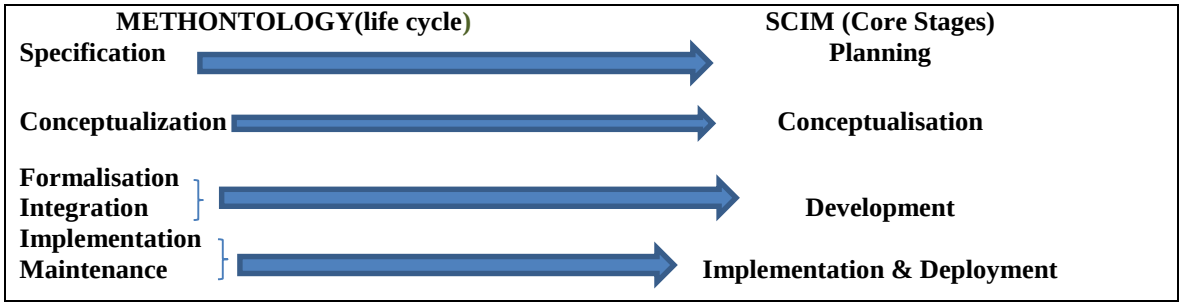


Figure 19: Mapping of SCIM with METHONTOLOGY life cycle

SCIM has the characteristics of both the linear waterfall process and the iterative Rational Unified Process (RUP), a couple of well proven and widely accepted software engineering process models (Kruchten 2003). Since the RUP is one of the process models of SCIM, it adopts the approaches of the RUP such as cycles, iterations, phases and workflows. However, the activities and techniques used in each workflow have been customised to cater for the needs of ontology development. In the RUP, the workflows and phases (i.e. Inception, elaboration, construction and transition) are independent in principle. Moreover, the contribution of the workflow per iteration varies from phase to phase. Figure 20 a. shows the conventional RUP diagram with the disciplines used in software engineering.

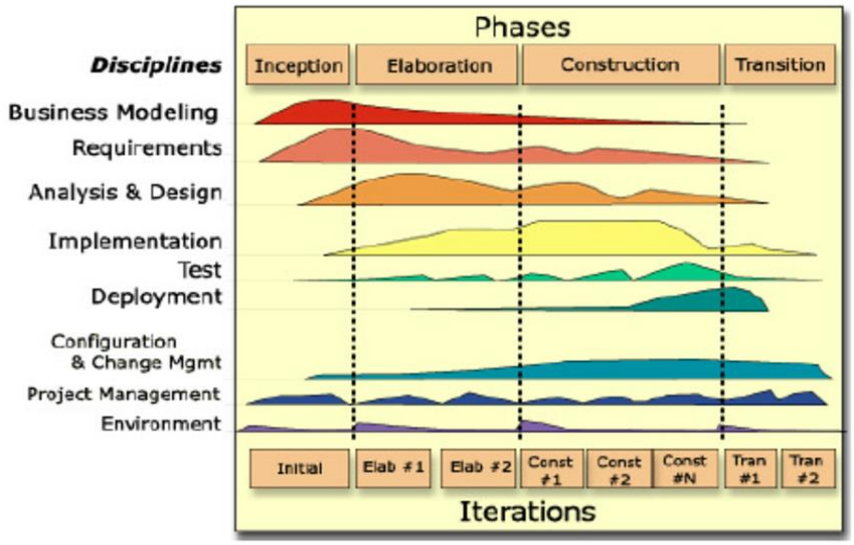


Figure 20 a: Conventional RUP diagram (Kruchten 2003)

The five customised workflows of SCIM that have been mapped against the RUP are *Requirement analysis, Domain Analysis, Conceptual Design, Implementation* and *Evaluation.* The activities and techniques of each workflow are described in Section 4.5. The number of iterations varies per workflow and the domination of activities per cycle is directly related to the workflow. The initial phase focuses more on pre-development whereas the later phases focus on the development and post development of the ontology. Figure 20 b. presents the customised framework of SCIM from the RUP with an abstract-level indication of the deliverables per phase.

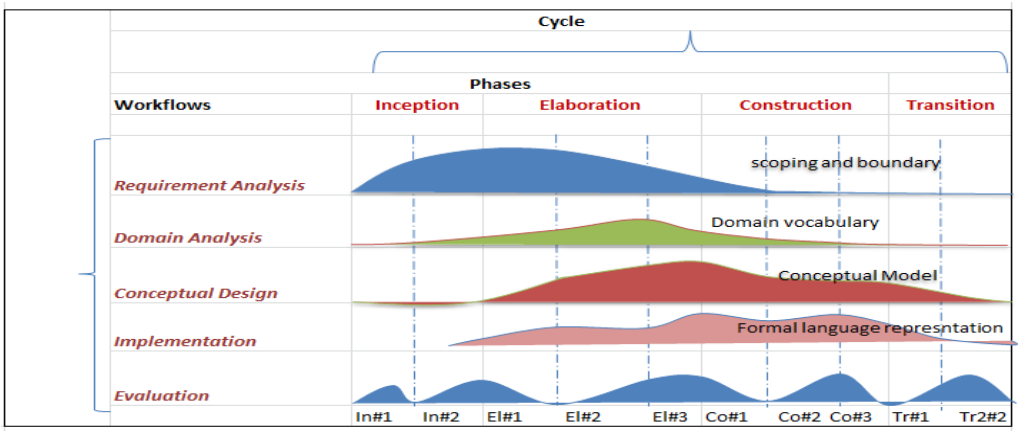


Figure 20 b: Customised SCIM workflows and deliverables per phase

The inception phase focuses mainly on the period up to the finalising of the requirements for the ontology. In the elaboration phase, a domain vocabulary is produced after the completion of the iterations. There are more iterations in the construction phase because the main

conceptual model is being built. During the transition phase, the ontology deployment and post-implementation activities take place. This period can range from the maintenance stage through to the deployment of the ontology. As with the lightweight agile software-engineering methodologies, evaluation is embedded in all the phases. The Domain Expert (DE) and Knowledge Engineer (KE) are involved in the various activities bound with the respective workflows, as shown in Figure 21.

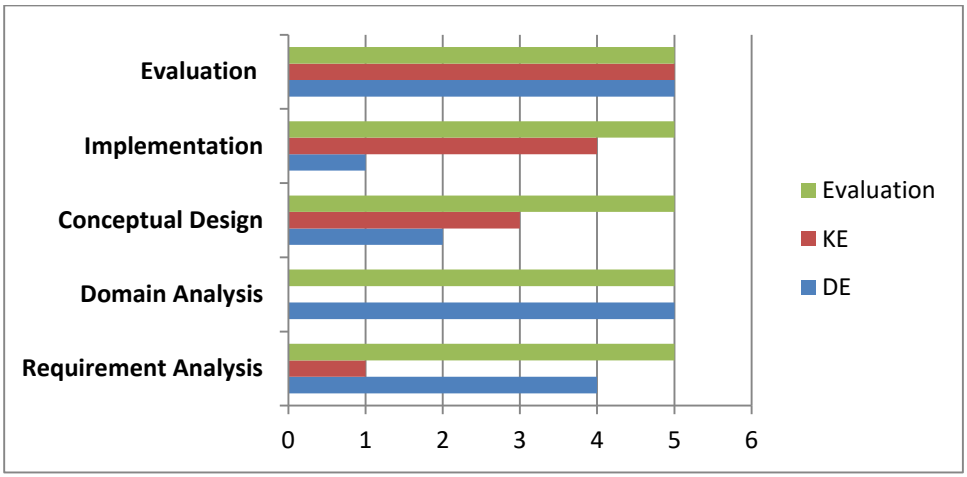


Figure 21: Involvement of key experts in the workflows of SCIM

SCIM uses UML for conceptual modelling as well as for blue-printing in a few other ontology development activities. UML has already shown its usefulness in ontology development (Guizzardi, Herre and Wagner 2002) and the RUP's hand in hand support with UML has been well recognised in SCIM. This will be an added benefit for software designers and practitioners in the development of ontologies (ElenaTudoroiu, Cretu and Paquet 2009).

What uniquely distinguishes SCIM from other methodologies is that it is a hybrid model derived from the underlying principles of well-proven software engineering process-models. Both the philosophical and engineering aspects of SCIM have been synchronised with the existing methodologies and standards. The hierarchy of SCIM consists of stages, workflows, activities and techniques. An incremental approach has been applied between the stages and an iterative approach has been applied between the activities within specific workflows. The core components of SCIM are generic, so it is not designed for a specific use. This means that is a generic domain ontology development, unlike UPON and other software-centric ontology-development methodologies. The activities assigned to each workflow and the deliverables against them are not problem-based. Techniques used in lightweight agile methodologies are embedded within the activities of the SCIM for many reasons. These reasons include ensuring the involvement of the end-user community in the

design process and the delivery of evolutionary prototypes at the completion of each iteration if required. The involvement of the end-user community from the beginning significantly reduces the role of KE and helps to evaluate the completeness of the ontology at every stage. A usable part/model of the ontology can be released earlier than with conventional ODMs.

4.4 Ontology Development Life Cycle of SCIM

Software-development methodologies have a life cycle, with a specified set of phases and tasks to be to be followed for the successful realisation of the software development process. It has been observed from the evaluation of the existing ODMs in Section 2.3 that most of them did not have a well-defined ODLC and the support for the ontology development was limited to a set of guidelines. From the evidence provided by the literature and case studies described in Chapter 2 of this thesis it was clear that the ontology-development process needed a set of stages, activities and tasks. The order of the execution of each of these components was realised in the ODLC which specified the specific stages for the ontology development. In the case of SCIM, the stages are *Planning, Conceptualisation, Development, Implementation and Deployment***.** The typical ODLC should have the components for the transformation of domain concepts to a well-documented and evaluated domain ontology and should clearly specify the sequence of activities to be followed within each stage.

The underpinning philosophy of SCIM is the software-centric approach which has been embedded into a hybrid model derived from two well-proven software-engineering process models. The main components of SCIM are the stages, workflows and activities, which follow the linear waterfall model, and the iterative activities within the workflows. Since ontology development is an iterative process, SCIM has the features of RUP for the overall development process which includes cycles, iterations, phases and workflows. Process mapping has been applied between the workflows of the RUP and SCIM to ensure the inclusion of all the disciplines of conventional RUP in the SCIM, together with the necessary customisation/grouping. After careful analysis, nine software-engineering related disciplines of conventional RUP have been mapped to five ontology development workflows. These workflows were placed within the respective stages of SCIM and subsequently the activities of SCIM have been embedded within the workflows. Guidelines specifying the various techniques to be applied in each activity are given in the ODLC. Since deliverables against milestones are measurable entities, the respective deliverables are proposed against each workflow. Figure 22 illustrates the abstract view of the ODLC for SCIM.

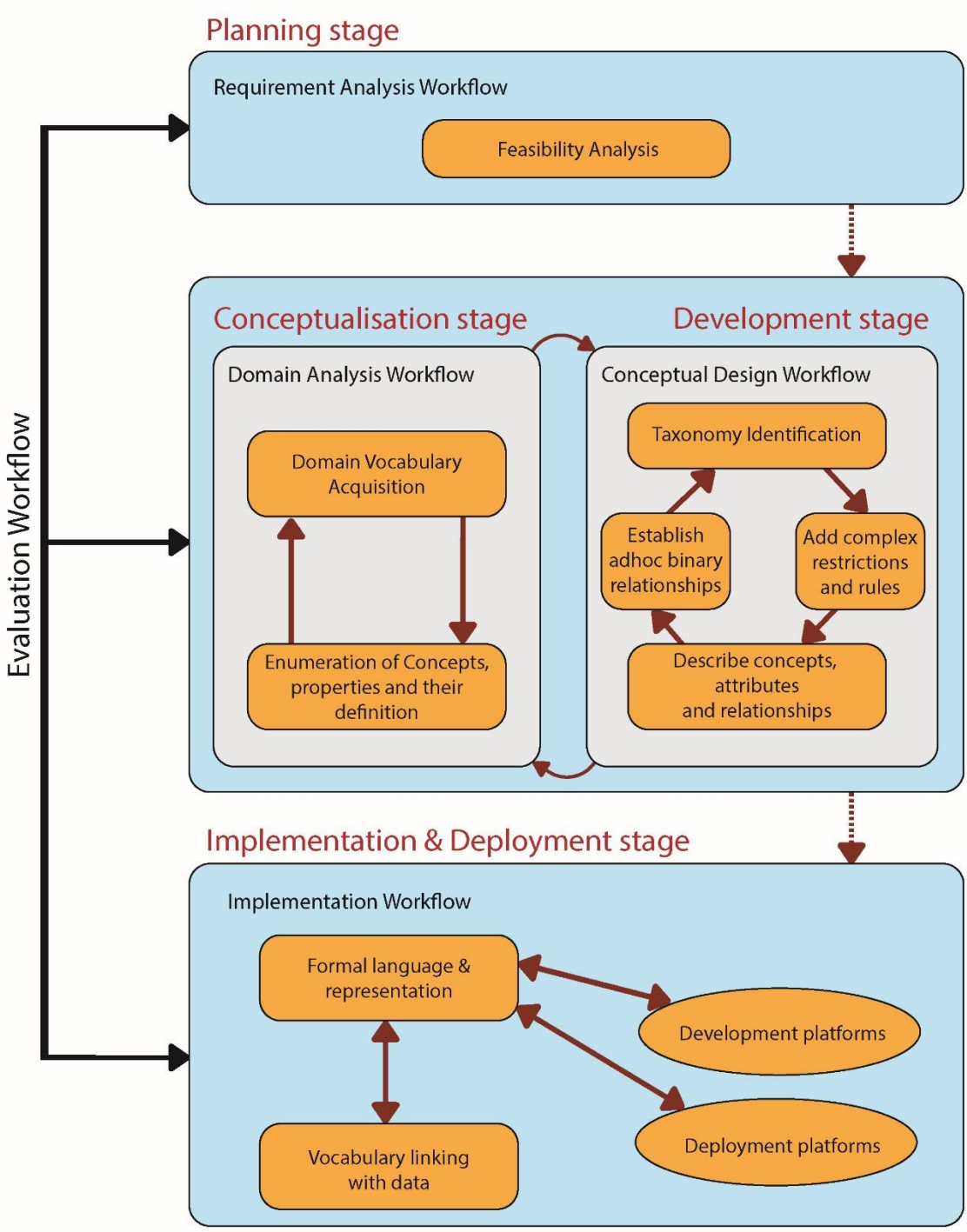


Figure 22: Abstract view of the ODLC for SCIM

4.4.1 Stages of the SCIM

This section describes the main purpose and deliverables of each of the stages of SCIM. These stages follow the sequential waterfall approach in their order of execution and the outline for the workflows and the activities integrated into each stage are described.

4.4.1.1. Planning Stage

The ultimate goal of this stage is to produce a semi-formal, formal or mixed mode Ontology Requirement Specification Document (ORSD), the extent of formalism varying according to the techniques applied for the preparation. One of the pitfalls observed in the literature is that the guidelines offered by existing ODMs are inadequate for defining the domain requirements. Few of the older methodologies such as the Grüninger and Fox methodology (Grunninger and Fox 1995), the On-To-Knowledge methodology (Staab et al. 2001), and the Unified methodology (Ushold and King 1995), have identified the importance of the ORSD to describe the purpose, the intended users, and have provided a set of requirements for the ontology. However, other than describing a few high-level steps they have failed to provide enough detail on the systematic techniques needed to produce the ORSD. Other than the usage of competency questions (CQs), only a few techniques have been applied for the requirement evaluation in these previous methodologies, but in SCIM, a systematised approach using agile techniques has been proposed for the preparation of the ORSD.

An ORSD can be prepared in many forms, which are formal (in natural language), semi-formal (for the use-case description), or mixed mode (where the use-case descriptors are supplementary to the document prepared in natural language). As part of the process mapping stated in Section 4.3, the requirement-analysis workflow is mapped to the planning stage of SCIM where the sole activity is the feasibility analysis. The scope and boundary of the proposed domain ontology is decided during this activity. The identification of the first-level concepts is a part of scoping, therefore, appropriate techniques have been proposed for the completion of the feasibility analysis in SCIM. Unlike other ontology development methodologies, SCIM uses software development techniques for the scoping and boundary finalisation. As in the software-engineering process models, agile techniques are proposed for the completion of the feasibility-analysis activity. These agile techniques include: Joint Ontology Development (JOD), motivational scenarios, and the usage of complex and simple competency questions.

For the methodological guidelines of the ORSD, SCIM follows the context of the existing NeOn methodology (Gómez-Pérez and Suárez-Figueroa n.d.), which considers the availability of existing ontologies in ontology networks. The Neon methodology also takes into account the reusability of existing ontologies and collaborative ontology-development which are also incorporated in SCIM. Taking this into consideration, a filing card activity (Suárez-Figueroa , Gómez-Pérez and Terrazas 2008) is proposed as the base for the stage one deliverable. A well-documented ORSD should include the description of the domain concerned, the purpose of the ontology, the scope and boundary of the domain ontology, domain-specific assumptions, the level of formality, details of techniques applied, ontology requirements and the list of sources used for the extraction of concepts. Figure 23 shows the template of the ORSD.

ONTOLOGY REQUIREMENTS SPECIFICATION DOCUMENT (TEMPLATE)
Domain: **Date of Creation:** **Version No:** **Author(s):** **Explicit Domain Assumptions if any:**
Purpose: The main goal of the ontology should be stated here with rationale
Level of formality: Indication of the proposed formal language for ontology implementation
Scope: The general coverage of proposed ontology at least in terms of key concepts and attributes
Boundary: The Specific coverage of proposed ontology in terms of all concepts, attributes and formal axioms
Integration with any other existing ontology:
Intended Users: List of end users expected for the ontology. Motivational scenarios can be applied for the identification of intended users.
Ontology Requirements: • *Functional requirements:* The specific set of requirements that the ontology should fulfill for its Intended users including optional priorities • *Non-functional requirements*: The general requirements that the ontology should fulfill for its Intended users including optional priorities
List of sources: Resources used for the concept extraction
Techniques applied: State the technique applied for concept extraction e.g., competency questions, survey etc.

Figure 23: ORSD template.

4.4.1.2. Conceptualisation Stage

The conceptualisation stage focusses on the identification of the concepts that exist in the domain together with their properties. The goal of this stage is to produce a conceptual model,

which can be delivered in the form of a domain vocabulary in its minimal form. As stated in 4.2, the domain analysis workflow has been mapped to this stage in SCIM. The activities embedded in this workflow are *domain vocabulary acquisition*, together with the enumeration and definition of *concepts*, and *properties*. The deliverable after the completion of this stage will be a glossary of terms, with a definition of all the concepts and the details of their properties. For the fulfilment of the affiliated activities, techniques such as the use of a survey, the extraction of concepts from documents, and the deployment of concept-mapping tools (Novak and Canas 2014) can be applied at this stage. Concept maps are used to capture the concepts of a domain in a more formalised manner than mind mapping. They present the knowledge in a meaningful form which helps for reusability and shareability. Automated tools (e.g. CTOOLs) can also be applied at this stage for the concept and relationship extraction. Both data and object properties belonging to the concepts will be listed during this stage. Software-centric object-oriented analysis approaches, such as the noun identification technique and UML class models can be applied for the creation of the internal structure of the concepts. Figure 24 shows the representation of the internal structure of a concept.

CONCEPT REPRESENTATION STRUCTURE	
Concept Name	Description
Attribute/Property 1 Attribute/Property 2 .. Attribute /Property n	Description of attributes: This includes the nature of property such as class property or instance

Figure 24: Internal structure of concept

4.4.1.3. Development Stage

The development stage is very much related to the conceptualisation stage, which are also interconnected. The goal of this stage is to transform the conceptual model of the ontology defined at the conceptualisation stage to a semi-formal model. Therefore, an iterative interconnectivity is defined between these two stages as shown in the abstract view of the ODLC for SCIM (Figure 22). As a result of this interconnectivity, the conceptual-design workflow has been mapped to the development stage. The core activities embedded in this workflow are for: *taxonomy identification, building of ad-hoc binary relationships, adding complex restrictions/rules and describing concepts, attributes and relationships*. These are generic activities for the development of the domain ontology regardless of the domain concerned. As part of the conceptual model development, this stage emphasises the main

elements of the ontology such as *concepts, relations, attributes, instances, constants, axioms and rules*.

The taxonomy-identification activity sets up of a concept hierarchy, to create an ontological structure between the concepts. Although it is not possible to develop the conceptual model in a sequential manner completely, it does need to have an order to work with, for model representation. For the development of the concept taxonomy, the SCIM follows the relations *Subclass-of, Disjoint-Decomposition, Exhaustive-Decomposition* and *Partition.*

The 'Build ad-hoc binary relationships' activity defines the inverse, transitive, symmetric and reflexive relationships between the concept taxonomy. The activity describes *concepts, attributes and relationships* in a concept lexicon which contains the domain concepts, relations, instances and the class (or instance properties). Once the concept lexicon has been built, it can be documented in detail by the development team.

The *add complex restrictions and rules* activity defines and states explicitly the formal axioms for the proposed domain ontology. The appropriate mechanisms such as first-order logic can be applied to describe the formal axioms.

4.4.1.4. Implementation and Deployment stage

This stage of SCIM involves the usage of an appropriate Ontology-Development Environment (ODE) just as software development requires an Integrated-Development Environment (IDE). The selected ODE should support the formal languages, as the deliverable for this stage is an evaluated formal-language coded ontology. Therefore, for the conceptual model, SCIM recommends an ODE with a formal language code-generation facility. To ensure the correctness of the ontology development, the ODE used the reasoner. In SCIM, the implementation workflow has been mapped against this stage with the necessary activities. *Formal language representation* and *vocabulary linking* with the data are the activities integrated with the implementation stage.

4.5 Workflows and Activities of SCIM

This section describes in detail the workflows of SCIM and the activities integrated into them. The phases of RUP are applied in all the workflows, so the activities attached to each workflow are iterative as discussed in section 4.2.

4.5.1 Requirement Analysis workflow

The feasibility analysis is the only activity proposed for this workflow. As shown in figure 20b, it has been mapped against the first two conventional workflows of RUP which are business modelling and requirements definition. This activity covers the requirement analysis by specifying the semantic requirements of the ontology users. It also includes the identification of scope, purpose, boundary, intended users and ontology requirements. A mixed mode ORSD will be produced at the end of the activity which becomes the deliverable from the planning stage. A number of techniques are applied in this activity to define the degree of formality of the ORSD.

Also, during this activity, there is a significant involvement of the Domain Expert (DE) and users. A variation of the agile technique followed by the use of lightweight software-methodologies such as Joint Ontology Development (JOD) sessions have been proposed in SCIM for the users and DE, to define the purpose, scope and boundary of the ontology. As an optional requirement, the selection of an implementation language can be decided in this activity, subject to the availability of KEs for the JOD sessions. During the JOD sessions, the end-users of the proposed domain ontology will be finalised. Other variations of agile techniques used in this workflow are motivational scenarios, the usage of competency questions and the modelling of the analysis use-cases, to complete the identification of the ontology requirements. As with software engineering, functional and non-functional requirements are defined during the JOD sessions with the users, using informal interviews and competency questions. The functional requirements are domain-specific and refer to the knowledge represented in the ontology, whereas non-functional requirements are generic and refer to knowledge, which is not specific to the domain concerned, but which should have been satisfied by time the ontology is completed. A Set of UML use-case descriptors and a diagram can be used to express the requirements and should be prepared in natural language for better understanding, and attached as a supplement to the ORSD. As evaluation is associated with all the workflows, the validation of the requirement is a technique embedded in the activity. The requirements validation will be carried out using the competency questions and relevant criteria to identify the conflicts, missing information and contradictions in the domain knowledge. The overall tasks of the feasibility analysis activity are presented in Figure 26 as a use-case diagram.

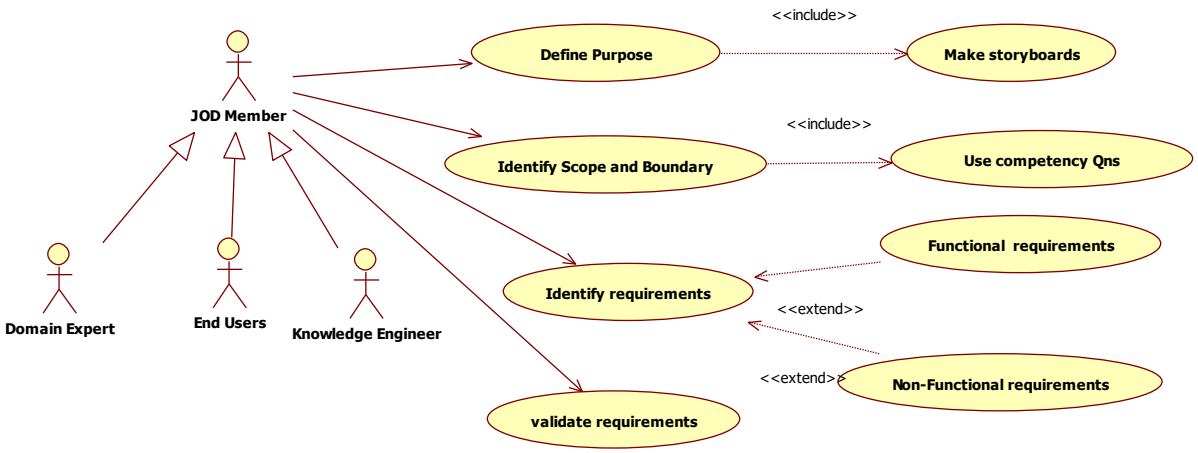


Figure 26: Use Case diagram of feasibility analysis activity

The scope and boundary definition for the proposed ontology includes the identification of the abstract areas for the domain concerned. The concepts belonging to those areas also need to be represented in the ontology in detail. Defining the business purpose of the ontology with motivational scenarios finalises the categories of ontology users. The usage of competency questions ensures the ontology meets certain criteria at the conceptual level and can also be used for the evaluation of the workflows.

Apart from studying the environment in which the ontology is to be deployed, the possibilities for integrating the ontology into other systems are also reviewed in this workflow. Agile techniques ensure the involvement of the DE who is a key resource for this workflow.

4.5.2 Domain Analysis workflow

This workflow is mapped against the "analysis and design" workflow of the conventional RUP as part of the process mapping discussed in Section 4.2. The development of an ontology starts with the definition of concepts related to the scope of the domain concerned. *Domain vocabulary acquisition, enumeration of concepts, and the definition of properties* are the main activities attached to this workflow, which refine the requirements finalised in the requirements workflow. As activities embedded within the workflow are completed, deliverables will be produced which include a generic domain vocabulary (containing a finer list of domain concepts) and a list of properties. The deliverables will then be listed in the final glossary of terms, with a definition of the concepts.

4.5.2.1. Domain vocabulary acquisition

Classes are the key elements of domain ontologies which are derived from the concepts after refinement, which in the case of SCIM is a result of the vocabulary acquisition activity. The DE plays a vital role in this activity being the resource with the best knowledge of the domain. The vocabulary is built after the detailed identification of the domain specific concepts, where UML represents the concept structure and the related attributes. The more refined vocabulary eventually extends to a glossary which is built on top of the concepts extracted from the existing domain-specific documents such as reports, policies, procedures and standards. The glossary can be produced in a different form for each methodology but SCIM uses the same format as the legal domain ontology (Corcho et.al 2005) developed with METHONTOLOGY. The glossary includes all the relevant domain-specific terms which include the synonyms, acronyms and the description of concepts. A survey of the domain users was performed to provide the detail for the concept identification. The development of the glossary can be supported by the use of automated concept-mapping tools. Figure 27 shows the proposed SCIM template for the glossary of terms.

Concept Name	Synonyms	Acronyms	Natural language Description	Concept type

Figure 27: Glossary template

The application-specific and domain-specific vocabularies (i.e. concepts from the application and domain) are the two kinds of vocabulary identified during this activity. Similar to the mechanisms followed by software engineering approaches for ontology building, SCIM proposes an introduction to the intermediate vocabulary (De Nicola, Missikoff and Navigli 2009). As the final deliverable of the activity, prior to the development of the domain vocabulary, an intermediate vocabulary will be prepared by combining the application-oriented, domain-oriented and common concepts. Figure 28 illustrates the technique behind the intermediate vocabulary creation using a Venn diagram.

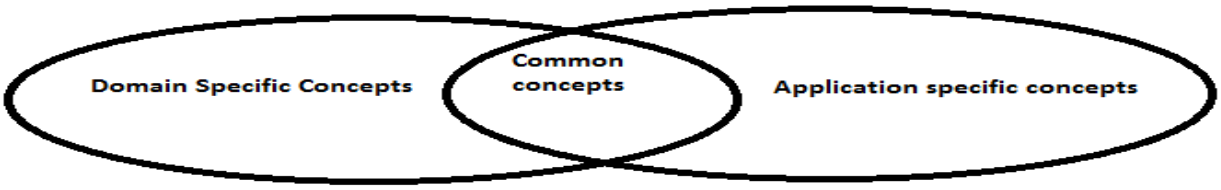


Figure 28: The technique behind intermediate vocabulary creation

Intermediate vocabulary construction can be completed with all the common, domain concepts after the approval of the DE. Then the relevant application-oriented concepts will be incorporated after the approval of the KE. Definitions will then be added to the concepts to form a more refined domain vocabulary.

4.5.2.2. Enumeration of Concepts, Properties and their definition

The purpose of this activity is the listing of concepts/classes and properties for the concepts included in the refined glossary, developed in the previous activity. The properties (which provide the structure to the concepts) are enumerated, being represented as either *atomic* or *complex*. Atomic properties can be intrinsic or extrinsic, based on their nature and they model first level information (e.g. phone no) whereas complex attributes model more structured information. The KE together with the DE can perform activities iteratively if required. In formal representation-languages, properties are of two major categories, which are: *data properties* and *object properties*. Data properties describe value of the property with its value, whereas, object properties link an instance of the concept to another instance. This workflow leads to the derivation of both data properties and object properties. The concepts identified, with the support of properties, make possible a semantic translation from one source into another, and the property values help to achieve the needed shared vocabulary.

To summarise, the domain-vocabulary acquisition activity is followed by the activity which enumerates the concepts and decides the properties and definitions, which together deliver a well-refined glossary of the domain, to be used by the remaining workflows, which finalises the internal structure of the concept.

4.5.3 Conceptual Design Workflow

This workflow associated with the conceptualisation stage is focused on the development of a conceptual model with the proposed domain ontology as its deliverable. *Taxonomy Identification, add complex restrictions and rules, describe concepts, attributes and relationships and establish ad-hoc binary relationships* are the main activities of this workflow. These activities can be performed iteratively within the workflow which is the underpinning philosophy of SCIM. Although the sequence is not important, the taxonomy identification has been chosen as the first activity to start off the conceptual design workflow.

4.5.3.1. Taxonomy Identification

After the glossary of terms is filled with a reasonable set of refined concepts, the KE starts to work on the taxonomy-identification activity. This activity defines the concept hierarchies of the proposed ontology by building a concept taxonomy. The concepts/classes are organised in a hierarchical structure which is mainly based on the taxonomic relationships and makes use of the four formal relationships proposed by SCIM. The four relationships used in this activity are *subclass-of, disjoint-decomposition, exhaustive-decomposition* and *partition* as stated in the development stage. The *subclass-of* relationship abides by the principle that an instance of a subclass will be an instance of the superclass. A top-down development process starts with the definition of the most general concepts of the domain followed by the specialised concepts. A bottom-up development process starts with the definition of the most specific concepts as the "leaves" of the hierarchy, with the subsequent grouping of the more general concepts (Uschold and Gruninger 1996). A hybrid development process is a combination of the top-down and bottom-up processes, such as the "middle out" process where a few core concepts are picked out and then the taxonomy is built on top of that. The disjoint-decomposition relationship follows the principle that a concept is a set of subclasses of a superclass that don't have a common instance. They may be the instances of the set concept, but they should not be the instances of the individual concepts in the decomposition. An exhaustive-decomposition relationship abides by the principle that a concept is a set of subclasses that may have common instances and subclasses. As part of the evaluation, the KE has to validate the concept hierarchy to ensure its correctness, prior to proceeding with the specification of new knowledge. The developed taxonomy can be extended to domain-specific relationships in the subsequent activities within the workflow.

4.5.3.2. Add complex restrictions and rules

This activity identifies and describes explicitly the *formal axioms* required in the proposed ontology. The intermediate representation of each formal axiom, as with METHONTOLOGY, includes the name of the axiom, the description (in natural language), a logical expression using first order logic, the concepts, properties and ad-hoc relationships to which the axiom refers, and the variables used. Figure 29 shows the template of a formal axiom representation proposed by SCIM.

Axiom Name	Description in NL	Logical expression	Referred Concepts	Referred relationships	Variables

Figure 29: Template of a formal axiom representation

During this activity, property constraints can be set which limit the set of possible values for a property. These include the value types, allowed values, and the mandatory number of values (cardinality). For example, the value of a name slot is a single string (i.e. the name is a slot with a value type string). The domain and range for a property can be specified by explicitly stating the possible or enumerated values. A slot can have multiple values and the values are instances of the class. This activity also identifies the rules required in the ontology. An instrument-rule table is used for the representation of ontology rules which contain: the rule name, description, expression, concepts, referred attributes, referred relationships and expression, similar to the formal axiom-representation template.

4.5.3.3. Establish ad-hoc Binary Relationships

This activity establishes the appropriate semantic relationships between the identified concept hierarchies for the domain. During this activity, the binary relationship seeks to establish an ad-hoc relationship, between the same or different concepts, from the conceptual model produced in the previous workflow. The principle of object-oriented inheritance is used when deriving the semantic structural relationships. The ad-hoc binary relationships include *inverse, transitive, symmetric and reflexive* relationships for the concept taxonomy being built. For every ad-hoc binary relationship, the KE must specify the name of the relationship, the source and target concepts, and the cardinality. Figure 30 shows the structure of an ad-hoc binary relationship table.

elationship Name	Source Concept	Cardinality	Target Concept	Inverse relation

Figure 30: Structure of an ad-hoc binary relationship table

4.5.4 Implementation workflow

This workflow has been mapped to the implementation and deployment stage of SCIM. The implementation of the ontology requires an ODE. Protégé is an open source, widely used ODE, developed by Stanford University. SCIM proposes that any stable version of Protégé can be used for the formal language representation but this does not preclude the developer from using an alternative ODE. During this workflow, the encoding of the ontology with the formal language takes place. The *Formal language representation* and the *vocabulary linking with data* are the two activities associated with the implementation workflow. The expressive power and computational complexity of the associated reasoning method and the level of acceptance are the core parameters to be considered for choosing the formal language. OWL (Antoniou and Harmelen 2004) is one of the best available formal languages to encode an ontology, even for the semantic web context. Protégé can auto-generate OWL codes against the ontology designed. Therefore, the output of the implementation workflow is an OWL implementation of the proposed ontology. The KE is the key role player in this workflow as the implementation delivers a physical model of the ontology as its output.

4.5.4.1. Formal Language Representation

During this workflow, the proposed ontology has been represented in one of the formal languages available. The ODE will accommodate this activity as most of the modern versions of these have the capability to generate the formal language for the conceptual model of the ontology. This activity focusses on the formal representation of both the data and object properties. SCIM always recommends Protégé as the primary choice of ODE. More details of this workflow are made available in section 5.2.4 as this is a core implementation-specific activity. OWL codes are recommended by SCIM for formal language representation as justified in the earlier part of this thesis.

4.5.4.2. Vocabulary Linking with Data

This activity creates individual instances of the classes in the hierarchy. Defining an individual instance of a class requires choosing a class, creating an individual instance of that class, and filling in the slot values. For each instance, an instance table which defines the name of the instance, the name of the instance it belongs to, and the values of the attributes can be developed. At this stage the vocabulary is linked absolutely with the real data. In

SCIM, the ontology development activities within their respective workflows can be iterative. Figure 31 shows the template of the instance table

Instance Name	Concept Name	Attribute	Values

Figure 31: Template of instance table

4.5.5 Evaluation workflow

According to the framework of SCIM, this workflow has been implicitly embedded within all the workflows for each activity. This has been elaborated in detail along with the description of each activity. Moreover, on each iteration of the activities, the deliverable produced is a piece of evaluated work by the people involved in that activity such as the DE, KE, User and member of the ODT as illustrated in Figure 21 of section 4.3.2.

4.6 Final Framework of SCIM

The final framework of SCIM accommodates the four stages of ontology development, in a linear manner, into the phases of the RUP, which is one of the most popular iterative methodologies of software engineering. The incremental and iterative development approach of the RUP with the scope for risk mitigation provides a disciplined approach for assigning tasks and responsibilities within an ODT. The RUP captures many of the best practices in modern software engineering in a form that is also suitable for ontology engineering, thus bridging the gap between two complementary branches of engineering.

The phases and stages of the proposed ODM are fitted into the four phases of the RUP, which are: *inception, elaboration, construction and transition.* The overriding goal of the inception phase which maps to the feasibility study is to achieve agreement by all the stakeholders on the objectives of the project. The purpose of the elaboration phase is to analyse the problem domain, establish a sound architectural foundation, develop the project plan, and help to eliminate the high risk of the project elements. During the construction phase, all components and application features are developed and integrated into the product, and all the features are thoroughly tested. These objectives are then mapped to the ontology definition phase. Finally, the ontology implementation is mapped to the transition phase. Figure 32 illustrates the final framework of the proposed SCIM.

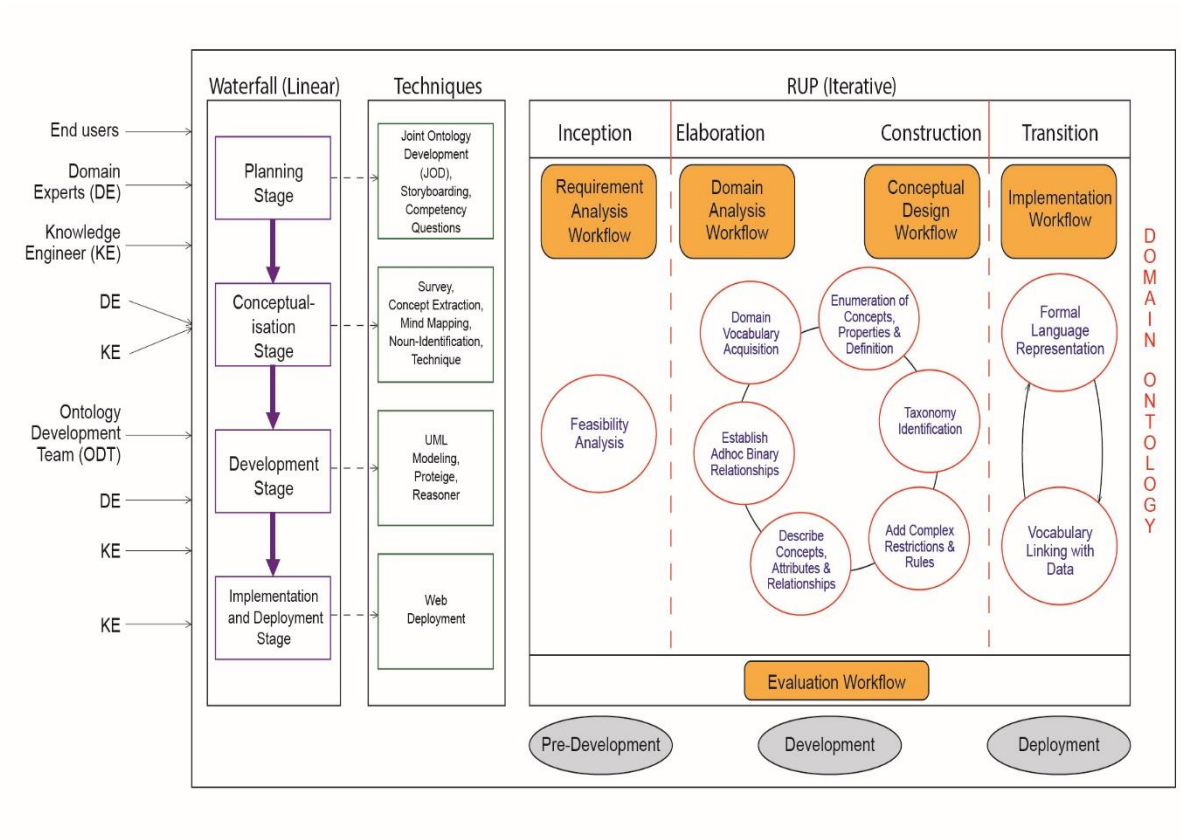


Figure 32: Final framework of the SCIM

4.7 Chapter Summary

A detailed description of the proposed software-centric methodology has been presented in this chapter. Detailed coverage of the context, underpinning philosophy, hierarchy of components and ontology development life cycle has also been presented. Each component of SCIM has been described with the support of the necessary illustrations. The chapter ends with an illustration of the final framework of SCIM which consolidates all aspects, inclusive of stages, activities, techniques and the placement of the members of the ODT. Thus, this chapter offers a complete conceptual coverage of the proposed methodology. The application of SCIM to a specific case study is covered in the next chapter where the conceptual aspects discussed in this chapter have been applied.

5. Application of the Proposed Methodology

This chapter of the thesis provides a detailed coverage of the application of the proposed methodology presented in Chapter 4. The application of the methodology has been carried out as a proof-of-concept for the realisation of the conceptual aspects of SCIM described in Chapter 4. The detail and rationale for the chosen domain are described first, followed by the four stages of SCIM. Detailed descriptions of the workflows, assigned activities, techniques applied, deliverables, and the software platform, supported by the appropriate screenshots, are given in the following sections.

5.1 Domain Overview

Recent trends in education place more emphasis on the needs of the learner with the appearance of learner-centred approaches for the delivery. E-learning, as well as introducing new technology, provides an opportunity to learn when it is convenient for the user. New approaches, such as semantic-web based educational systems are providing more adaptive, personalised and intelligent learning-environments by making use of the concept of ontologies (Yarandi, Jahankhani and Tawil 2013). The concept of taxonomy which can be defined as a means for grouping things in a hierarchical structure, was effectively implemented in ontologies, which are making use of the taxonomy of domain concepts and the semantic structural-relationships between them.

The growth of e-Learning educational systems has paved the way to learn at the convenience of the user by making the best use of web technologies. Semantic-web based educational systems using semantic-web technologies and ontologies, provide more personalised learning (Dicheva 2008). They support the sharing of knowledge in a standard format with common semantics as a knowledge base. Ontologies can be used as a skeletal foundation for such a knowledge base that will allow different applications/software agents to speak the same language.

The proposed SCIM has been applied to the development of an ontology for the teaching and learning of introductory Java-programming. The motivation behind building an ontology for Java programming was an attempt to organise the learning aspects of a widely adopted industry language. When teaching the introductory Java-programming modules, teachers introduce Java according to their own individual preferences. The use of an ontology ensures that the basic hierarchical semantic-structure of the language is not violated, even though the way different teachers present the material varies. The different stages of the

proposed methodology have been applied to the development of the domain-ontology prototype, which has been called the Java Learning Educational Ontology (JLEO). Three different variations of Java programming from the undergraduate curriculum of a premier higher-education institution in Oman were selected as a case for the design of the JLEO. Using these three Java based modules of Middle East College (MEC)'s computing curriculum as a guideline, the basic learning units of the JLEO have been defined. These modules are "Introduction to Programming", "Object Oriented Programming" and "Internet Programming". The intended contributions of the JLEO include:

- Establishing the semantic structural-relationships for the introductory Java learning units.
- Creating learning units for building a knowledge-based system on top of the conceptual model.
- Designing a prototype ontology that can be used as a base for the implementation of a knowledge-based system for basic Java learning.

5.2 Application of SCIM Stages and Workflows

SCIM proposed four stages together with their respective workflows as part of the Ontology Development Life Cycle (ODLC) as illustrated in figure 22 of chapter 4. The linear waterfall approach (borrowed from software development) has been adopted by SCIM which generally proceeds in a sequential manner from one stage to another. The exception to this is the back and forward approach which has been adopted by the second and third stages. The flexibility of SCIM makes the methodology for practitioners less complex and subject to fewer risks.
The following subsections best describe the application of the stages, workflows and activities of SCIM in the JLEO prototype development together with the evidence for the various techniques applied.

5.2.1 Planning Stage

According to the illustration of the framework of SCIM shown in Figure 32 of Section 4.6, the pre-development phase for the development of the ontology takes place at this stage. Therefore, the application of the proposed workflow and activity belonging to the planning stage of the JLEO development has explicitly emphasised the completion of the pre-development phase deliverables. Since SCIM follows the footprints of software methodologies, the feasibility of the JLEO is the major concern of the planning stage. The

ultimate formal deliverable of this stage is the detailed Ontology Requirement Specification Document (ORSD). The filing card activity from the NeOn methodology (Baonza 2010) has been used as a reference instrument to prepare the key-slots card for the JLEO ontology-requirements specification, prior to the preparation of the final ORSD. The filing card template used for the creation of the ORSD for the JLEO can be seen in Figure 33. The ORSD contains the details of the domain, including the purpose, scope, boundary, domain-specific assumptions and the techniques applied and can be viewed in Appendix B1.

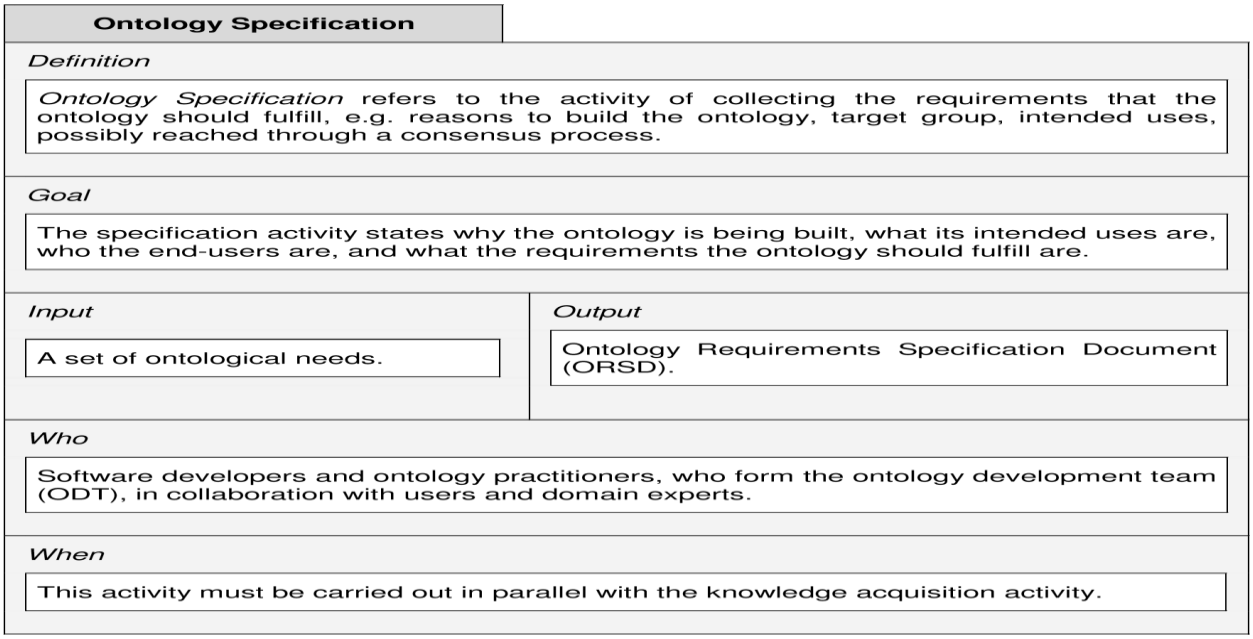


Figure 33: Filing card template (Baonza 2010)

The hierarchy of the components of SCIM are organised into stages, each stage containing a workflow with one or more associated activities. The activity assigned to the requirements-analysis workflow of the planning stage is the *feasibility analysis*. Before development begins, this activity investigates the possible integration of any existing ontologies, The specification of the proposed ontology is also finalised at this stage together with the respective formal deliverables as guided by SCIM. The techniques in this activity can be used iteratively (originating from the inception and elaboration phases of conventional RUP) as part of the underpinning philosophy of SCIM. (as shown in Figure 20b of Chapter 4). A number of techniques have been recommended by SCIM for practitioners to follow when working on the feasibility analysis. This includes agile techniques originating from software engineering. The details of the requirement analysis workflow are described in the next section.

5.2.1.1 Requirement Analysis Workflow

The workflow proposed for the planning stage of SCIM is the requirement analysis as described in section 4.5.1 of Chapter 4. This workflow focusses on the gathering of requirements for the proposed ontology and includes the activities for the completion of the ORSD which is the main deliverable from this stage..

5.2.1.1.1 Feasibility Analysis Activity

The feasibility analysis is the sole activity proposed by SCIM for the requirement analysis workflow. A set of techniques such as the Joint-Ontology Development (JOD) sessions, informal interviews, motivational scenarios and the use of Competency Questions (CQs) have been recommended by SCIM for the fulfilment of the activity. As part of the requirements analysis, initial attention has been given for the completion of the key-slots card for the ontology-requirements specification. After the necessary iterations of the above techniques have been completed, the JLEO key-slots card (see appendix B2) is produced. For the identification of the purpose, scope, boundaries and level of formality of the JLEO, a JOD session was conducted on 13th August, 2017 in one of the MEC meeting rooms.

Joint Ontology Development Session

The JOD session is the counterpart of the Joint Application Development (JAD) methodology employed for software development. Since SCIM adopts agile techniques for its activities, the JOD session has been used at the JLEO planning stage to determine the fields required for the ORSD. These fields include the purpose, scope, boundary and intended uses of the JLEO. The representatives of the end-user comminity, domain expert (DE) and knowledge engineer (KE) were the participants of the JOD. A detailed description of the JOD can be viewed in Appendix B3.

Informal Interviews

According to the action plan for the JOD session, informal interviews were conducted for the module leaders of the three introductory Java modules to identify the intended users of the JLEO. After obtaining the necessary ethical approval, before the interviews took place, the interview questions were sent to each one of the participants. The interview findings have been used for the identification of the learning units. The commentary from each of the three interviews can be viewed in Appendix B4.

Motivational Scenarios

The motivational scenarios exhibit the problems that occur when users need information which the system can't provide (Chaware and Rao 2010). They also describe the solutions to the semantic aspects of these problems. In the case of the JLEO development, three motivational scenarios have been created for the identification of the intended users. An appropriate template has been used for presenting the scenarios. The entries in the template are *name of the scenario, Actors, description and abstract level resources.* The templates used in the scenarios for the JLEO development are shown in Tables 7,8 and 9 below.

Name of the Scenario: Module Descriptor Development of Introductory Java courses
Actors (Active) : Faculties/teachers handling introductory Java courses (ITP, OOP, IP)
Description of the Scenario: The proposed scenario requires a faculty or a group of faculties with experience of the modules to participate in the Module Descriptor (MID) development. This task can be carried out with the support of both open and closed corpuses. Approved module descriptors for similar courses of other Higher Education Institutions (HEI) should be referenced. Individual faculties who deliver the modules can make their drafts based on the core concepts of the respective modules. The academic committee which defines the policies for the delivery of the modules should finalise subject to approval by the Module Reviewer.
Abstract level resources: Aims and objectives, indicative contents, assessment types, learning resources, reference texts.

Table 7: Motivational Scenario 1

Name of the Scenario: Find the concepts and details from the knowledge model
Actors (Active): Student/students who registered any of the introductory Java courses (ITP, OOP, IP)
Description of the Scenario: The proposed scenario requires a student or a group of students who were registered on any of the introductory java courses listed. The students need the scenario to find the module-specific concepts of the semantic model. This task should be carried out with the support of the ontology. Using the scenario, the students can find the core module-specific concepts and the relationships between them to improve their understanding of the respective module.
Abstract level resources: Java language concepts, object-oriented terms, Java networking concepts

Table 8: Motivational Scenario 2

Name of the Scenario: Review and updating of the concepts and relationships for the knowledge model.
Actors (Active): Reviewer/Module Leader of any of the introductory Java courses (ITP, OOP, IP)
Description of the Scenario: The proposed scenario requires a leader/reviewer or support teacher who has expertise in the module and contributed to the finalisation of the conceptual structure of the ontology. The module leader/reviewer of any one of the modules can find the core module-specific concepts and the relationships among them and can make amendments when required.
Abstract level resources: Concept review reports, moderation reports

Table 9: Motivational Scenario 3

According to the specified outcomes of the JOD session, proven techniques such as the usage of competency questions (Brusa, Caliusco and Chiotti 2006) and survey have been applied for the identification of the context, purpose, boundaries and scope of the JLEO, which are described in the following sections.

5.2.1.1.2 Background/Context of the JLEO

The challenges for the tutors to guarantee a high-quality knowledge transfer is demanding in the modern academic environment. One of the practical solutions to this challenge is the creation of knowledge-based systems where knowledge is created in a shareable and reusable form. Since knowledge base systems start from where ontologies end, ontologies can play a vital role in their development. The visual representation of domain-specific concepts and the semantic structural-relationships between them, gives the students both syntactic and semantic knowledge of the domain.

The goals for building the Java Learning Educational Ontology (JLEO) were:

- To use as a learning ontology for providing a structural-knowledge representation model for teaching Java based programming modules.
- To build a conceptual structure of Java learning units that can act as a mind-mapping tool for the effective teaching of the modules;
- To build a visual navigation interface to the learning units;

- To assist the students who are learning the modules to find the concepts and their conceptual hierarchy; and;
- To be used as a reference ontology to develop a Java learning system for both operational and domain knowledge.

5.2.1.1.3 Intended Benefits

The JLEO can be used by higher education institutions for organising the learning units of Java-based modules into a conceptual structure. It can also be used in the future as a foundation for developing semantic-web based educational systems. The ontolological structure will assure the taxonomical structure for the concepts of Java programming, regardless of the preferred method of delivery by the tutor. Furthermore, the development of a knowledge-based/semantic-web based educational system on the top of the JLEO could guarantee the consistent sharing of knowledge across the Java faculties.

5.2.1.1.4 Scope of the JLEO

The prototype version of the JLEO focusses on three variations of modules for teaching the principles of the Java 2 Standard Edition (J2SE) release of the Java language. It has been applied in a higher-education institution where different perspectives were held with regard to teaching and learning. The feasibility of the ontology was tested in seven modules from the MEC curriculum which directly deal with Java programming, with 20 different faculty members delivering modules which specialised in software technology, computer science and information systems. A fully developed ontology could be used as a foundation for a knowledge-based system development, which could integrate with a Learning Management System (LMS), e.g. Moodle, for guidance within different learning paths, as a precursor to adaptive learning.

The vocabulary for the JLEO was taken from various resources such as: open/closed corpora, surveys/interviews conducted with the Java teaching staff and the course materials for the modules.

5.2.1.1.5 Boundary of the JLEO

The Boundary of the JLEO prototype was limited to three streams of Java programming. The first of these dealt with an introduction to programming with the Java language. The concepts in this stream were taken from a spectrum ranging from the basic data types supported by the

language, through to program execution. The second stream discussed object-oriented programming with Java. The topics included the core concepts of object orientation and their Java implementation together with Java I/O streams. The subject of the third stream was networking, and the Java topics included were: the GUI, applets, URL programming, the DNS, socket programming, remote method invocation and object serialization with multi-threading. The abstract-level CQs used for the finalisation of the JLEO scope and boundary were:

1. To what extent, are variations of Java Programming exposed in the curriculum of the undergraduate programmes of MEC?
2. What are the important variations of Java Programming accommodated in the Introductory Java courses of MEC?
3. How many of MEC's programming modules are Java based and how are they categorised?
4. What are the java programming streams to be considered for the development of the knowledge based system for the teaching and learning of the introductory Java courses?
5. Considering the wide scope of Java as an industry language what concepts should be excluded in the first version of the knowledge based system for teaching and learning?
6. Does a knowledge model of Java programming make teaching and learning more flexible for teachers and students?

5.2.1.1.6 JLEO Requirements

The requirements that the ontology should satisfy are defined in the ORSD as the deliverable of the planning stage. An amalgamation of both functional and non-functional requirements are considered for the JLEO both of which are described in the following paragraphs.

Functional requirements: These are the specific set of requirements that are mandatory for the users of the JLEO, particularly, the knowledge to be represented in the ontology which should include language fundamentals, object-oriented concepts and network programming.

Non-functional requirements: This category refers to the requirements which are non-mandatory and include: the standard of resources used for concept extraction, naming conventions, and standards, etc. The abstract-level development scope of the JLEO is shown as a use-case diagram in Figure 34 below.

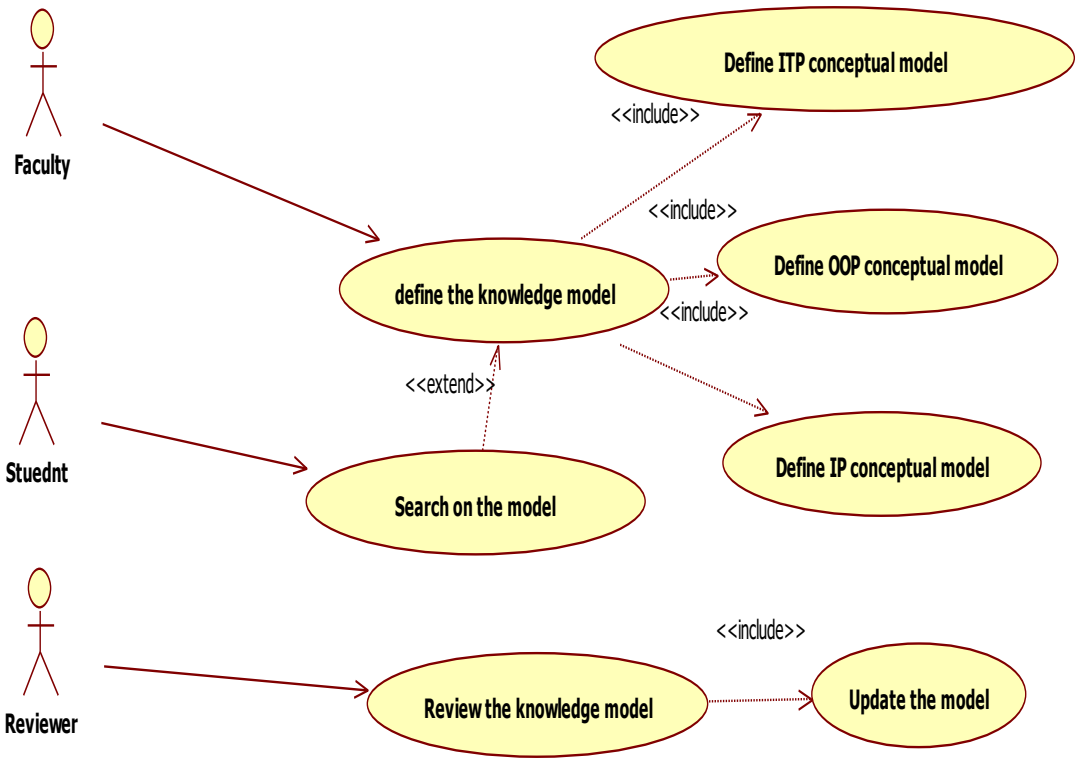


Figure 34: Use case diagram of the JLEO development scope

5.2.2 Conceptualisation Stage

The *development* stage for the JLEO is preceded by the conceptualisation stage. Therefore, in the final framework of SCIM (see Figure 32 of Chapter 4), the workflows and the activities of both the conceptualisation and development stages are represented in the design and development part of the ontology. The conceptualisation stage follows the planning stage as the stages of SCIM follow the linear philosophy of the underpinning waterfall methodology (Powell-Morse n.d.) of software engineering. Unlike other existing methodologies, a strong inter-connectivity exists between the conceptualisation and development stages. As these two stages focus on the design and development aspects of SCIM, many iterations of these two stages can be performed, giving an added advantage to the development process.

The abstract level concepts of the domain identified as part of the planning stage for ontology development are further taken up for refinement at this stage. During the conceptualisation stage, the KE will structure the conceptual model according to the judgement of the DE and and other techniques recommended by the SCIM. Other than those provided by the leading existing methodologies such as METHONTOLOGY and UPON, there are few other choices of techniques available for conceptual model creation of

ontologies. METHONTOLOGY employs the conventional technique of Glossary of Terms (GT) preparation (Fernández-López, Gómez-Pérez and Juristo 1997). UPON uses software-centric techniques such as case-diagrams, application design, domain and reference lexicon creation, etc. However, alhough these techniques are software-centric, a significant learning curve can be experienced when software practitioners to apply them to ontology engineering.

According to the life cycle of SCIM (see Section 4.5 of Chapter 4), a number of formal deliverables are offered by this stage which range from a minimal-model domain vocabulary-of-concepts to a Glossary of Terms (GT), inclusive of the concepts with their associated definitions and properties. None of the existing ontology-development methodologies offers more than a single deliverable at the conceptualisation stage. The domain vocabulary-of-concepts is a pathway to a formal GT development in SCIM. This deliverable can be considered as a mechanism to ensure that all the major domain specific concepts in the GT are created. The complete GT of the *IntroductionToProgramming module* in the prescribed format is provided in Appendix B5. Iterations have been applied to the activities in both the conceptualisation and development stages to design the conceptual model of the JLEO, which was developed using a modelling language and UML modelling tool.

As defined in the the hierarchy of SCIM components, the workflow mapped to the conceptualisation stage is the *Domain Analysis Workflow* which identifies the concept/knowledge for the introductory Java learning domain, comprising the main part of the JLEO development. The activities assigned to this workflow are the *domain vocabulary acquisition,* the *enumeration of concepts, and the definition of properties*. After completing the activities proposed by SCIM, the main concepts of the JLEO for the three variations of the introductory java modules will have been identified. The activities in SCIM can be executed iteratively, an approach based on the philosophy of the RUP methodology used in software engineering. Because the design of SCIM has been based on software-engineering methodologies, the domain-analysis workflow has been mapped against the *analysis and design* discipline of the RUP. The techniques applied for both the conceptualisation and development stages take place in the elaboration and construction phases of conventional RUP. As shown in Figure 22b of chapter 4, activities can be performed iteratively in both of these stages. Besides using the judgement of the DE, many other techniques have been recommended by SCIM for practitioners to follow while working on the activities associated with the respective workflow. These include: the usage of competency questions, formal text analysis and brainstorming. The techniques recommended include agile techniques, which

have been succesfully used in software engineering. The details of the workflow and activities are described in the following subsection.

5.2.2.1 Domain Analysis Workflow

Although knowledge acquisition is an independent process (Fernández-López, Gómez-Pérez and Juristo 1997), it is one of the crucial steps in domain-ontology development and the primary objective of the domain-analysis workflow. In fact, the practical part of domain-ontology development starts from this workflow, where the key steps are the processes for collecting domain-specific information from the identified sources and the analysis by domain experts. Although SCIM follows a similar approach to METHONTOLOGY and UPON, by applying validation techniques, it provides more disciplined and proven techniques than its predecessors.

As a common practice, during the process of knowledge acquisition, the DE extracts domain-specific concepts by following the mechanisms recommended by the methodology. Most of the available methodologies, as observed by the literature review, failed to offer specific activities for knowledge acquisition, together with an explicit list of techniques, in their ODLC. However, SCIM does provide as part of its life cycle the domain analysis workflow which includes a set of activities and techniques for fulfilling knowledge acquisition. The construction of the domain ontology with a number of domain concepts and the structural relationships between them fulfils the minimum requirements for the development of a dynamic semantic knowledge-based system, made possible by the ontology-centric approach.

The Domain Analysis workflow further refines the scope of the ontology defined in the ORSD by the usage of CQs, brainstorming, knowledge-acquisition tools, the judgement of the DE, the analysis of the published handouts of the domain, domain-related web pages, and published figures and tables. Other published ontologies are also recommended either as a technique or as a source for the concept extraction by SCIM. For the completion of this workflow, a domain vocabulary has been developed for the three introductory Java modules of the MEC curriculum. Details of the activities recommended for this workflow and the techniques applied for the completion of activities for JLEO development are described in the following subsections.

5.2.2.1.1 Domain Vocabulary Acquisition

The domain vocabulary acquisition is the first activity belonging to the domain-analysis workflow which focusses on the identification of classes after the refinement of the domain concepts collected. Although SCIM provides different techniques for knowledge acquisition, the KE is responsible for choosing those that are the most appropriate for the domain. However, for the JLEO development, these have been selected and applied by the KE and DE. The initial deliverable from this activity is the domain vocabulary which is created from the domain concepts collected.

The techniques applied for the completion of this activity are the *usage of Competency Questions*, *Domain expert(s) judgement* and *Formal Text Analysis*. Considering the academic nature of the domain, emphasis has been given to the concept identification from the module guide, online course articles and text books. This has provided a more concise and higher density of domain knowledge, which was accepted by the target community.

As well as these sources, traditionally educational institutions also applied their own teaching and learning materials to strengthen the teaching and learning process, so, in the case of JLEO development, the approach followed by MEC such as the usage of module descriptor, and the module guide/lecture slides for teaching the introductory Java modules have been adopted for concept identification. The module descriptor of the three modules explicitly states the critical terminologies related to the respective modules and the topics that have been incorporated in the module descriptors offer the scope for more detailed knowledge acquisition of the concepts belonging to the core topics. A detailed description of each of the techniques applied for the completion of this activity follow.

Usage of Competency Questions (CQs)

The use of CQs, which has been recommended by most of the existing ontology-development methodologies, is one of the most common techniques for identifying domain concepts in ontology development. Unlike other methodologies, the use of two levels of CQs is proposed by SCIM. The first level is more generic as it was intended for identifying the scope and boundary of the domain, which has been carried out as part of the feasibility-analysis activity. The second level of CQs is more specific in nature and used for identifying the domain concepts. The CQs in general assist the KE and DE to define the ontology scope and verify that it contains all the answers in concept form that are required to build the domain ontology. CQs make the ontology more expressive as the answers to the CQs are mainly terms/concepts and it is rare to use a domain concept in an ontology if it doesn't answer any

of the CQs. Moreover, the CQs, by the use of composition and decomposition processes can define a hierarchy so that an answer to a question may also reply to others with a more general scope.

The scope of the ontology can be determined by drafting a list of generic CQs that address the domain of the knowledge-based system. Ideally, the ontology, for a consistent communication, will be capable of answering the CQs without ambiguity (Gruninger and Fox 1995). CQs will provide a single factor (such as a fact or event) to make a decisive test of the narrative idea. They are as follows:

1. ***Does the ontology contain sufficient information to answer these categories of questions?***
2. ***Do the answers need a certain level of detail for the representation of a particular topic?***

In the case of the JLEO design and development, the usage of domain-specific CQs have been used as the primary formal-technique for concept extraction. The collection of domain specific concepts with the support of a formal technique reduces the scope of personal judgement which eventually leads to inconsistency in communication. Moreover, the findings from CQ analysis helps the KE to state the domain assumptions explicitly, when designing the ontology. However, considering the academic nature of the chosen domain, as well as the technical information, the tacit knowledge of instructors/tutors has also been considered. The answers to the CQs circulated to the respective faculties were used to identify the core concepts from the module content of the three introductory Java modules. Most of the answers received back from the faculties have provided the domain concepts for the proposed JLEO. Examples of CQs used for JLEO domain concept acquisition are:

1. *What are the categories of primitive data type?*
2. *What are the variations of non-numeric data type in Java Language?*
3. *What are the numeric data types in Java Language?*
4. *List the different floating-point data types of the Java Language?*
5. *What are the categories of variables in Java Language?*
6. *What are the three types of classes available in Java?*
7. *What are the three types of methods available in Java?*
8. *How does the static variable differ from the instance variables?*
9. *What is an instance of a class?*
10. *What is the first mode of client-server communication implementation in Java?*

11. What are the types of sockets supported by Java?

12. What are the core classes used for stream-socket implementation?

13. What are the core classes used for datagram-socket implementation?

14. What is the mechanism for broadcasting in Java?

In contributing towards the list of CQs, the DE had focussed on three different uses of the Java language which were general programming, object-oriented programming and network programming. The module leaders/instructors of the respective modules at MEC were selected as the target group of participants to answer the CQs because they were responsible for the module descriptor drafting, curriculum development, and incorporation of the core topics, for the respective modules. The CQs had been sent to them by email and the answers received provided the concepts for developing the JLEO prototype. In total, 40-50 questions per chosen module were circulated to the three module leaders. The answers, being small in number and very specific were analysed by the DE and validated by the module leaders which provided the key concepts and essential mappings required for the knowledge base.

Thus, the use of CQs has provided a systematic way of collecting the concepts used for the development of the domain ontology which has impacted the expressiveness of the concepts in the JLEO. The module leaders have provided the concepts for the JLEO from the introductory Java modules which were produced from the answers to CQs designed from the perspective of teaching and learning.

Domain Expert's Judgement

Taking the judgement and expertise of the DE in relation to the ontology-development process into account, was an additional means for extracting the concepts for the ontology development team. By applying knowledge elicitation techniques to identify the relevant knowledge sources and discover information directly from the DE can be considered as an effective technique for the refinement of the concepts. It has been observed from the literature that the involvement of the DE has been utilised in many of the previous ontology development initiatives.

The DynamOnt methodology (Gahleitner et al. 2005) suggested acquiring guidance from the DE who acted as the knowledge expert (KE) for producing the resultant ontologies with the qualities of foundational ontologies such as DOLCE (Gangem et al. 2002) and by making use of existing ontology-development methodologies (Schreiber and Akkermans 1999).

HCOME, one of the older ontology-development methodologies (Kotis and Vouros 2006) relied on the KE for the development, maintenance and evaluation of ontologies with minimal involvement of the DE. It proposed supporting the DE by encouraging them to collaborate with a community of KEs for the construction of ontologies.

Although the involvement of the DE varies in both of these methodologies, they both strongly recommended the combined involvement of both the DE and KE and assuring the active involvement in ontology authoring by giving the DE a key role in ontology evaluation.

Another holistic approach (Denaux et al. 2011) for ontology development also strongly recommends the involvement of the DE to provide a higher level of expressiveness in the development of ontologies.

Most of the advanced ontology-development environments support collaborated ontology-development to build ontologies, e.g. HCONE (Kotis and Vouros 2006) and Web Protégé (Tudorache et al. 2008). These tools offer web-based techniques, such as forums, supporting users to propose, document and implement changes to ontologies. The benefit of these is to inspire the establishment of a group, by encouraging the DEs and KEs to collaborate together in ontology development. The latest web tools provide interfaces for the KE to design the ontology in an effective manner as well as to generate RDF/OWL codes (Schober, Malone and Stevens 2009).

The SCIM methodology also strongly recommends the involvement of both the DE and KE in ontology development. Therefore, seeking the judgement of the DE on the answers to CQs is an instrument for concept finalisation of the JLEO development providing a means of refining the concepts that have been used. The dialogue for the interview with the DE is shown in Appendix B6. The conclusion of the interview with the DE justifies the concepts that have been extracted from the various sources. The extracted key concepts which form the essential mappings for each of the Java modules are a result of the analysis by the DE. A summary of the interview with the DE follows

Summary of the Interview with Domain Expert

As discussed previously the idea of developing a prototype of the JLEO to apply the proposed SCIM is a viable one. As discussed, the scope identified for the JLEO prototype was appropriate as the proposed prototype was intended to be a proof-of-concept. The three introductory Java learning modules chosen as the base for the JLEO development were appropriate because they covered three different variations of introductory Java programming. The DE had agreed the merit and significance of the concepts, but recommended that a limit was placed on the scope of the main concepts for the prototype development. However, the judgement of the DE ensured that the concepts were properly defined.

The concepts for the introductory java programming module were divided into categories which were the data models, tokens, control structure and development/implementation. The representation, specific functionality, and general semantic structure for the fundamentals of the Java language were included in these concepts.

The DE made the following general observations:

- The looping concept should be split into more precise types for semantic clarity.
- Since data models were crucial in any programming language for knowledge representation, the concepts for this category were complete.
- The testing subcategory under the development/implementation category should be divided into testing strategy and testing technique.
- The concepts within the tokens category were appropriate
- Additional detail be added to the Domain Naming Service (DNS) under the InetAddress concept.
- GUI concepts should be added to Networking because the module descriptor was considered to be one of the core instruments used for concept identification.
- Socket programming and RMI should be split as they were two different levels of communication
- The object serialization concept must be included.

For the Object-Oriented Programming (OOP) category the DE made the following specific observations:

- The concepts identified under the OOP category mainly highlighted the features of the object-oriented model to build a simple console application. Considering the nature of Java programming language, some overlapping had occurred between the concepts of Java language and OOP. Due to the nature of the chosen programming language most of the essential OOP concepts had been taken into consideration.
- Although the method is a minimal thing in the Java language, it should be included as a kind of concept in OOP that is different to other special kinds of methods.
- Special features such as packages, interfaces, exception handling and streams emphasised the object-oriented programming concepts.
- Though the core aspects of packages were considered, extra packages and advanced topics such as SWING and JDBC were not recommended for the first version of the JLEO.

To summarise, the DE provided many constructive comments and approved most of the concepts to help the developers produce a well-structured JLEO.

Formal Text Analysis

The knowledge acquisition from the formal text analysis includes the concept extraction from the module descriptors, web-based Java learning sites, published text books, MEC module specific handouts and lecture slides. The text book preferred for the particular module has been encapsulated by the module descriptor. A good module descriptor is built on the top of a solid foundation of course design (Riviere, Picard and Coble n.d.) This encompasses the list of learning topics. The descriptors offer an introduction to information for both the student and faculties and are also a valuable source of reflective teaching practice for colleagues, including the review or search committees.

For the development of the JLEO, many online websites have been referred to in addition to the Java API to ensure that no major concept was dropped from the respective variations of Java programming. Examples of the sites referred to were:

- https://docs.oracle.com/javase/7/docs/api/
- https://www.javatpoint.com/java-tutorial;
- https://www.tutorialspoint.com/java/java_variable_types.htm;
- https://www.w3schools.in/java-tutorial/;

Further to the usage of module descriptor and online sites, Java texts were used to complete the preparation of the final deliverables of the domain vocabulary and GT activities. The two titles referred to for this information were:

- Java 2 : The Complete Reference by Herbert Schildt 8th Edition;
- Programming with Java: A Primer – Balagurusamy

5.2.2.1.2 Enumeration of Concepts, Properties and their Definition

This activity is complementary to its predecessor activity. During this activity, the listing of the properties belonging to the refined concepts of the Glossary of Terms (GT) takes place. Properties are to define the internal structure of the classes. According to the recommendations of SCIM, the first-level information of the properties used the atomic properties, and the detailed structured-information used complex properties for the conceptual model of the ontology. Both atomic and complex properties have been assigned to most of the classes incorporated in the GT. Subject to the nature of the property, atomic properties can be intrinsic or extrinsic. Intrinsic properties exist without depending on other things whereas the extrinsic property depends on its relationship with other things.

As recommended by SCIM, this activity can be performed iteratively. Both the DE and the KE take part in the iterations as SCIM strongly recommends their involvement in line with the DE's judgement discussed in section 5.2.1. Since SCIM is software-centric, the UML class model is recommended as one of the candidates for the creation of the internal structure of the concepts, hence this activity can be accommodated in the model. A template has been proposed for the representation of the internal structure of the concepts. The internal structure of all the concepts listed in the GT after refinement can be represented by making use of the template. An instance of the concept's internal-structure representation and its attributes are presented in Figure 35 following.

Concept Representation Structure			
Concept Name	**Description**	**Concept type**	
Lecturer	A person who gives lectures at collegea/higher education establishments who helps knowledge seekers to acquire knowledge by delivering subject-speci sessions and activities	Sub Concept	
Attribute Name	**Attribute description**	**Attribute type**	**Property type**
Lecturer ID	Identity number of the Lecturer	Atomic (Intrinsic)	Data Property
Lecturer FirstName	First Name of the Lecturer	Atomic (Intrinsic)	Data Property
Lecturer Last Name	Last Name of the Lecturer	Atomic (Intrinsic)	Data Property
Taught Module	Name of the module taught by the lecturer	Atomic (Extrinsic)	Data property
teaches	Core activity of lecturer	Complex	Object Property
studies	An activity that occurs with a student	Complex	Object Property

Figure 35: Concept representation structure

There are mainly two types of properties, which are data properties and object properties. Object properties are specifically used to create links between the individuals, but data properties are used to describe the relationship between an individual object and its data values. The types of properties are further explained in the activities of the development stage which follows.

5.2.3 Development Stage

This is the third stage of the proposed methodology, the core objective being the development of the concepts, attributes and relationships for the conceptual model of the domain ontology. Therefore, it is implicit that this stage occurs in the development part of the final framework of SCIM as shown in Figure 32 of chapter 4. The completion of the conceptual model takes place at this stage from the information provided by both the peer and conceptualisation stages.

Although the stages of SCIM are generally sequential, the conceptualisation and development stages can be performed iteratively as shown in Figure 32 of Chapter 4. This is because ontology development is an iterative activity (Ahmed and Gerhard 2007) and is heavily dependent on the domain analysis, which is the main objective of the conceptualisation stage. SCIM is the first methodology to propose an iteration between these stages. The number of iterations applied ensures the accuracy of the ontology by making corrections to the workflows or activities of the stages. Therefore, the conceptual model

produced as the deliverable from the development stage should be an accurate model for formal-language representation.

During this stage, the DE, KE and members of the Ontology Development Team (ODT) are involved in the creation of the conceptual model. The involvement of the KE is vital for applying the software-centric techniques of the model which include *UML modelling*, *construction of static models*, and the *usage of ODE*s, The UPON methodology used a combination of static and dynamic UML models but SCIM has proposed a static UML model (class diagram) (for representing the conceptual model), UML static realtionships, and semantic relationships. The refined concepts with attributes, derived from the conceptualisation stage are connected by completing the activities using the appropriate techniques. Unlike METHONTOLOGY, software-centric techniques are proposed by SCIM for the conceptual model development, which means that software practitioners can be used as KEs for the ontology development. In the case of METHONTOLOGY, an additional deliverable called the *integration document* was proposed to define the meta-ontology for including a concept that was not defined in the *ontology server*. As SCIM provides an iterative approach, there is no need for this extra creation of an integration document as the new concept can be incorporated in one of the iterations.

According to the Ontology Development Life Cycle defined by SCIM (Chapter 4, section 4.4), the one formal deliverable from the development stage was a static conceptual model of the domain ontology in the form of a UML class diagram. The structural model of the concepts was defined in the predecessor and peer stages used for the construction of the class diagram. Iterations were applied for the activities belonging to both the conceptualisation and development stages to design the conceptual model which was created by the use of STARUML, a proven ontology editing environment. The conceptual design workflow was mapped to the development stage of SCIM as illustated in the hierarchy of components in Figure 22 of Chapter 4 . Four activities were proposed for the conceptual design workflow, which were: *Taxonomy identification, Complex restrictions and rules, Describe concepts attributes/relationships* and *Establish ad-hoc binary relationships*. Conceptual models, in the form of static class diagrams, were created at this stage, for the three variations of introductory Java modules from the Middle East College (MEC) curriculum, This was achieved by completing the acivities within the conceptual-design workflow, which originated from the analysis & design discipline of conventional RUP. The techniques applied for both the conceptualisation and development stages can be found in the elaboration and construction

phases of RUP. In the proposed SCIM however, the activities within the workflow in both of these stages can be performed iteratively, to ensure the accuracy of the developed ontology. SCIM recommends techniques for the practitioners to follow when working on the activities associated with the conceptual-design workflow. A detailed description of the workflow and activities is described below, together with the evidence of their applicability to the JLEO development.

5.2.3.1 Conceptual Design Workflow

The conceptual model development is based on requirements gathering which is a vital process in any form of software development. In fact, the logical model development prior to the realisation of the physical model was encouraged by most of the traditional software-engineering methodologies (Rajadorai, Hassan and Admodisastro 2012). To a considerable extent, the provision of logical model development was noted as one of the reasons behind the success of software engineering as it ensured the accommodation of the requirements in the physical system. Agile software engineering methodologies such as Rapid Application Development (RAD) encourage the use of conceptual model development as an initial prototype and for that to then evolve into the real system. Unlike METHONTOLOGY, SCIM proposes combining the conceptualisation and development stages for the conceptual model development, suggesting that they be performed iteratively. According to the lifecycle of SCIM, the development stage contains the conceptual design workflow. In existing ontology-development methodologies, the conceptual-design workflow focusses on the development of a conceptual model of the domain ontology. However, in the case of SCIM, the conceptual model is created, based on the refined concepts identified and incorporated in the Glossary of Terms (GT) produced by the predecessor stage. Unlike many other methodologies, software-centric techniques have been used for the conceptual model creation. Moreover, the activities of the conceptual-design workflow in SCIM encouraged a disciplined approach with more proven techniques than its predecessors.

This workflow primarily offers the provision for the KE to incorporate the refined concepts from the conceptualisation stage in the conceptual model. In addition to the KE, the members of the Ontology Development Team (ODT) and the DE also play pivotal roles in the activities associated with the conceptual design workflow. In line with the underpinning software-engineering approaches, SCIM uses software-modelling techniques for the development of the conceptual model. Although conceptual modelling is a part of many

existing ontology-development methodologies, none of them have proposed software modelling techniques for the conceptual-model development except UPON which recommends UML modelling for ontology design at an abstract level. Unlike UPON, SCIM has recommended UML-based modelling activities for conceptual model development, incorporated within the ontology development life cycle. Unlike other methodologies, a list of formal techniques have been proposed by SCIM to complete the recommended activities of the conceptual design workflow. Within this workflow, the activities focus on the hierarchical arrangement of the domain concepts based on: the appropriate relationships, explicit descriptions of the formal axioms, formal description attributes and the establishment of semantic structural relationships between the concepts. As the semantic structural relationships between the concepts fulfill the minimum requirements for knowledge-based system development, they are vital in the construction of domain-ontology development. The description of the attributes is an extension of the *enumeration of concepts, properties and definition* activities of the predecessor stage. To summarise, the conceptual-design workflow ensures that the refined concepts and the relationships between them for the conceptual model of the proposed domain ontology are within the scope, defined in the ORSD, as the deliverable of the planning stage.

The *Usage of UML class modelling, usage of defined templates* and *ODEs* are recommended as the techniques for the conceptual-model development. This workflow is the counterpart of the implementation discipline for traditional RUP. In the case of JLEO development, upon the completion of this workflow, a conceptual model has been developed in the form of a class model for the domain of three introductory Java modules from the MEC curriculum.

Details of the activities recommended for this workflow and the techniques applied for the completion of activities for the JLEO development are described in the subsequent sections. All of these activities are performed iteratively which allows the conceptual model to evolve through each iteration.

5.2.3.1.1 Taxonomy Identification

As mentioned in the previous section, the first activity belonging to the conceptual design workflow was the *taxonomy identification.* During this activity, the DE starts to build the concept taxonomy which defines the hierarchy of the concepts defined in the GT. To build the concept taxonomy, the taxonomical relationships have been used by this activity. In the case of the JLEO, the domain concepts defined in the GT have been organised into a

hierarchical structure based on four formal taxonomic relationships. These relationships have been taken from two previously defined ontologies, such as the Frame Ontology (Tudorache et al. 2010) and the OKBC Ontology (Kotis and Vouros 2006). METHONTOLOGY also used these relationships for concept-hierarchy development, however, the techniques used within the SCIM, such as static modelling with UML, and the usage of an ODE for conceptual model development vary significantly from METHONTOLOGY. The four relationships used for taxonomy identification are *subclass-of, disjoint-decomposition, exhaustive-decomposition* and *Partition* (Vilches-Blázquez et al. 2007) .

i) Subclass-of

If a *conceptA* is a *subclass* of another subclass *conceptB*, and each instance of *conceptA* is also an instance of *conceptB*. In the case of the JLEO, the *Subclass-of* relationship has been applied to concepts at various places in the conceptual model (Vilches-Blázquez et al. 2007). For example, as Figure 37 shows, every instance of the concept *object* is an instance of the term *OOPconcept*. Moreover, a concept can be a *subclass of* multiple concepts in the concept hierarchy. The most commonly used formal taxonomic relationship in JLEO conceptual model is *subclass-of.* This is due to the fact that the *subclass-of* relationship best illustrates the concept hierarchy, and the representing semantic is a relationship between the concepts. For example, the relationship in Figure 36 can be semantically rendered as the object is an *OOPConcept*.

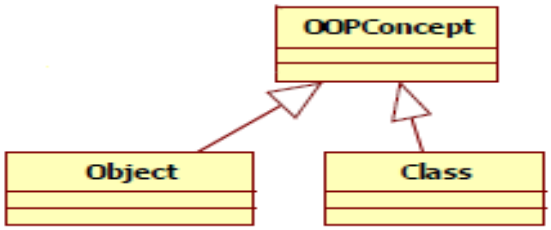


Figure 36: Subclass-of relationship

ii) Disjoint-Decomposition

Disjoint-decomposition is a formal taxonomic relationship where *conceptA* is a super class of a set of *disjoint* subclasses. There cannot be an instance common to the given subclasses, but there can be an instance of *conceptA* when neither instance represents the chosen subclasses (Vilches-Blázquez et al. 2007). This means there can be instances of conceptA that are not instances of any of the concepts in the decomposition but are an instance of another subclass. Figure 37 shows that the concepts *Student* and *Lecturer* are forming a *Disjoint-Decomposition*

of the concept *Person*. This is due to the fact that no *person* can be simultaneously a *lecturer*, and a *student*. In addition to that, there could be instances of the concept *Person* that are not instances of both *disjoint* classes.

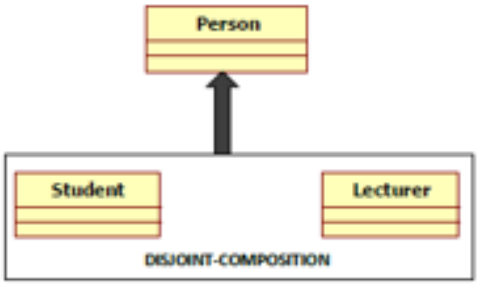


Figure 37: Disjoint-Decomposition relationship

iii) Exhaustive-Decomposition

Exhaustive-Decomposition is a formal taxonomic relationship where *conceptA* is a set of subclasses that cover *conceptA* and may have common instances of subclasses. This means there will not be any instances of the *conceptA* that are not the instances of at least one of the concepts in the decomposition (Vilches-Blázquez et al. 2007). Figure 38 shows that the concepts *Single level*, *Hierarchical*, *Multiple* and *Multilevel* makes an *exhaustive-decomposition* of the concept *Inheritance* because there are no instances for the concept *Inheritance* that are not an instance of at least one of the concepts from the decomposition.

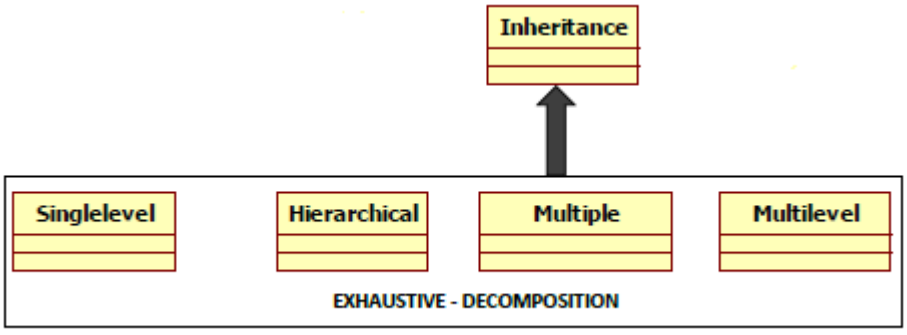


Figure 38: Exhaustive-Decomposition

iv) Partition

Partition is a formal taxonomic relationship where *conceptA* is a set of subclasses that cover *conceptA*, but do not have any common instances. This means there are no instances of *conceptA* which are not instances of any one of the concepts in the *partition* (Vilches-Blázquez et al. 2007). Figure 39 shows the concepts *primitive* and *non-primitive* which make up the partition of the Concept *Datatype.* This means that the concept *Datatype* can be either a *Primitive* or a *Non-Primitive.*

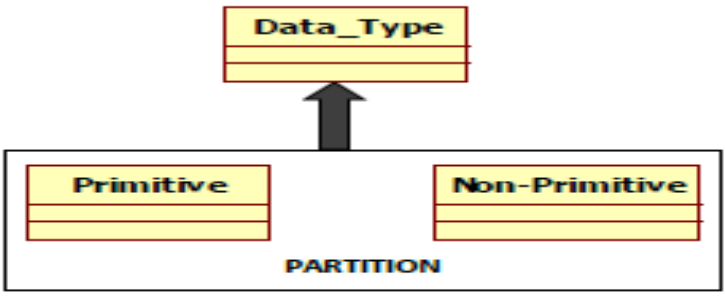


Figure 39: Partition relationship

An excerpt of the core concepts and formal taxonomical relationships of the conceptual model is illustrated in Figure 40. UML static modelling has been applied which reduces the learning curve for software practitioners. STARUML was used as one of the modelling tools for the creation of the conceptual model.

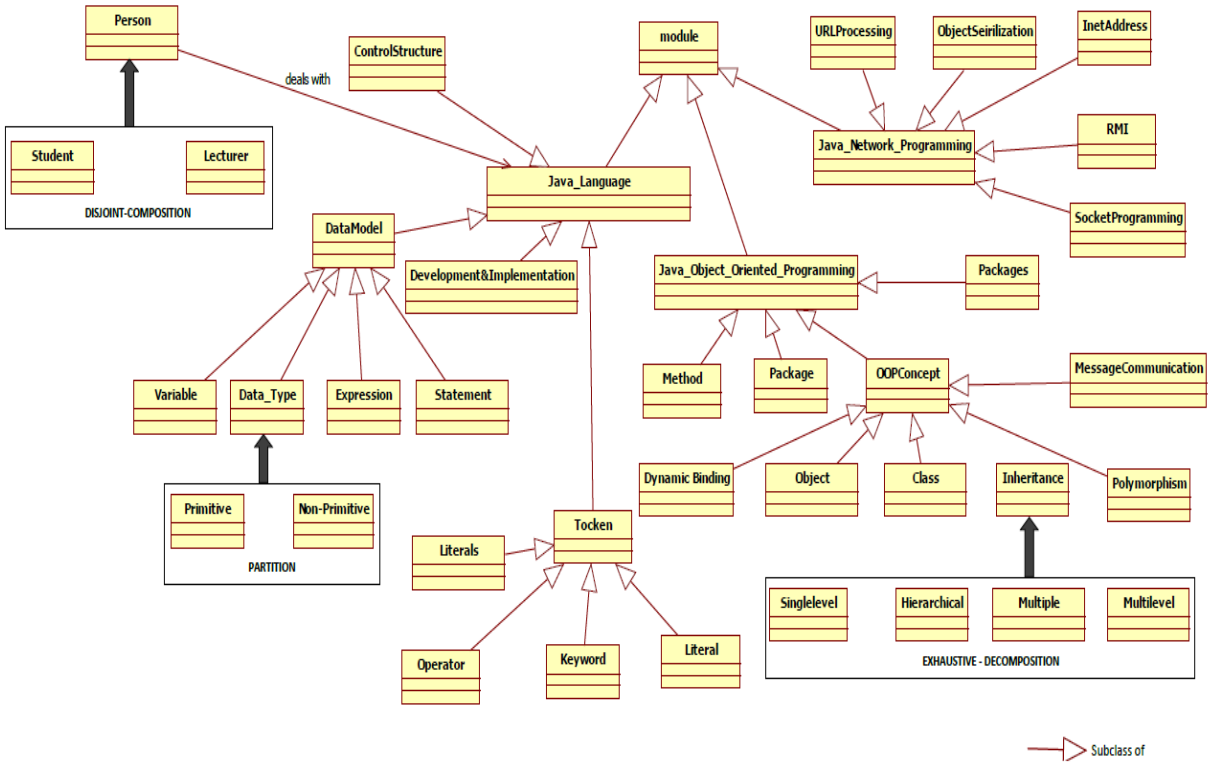


Figure 40: Excerpt of the conceptual model of JLEO

5.2.3.1.2 Add Complex Restrictions and rules

This activity primarily focuses on the identification and definition of the formal axioms and rules to be followed explicitly in the design of the ontology. This activity takes place after the finalisation of the domain-specific concepts and taxonomy. The formal axioms are usually logical expressions that have a boolean value of 'true' when used for evaluation within the ontology. They play a leading role as they are used to specify the constraints in the ontology. With ontologies used for representing the business-domain concepts and attributes in a structured manner with consolidated axioms, the incorporation of formal axioms is inevitable. Axioms are one of the most effective mechanisms for deriving new information from the ontology and are usually expressed by a formal representation. The main categories of axioms are *epistemological*, *consolidated* and *derived*. In the case of the JLEO, the epistemological axioms have been defined to enforce the use of constraints to structure the concepts (e.g. *subclass-of*, *partition*, *cardinalities*, etc.) Later by using OWL, a minimum number of consolidated and derived axioms have been implemented. The KE is the key person who identifies and defines the formal axioms of the ontology in accordance with the opinion of the

DE. An instance of a formal constraint used in the JLEO would be, "*a Person cannot be a Lecturer and Student at the same time"*

SCIM proposed a table (*Section Chapter 4.5.3.2)* to consolidate the formal axioms of the JLEO. An excerpt of the JLEO axiom table is shown in Table 10. The SCIM recommends First-Order Logic (FOL) for representing formal axioms as they are more expressive in description logic. FOL is similar to natural language and represents the world as concepts, relations and functions. The second row of the table shows that the datatype of a variable or a method in Java programming is either primitive or non-primitive, but it cannot be both during the declaration of either the variable or the method. The columns for referred concepts and relationships show the names of the concepts and relationships used in the formal axioms. The first order variables represented as? X and ?Y have been used as the variables for *datatype* and *java-programming.* Similarly, some formal axioms have been incorporated in the JLEO prototype, but, unlike other methodologies, rules have also been defined in this activity.

Formal Axiom Name	Description in English	First Order Logic	Referred Concepts	Referred relationships	Variables
Unsuitability lecturer student	A Person cannot be Lecturer and Student in the Java_ programming	not(exists(?A,?B)(person (?A) and java-programming (?B) and [is lecturer](?A,?B) and [is student](?A,?B)))	Person Java-programming	is lecturer is student	?A ?B
Incompatibility primitive non-primitive	The datatype of a variable cannot be primitive and non-primitive in Java_ programming	not(exists(?X,?Y)(primitive (?X) and java-programming (?Y) and [is primitive](?X,?Y) and [is non-primitive](?X,?Y)))	Datatype Java-programming	is primitive is non-primitive	?X ?Y

Table 10: Excerpt of the Axiom table of JLEO

An asserted class hierarchy generated by the reasoner for the concept expression of JLEO is shown in Figure 41. This illustrates that the reasoner-generated diagram follows the constraints enforced by the formal axioms.

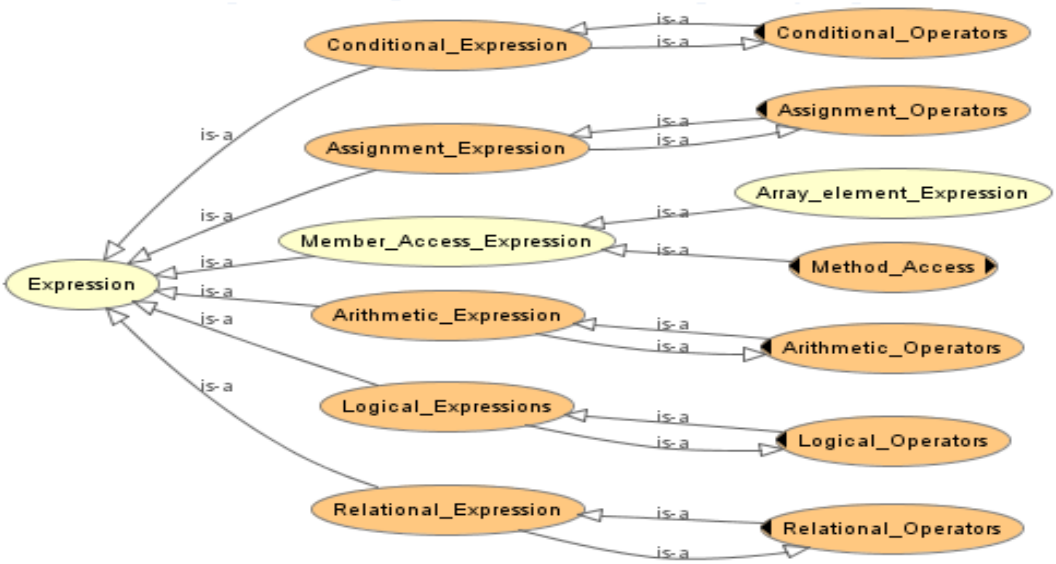


Figure 41**:** Asserted class hierarchy of the concept 'expression'

To infer knowledge in the ontology, rules are generally used by ontology designers, which have been applied in the form of the values of attributes, and the instances of the relation. For example, if the *visibility control* of a variable/method is not specified explicitly during the declaration, by default it will be set to *package*. Different ontology development methodologies have proposed different ways to define the rules incorporated in the ontology. A name rule-table has been proposed by METHONTOLGY containing the rule and its description in natural language together with a formal expression used for its application. SCIM has adopted a template which is more like an algorithm where the commonly used selection structure *if..then* was used.

"If the visibility control of any variables and methods is not declared explicitly, then by default the package visibility control will be applied for such variables and methods".

The rule would be:

If <visibility control not defined> then <package visibility control will be assigned automatically>

5.2.3.1.3 Describe concepts attributes and relationships

This is a complementary activity to the *Domain Vocabulary Acquisition* and *the Enumeration of Concepts, Properties and their Definition* activities of the predecessor stage. The existence of this activity is mainly due to the underpinning iterative-philosophy of SCIM

for the ontology development. Further to that, this activity complements the conceptual model created as part of the conceptualisation stage with the inclusion of the *instance* and *class* attributes as part of the refinement process. The instance attribute is a property of the individual objects within the concept, but the class attributes are the properties of the respective class independent of the instance.

For example, the value of the *studentId* attribute varies for the individual *student1* and individual *student2*, whereas the *person_type* attribute of the *person* concept is a *class attribute* which had an enumeration of *value_ type* such as *lecturer* or *Student.*

The description of both kinds of attribute is crucial in conceptual-model development as they are interpreted in the formal language during the implementation stage. SCIM strongly recommends incorporating both class attributes and instance attributes in the glossary of terms (GT) for their perusal in this activity. Most of the existing methodologies including UPON have not proposed a mechanism for describing the details of attributes at the development stage. Instead, they offer the necessary guidelines to interpret the details of attributes in formal language as part of the implementation. Hence, the expressive power of the formal language used in the description of the attribute details is important. In the case of METHONTOLOGY, two exclusive mechanisms in the form of an instance-variable table and class variable table are provided to describe the attribute details. Although it offers a specific approach, the preparation of two such tables are a time-consuming process for the KE and ODT. Considering the pros-and-cons of both UPON and METHONTOLOGY methodologies, SCIM proposed a mechanism which offered an explicit step for describing attributes by proposing a single table called an attribute table which contains the details of instance and class attributes.

As part of the development of JLEO, samples of instance and class attributes have been incorporated in the GT prepared as the deliverable of the conceptual model. Samples from the GT were used for the attribute table prepared by the KE for this activity. Table 11 shows an excerpt for the attribute table of the JLEO. The column *Concept name/Defined concept* shows the source of the definition of the concept. The value type represents the enumeration of concepts according to the *subclass-of relationship* for the class attributes. The *values* column represents the specific type of primitive values for instance attributes, and user-defined values for class attributes. The *range of values* and *cardinality* represents the minimum and maximum values of the respective attributes and the number of possible instances of the concept respectively.

Attribute type	Attribute Name	Concept Name/Defined Concept	Value type	Values	Range of values	Cardinality
Instance attribute	StudentId	Student	integer		1..1	(1,1)
Instance attribute	StudentName	Student	String		1	(1,1)
Instance attribute	Modulesregisterd	Student	integer		1..3	(1,3)
Instance Attribute	ModuleTitle	Module	string		1..1	(1,1)
Class Attribute	Person_Type	Lecturer	[lecturer, student]	lecturer	2	(1,2)
Class Attribute	Person_Type	Student	[lecturer, student]	student	2	(1,2)

Table 11: An excerpt of the attribute table of JLEO

The entries in the attribute table have been interpreted into the JLEO with the help of Protégé, the chosen ontology editor, as shown in Figure 43 shows two of such interpretations. In summary, the completion of this activity as recommended by SCIM ensures that all the concepts and the relationship between them, together with their attributes are described.

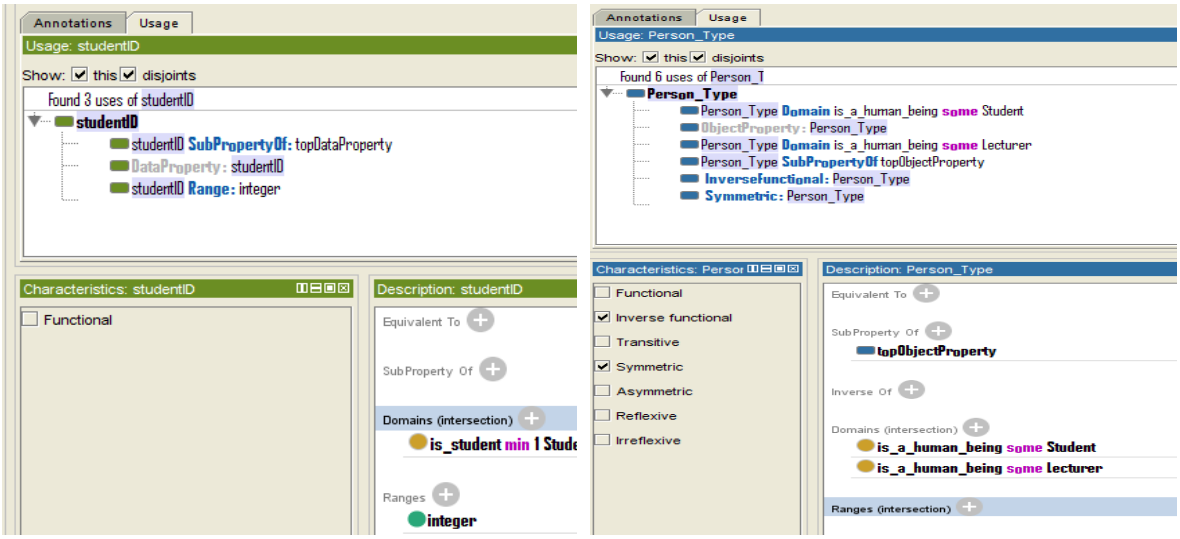


Figure 42: Excerpt of Protégé interpretations of JLEO attribute table

5.2.3.1.4 Establish Ad-hoc binary relationships

This is the final activity of the development stage which mainly focuses on defining the semantic ad-hoc binary relationships in detail. During this activity, the KE determines the ad-hoc relationships between the concepts defined in the GT. Figure 43 shows the ad-hoc binary relations *is-a* and *has-a* with their inverse relations *isPartOf* and *haspartA*. These relations connect the classes and needs to be checked to ensure that no errors have occurred.

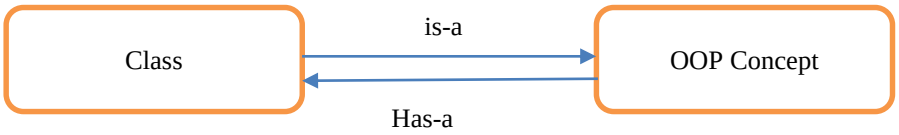


Figure 43: An excerpt of ad-hoc binary relationship

In the JLEO, an object property *'teaches'* links the individual *lecturer1* to the individual *OOP*. The OWL properties defined for the *teaches* object property are *inverse, functional, asymmetric* and *reflexive*.

Inverse property: if a property links individual x to individual y, then its inverse property will link individual y to individual x. In the case of JLEO, the property is *teaches* and its inverse property is *istaught*. If lecturer1 teaches the *OOP* module, then because of the inverse property, it can infer that the *OOP istaught* by lecturer1.

Functional property: For a given individual, if there must be at least one individual that is related to the other individual via the property. If lecturer1 teaches the *OOP* module, and lecturer1 also teaches the *Internet programming* module, then it implies that *OOP* and *Internet programming* are the individuals of Module.

Asymetric property: The property *teaches* relates the individual *lecturer1* to the individual OOP but the individual OOP is not related to the individual *lecturer1* via the *teaches* property. Various OWL property characteristics have been applied to the object properties defined in the JLEO as shown in Figure 44.

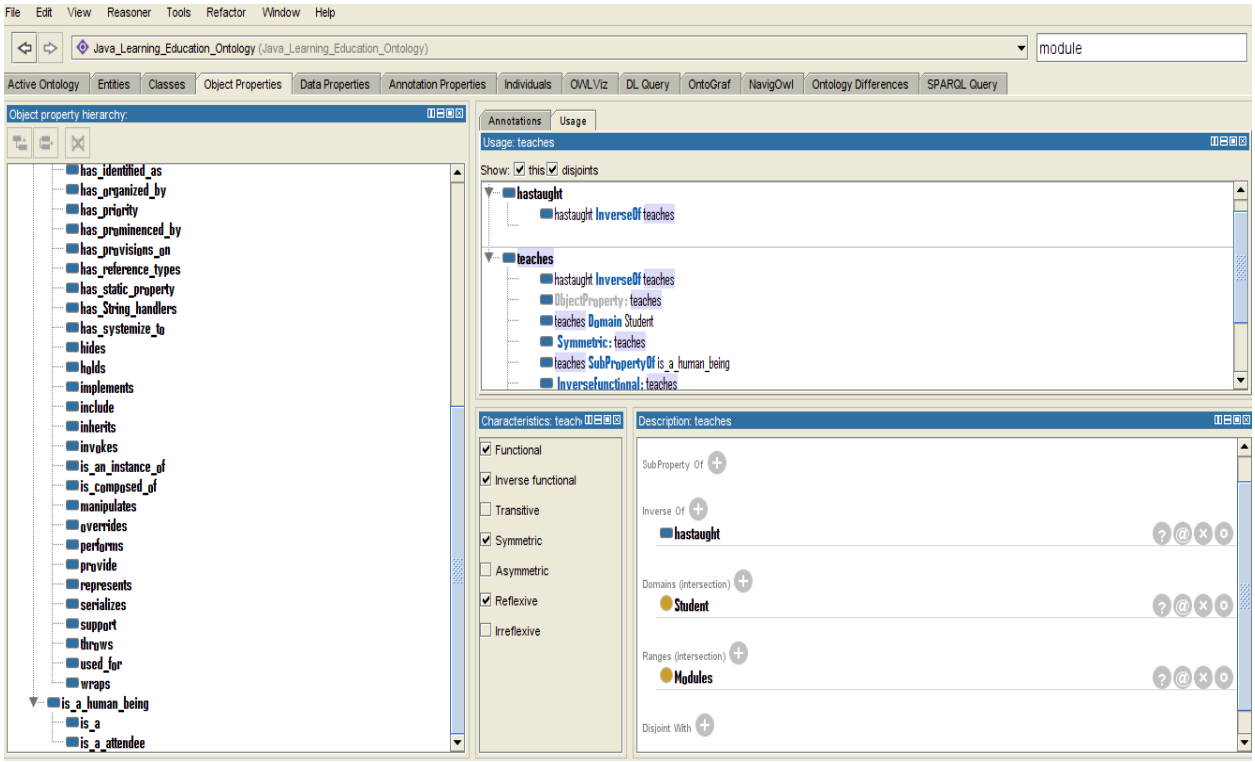


Figure 44: An excerpt of the object property 'teaches' of JLEO

SCIM proposed an ad-hoc binary relationship table to record the relationship details. The table contains the name of the relation, the names of both source and target concepts together with their cardinality and inverse relation respectively. All the ad-hoc binary relationships should be incorporated within the table. An excerpt of the of the ad-hoc binary table is shown in Table 12.

Relationship Name	**Source**	**Cardinality(Maximum)**	**Target**	**Inverse Relation**
teaches	Lecturer	3	Module	isTaught
isLecturer	Person	N	Lecturer	hasLecturer
isStudent	Person	N	Student	hasStudent

Table 12: Ad-hoc binary relationship table of the JLEO

5.2.4 Implementation and Deployment stage

The last stage of SCIM is the *Implementation and deployment stage*. As the name indicates, this stage focusses exclusively on the implementation of the proposed domain ontology. This encompasses the transformation of the conceptual model developed in the previous stage to

its formal language representation which implicitly transforms the human-understandable conceptual model to a machine-readable representation. The final deliverable upon the completion of this stage is a validated formal-language representation of the domain ontology. Considering the features offered by the popular Ontology Design Environment (ODE), SCIM strongly recommends the use of an appropriate ODE for ontology implementation at this stage. Unlike other existing ontology development methodologies such as METHONTOLOGY, which use native ODEs, SCIM recommends the use of any ODE which encompasses the essential features such as the support for the generation of different formalisms, the availability of a reasoner, the easy integration of visualization tools, etc. The Resource Description Framework (RDF) or Web Ontology Language (OWL) are recommended for the formal language representation of the conceptual model, where the KE, together with the ODT, plays a pivotal role. SCIM recommends the use of both RDF and OWL as the modelling languages for the description of data of the model.

According to the hierarchy of the stages and workflows of the SCIM, the only workflow attached to this stage is the *implementation workflow*. The two activities assigned to this workflow are the *formal language representation* and the *vocabulary linking with data*. For the JLEO development, Protégé 4.3 was chosen as the ODE and OWL was chosen as the formal language for the ontology representation. Both the activities assigned to this workflow can be performed iteratively in accordance with the philosophy of SCIM. The formal language interpretation of the JLEO is in line with the deployment discipline of conventional RUP and the workflow activities of this stage originate in the construction and transition phases of that methodology.

The details of the workflow and activities applied for the development of the JLEO have been described in detail in the subsequent sections along with the necessary screenshots. Selected code snippets have been presented to show the formal language representation of the respective parts of the JLEO.

5.2.4.1 The development platform

Protégé is a free open-source platform introduced by Stanford University School of Medicine and is considered to be one of the best available ODEs for domain ontology design and development (Protégé: Stanford University 2005). Considering the fast-growing user community (30000 plus), Protégé is one of best choices for the developer to design and develop ontologies. It contains a suite of easy-to-use tools for the KE and members of the ODT to build

domain models and ontologies. The modelling techniques used in Protégé-had a close resemblance to object-oriented (i.e. frame-based) techniques used in software developments. Therefore, one of the reasons for choosing it for JLEO development was to reduce the learning curve of software practitioners converting to ontology development.

Unlike other ODEs such as Apollo, Protégé supports graphical views, web information extraction (Web-protégé) and collaborative ontology development. It implements knowledge-modelling structures and actions that support the construction, visualisation, and manipulation of ontologies in various formalisms. Moreover, the philosophy behind the design of Protégé is very much synchronised with the rationale behind SCIM development, which was to give guidance to the ontology developers and DEs, thereby reducing the time taken for ontology development. Protégé has an open architecture that allows different modelling languages to be used. Figure 45 shows the abstract level classes of the JLEO in Protégé representation. The graphical model generated by OWL-Viz, a plugin for Protégé is shown in Figure 47.

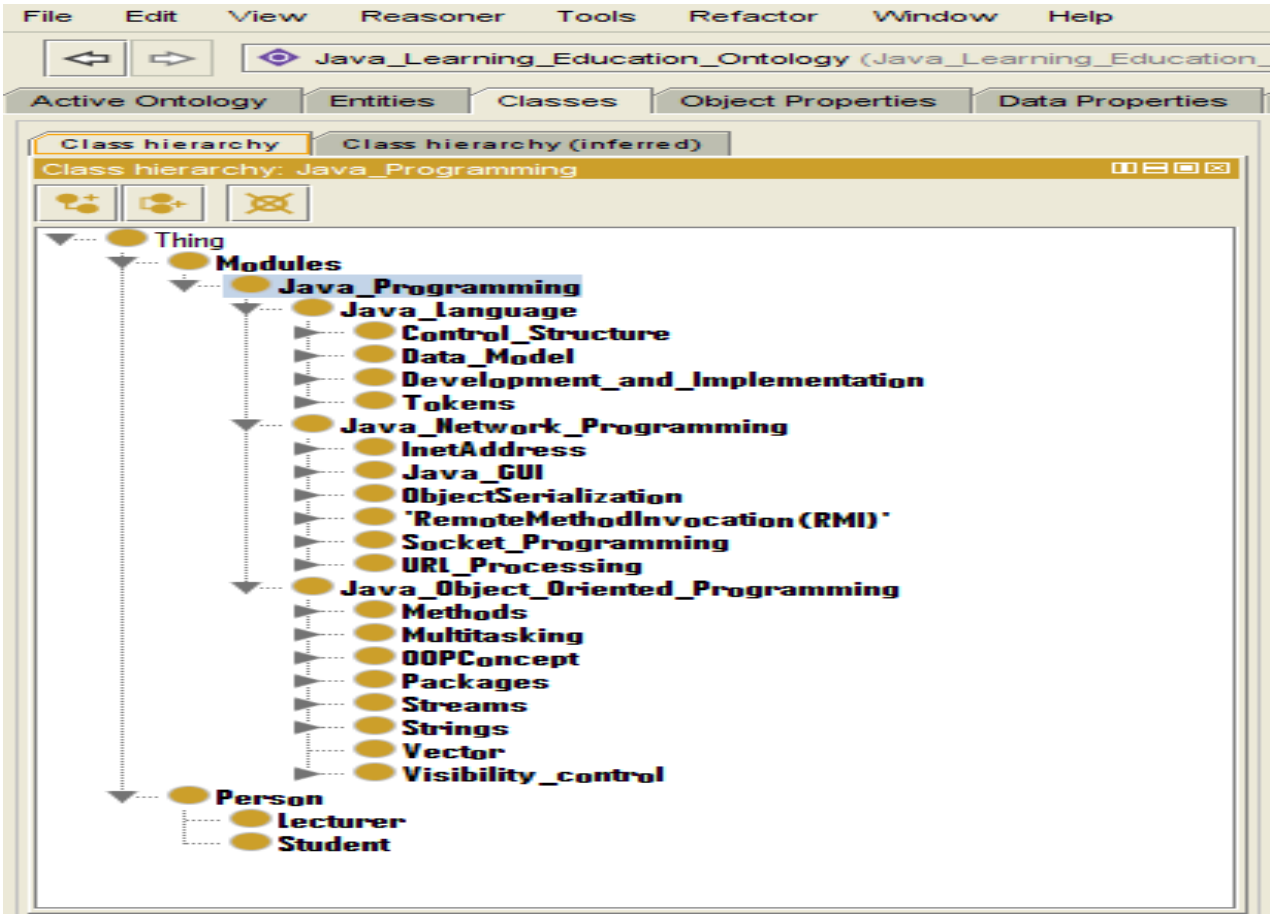


Figure 45: Abstract level classes of JLEO in protégé

One of the other significant benefits of Protégé as the ODE for JLEO development is its scalability and extensibility. Protégé offers an environment to build and manipulate enterprise ontologies in an efficient manner (Cardoso and Escórcio 2007). The extensibility of Protégé by the addition of plug-ins, was the reason it was adopted and customised for the requirements of the JLEO. The most useful were the tab plug-ins. Figure 46 shows all the available tabs for the version of Protégé which was used for the JLEO development.

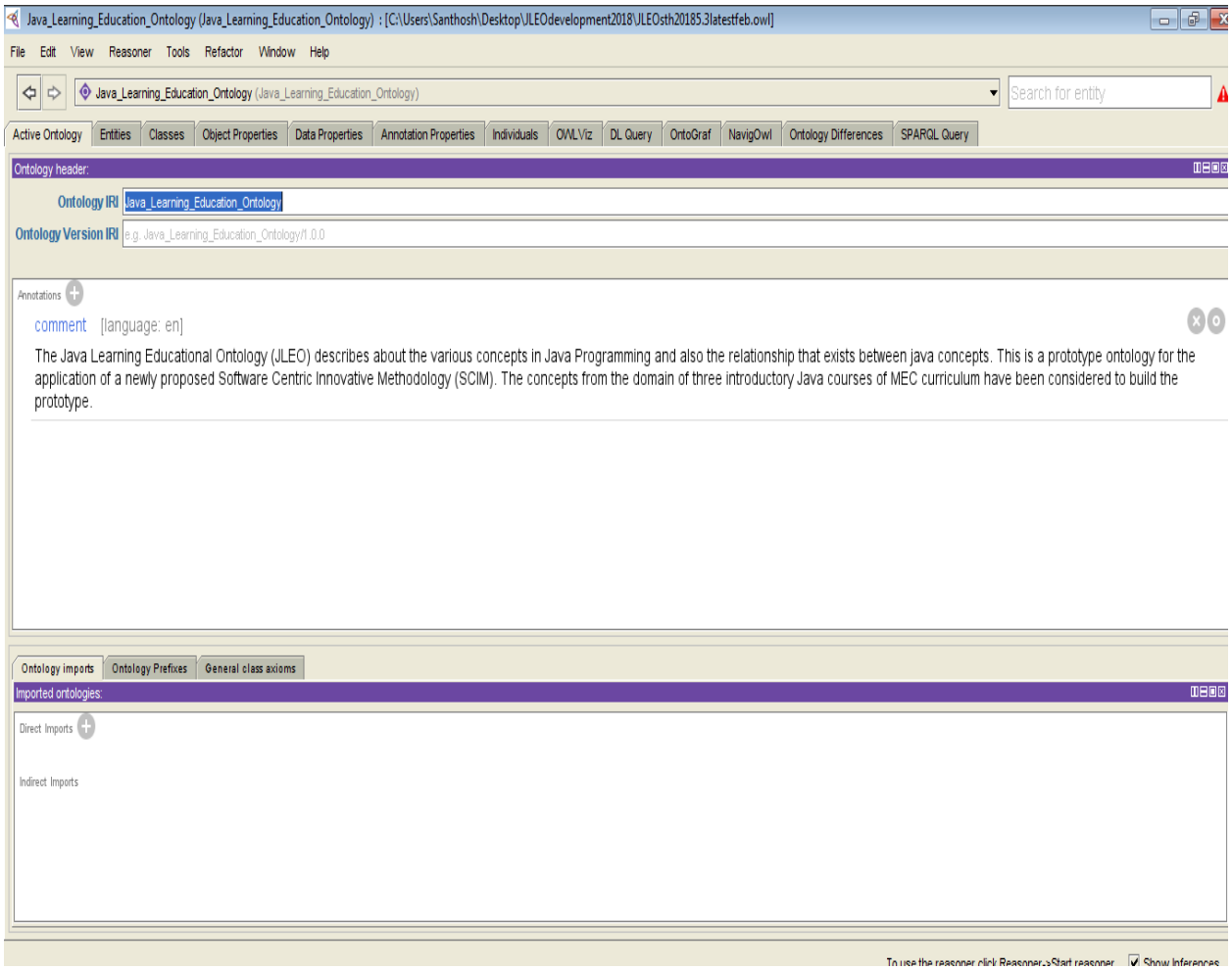


Figure 46: Tab plug-ins of Protégé

The currently available tabs included facilities for visualisation, graphical representation, ontology merging and querying etc. The tabs such as OWLViz, OntoGraph offered mechanisms to present the graphical views of the JLEO. The OntoGraph tab supported the customised navigation, zooming and searching of particular elements in the knowledge structure. Moreover, it presents different layouts of nodes in a graph which highlights the connections between clusters of data. Figure 47 and Figure 48 shows instances of OWLViz and OntoGraph representation for the JLEO.

Figure 47: OWLViz instance of JLEO

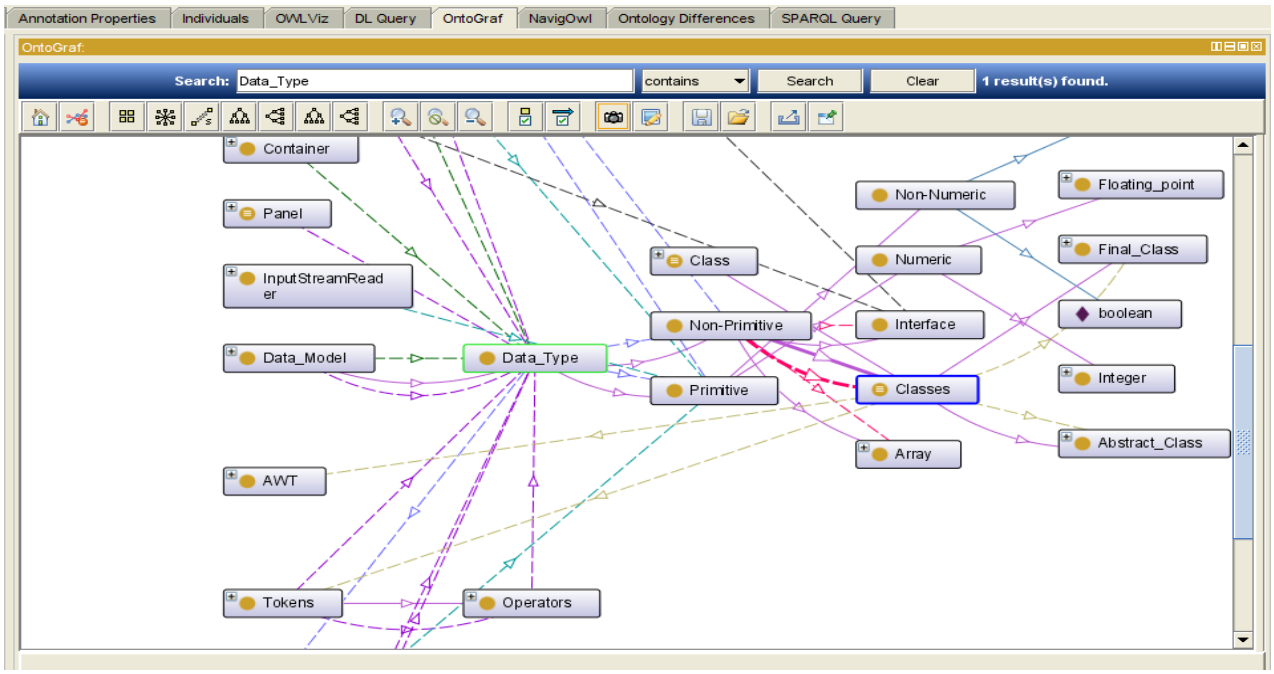


Figure 48: OntoGraph instance of JLEO

5.2.4.2 Implementation workflow

This workflow primarily focusses on the conversion of the conceptual model developed in the predecessor stage to a formal language representation. During this workflow, by considering the necessary parameters, the selection of ODE and choice of formal language takes place. The different ODEs referred to in the literature review were considered as part of this workflow prior to choosing Protégé. The brief description of the development platform given in the previous section covered the rationale behind the selection of the chosen ODE with the necessary screen shots.

5.2.4.2.1 Formal Language Representation

This is the first and foremost activity belonging to the Implementation workflow. During this activity, the KE together with the members of the ODT make the best use of the chosen ODE. This is mainly for the transformation of the conceptual model of JLEO designed in the previous stage to a formal language representation. Protégé, the chosen IDE supports the generation of formalisms based on the designed ontology. The Resource Description Framework Schema (RDFS) and OWL were the two formalisms considered for the language representation of JLEO. Both RDFS and OWL are W3 Specifications (Staab and Studer 2004). RDFS offers a mechanism for the users to denote the semantic structural relationships between the data by specifying them in a subject-predicate-object format. OWL is more expressive compared to RDFS and has its own formal semantics (Group 2012). Therefore, OWL can be used to capture the knowledge in a machine-understandable form.

The three core constructs of OWL which include *class*, *individual* and *property*, best describe the semantic structural relationships among the concepts of the chosen domain (Horridge 2011). In OWL semantics, classes are a set of *individuals* and *individuals* are the *objects* in the domain. Relationships in OWL are binary which can be represented in the same subject-predicate-object form as RDFS. Protégé supports OWL through the Protégé-OWL plugin. Its core functions are based on object-oriented modelling (i.e. frame-based) which reduces the learning curve for software practitioners to become ontology designers as many of them are comfortable with Object Oriented Software Engineering (OOSE). Protégé -OWL has an open architecture which allows other modelling languages to build on the top.

JLEO classifies the various learning objects used in the three introductory Java modules of the MEC undergraduate curriculum into a well-defined hierarchical structure. The abstract-level ontology is defined in the *jleo.owl* file which defines the hierarchical structure

of the concepts as shown in Figure 45. Figure 49 shows the super class, subclass and individuals at an abstract level. The instances of *student* are *student1*, *student2* and *student3*. *Student* is also a subclass of *person*. *Module* has the subclass J*ava programming* and an individual *network programming* and *person* has subclasses of *lecturer* and *student*, the relationships being represented by different kinds of arrow lines.

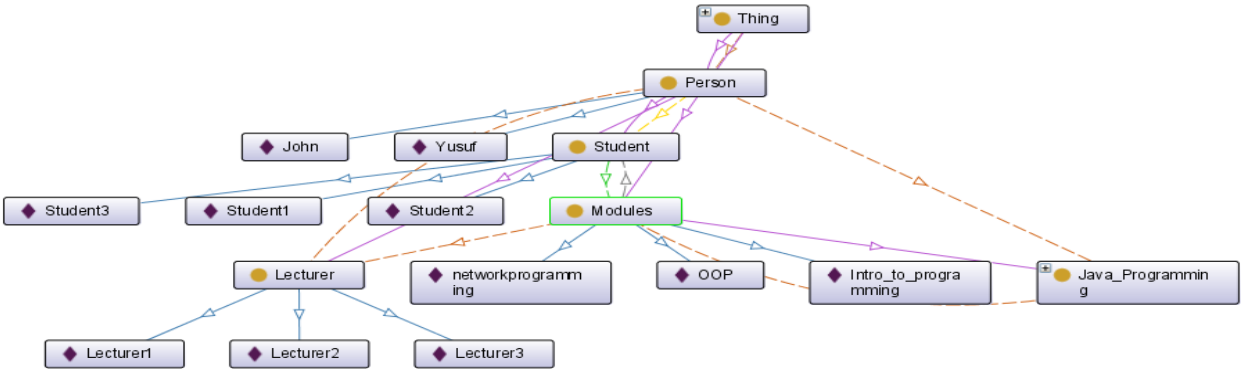


Figure 49: Excerpt of JLEO with semantic relationships

Figure 50 shows the owl code snippet generated by Protégé for the creation of the classes, operators, person and polymorphism. All the classes of the JLEO have also been declared with the names stated in the Glossary of Terms defined in the conceptualisation stage.

```
<Declaration>
  <Class IRI="http://www.semanticweb.org/ontologies/2017/8/untitled-ontology-20#Operators"/>
</Declaration>
<Declaration>
  <Class IRI="http://www.semanticweb.org/ontologies/2017/8/untitled-ontology-20#Person"/>
</Declaration>
<Declaration>
<Class IRI="http://www.semanticweb.org/ontologies/2017/8/untitled-ontology-20#Polymorphism"/>
</Declaration>
```

Figure 50: OWL Snippets of the JLEO classes

The owl codes of subclasses were generated in line with the JLEO designed at the development stage. Table 13 shows the OWL code snippet of the three subclasses of *component* class and

their OWLViz graph. In the conceptual model of the JLEO, *CheckBoxGroup*, *CheckBox* and *Choice* are the subclasses of Component class.

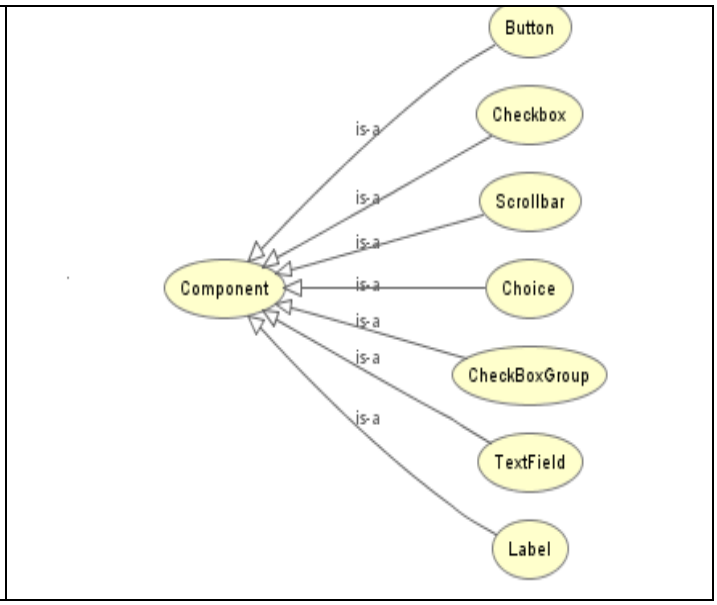
…….. <SubClassOf> <Class IRI="#CheckBoxGroup"/> <Class IRI="#Component"/> </SubClassOf> <SubClassOf> <Class IRI="#Checkbox"/> <Class IRI="#Component"/> </SubClassOf> <SubClassOf> <Class IRI="#Choice"/> <Class IRI="#Component"/> </SubClassOf> …….	

Table 13: OWL snippet of three subclasses of Component class and OWLViz

The *disjoint* classes defined in the JLEO conceptual model have been represented in formal language. The snippets of *disjoint* classes are shown in Figure 51. The *disjoint* classes represented in the snippet belong to the *layout* class. The formal language interprets the *disjoint* classes in such a way that no single instance of any *layout* will be an instance of any other *layout* class.

```
<DisjointClasses>
    <Class IRI="#GridBagLayout"/>
    <Class IRI="#GridLayout"/>
 </DisjointClasses>
 <DisjointClasses>
    <Class IRI="#GridBagLayout"/>
    <Class IRI="#GroupLayout"/>
 </DisjointClasses>
 <DisjointClasses>
    <Class IRI="#GridBagLayout"/>
    <Class IRI="#ScrollPanelLayout"/>
 </DisjointClasses>
 <DisjointClasses>
    <Class IRI="#GridBagLayout"/>
```

```
    <Class IRI="#SpringLayout"/>
  </DisjointClasses>
  <DisjointClasses>
    <Class IRI="#GridLayout"/>
    <Class IRI="#SpringLayout"/>
  </DisjointClasses>
  <DisjointClasses>
    <Class IRI="#GroupLayout"/>
    <Class IRI="#ScrollPanelLayout"/>
  </DisjointClasses>
```

Figure 51: An excerpt of the OWL code snippets of the *disjoint* classes

Both data-properties and object-properties defined in the conceptual model of the JLEO have been represented in terms of the formal model. The instances of the formal language representation of inverse properties *'teaches'* and *'istaught'* for the instances of *lecturer* and *student* objects and the functional object properties achieved by the *and assigns* are shown in the first column of Table 14. In the second column, an instance of the data properties defined for the class *module* is represented. The data properties moduleCode, moduleID, and moduleTitle best describes the concept of *module*.

Object Properties	**Data Properties**
<InverseObjectProperties> <ObjectProperty IRI="#istaught"/> <ObjectProperty IRI="http://www.semanticweb.org/ontologies/2017/8/untitled-ontology-20#teaches"/> </InverseObjectProperties> <FunctionalObjectProperty> <ObjectProperty IRI="#achieved_by"/> </FunctionalObjectProperty> <FunctionalObjectProperty> <ObjectProperty IRI="#assigns"/> </FunctionalObjectProperty>	*<Declaration> <DataProperty IRI="#moduleCode"/> </Declaration> <Declaration> <DataProperty IRI="#moduleId"/> </Declaration> <Declaration> <DataProperty IRI="#moduleTitle"/> </Declaration> <Declaration> <DataProperty IRI="#moduleregistred"/> </Declaration>*

Table 14: Object and Data properties representation

The individuals defined in the conceptual model of the JLEO have been linked with their respective classes in the formal-language representation. The snippet representing object and data properties for the individuals of *lecturer* and *student* is shown in Figure 52.

< DataPropertyAssertion>

<DataProperty IRI="#LecturerFirstName"/>

<NamedIndividual IRI="#Lecturer1"/>

<Literal datatypeIRI=”&rdf;PlainLiteral”>John</Litearl>

</ DataPropertyAssertion>

< DataPropertyAssertion>

<DataProperty IRI="#studentName"/>

<NamedIndividual IRI="#Student3"/>

<Literal datatypeIRI=”&rdf;PlainLiteral”>Mohammed </Litearl>

</ DataPropertyAssertion>

Figure 52: OWL snippet of individual

5.2.4.2.2 Vocabulary Linking with Data

This activity primarily focusses on the creation of instances. In addition to the creation of instances, the techniques that can be used for the deployment of the JLEO have also been incorporated. Instances of the domain concepts are the building blocks of ontologies and lead to ontology-based knowledge systems. In the case of the JLEO, the instances of various Java programming concepts have been created. This has been done with the support of the features offered by Protégé. SCIM relied on the conceptual model created as the deliverable of the predecessor stages for creating the relevant instances/objects. The KE in consultation with the DE defines the relevant instances of the JLEO. While creating the individuals, the ODE is given provision to assign the values for both the defined object and the data properties. For an instance of the concept module, such as the *Introduction to programming*, the semantic relationship *isTaughtby* has been defined as an object property. As described earlier, this object property establishes the relationship between the individuals of *module* and *lecturer* by the semantic structural relationship Introduction_to_programming isTaughtby *Lecturer1***.** The data property *moduleCode* has been defined with its permitted data type. Figure 53 shows the screenshot of the steps used for the creation of the individuals of the *Introduction_to_programming* concept.

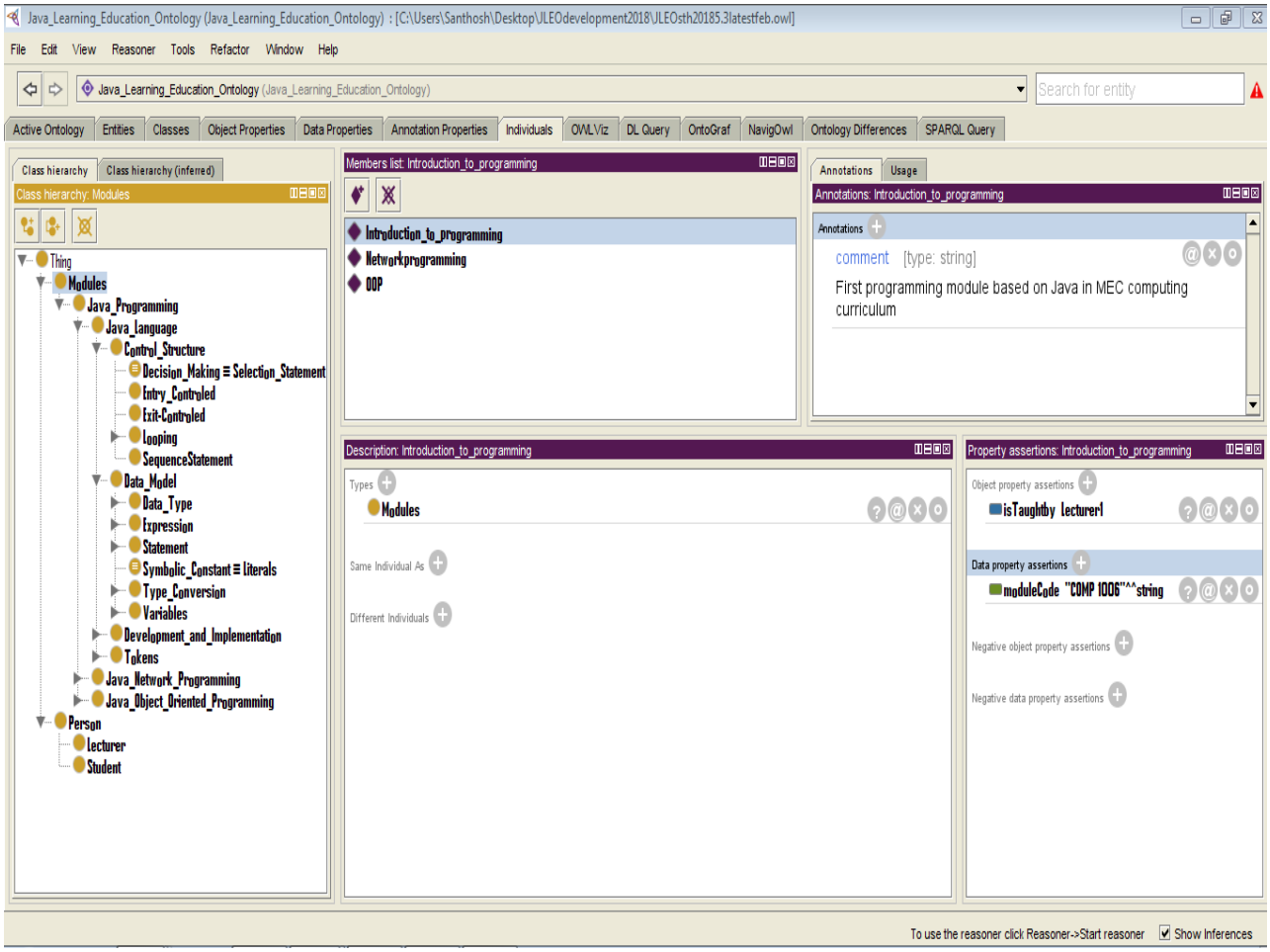


Figure 53: Screenshot of the individuals in Protégé

The SCIM proposed a table called *instance table* to maintain the details of the instances defined in the ontology. The KE can fill the entries in this table and this gives an overview of the detail of the instances. Figure 54 shows the details of the instances of the class *student*. An excerpt of the Instance table of the JLEO for the *student* class are shown in Table 15. The SCIM recommends that the practitioners incorporate all of the instance details in the proposed instance table.

Instance Name	Concept Name	Attribute	Values
Student1	Student	StudentName	abc
Student2	Student	StudentID	S101
Student3	Student	Student_type	UGStudent

Table 15: An excerpt of the instance table of the JLEO Ontology

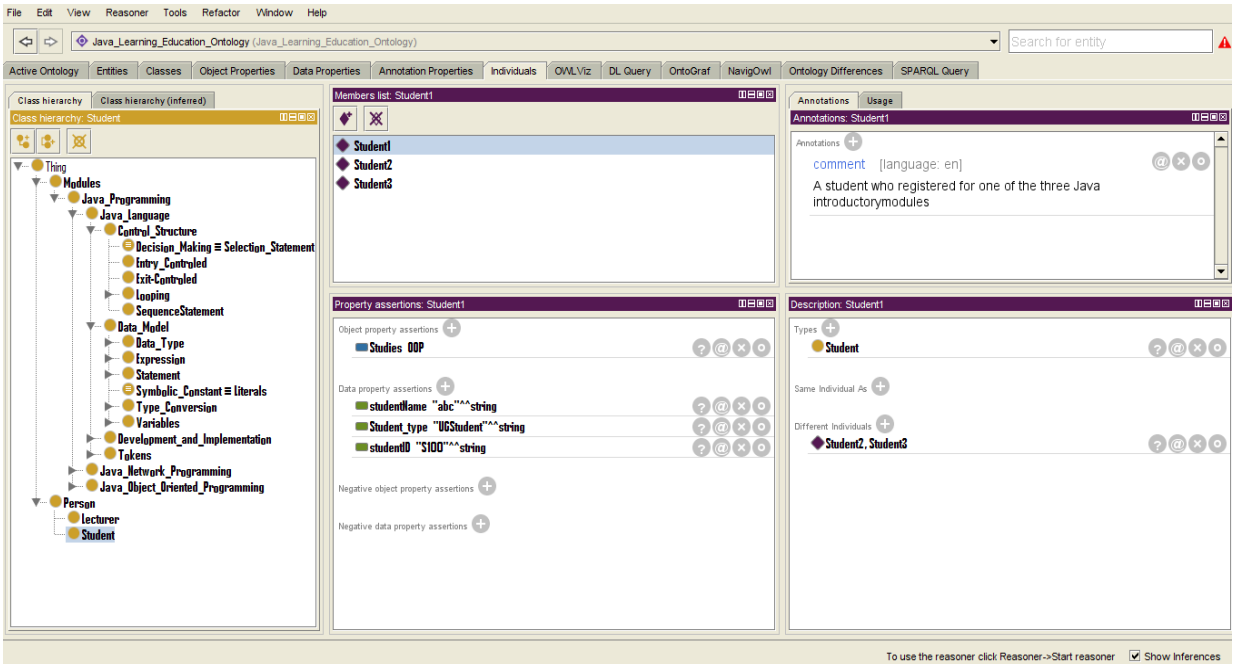


Figure 54: Student class's instances

5.3 Deployment of JLEO

The Deployment of the JLEO originates from the transition phase of the underpinning RUP methodology. It can be deployed either to the web or on a standalone machine. A web-based ontology was not required as to give support to the application to the stages and activities of SCIM a stand-alone version was sufficient. However, a web-based ontology could be developed in the future to provide a knowledge-based java learning system for MEC students. The prototype of JLEO was developed and demonstrated on a local machine.

The abstract-level OWLViz knowledge representation of the JLEO has been shown as evidence of the deployment. Figure 55 shows the abstract-level view of the JLEO development for the three variations of Java programming. Figure 56, 57 and 58 shows the abstract-level knowledge representation for the Java language, Java network programming and Java object-oriented programming modules respectively.

Figure 55: Abstract level OWLViz representation of the JLEO

Figure 56: Abstract level OWLViz representation of ITP module

Figure 57: Abstract level OWLViz representation of the IP module

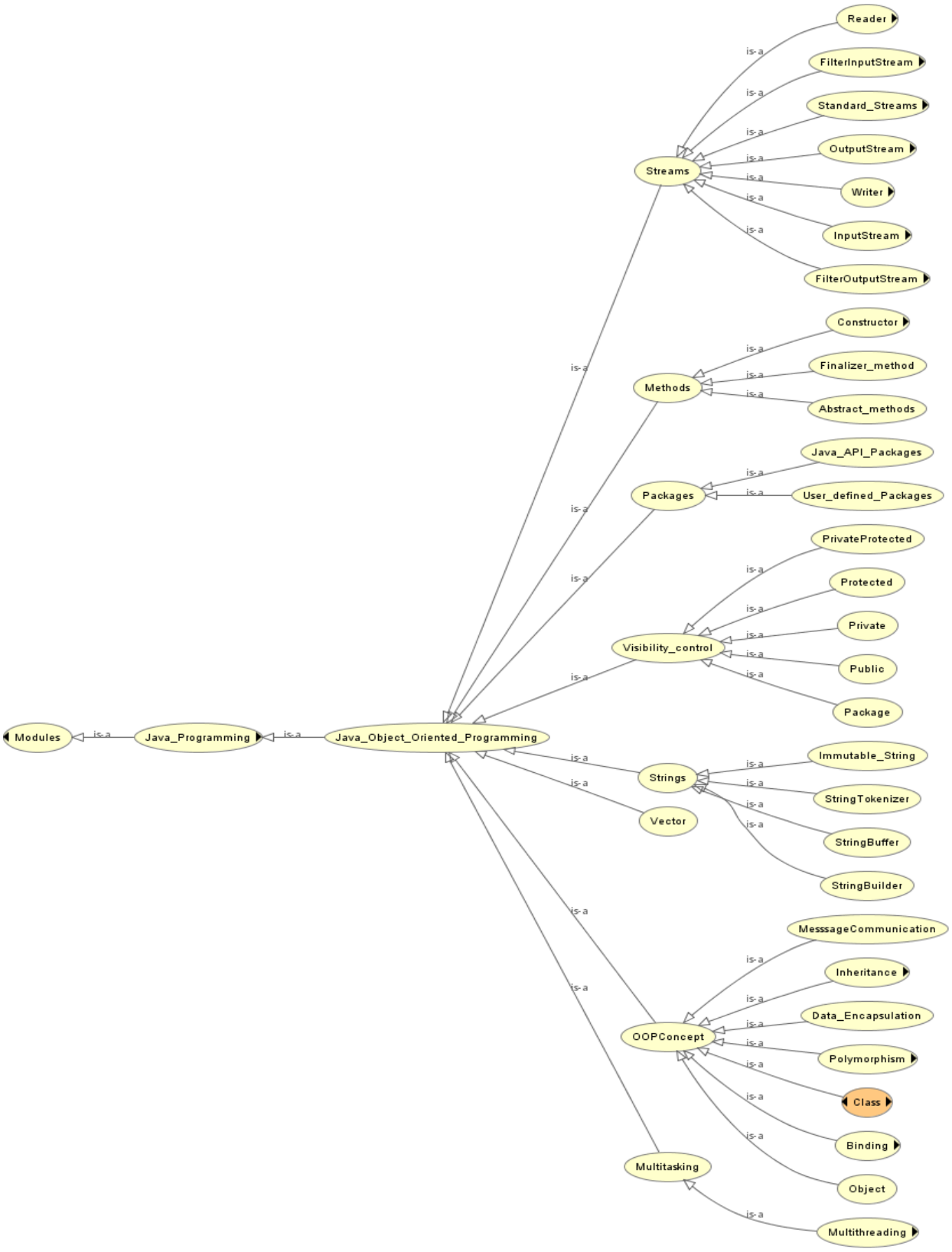


Figure 58: Abstract level OWLViz representation of the OOP module

5.4 Chapter Summary

A detailed description of the application of the proposed SCIM methodology for the domain of Java learning, in the form of the JLEO (Java Learning Education Ontology) has been presented in this chapter. Detailed coverage of the application of SCIM in terms of its different stages, workflows and activities have been incorporated with the necessary figures, tables and code snippets. The techniques used in the JLEO development for the SCIM have been presented with the necessary evidence of their application to the domain.

6. Evaluation of Proposed Methodology

This chapter provides a detailed description of the evaluation of SCIM. The evaluation of research findings is an integral part of any form of research. This chapter describes and illustrates the evaluation framework, detailing and providing evidence of the techniques used. Details of the target group with a brief description of the profiles of experts selected to perform the evaluation are then given. This is followed by the presentation and analysis of the results from the questions given to the evaluators to support the evaluation. A consolidated matrix comparing the SCIM against other methodologies is then presented which is followed by a chapter summary at the end.

6.1 Introduction to the evaluation framework

The objective of the evaluation process is the validation of the research findings against the problem that has been addressed by this study. The results of the evaluation process examine whether the research conducted provided an optimal solution to the research problem. Although many pre-defined techniques are available for evaluation including those for exploratory research, there is no hard and fast rule to follow. Very often, improvised and off the shelf techniques are used in the evaluation of exploratory research. The purpose of the evaluation is to demonstrate the relevance of the research, that it has been conducted properly, that it has been structured with sufficient precision, and that it provides a real contribution to the specific research field. The evaluation parameters are discussed in the following section.

6.2 Evaluation Parameters

To study and evaluate the exploratory research, a set of parameters have been suggested by Burstein and Gregor (Burstein and Gregor 1999). The five suggested parameters are *Significance of the study*, *internal validity*, *external validity*, *Objectivity* and *Confirmability*. These five criteria and their application to this study will be examined in greater detail below.

The significance of the study was primarily concerned with the importance and the novelty of research and the contribution and relevance of the results. For the research to be significant, it is mandatory that it should deliver a better solution to the research problem than had been proposed previously. Therefore, the research should contribute some new knowledge to the specific research domain. The necessary techniques have been

accommodated in the SCIM evaluation framework to examine the significance of the research by incorporating the appropriate questions in the survey conducted with the target group.

Internal validity focusses on the research artefacts and the availability of reliable and proven evidence to support the research findings, complemented with robust and credible research arguments. Credibility can only be demonstrated by rigorous experiments and the research findings support the research conclusions. However, the credibility level goes down if the negative comments from the evaluation process are ignored. For the SCIM evaluation credibility was assured by providing the prototype ontology together with the artefacts produced from each stage of the development life cycle together with a set of relevant questions for the evaluators to answer.

External validity is concerned with the generalisability of the research findings and is usually expressed as being either low or high. A low reading of external validity indicates that the result findings can only be applicable to the specific cases used for the evaluation. On the other hand, for the external validity to be high it needs to be shown that the research findings could be applied more widely to cases outside the specific domain chosen for the research. To achieve this the experts who had experience in multiple ontology-based projects were asked in the survey to answer questions which were designed to confirm the level of external validity. The questions were:

- ✓ Does the development of SCIM confirm and support the existing theories of ontology development?
- ✓ Were the assumptions, proven practices, and the views, personal opinions and preferences of the experts stated clearly and fully analysed?
- ✓ Was the procedure followed stated clearly with supporting evidence?

Objectivity is demonstrated when all the assumptions, subjective judgements and opinions of the research are explicitly stated.

Confirmability of research is shown when it can be considered by other researchers to be reliable.

The following questions were issued for the target group of experts to answer, to confirm both the objectivity and confirmability of the findings of the research:

- ✓ Are the research objectives and research questions clear and well-connected?
- ✓ Does the research follow a well-defined research design?
- ✓ Are the methods, approaches, experiments followed properly and described clearly?
- ✓ Are the data/parameters used for the experiments relevant?
- ✓ Are the limitations of the research considered in the presentation of the research findings?

The research objectives and the research questions were given to the experts during the briefing session and appropriate questions were incorporated in the survey questions for them to answer. These measures were considered to be sufficient to test the objectivity and confirmability of the research.

6.2.1 Evaluation approach

For the evaluation of the SCIM, a, customised Goal-Question-Metric (GQM) approach (Bandeira et al. 2017) was followed by embedding the parameters presented in Section 6.2.1 in the evaluation framework. The approach is discussed in more detail in the following subsection.

6.2.2 Customised Goal-Question-Metric (GQM) approach

Any engineering process requires feedback and evaluation for measuring its maturity (Basili 1992). The evaluation of an engineering methodology is more complex than those performed on products by customers or end-users. This is because a methodology must produce a working system and many steps are required in the evaluation process to achieve that. For a developer/practitioner to evaluate a methodology properly, the goals/objectives of the methodology have to be specified clearly at the beginning. The GQM approach supports this by defining the goals and objectives which are then refined by a set of questions which collect the judgements of developers or practitioners, together with their qualitative opinions, and offers a final framework for interpretation.

To apply GQM for the evaluation of SCIM some customisation was required to meet the goals of the research. The evaluation parameters discussed in 6.2.1 were used for forming the evaluation framework and a set of questions were used to refine the evaluation parameters to assist the evaluator to complete an evaluation matrix. The matrix was prepared to compare SCIM with other leading ontology development methodologies based on the qualitative opinions of experts. Seven parameters were defined for the comparison and the rationale for

those parameters is described in section 4.1.2 of Chapter 4. The set of questions were circulated to the target group of evaluators to collect their quantitative judgement supported by qualitative justification. Both the parameters and the questions were used for the consolidation of the evaluation matrix.

The primary selection criteria for the evaluation was that the evaluators were chosen from the field of ontology engineering, each with a solid track record as a research investigator in internationally funded projects and having large-scale ontology development experience. A supplementary criterion for selection was that the evaluator had experience of ontology within the education sector in the selected domain or significant experience in ontology development in industry.

The target group was formed with a mixture of evaluators who best met the parameters specified. A list of the evaluators in the target group together with a brief summary of their expertise is presented in Section 6.3.4. The practitioners in the target group had experience in ontology development, funded projects, industrial developments, research and should have publications in top level journals and conferences on ontology engineering. They were asked to evaluate the artefacts, processes and components of the methodology and the relationships between them. Also, they needed to verify how well the proposed methodology was applied in the development of the prototype ontology. By completing the survey questions, the evaluation mechanism offers an opportunity for all the evaluators to give their judgement and the justification for their answers. In addition to that, each evaluator was asked to fill in the evaluation matrix in line with their qualitative statements. The framework used for the evaluation of SCIM is shown in Figure 59.

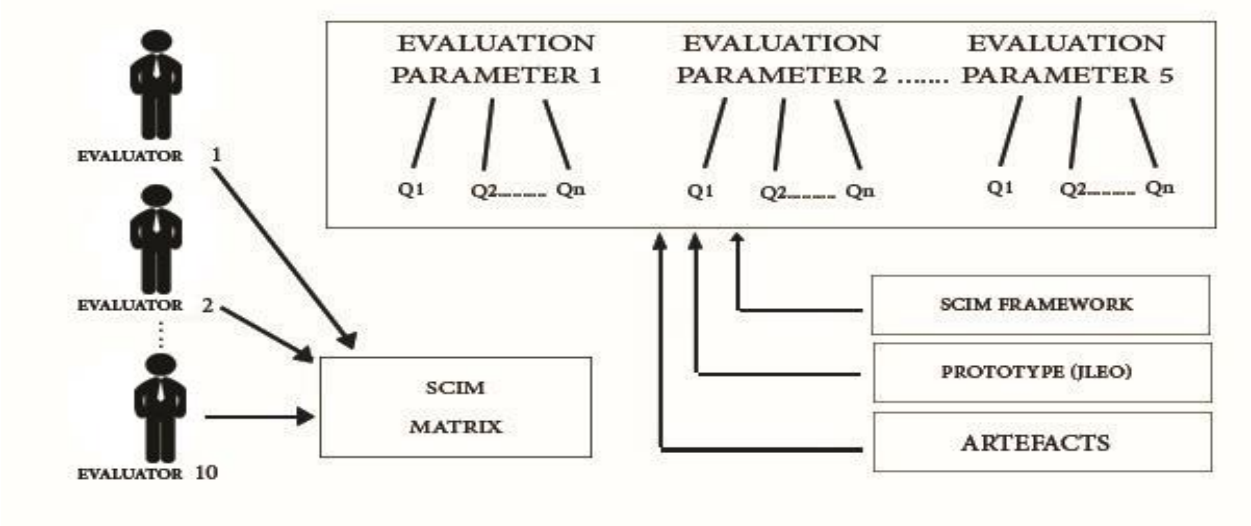


Figure 59: Evaluation Framework of SCIM

According to the customised GQM approach followed, five evaluation parameters mentioned in section 6.2.1 were used as the basis for preparing the set of questions circulated among the evaluators. To assist the evaluation, the final framework of SCIM, the JLEO and the artefacts for each life cycle stage of the JLEO were given to the evaluators as supplementary aids. The details of the process completed are given in the subsequent sections.

6.2.2.1 Briefing Session

A briefing session on the SCIM evaluation was conducted on Wednesday 13th June 2018 for the evaluators. The session was conducted to facilitate the evaluation process to initiate the evaluation workflow for the SCIM. The objectives of the research and research questions were introduced to the evaluators by the researcher. An overview of the proposed methodology specifying the hierarchy of components, the deliverables from each stage of the development life cycle, workflow and activities was then given. Further to that, the JLEO produced using the SCIM was demonstrated as a proof-of-concept. After the demonstration the researcher introduced the matrix to the evaluators and explained the rationale for the parameters used for the ontology comparison. At the end of the session, the researcher distributed the survey questions, evaluation matrix, supplementary materials (e.g. an overview of SCIM), JLEO artefacts and the OWL representation of the JLEO (i.e. the jleo.owl file). Brief lists of the benefits of the session were:

- ✓ It served as a core forum for the researcher to brief the evaluators regarding the research questions and objectives.
- ✓ It provided a forum for the evaluators to clarify their role with the researcher and to ask any questions about the evaluation.
- ✓ It helped the researcher to demonstrate the prototype ontology developed by using SCIM.
- ✓ It provided a forum for the researcher to describe the evaluation matrix prior to distributing to the evaluators.
- ✓ It gave the researcher an opportunity to distribute the survey questions and supplementary documents to the evaluators.

6.2.2.2 Evaluation Questionnaire

A set of fifteen questions were circulated among the evaluators together with the matrix to be completed which compared the SCIM with other existing ontology development

methodologies. As illustrated in the Evaluation framework (Figure 60), the questions were prepared taking into account the five evaluation parameters which were significance, internal validity, external validity, objectivity and confirmability**.** Table 16 shows the mapping of the questions against the parameters. An open-ended question was provided at the end to allow evaluator to add any additional remarks which has not been included in the table.

Parameter #	**Questions**	**Qn #**
1	Do you feel that the existing ontology development methodologies are lacking the software engineering flavour to a considerable extent, though there are significant resemblances between Ontology Engineering and Software Engineering?	1
1, 2	Do you think the stage-based approach followed in the SCIM as the mode of development is an appropriate choice compared to other modes such as modular development and evolutionary prototype? Why?	3
1	To what extent does the hierarchy of components for the SCIM support the practitioners for faster ontology development?	4
1, 2	In your opinion, does the SCIM meet all the basic requirements of an ontology development methodology?	2
2	Assess the relevance of the stages of the SCIM and the proposed deliverables for each stage, within the context of large-scale ontology development.	6
1,3	Do you think the idea of using proven software engineering process models for ontology engineering will boost the process of large-scale ontology development, given that software development has a rich set of resemblances with ontology development? How?	5
3	In your opinion, does the suggested list of techniques to complete the activities of the SCIM support collaborative ontology development? Are there any modes for joint development?	7
3	To what extent does the SCIM encourage re-usability? Does SCIM have any provisions to integrate/accommodate existing ontologies?	8
3	In your opinion, does the hybrid model combining both linear and iterative process models support the developers for easier ontology development?	9
3	In your opinion, are the activities mapped to the different workflows of the SCIM appropriate? Please provide your comments to support your answer	10

4	Does the mechanism of the segregation of application specific concepts and domain specific concepts of the SCIM support the extent of application dependency?	11
4	In your opinion, to what extent has the components of the SCIM incorporated an Ontology Development Life Cycle (ODLC)?	12
5	What is the level of the coverage of activities and techniques employed in the SCIM?	13
5	Would you recommend/select the SCIM as a methodology for future ontology development projects? Why	14

Table 16: Mapping of survey questions against evaluation parameters

Questions have been designed in such a way that they give space for the evaluators to state their opinions in a measurable manner along with qualitative comments. Therefore, most of the questions offer scope for the evaluators to mark their Boolean mode Yes/No with subjective justifications. The questions have also been designed so that the answers contribute to the completion of the given matrix. A detailed analysis of their answers is described in Section 6.3.4.2.

6.2.2.3 Target group specifics

As illustrated in the evaluation framework, evaluators with experience in ontology engineering ideally from the education sector, were the target group chosen for the SCIM evaluation. The target group consisted of 10 members matching the selection criteria. 40% of the members had an advanced level of ontology-development experience in funded projects with an appropriate research profile. 20% of the members had considerable experience of ontology development for industry. 30% of the members were advanced level ontology practitioners with strong academic and research experience. One evaluator had an exceptional level of teaching experience in the domain chosen for the prototype development. A profile of the evaluators follows.

6.2.2.4 Profile of target group evaluators

Evaluators 1-3 were the co-principal investigators of a project funded by the Oman government entitled *"Ontology driven Decision Support System for detection and risk assessment of Hypertension in related diseases in Sultanate of Oman".* This project, valued at one and a quarter million pounds was funded by The Research Council of Oman (TRC) in

2015-2016. The evaluators were also involved in many international ontology-based projects as consultants/developers/mentors. Besides their development experience, their research profile also included publications on Ontology Engineering in highly-rated journals and international conference proceedings.

Evaluator 4 played a leading role in ontology-based projects which included: *A framework of Ontology based Ranking, Classification and Clustering of documents (2010)* and *Development of a domain ontology for Tourism (2014)*. In addition to these projects, the evaluator was also a team member for the *CLIA Project – UNL based search engine (2015).*

Evaluator 5 was affiliated to the Oman National Hydrographic Office as a knowledge-base developer.

Evaluator 6, was a senior IT engineer affiliated to the Ministry of Regional Municipalities and Water Resources (MRMWR) in the Sultanate of Oman and possessed strong ontology development experience.

Evaluators 5 and 6 were free-lance software developers who were also involved in the development of ontologies in other domains. Moreover, they had experience of the methodologies used for comparison in the SCIM evaluation.

Evaluator 7 had a PhD in text mining and machine learning. from an Indian University and also had various publications for this domain.

Evaluators 8 and 9 had a strong academic and research profile in ontology engineering and were involved in ontology-based projects. Both of them were pursuing a PhD in ontology engineering.

Evaluator 10 had more than 8 years of teaching experience in computer-science subjects and has specialised in Java-based modules, such as Java programming, Java network programming, Internet Programming and Software Application development. Evaluator 10 was a member of the 'Special Interest Group for Ontology and the Semantic Web' from Middle East College.

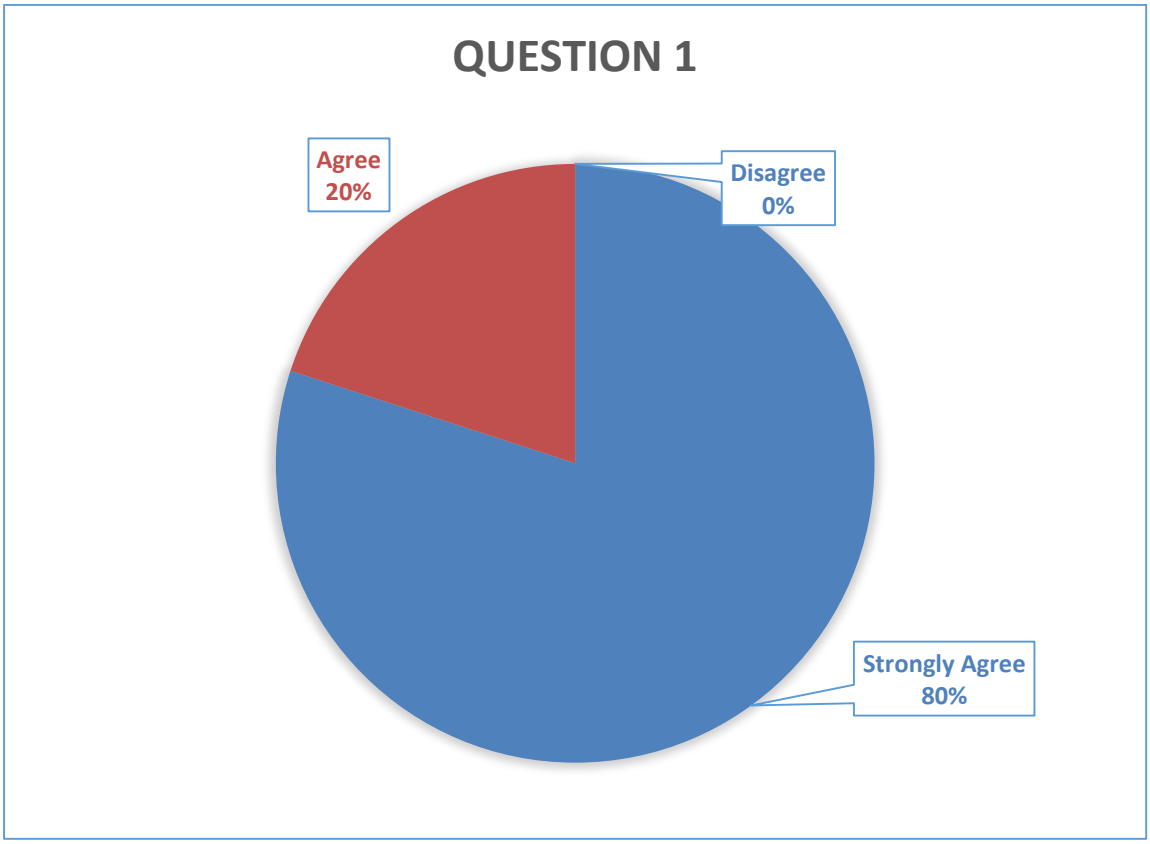
6.2.3 Evaluator's Answer Analysis

The answers to the questions given to the evaluators were collected by email and analysed by plotting the 14 questions on an Excel-based graph. The answers to the questions were grouped into two categories. The questions with YES/NO answers were grouped together in one category. The other category grouped the remaining questions under an appropriate title based

on the qualitative justification given by the evaluators. The answers from for each question have been used for plotting the graphs. In addition to that, the justifications given by the evaluators for the answers to each question have also been analysed. The analysis of the answers given for each question are presented below.

Question 1

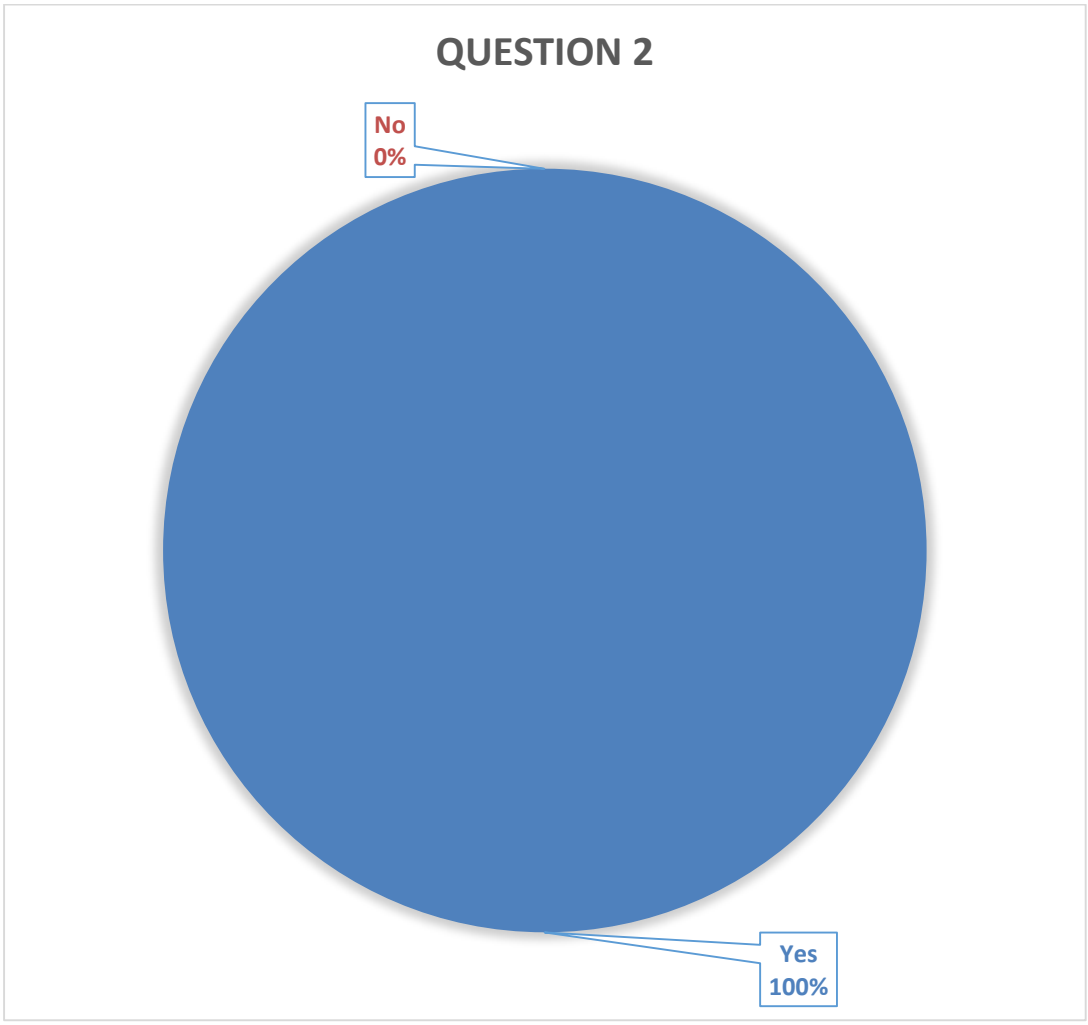
Do you feel that the existing ontology development methodologies are lacking the software engineering flavour to a considerable extent, though there are significant resemblances between Ontology Engineering and Software Engineering?



All the evaluators agreed that the existing ontology development methodologies were lacking the software engineering aspects. 80% of the evaluators strongly agreed that this gap should be bridged. In addition to their answers, evaluators had pointed out that only limited attempts have been made to use proven SE methodologies for ontology engineering. This shows the significance of the research explicitly and justifies the research gap that was addressed by the research.

Question 2

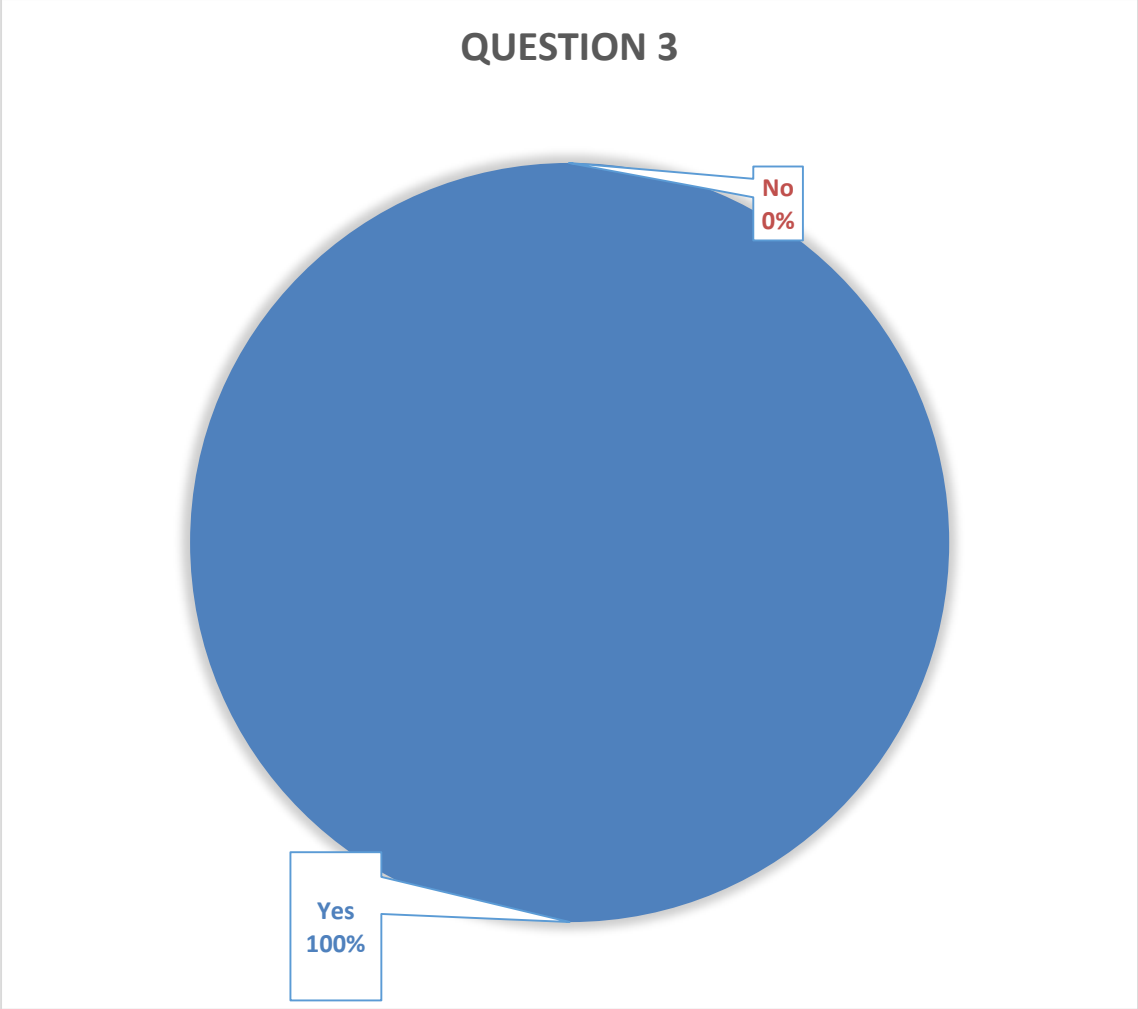
In your opinion, does the SCIM meet all the basic requirements of an ontology development methodology?



All the evaluators were of the opinion that the SCIM met all the basic requirements of an ontology development methodology. In fact all the evaluators supported their claim with enough justification. Instances of such justifications pointed that the SCIM not only met the basic requirements of a methodology, but was also well supported by a strong undepinning philosophy. For an engineering process like the proposed methodology, the availability of a strong philosophical approach further confirms the significance and internal validity of the research.

Question 3

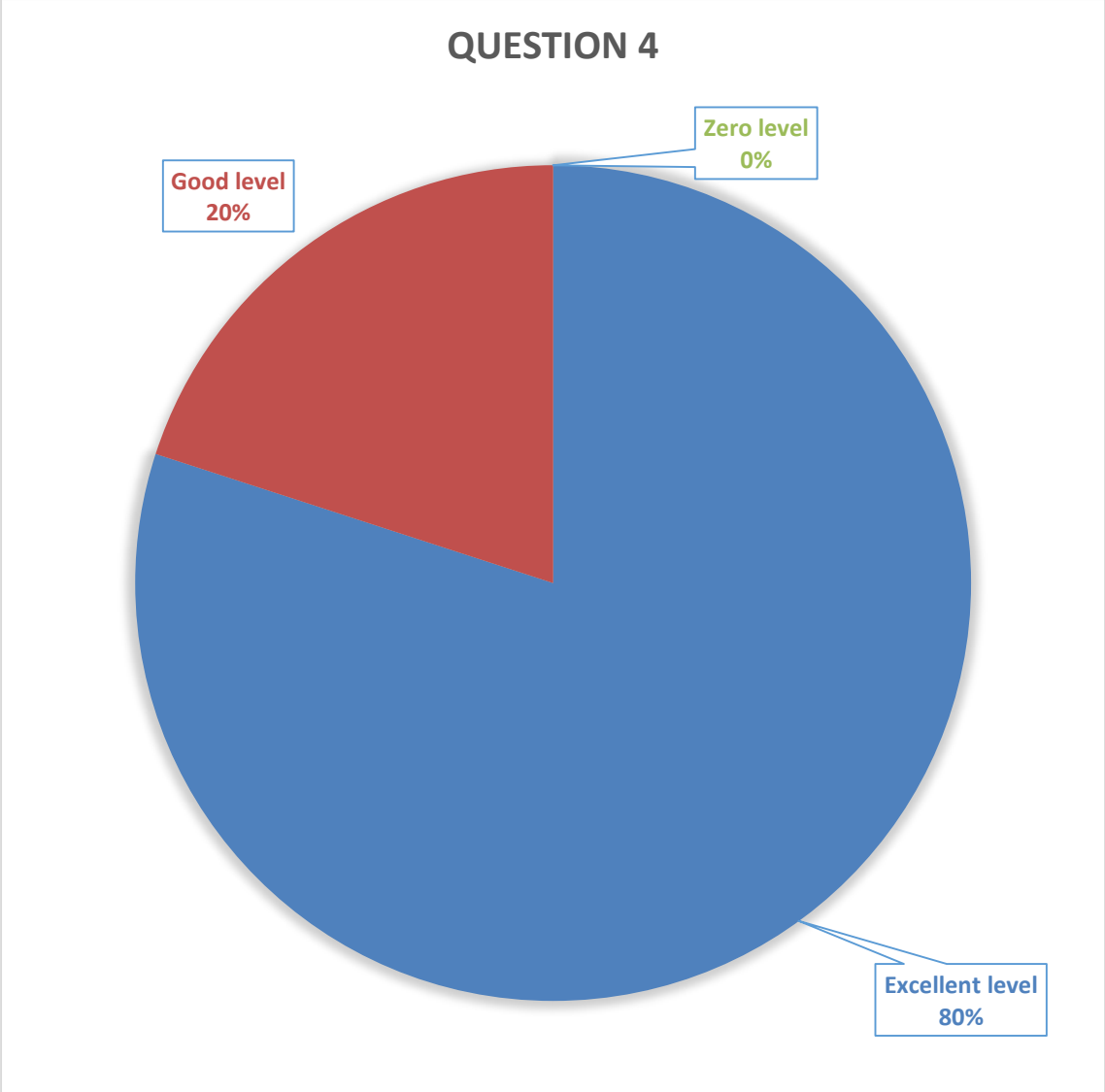
Do you think the stage-based approach followed in the SCIM as the mode of development is an appropriate choice compared to other modes such as modular development and the evolutionary prototype? Why?



All the evaluators agreed that the stage-based approach proposed by the SCIM was much better than the modular development and evolutionary prototype approach followed by the existing methodologies. Moreover, the stage-based supported generic ontology development unlike other modes more suited for the development of problem-specific ontologies. Further to that, some evaluators were of the opinion that the stage-based approach was more appropriate for modern ontology development to facilitate the quick integration with decision support systems. This result shows the significance of the linear model applied to the stages of the SCIM and supports the internal validity of the research.

Question 4

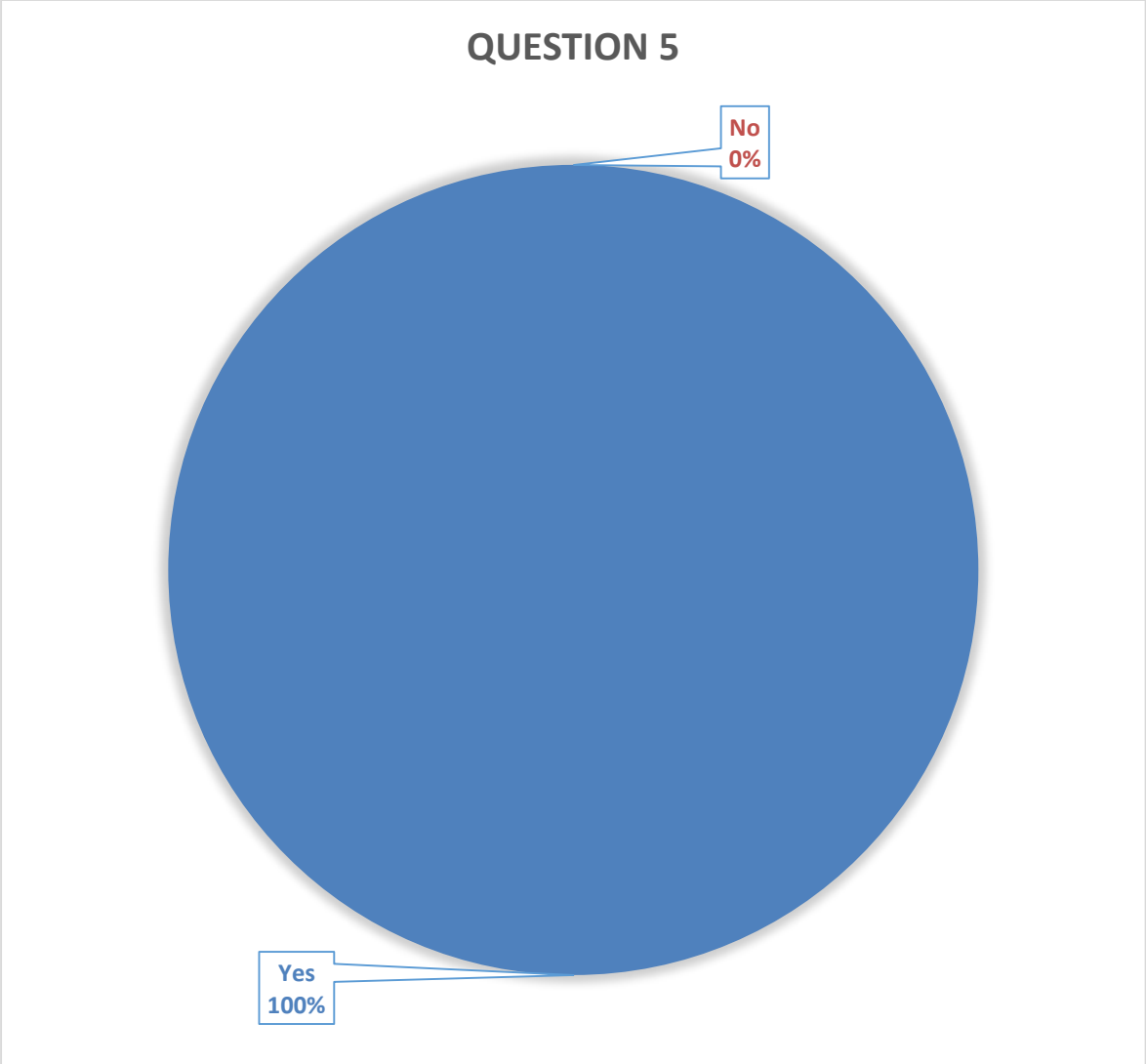
To what extent does the hierarchy of components for the SCIM support the practitioners for faster ontology development?



All the evaluators answered that the hierarchy of the components of the SCIM, gave at least good support to the practitioners for faster ontology development. 80% of the evaluators strongly agreed that SCIM had a very well defined hierarchy of components to help developers to achieve faster ontology development. Furthermore, the experts in their justification statements, explicitly stated that the well-defined hierarchy and unambiguous descriptions of the SCIM components accelerates the process of ontology development.

Question 5

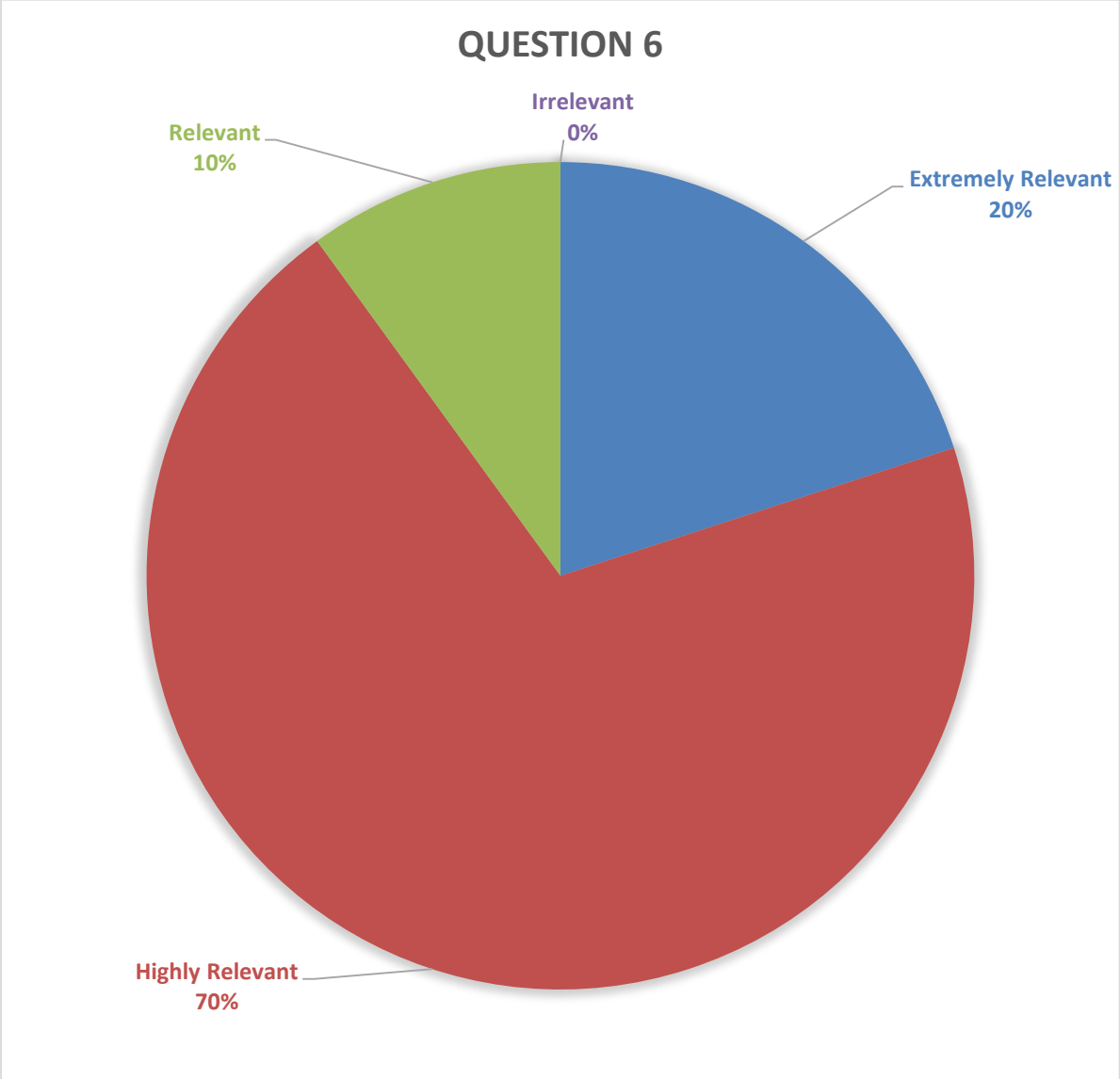
Do you think the idea of using proven software engineering process models for ontology engineering will boost the process of large-scale ontology development, given that software development has a rich set of resemblances with ontology development? How?



All the ten evaluators unanimously agreed that the proposed idea will boost the development process of large-scale ontology development. Evaluators pointed that by using software process models to ontology engineering, the learning curve for software practitioners to convert to ontology development is reduced considerably, accelerating the ontology-development process. This result demonstrated the significance and external validity of SCIM.

Question 6

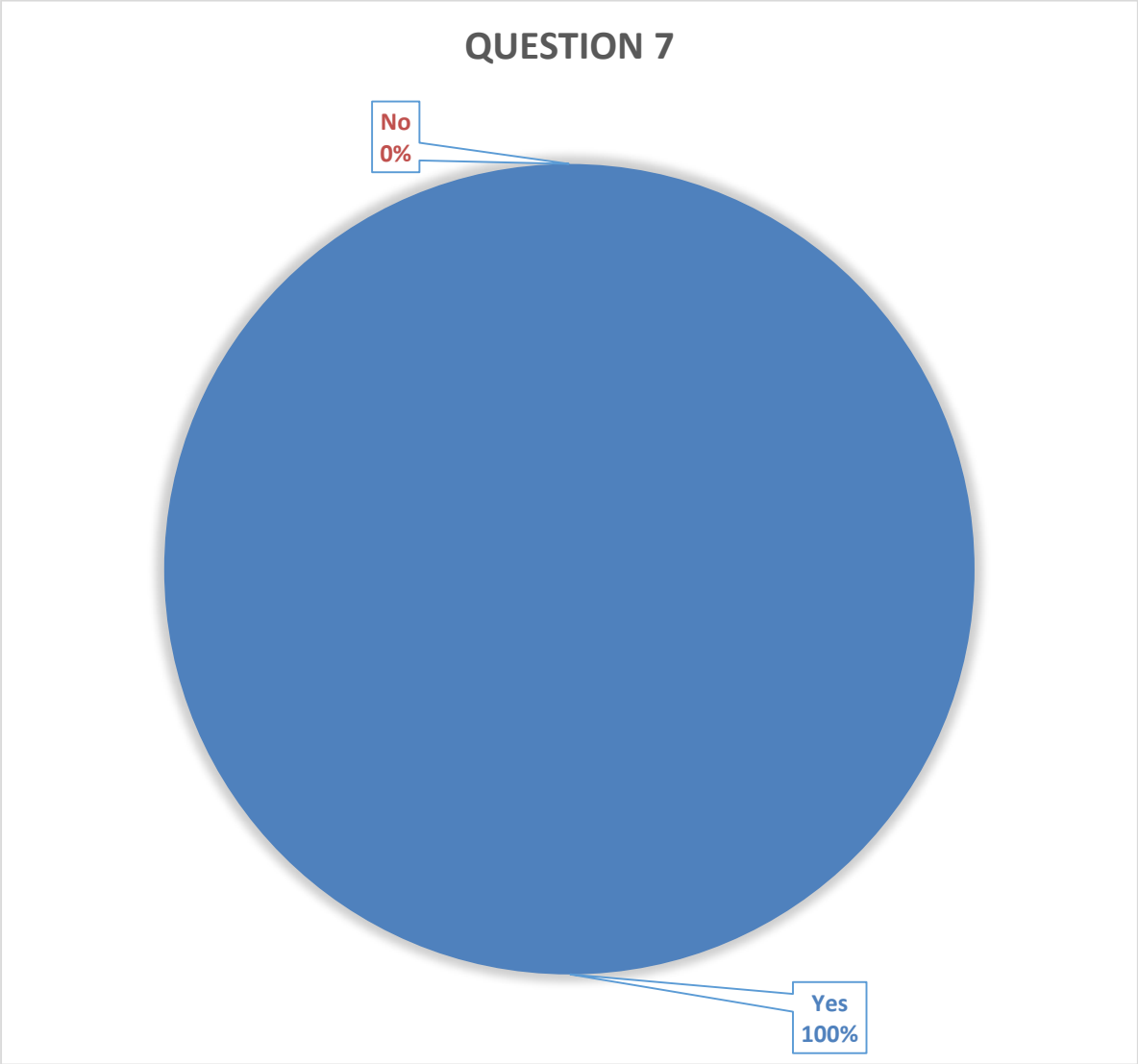
Assess the relevance of the stages of the SCIM development life cycle and the proposed deliverables for each stage, within the context of large-scale ontology development?



All the evaluators agreed on the relevance of the stages of SCIM. However, the level of assessment varies slightly. Evaluators made use of the artefacts supplied to them for the assessment of relevance. Since 90% of the evaluators gave the ranking of either extremely high relevance or high relevance it is evident that SCIM assures the internal validity significantly. 10 % of evaluators expressed minor concerns in their justification statements regarding deployment.

Question 7

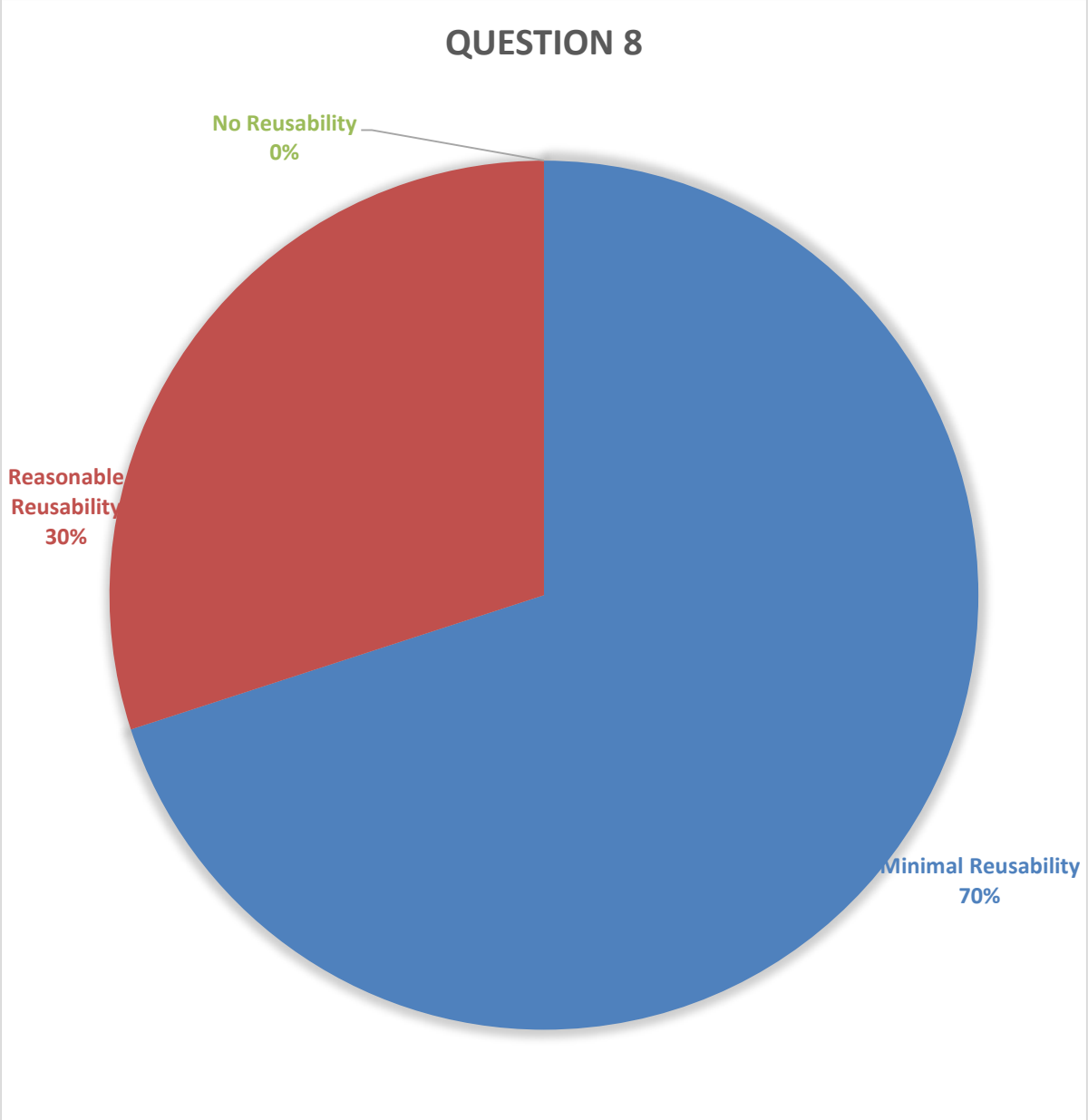
In your opinion, does the suggested list of techniques to complete the activities of the SCIM support collaborative ontology development? Are there any modes for joint development?



All the evaluators were of the same opinion that the techniques proposed by the SCIM supports collaborated ontology development. Evaluators pointed out explicitly that the techniques such as JOD and the use of the ODT were instances of collaborative development.

Question 8

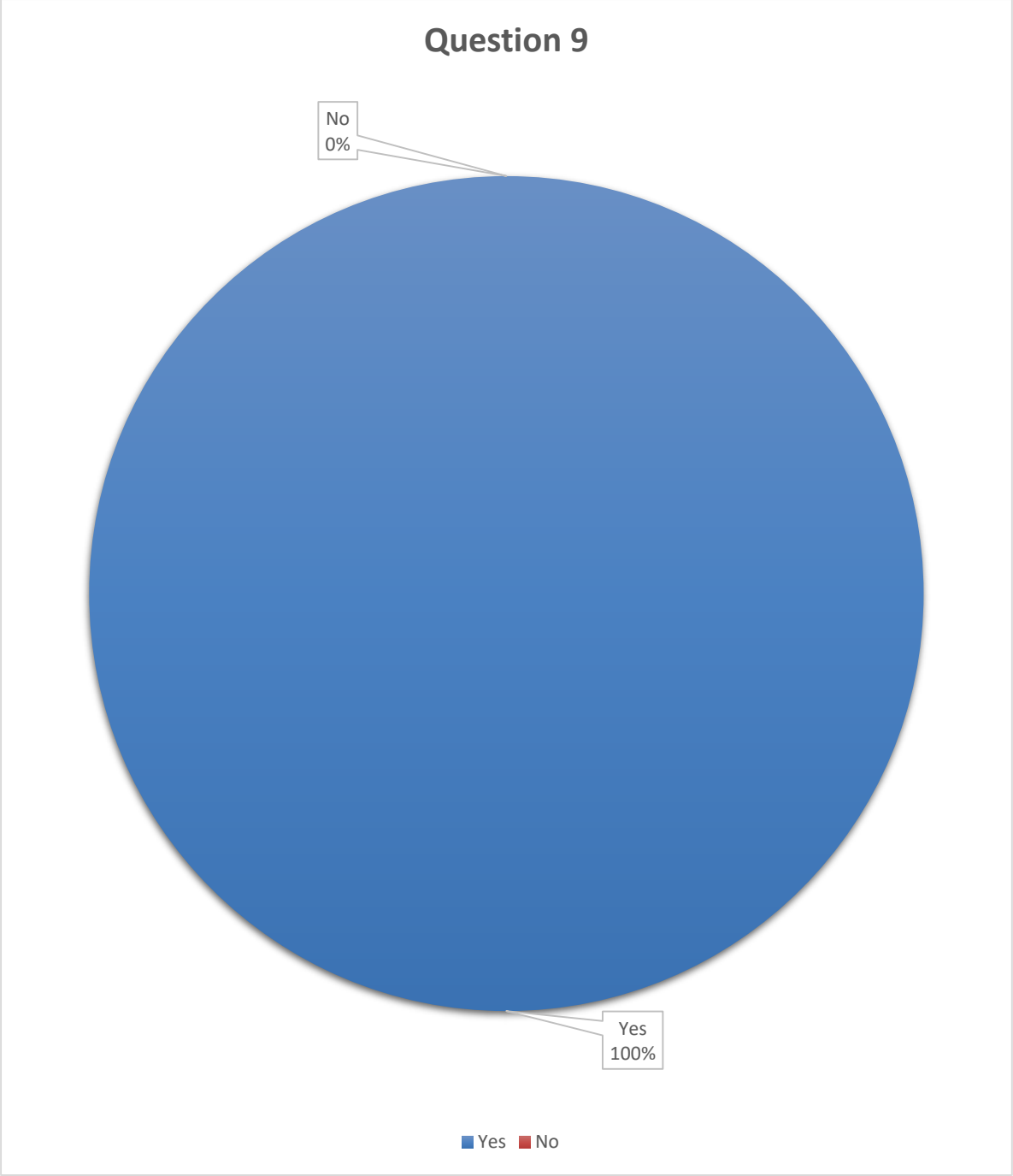
To what extent does SCIM encourage re-usability? Does SCIM have provisions to integrate/accommodate existing ontologies?



All the evaluators were of the opinion that SCIM significantly encouraged re-usability. 30% of the evaluators agreed SCIM supports the reusability. However, 70% of evaluators pointed out that the scope of reusability was minimal in SCIM. They suggested investigating the feasibility of incorporating a facility for the integration of other domain ontologies with SCIM.

Question 9

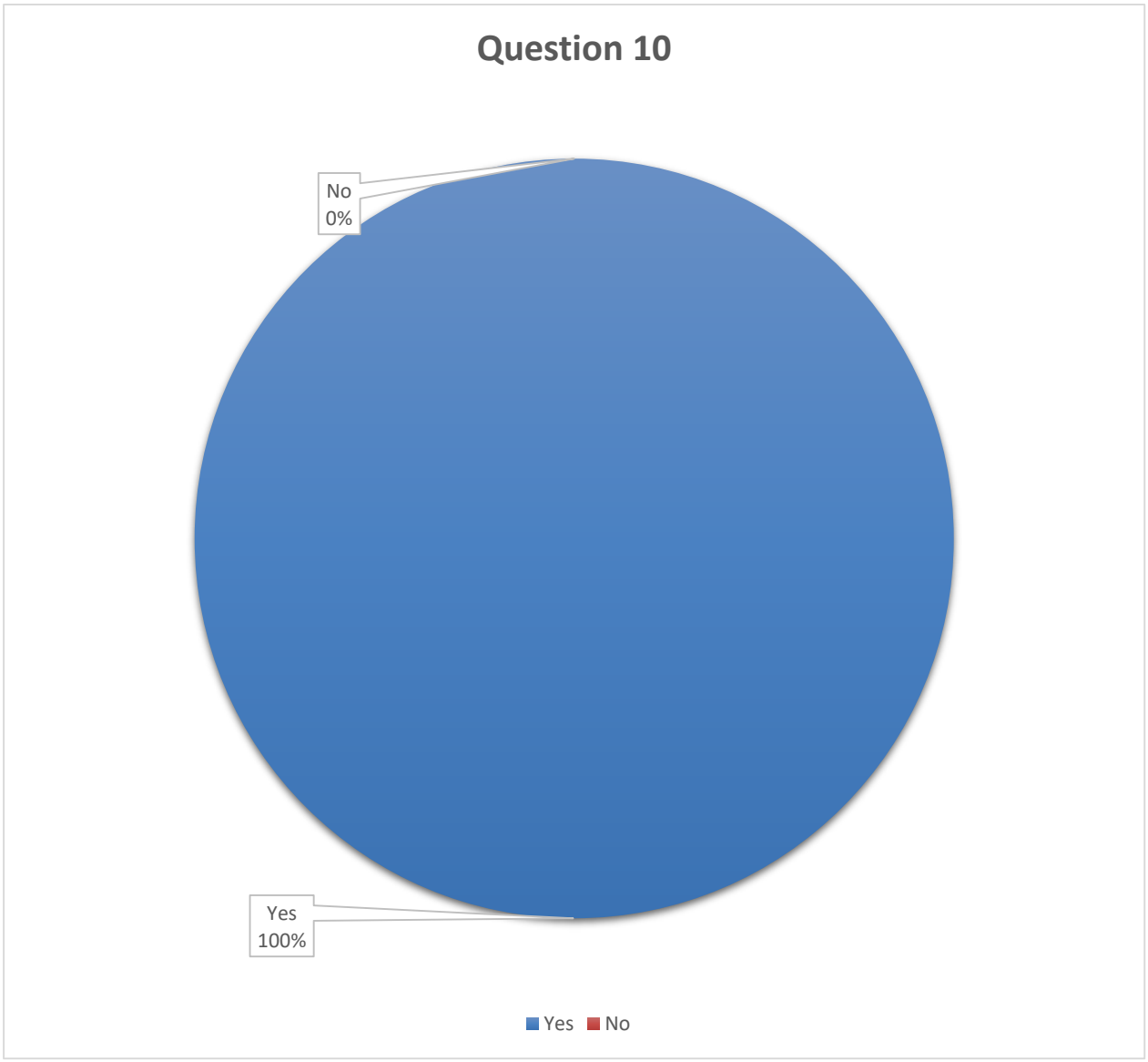
In your opinion, does the hybrid model of combining linear and Iterative process models support the developers for easier ontology development?



All the evaluators were of the opinion that the hybrid model of linear and Iterative process models applied in SCIM supports the developers for easier ontology development. This shows that the philosophy of SCIM (i.e. the amalgamation of linear and iterative approaches as a hybrid model) supports the developers for easier ontology development. Moreover it shows the external validity of SCIM is demonstrated by the use of other existing theories such as waterfall and RUP. Furthermore. Some of the evaluators suggested that the proposed model should mitigate risk considerably.

Question 10

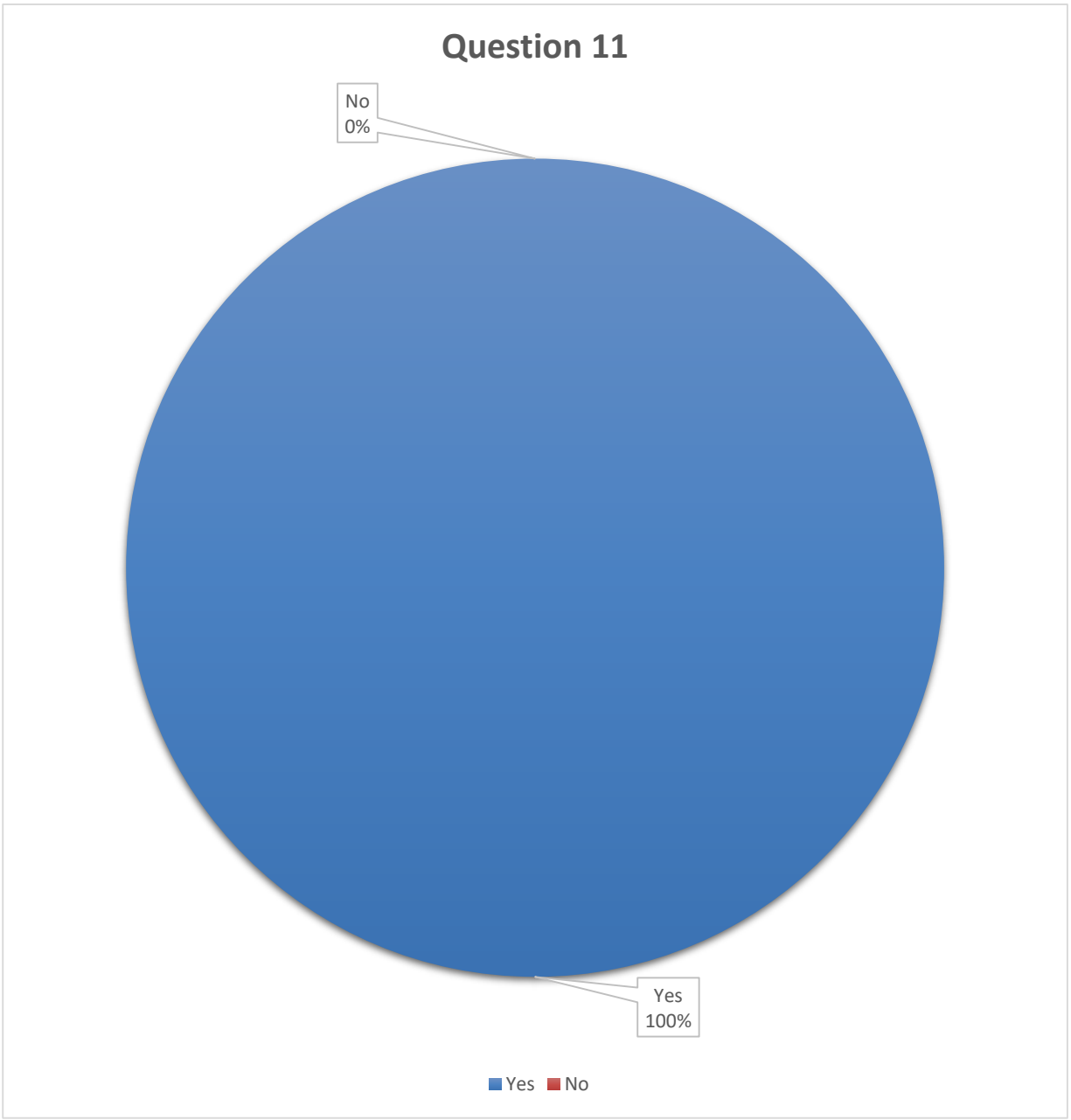
In your opinion, are the activities mapped to the different workflows of SCIM appropriate? Please provide comments to support your answer.



All the evaluators agreed that the activities were well mapped to the different workflows of SCIM ontology development. This shows a high level of conformance to the internal validity parameters set for SCIM.

Question 11

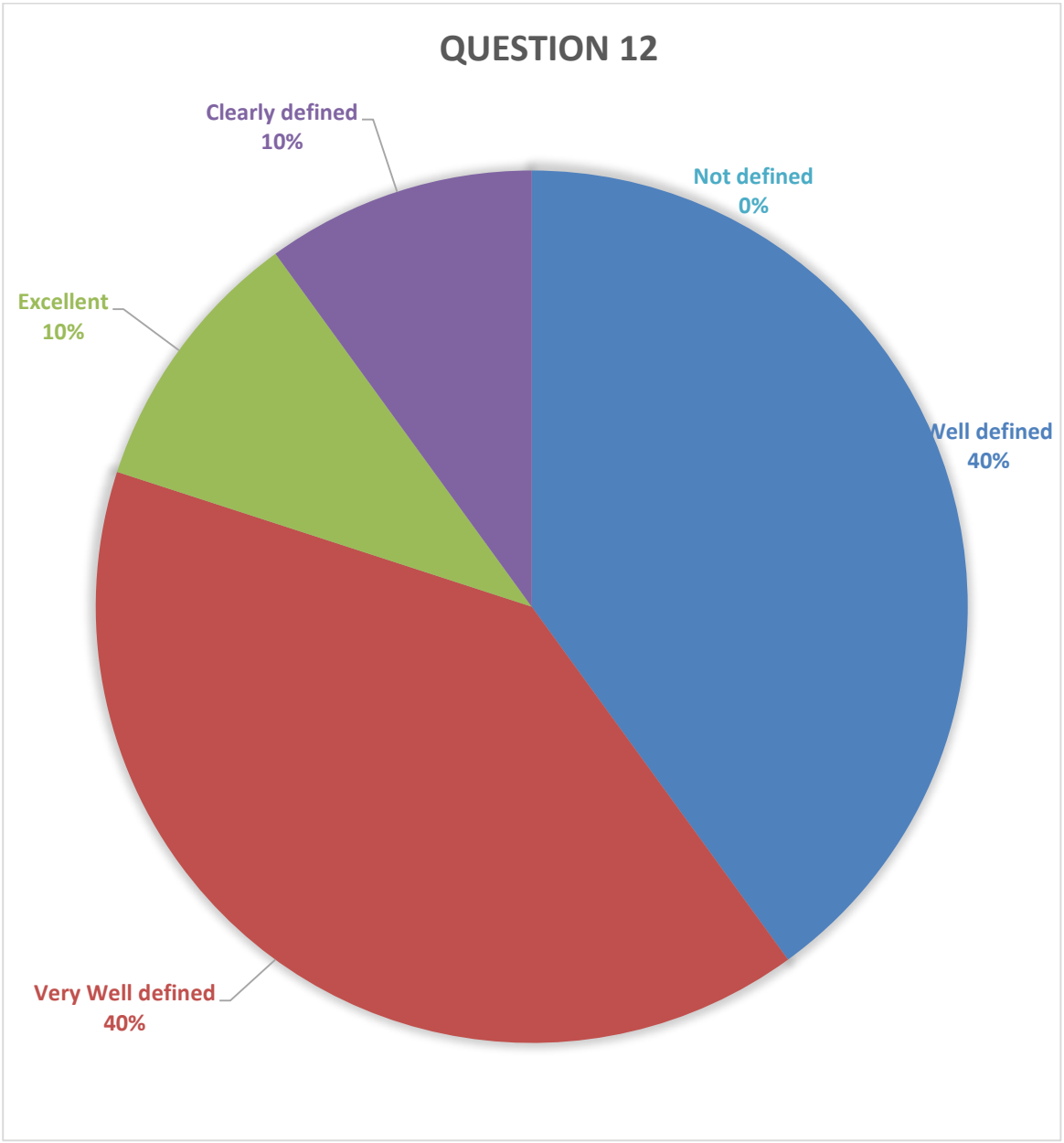
Does the mechanism of the segregation of application specific concepts and domain specific concepts of SCIM support application dependency?



All the evaluators agreed that SCIM significantly supports the extent of application independency which ensures reusability and interoperability showing the external validity with existing theories.

Question 12

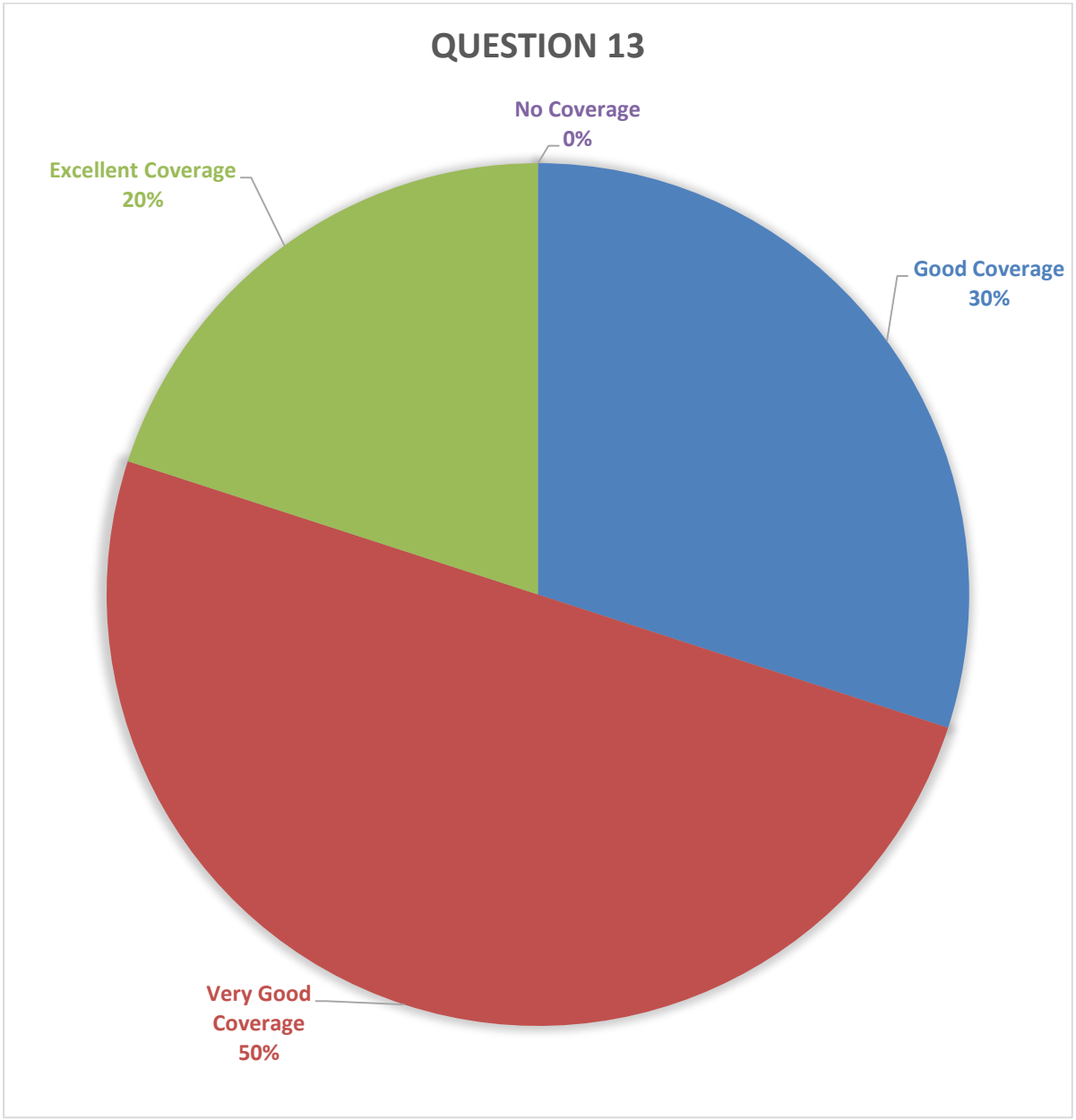
In your opinion, to what extent has SCIM a clearly defined Ontology Development Life Cycle (ODLC)?



All the evaluators were satisfied with the definition of ODLC for the SCIM. 90 % of the evaluators had pointed out that ODLC had either a well-defined, very well defined or excellently defined ODLC. In their justification statements, some evaluators mentioned that the ODLC of SCIM was much better than that of existing methodologies. This shows a clear accommodation of a high level of external validity.

Question 13

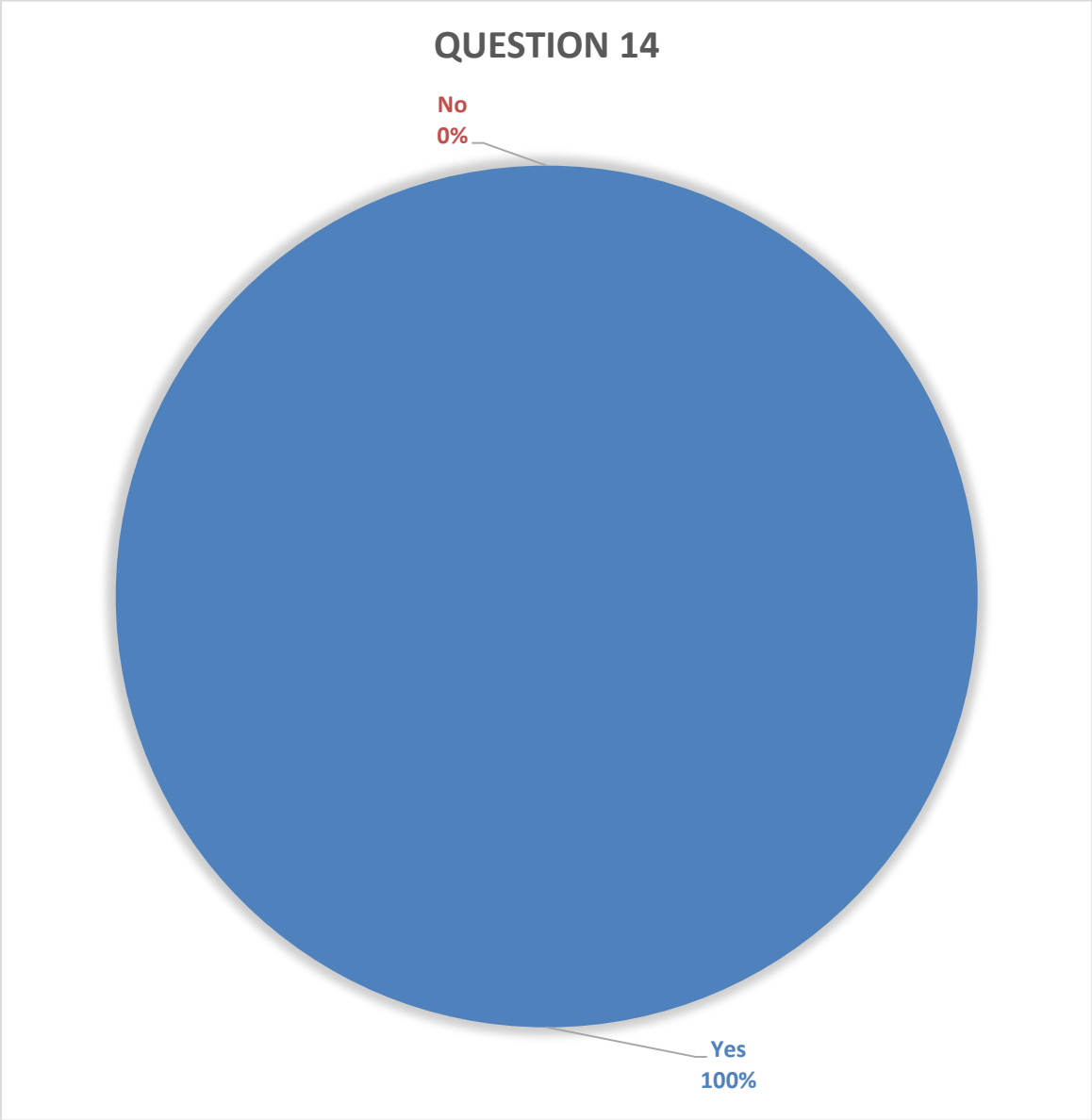
What is the level of the coverage of activities and techniques employed in SCIM?



All the evaluators agreed that the level of coverage of the employed activities and techniques in SCIM was either good or better with 20% of them giving a rating of excellent. The evaluation experts made use of the artefacts and prototype ontology for making their judgement which ensured the objectivity and confirmability of SCIM.

Question 14

Would you recommend/select SCIM as a methodology for future ontology development projects? Why?



All the experts were of the opinion that they would recommended the adoption of SCIM for future ontology development projects. However, all of them pointed out in their comments that they would recommend it to be used initially for small ontology development projects. This shows the applicability of SCIM for large scale ontology development projects, once its maturity has been proven in few small projects.

6.2.3.1 Consolidated Analysis

In addition to the question-wise analysis, a consolidated graph was plotted showing the answers given to each question and the number of evaluators who gave that answer. This is shown in Figure 60.

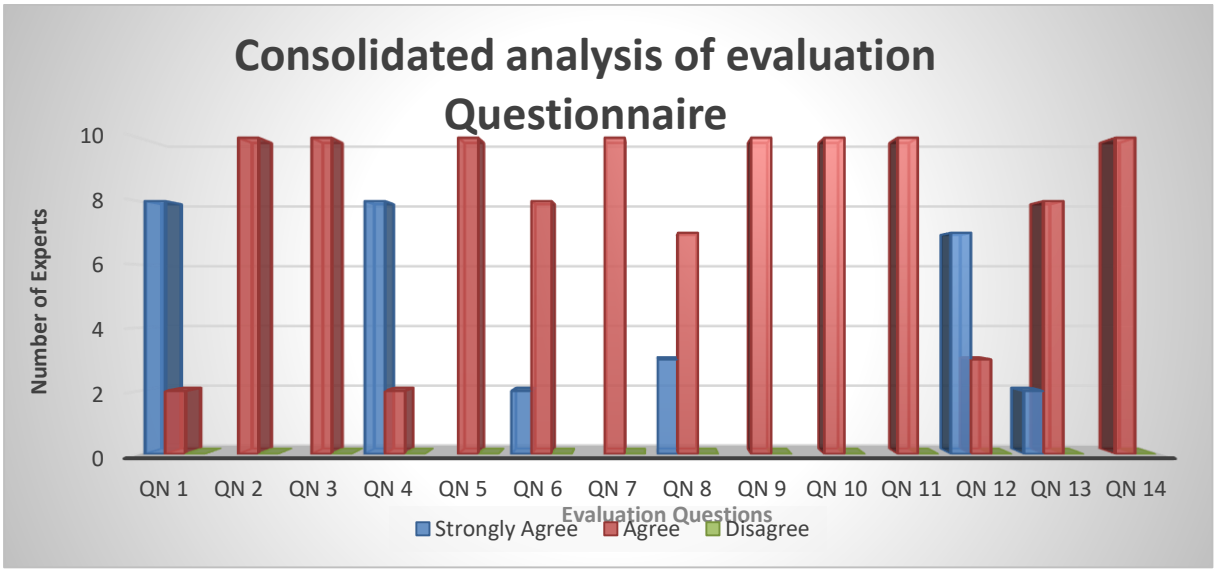


Figure 60: Consolidated graph of the answers of evaluators

As the second part of the evaluation a matrix was prepared which compared SCIM with ten of the most widely used ontology development methodologies. The existing methodologies listed in the matrix were obtained from the literature and are discussed in Chapter 4.2.1. The columns of the matrix specify the criteria for comparison according to the recommended features for an ontology development methodology resulting from the analysis of the literature. The recommended features are also discussed in the same chapter. Each evaluator was asked to mark the columns of the matrix for SCIM in the last line of the matrix. It was noted from the matrix filled out by each expert that the seven criteria supplied for comparison were well met by SCIM. Moreover, many of the limitations of the other methodologies had been overcome in the design of SCIM. The matrix is shown in table 17 with the last line highlighted in blue showing the consolidation of the responses of all the evaluators for SCIM.

Name of Methodology	Mode of development	Support for collaborative ontology development	Support for re usability	Support for interoper -ability	Extent of Application dependency	Ontology Life Cycle support	Coverage of employed methods and activities
Ushold and King (KEM)	Stage based	No	No	No	Application dependent	No	Limited coverage available for purpose identification , ontology building and evaluation.
Gruninger and Fox (TOVE)	Stage based	No	Yes	No	Application Semi-independent	No	Limited coverage available for informal specification, formulation of competency question…
METHONTOLOGY	Evolutionary prototype	No	Yes	No	Application independent	Yes	Sufficient coverage for specification, conceptualization, formalization , integration, implementation and maintenance
Ontoligua	Modular development	No	Yes	Yes	Application independent	No	Limited coverage on ontology development and integration.
Common KADs and KACTUS	Modular development	No	Yes	No	Application dependent	No	Limited coverage on ontology design and development
On-To-Knowledge	Evolutionary prototype	No	No	No	Application dependent	Yes	Limited coverage on ontology design and development
UPON	Evolutionary prototype	No	Yes	No	Application independent	Yes	Limited coverage on ontology design and development
XP.K	Evolutionary prototype	Yes	No	No	Application independent	No	Limited coverage on ontology development

EXPLODE	Evolutionary prototype	Yes	No	No	Application independent	Yes	Limited coverage on ontology development
RapidOWL	Evolutionary prototype	Yes	Yes	No	Application independent	No	Limited coverage on ontology development
SCIM	Stage based	Yes	Yes	Yes	Application independent	Yes	Very detailed coverage off ODLC methods and techniques.

Table 17: Consolidated matrix of SCIM

6.2 Analysis Conclusion

By conducting a survey of the selected evaluators in the field, it has been proved that the SCIM development has effectively accommodated the evaluation parameters required by the evaluation framework. It has been observed that the briefing session and the supplementary materials provided to the evaluators assisted them considerably for the evaluation. The framework of SCIM and the artefacts produced from the stages of the life cycle of the JLEO also added to the effectiveness of the evaluation. All the evaluators have examined the JLEO on their own workstations using the chosen ODE (i.e. protégé). Further to the evaluation, all the evaluators unanimously agreed on the viability of the proposed methodology for ontology development. The target group of evaluators agreed the significance and relevance of SCIM for the domain considered. The evaluators highlighted the availability of a well-defined Ontology Development Life Cycle, the hierarchy of the components of SCIM, the stage-based approach for development and the mapping of artefacts against the stages of the ontology development life cycle. However, they expressed minor concerns on the deployment and reusability of SCIM in regard to its use for large-scale commercial ontology development. Based on the evaluation findings, the evaluators concluded that the current version of SCIM can be used effectively for the development of small-scale ontology development. To summarise, SCIM can claim to be a software-centric methodology for ontology development which in some respects is the equivalent but in other respects better than existing ontology-development methodologies.

6.3 Chapter Summary

A detailed description of the evaluation mechanisms applied for SCIM evaluation was presented in this chapter. The evaluation framework was defined and the criteria for evaluation discussed. This was followed by an explanation of the process used for selecting the evaluators and the criteria used for selection. The results from the evaluation were then presented including individual results for each question that was given to the evaluators to answer. A graph of the consolidated results of the matrix also filled out by the evaluators was then presented.

The next chapter gives the conclusions to the research with suggestions for future work.

7. Conclusion and future work

'Our imagination is the only limit to what we can hope to have in the future.'
Charles F. Kettering

7.1 Chapter Overview

This chapter provides a brief description of the concluding marks in terms of knowledge contribution, future prospects and limitations of this research. Right from the identification of the research problem to evaluation, this research followed a customised Design Science approach as the research design. The research process has been defined in such a way to complement the research design which assists the researcher to achieve the objectives and propose solutions to the research problem. The interim findings of the research and parts of this research have been published in two international journals and in the proceedings of two International (ACM and IEEE) conferences. Since this research was exploratory, a detailed literature review was conducted which contributed significantly to the design of SCIM for ontology development. Section 7.2 describes the contribution of this research to the knowledge domain followed by a brief description on future research directions. The major limitations of this research are presented in Section 7.4 and this is followed by the reflections of the researcher in section 7.5.

7.2 Contributions to knowledge

In this thesis, the researcher addressed the problem of *how to develop a methodology for ontology development which bridges the gap between two complementary branches of engineering branches such as Software Engineering and Ontology Engineering*.

The main contribution of this research to the knowledge community is the proposal of a Software Centric Innovative Methodology (SCIM) for domain ontology development, The amalgamation of two proven software engineering process models as a hybrid model is the underpinning philosophy of SCIM which contributed significantly to the bridging of the gap between Software Engineering and Ontology Engineering by the combination of Linear waterfall and Iterative RUP process models. This philosophy makes possible the use of software developers familiar with these models for ontology development. SCIM has introduced software models as a novel approach for the intermediate representation of model elements. This has been done mainly by extending UML models for ontology modelling. Specifically, this research has proposed a UML class model for the representation of the

conceptual model of the ontology, with the domain concepts and the semantic structural relationships between them. This contribution could be exploited in other ontology development methodologies with certain constraints, making possible the potential application to large-scale ontology development.

The second contribution is an ontology development life cycle (ODLC) with a clear hierarchy of the components consisting of stages, workflows, activities and techniques which is not provided by existing ontology development methodologies. The ODLC proposes a linear flow among its four stages and an iterative flow among the activities within its five workflows. Unique templates have been designed as artefacts for the stages and activities and the Ontology Requirement Specification Document (ORSD) was a new document introduced at the planning stage. Two new workflows were introduced for domain analysis and requirements analysis. The full coverage of activities and techniques is another novelty with recommendations for the use of agile techniques and a joint ontology development (JOD) which was proposed for the domain analysis.

Another contribution of this research is the prototype development of an educational ontology named as Java Learning Educational Ontology (JLEO) based on three different types of Java programming. The various components of SCIM have been applied in the prototype development resulting in the development of an ontology represented in formal language (OWL). A new effective hybrid evaluation framework combining the approaches of both Goal Question Metrics (GQM) with the five suggested evaluation parameters from the work of Burstein and Gregor `(i.e. s*ignificance of the study, internal validity, external validity, objectivity* and *confirmability)* was used for the methodology evaluation .

A further contribution is the customised design science approach used for the research methodology which could be used by other researchers for exploratory research in the future.

Finally, the body of knowledge has already been increased as a result of the work on this thesis by the publishing of five research papers which have been included in AppendixC.

7.3 Discussion on future work

There are several suggestions for future research that have emerged from this study. They are as follows:

The prototype ontology developed as a proof of concept for the SCIM could be extended to a knowledge-based system. Apart from the three variations of Java programming used for the prototype other Java software components such as servlets and Java Server Faces (JSF) streams could be added.

Larger enhancements of SCIM and the application on multiple domains were outside the scope of this research. Future work could include the application of SCIM to the domains of real-world problems in which the knowledge community is geographically distributed and the output should represent concepts from a crowd. For example, domains like e-health where thousands of domain experts hold different views on the topic.

Another future direction of this research is the application of SCIM simultaneously to multiple domains as a further test of its viability. Mechanisms to strengthen the reusability of SCIM could also be incorporated into future work.

7.4 Limitations

The limitations of the research which should be considered when assessing the value of the findings of this research or its potential for generalised application are discussed in the following paragraphs.

Though many methodologies were available, and considering the time available for the review of the literature and the respective analysis, the researcher selected ten of the most commonly used existing ontology development methodologies for detailed analysis.

For the application of SCIM, an academic domain was chosen because of the ease of data collection by the researcher. The methodology may not be able to be generalised for ontology development in different domains without special consideration for the differences of the domain.

Apart from the use of offline and online Java learning materials Java experts from MEC were used for the concept collection for the domain. A wider selection of experts could have suggested more or modified concepts although there was generally widespread agreement on over the target group on the concepts that were included.

The proof of concept for the SCIM was demonstrated by developing an ontology for only three variations of Java programming modules so some amendments may be required to apply the methodology for teaching all types of Java modules. Although, the generic nature of Java means that the amendments would probably be minimal.

7.5 Researcher's reflection

As the author of the thesis, I can confidently state that this research wouldn't have seen light of day without an extensive literature review. Since the research problem was only finalised after tuning, a set of realistic research objectives and research questions defined provided a smooth progression for the research journey.

By conducting the extensive literature review, I gained the necessary knowledge to assist me in proposing the conceptual framework of SCIM. Although I came across many ontology development methodologies and their application on various domains as part of the literature review, it has been noted that each one of them was developed to serve a specific purpose. As a researcher, I took a long time to completely understand the existing ontology development methodologies and their components and to make a comparison between them based on a set of defined parameters. This was one of the most challenging parts of this research which I managed to overcome by relying more on secondary resources and case studies. I had a very cooperative and understanding Director of Studies who was always there for me to clarify my doubts and to take me in the right direction. My supervisory panel members pitched in with their support as and when was needed.

Learning OWL and Protégé was another memorable phase of the study. The user friendliness of Protégé reduced the time span for prototype development. The completion of the milestone deliverables belonging to the different stages of SCIM encouraged me to proceed further with enthusiasm. The finalisation of domain concepts and semantic relationships between them was yet another interesting part of this work.

I was honored to participate in five international conferences as present my papers to other researchers. I was delighted to get good acceptance on the various parts of my research work from the international research community. That experience made me more confident and gave me the ability to explain and justify my ideas in front of a large audience. Last, but not least, my next objective is to continue my research on the ontology domain and continue to contribute significantly to the body of knowledge.

REFERENCES

Ahmed, Z. and Gerhard, D. (2007) 'How Does Ontology Contribute in Semantic Web Development?'.

Alatrish, E. (2013) 'Comparison Some of Ontology Editors', *Management Information Systems*, vol. 8, no. 2, April.

Antoniou, G. and Harmelen, (2004) 'Handbook on Ontologies', in *Web Ontology Language: OWL*, Springer.

Avison, D. and Fitzgerald, G. (2003) *Information systems development: methodologies, techniques and tools.* 3rd edition Maidenhead, UK, : McGraw Hill

Babbie, Earl R. The Practice of Social Research*. 12th ed. Belmont, CA: Wadsworth Cengage, 2010; Muijs, Daniel.* Doing Quantitative Research in Education with SPSS*. 2nd edition. London: SAGE Publications, 2010, [Online], Available* http://libguides.usc.edu/writingguide/quantitative [11 Auguest 2016].

Bandeira, J., Bittencourt, Patricia Es, I., Espinheira, and Isotani, S. (2017) 'FOCA: A Methodology for Ontology Evaluation', 2 September, Available: https://arxiv.org/abs/1612.03353 [1 June 2018].

Baonza, M.C.S.d.F. (2010) *NeOn Methodology for Building Ontology Networks: Ontology Specification Scheduling and Reuse*.

Software Modeling and Measurement: The Goal/Question/Metric Paradigm, September(1992), [Online], Available: http://hdl.handle.net/1903/7538. [January 2018]

Beck, K. (2000) *Extreme programming explained: embrace change.* Boston,USA : Addison-Wesley Longman Publishing

Berners-Lee, T. (2006) *Linked Data - Design Issues*, 23 July, [Online], Available: http://www.w3.org/ [December 2015].

Bizer, C., Heath, T. and Berners-Lee, T. (2009) 'Linked Data - the story so far', *International Journal on Semantic Web and Information Systems,* vol. 5, no. 3, pp. 1-22.

Borgo, S. (2004) *Classifying (medical) ontologies. Tutorial for the Ontology Workshop at the Semantic Mining Summer School*.

Brickley, D. and Guha, R.V. (2001) *Resource Description Framework (RDF) Schema Specification*, [Online], Available: https://www.w3.org/TR/WD-rdf-schema/ [May 2016].

Bruijn, J.d. (2003) *Digital Enterprise Research Institute*, 30 October, [Online], Available: http://www.deri.org/fileadmin/documents/DERI-TR-2003-10-29.pdf [12 September 2015].

Brunnlieb, M. and Holzer, S. (2016) 'Language-aware XML Aggregation', XML London 2016 Conference, London, 38-49.

Brusa, G., Caliusco, M.L. and Chiotti, O. (2006) 'A process for building a domain ontology: an experience in developing a government budgetary ontology', AOW '06 Proceedings of the second Australasian workshop on Advances in ontologies, Hobart, Australia, 7-15.

Burieson, C. (2007) *Semantic Focus, All about the Semantic Web*, October, [Online], Available: http://www.semanticfocus.com/blog/entry/title/introduction-to-the-semantic-web-vision-and-technologies-part-2-foundations/ [August 2015].

Burstein, F. and Gregor, S. (1999) 'The Systems Development or Engineering Approach to Research in Information Systems: An Action Research Perspective', 10th Australasian Conference on Information Systems, Wellington, NZ, 122-134.

Cai, J., Eske, V. and Wang, X. (2003) *Semantic Web & Ontologies*, 23 October, [Online], Available:http://www.mpi-inf.mpg.de/departments/d5/teaching/ss03/xml-seminar/talks/CaiEskeWang.pdf [16 Jun 2015].

Calero, C., Ruiz, F. and Piattini, M. (ed.) (2006) *Ontologies for Software Engineering and Software Technology.* 1st edition Springer-Verlag Berlin Heidelberg

Cardoso, J. and Escórcio , A.L.N. (2007) 'Editing Tools for Ontology Construction ', *Semantic Web Services:*, March, pp. 1-27.

Carlos Blanco, J.L.E.F.-M.R.V.-G.A.T. (2011) 'Basis for an integrated security ontology according to a systematic review of existing proposals', *Computer Standards & Interfaces*, vol. 33, no. 4, pp. 372-388.

Ceccaroni, L. and Kendall, E. (2003) 'A Semantically-Rich Graphical Environment for Collaborative Ontology Development in Agentcities', Barcelona, Spain.

Chandrasekaran, B., Josephson, R. and Richard, B. (1999) 'What are ontologies, and why do we need them', *IEEE Intelligent Systems and their Applications*, vol. 14, no. 1, Feb, pp. 20-26.

Chaware, S. and Rao, S. (2010) 'Integrated Approach to Ontology Development Methodology with Casestudy', *International Journal of Database Management Systems*, vol. 2, no. 3, August, pp. 13-19.

Chen, H., Finin, T. and Joshi, A. (2003) 'An ontology for context-aware pervasive computing environments', *Knowledge Engineering*, pp. 197-207.

Christain B, T.H.T.B.-L. (2010) 'Linked Data-The story so far', *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, June, pp. 1-22.

Cimiano , and VÄolker, (2005) *Text2Onto: A Framework for Ontology Learning and Data-driven Change Discovery*, Lecture Notes in Computer Science. 3513

Colomb, R., Raymond, K., Hart, L., Emery,R., Welty, C., Xie, T., and Kendall, E. (2006) 'The Object

Management Group Ontology Definition Metamodel', *Springer*, February.

Corcho, O., Fernandez-Lopez, M. and Gomez-Perez, A. (2003) 'Methodologies, tools and languages for building. Where is their meeting point?', *Data and Knowledge Engineering*, vol. 46, pp. 41-64.

Corcho, O., Fernández-López, M., Gómez-Pérez, A. and López-Cima, A. (2005) 'Building Legal Ontologies with METHONTOLOGY and WebODE', in Benjamins, Richard, Selic, Bran and Gangemi (ed.) *Law and the Semantic Web,Lecture notes in Computer Science*, Springer Berlin Heidelberg.

Cover, R. (2002) *Ontology Interchange Language (OIL)*, 31 October, [Online], Available: http://xml.coverpages.org/oil.html [July 2015].

Cranefield, S. (2001) *UML and the Semantic Web*, [Online], Available: http://hdl.handle.net/10523/1005 [May 2015].

DAML Ontology Library (2004), April, [Online], Available: http://www.daml.org/ontologies/ [February 2015].

De Nicola, A., Missikoff, M. and Navigli, R. (2008) 'A Software engineering approach to ontology building', *Elsevier*, vol. 34, July, pp. 258-275.

Denaux, R., Dolbear, C., Hart, G., Dimitrovaa, V. and Cohna, G. (2011) 'Supporting domain experts to construct conceptual ontologies: A holistic approach', *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 2, July, pp. 113-127.

Denzin and Lincoln, Y., (2005) *The SAGE handbook of qualitative research.* 3rd edition Thousand Oaks : Sage Publications

'Design Science in Education', *MIS Quarterly* (March 2004), pp. 75-105.

Dicheva, D. (2008) 'Ontologies and Semantic Web for E-Learning', in Adelsberger, Heimo H, Kinshuk, Pawlowski and Ma, J. (ed.) *Handbook on Information Technologies for Education and Training*, Berlin, Heidelberg: Springer.

Ding L., Kolari P., Ding Z., Avancha S. (2007) Using Ontologies in the Semantic Web: A Survey. In: Sharman R., Kishore R., Ramesh R. (eds) Ontologies. Integrated Series in Information Systems, vol 14. Springer, Boston, MA

Donald Bell (2003) *An introduction to the Unified Modeling Language*, 15 June, [Online], Available: https://www.ibm.com/developerworks/rational/library/769.html [June 2015].

Dursun, E. (2015) 'The relationship between organizational trust, organizational support and organizational commitment'', *African journal of Business Management*, vol. 9, no. 4, February, pp. 134-156.

ElenaTudoroiu, Cretu, and Paquet, J. (2009) 'Investigations using the Rational Unified Process (RUP) Diagrams for Software Process Modeling', International Multiconference on Computer Science and Information Technology, 523-530.

Enterprise Integration Laboratory – EIL (2002), [Online], Available: http://www.eil.utoronto.ca/theory/enterprise-modelling/ [November 2015].

Farquhar, A. Fikes, R. and Rice, J. (1997) 'The Ontolingua Server: a Tool for Collaborative Ontology Construction', *International Journal of Human-Computer Studies*, vol. 46, no. 6, June, pp. 707-727.

Fernández-López, M., Gómez-Pérez, A. and Juristo, N. (1997) 'METHONTOLOGY: From Ontological Art Towards Ontological Engineering', Ontological Engineering AAAI-97 Spring Symposium Series, California, 33-40.

Fonseca, F., Câmara, G. and Davis Jr., C., (2003) 'Bridging Ontologies and Conceptual Schemas in Geographic Applications Development', *GeoInformatica*, vol. 7, no. 4, January, pp. 355-378.

Fortuna, B., Grobelink, M. and Maldenic, D. (2005) 'Semi-automatic Construction of Topic Ontologies', Joint International Workshops, EWMF 2005 and KDO 2005, Portugal, 121-131.

Funk and Wagnalls (1985) *Standard Desk Dictionary.* Harper & Row

Gahleitner, E., Behrendt, W., Palkoska, and Weippl , E. (2005) 'On Cooperatively Creating Dynamic Ontologies', ACM, Salzburg, Austri, 208-210.

Ganapathi, G., Lourdsamy, L. and Rajaram, V. (2011) 'Towards Ontology Development for Teaching Programming Language', World Congress on Engineering, Lodon, UK.

Gangem, A., Guarino, N., Masolo, C., Oltramari, A. and Schneider, L. (2002) 'Sweetening Ontologies with DOLCE', Springer, Berlin, Heidelberg, 166-181.

Gaševic, D., Djuric, D. and Devedžic, V. (2009) *Model Driven Architecture and Ontology Development*, Springer.

Get Research Design Assignment Help Now, [Online], Available: http://www.transtutors.com/homework-help/management/marketing/market-research-process/research-design/ [November 2016].

Gilbert, J.&.T.T. (1998) 'An Examination of Organizational Trust Antecedents ', *Public Personnel Management*, vol. 27, no. 3, p. 321–338.

Gómez-Pérez, A. and Corcho, O. (2002) *Ontology languages for the semantic web*, [Online], Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.7574&rep=rep1&type=pdf [June 2015].

Gómez-Pérez, A. and Suárez-Figueroa, M.C. (n.d) 'NeOn Methodology: Scenarios for Building Networks of Ontologies', 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008), Italy.

Gómez-Pérez, A., González, R. and Lama , M. (2004) 'ODE SWS: a framework for designing and composing semantic Web services', *IEEE Intelligent Systems* , vol. 19, no. 4, July-Aug, pp. 24-31.

Gregor, S. and Hevner, R.A. (2013) 'Positioning and Presenting Design Science Research for

Maximum Impact', *Management Information Systems Quarterly*, pp. 337-355.

Group, W.O.W. (2012) *OWL 2 Web Ontology Language Document Overview (Second Edition)*, December, [Online], Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/ [June 2016].

Gruber, T. (1993) *A Translation Approach to Portable Ontology Specifications*, April, [Online], Available: http://tomgruber.org/writing/ontolingua-kaj-1993.htm [March 2015].

Gruber, T.R. (1994) 'Ontolingua: A Mechanism to Support Portable Ontologies'.

Gruninger, M. and Fox, M.S. (1995) 'Methodologiy for the Design and Evaluation of Ontologies', in *IJCAI-95 Workshop on Basic Ontological issues in Knowledge Sharing*, Montreal.

Gruninger, M. and Lee, J. (2002) 'Ontology-Applications and Design', *Communications of the ACM*, vol. 45, no. 2, pp. 39-41.

Grunninger, M. and Fox, S.M. (1995) 'Methodology for the design and evaluation of ontologies', International Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI95), Montreal, 1-10.

Guarino, N., Staab, S. and Oberle, D. (2009) 'What Is an Ontology', in *Handbook on Ontologies*.

Guizzardi, G., Herre, H. and Wagner, G. (2002) 'Towards Ontological Foundations for UML', 1st International Conf. on Ontologies, Databases and Applications of Semantics (ODBASE 2002), Irvine, CA,USA.

Guyot, J., Falquet, G. and Teller, J. (2010) 'Incremental development of a shared Urban ontology: The urbamet experience', *Journal of of Information technology in Construction*, vol. 15, pp. 132-139.

Harper Collins (2005) *Collins Discovery Encyclopedia.* 1st edition HarperCollins Publishers Available: http://encyclopedia2.thefreedictionary.com/Ontology+(philosophy) [4 Jun 2013]

Hartog, D.N.D., Shippers, M.C. and Koopman, P.L. (2002) 'the impact of leader behaviour on trust in management and co-workers', *SA Journal of Industrial Psychology*, vol. 28, no. 4, pp. 29-34.

Horridge, M. (ed.) (2011) *A Practical Guide To Building OWL Ontologies Using Proteige 4 and CO-ODE Tools*, 13th edition, Manchester M13 9PL: The University Of Manchester.

Hristozova, and Sterling, L. (n.d) *An eXtreme method for developing lightweight ontologies*.

Ibrahim, A.A. and Ataelfadiel, A.M. (2017) 'Ontology Life Cycle: A Survey on the Ontology and its Development Steps', *International Journal of Science and Research (IJSR)*, vol. 6, no. 8, August, pp. 1352-1357.

'IEEE Standard Glossary of Software Engineering', *IEEE Xplore Digital Libray*, December 1990, pp. 1-84.

IEEE Standard Glossary of Software Engineering Terminology-Description (1990), [Online],

Available: http://ieeexplore.ieee.org/document/159342/ [Auguest 2015].

Ig Ibert Bittencourt, E.C.M.S.E.S. (2009) 'A computational model for developing semantic web-based educational systems', *Elsevier, Knowledge-Based Systems*, vol. 22, no. 4, May, pp. 302-315.

Iqbal, R., Azmi Murad, M.A., Mustapha, A. and Mohd Sharef, N. (2013) 'An Analysis of Ontology Engineering Methodologies: A Literature Review', *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 16, January, pp. 2993-3000.

Islam, N. and Sheik, G.S. (2015) 'A REVIEW OF TECHNIQUES FOR ONTOLOGY', *Journal of Information & Communication Technology*, vol. 9, no. 2, pp. 104-114.

John, S. (2010) 'Leveraging Traditional Software Engineering Tools to Ontology Engineering under a New Methodology', IEEE Xplore, 1-5.

John, S. (2014) 'Development of an Educational Ontology for Java Programming (JLEO) with a Hybrid Methodology Derived from Conventional Software Engineering Process Models', *International Journal of Information and Education Technology*, vol. 4, no. 4, Auguest, pp. 308-312.

John, S., Shah, N. and Smalov, L. (2016) 'Incremental and Iterative Agile Methodology (IIAM):Hybrid Approach for Ontology Design towards Semantic Web Based Educational Systems Development', *International Journal of Knowledge Engineering,* , vol. II, no. 1, March, pp. 13-19.

Jones, D., Bench-Capon, T. and Visser, P. (1998) *Methodologies for Ontology development*.

Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., and Hendler, J. (2005) 'Swoop: A Web Ontology Editing Browser', *Journal of Web Semantics*, vol. 4, no. 2, June 2006, pp. 144-153.

Keet (2009) *Ontology Engineering Top-Down and Bottom-Up*, [Online], Available: //keet.wordpress.com/2009/11/20/72010-semwebtech-lectures-34-ontology-engineering-top-down-and-bottom-up/ [May 2014].

Kent, E. (1999) 'Conceptual Knowledge Markup Language: The Central Core', Twelth Workshop on Knowledge Acquisition, Modeling and Management, Canada.

Khruahong, S., Kong, X. and Hoang, D. (2015) 'Ontology Design for Thailand Travel Industry', *International Journal for Knowledge Engineering*, vol. 1, no. 3, pp. 191-196.

Knublauch, H. (2002) *An Agile Development Methodology for Knowledge-Based Systems ncluding a Java Framework for Knowledge Modeling and Appropriate Tool Support*.

Kothari, C.R. and Grag, G. (2014) *Research Methodology Methods and Techniques.* 3rd edition Jaipur : New Age International Publishers

Kotis, K. and Vouros,G.A. (2006) 'Human-centered ontology engineering: The HCOME methodology', *Knowledge and Information Systems*, vol. X, no. 1, July, pp. 109-131.

Kreitner, R. and Kinicki, A. (1998) *Organizational Behavior.* Irwin McGraw-Hill

Kruchten, (2003) *The Rational Unified process: An Introduction.* 3rd edition Boston, USA : Addison-Wesley Longman Publishing Co

Lassila, O. and McGuinness, D. (2001) *The Role of Frame-Based Representation*, [Online], Available: http://www-ksl.stanford.edu/pub/KSL_Reports/KSL-01-02.html [June 2015].

Lee, Ming & Yen Ye, Ding & Wang, Tzone-I. (2005). Java Learning Object Ontology. 538-542. 10.1109/ICALT.2005.185.

Lopez, F. (1999) *Overview Of Methodologies For Building Ontologies,* [Online], Available: http://oa.upm.es/5480/1/Overview_Of_Methodologies.pdf [May 2015].

Lopez, M.F., Gomez-perez, A. and Sierrra, J.P. (1999) 'Building aChemical Ontology using Methontology and the Ontology design environment', *IEEE Intelligent Systems*, vol. 14, no. 1, January.

Loppez, F. (1999) 'Overview of Methodologies For Building Ontologies', IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 1-13.

Lord, P. (2010) *Components of an Ontology. Ontogenesis.*, January, [Online], Available: http://ontogenesis.knowledgeblog.org/514 [September 2016].

Lukasiewicz, and Straccia, (2008) 'Managing uncertainty and vagueness in description logics', *Journal of Web Semantics*, Available: http://w.websemanticsjournal.org/index.php/ps/article/viewFile/147/145 [20 July 2015].

Luke, S., Spector, L., Rager, D. and Hendler, J. (1997) 'Ontology-based Web Agents', First International Conference on Autonomous Agents (Agents97), New York, 59-66.

Malik, S.K., Prakash, N. and Rizvi, S.A.M. (2010) 'Developing an University Ontology in Education Domain using Protégé for Semantic Web.', *International Journal of Engineering Science and Technology*, vol. 2, no. 9, pp. 4673-4681.

Meenachi, M.N. and Baba, S.M. (2012) 'International Journal of Applied Information Systems', *International Journal of Applied Information Systems (IJAIS)*, vol. 4, no. 2, September, pp. 46-55.

Min-Kim, J., Choi, B.-I., Shin, H.-P. and Kim, H.-J. (2007) 'A Methodology for constructing of philosophy ontology based on philosophical texts', *Elsevier*, October, pp. 302-315.

Mizoguchi, R. (1998) 'A Step Towards Ontological Engineering', 12th National Conference on AI of JSAI, Japan, 24-31.

Natalya F. Noy, D.L.M. (2001) *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05.

Nicola, D. and Missikoff, M. (2016) 'A Lightweight Methodology For Rapid Ontology Engineering', *Communications of the ACM*, vol. 59, no. 3, March, pp. 79-86.

Novak, D.J. and Canas, J.A. (2014) *The Theory underlying concept maps and how to construct and use them*, April, [Online], Available: http://cmap.ihmc.us/Publications/ResearchPapers/

TheoryCmaps/TheoryUnderlyingConceptMaps.htm [December 2016].

Paliszkiewicz (2011) 'Trust Managment: Literature review', *Mangement*, vol. 6, no. 4, pp. 315-331.

Paliszkiewicz, J. (2013) 'Organizational trust as a foundation for knowledge sharing and its influence on organizational performance', *Online Journal of Applied Knowledge Management*, vol. 1, no. 2, pp. 116-127.

Pére, G., Asunción, Lopez, F., Mariano, Corcho, O. (2004) *Ontological Engineering.* 1st edition Springer

Pouchard, L.., Ivezic, N. and Schlenoff, C. (2000) 'Ontology Engineering for Distributed Collaboration in Manufacturing', AIS2000 conference.

Powell-Morse, A. *Waterfall Model: What Is It and When Should You Use It?*, [Online], Available: https://airbrake.io/blog/sdlc/waterfall-model [June 2016].

Proteige: Standford University (2005), [Online], Available: http://protege.standford.edu [June 2015] .

R.C. Solomon and F. Flores (2001) *Building Trust in Business, Politics, Relationships, and Life.* Oxford University Press

Rajadorai, P., Hassan, S. and Admodisastro, N. (2012) 'Critical evaluation on Software Development Process Models with respect to Mobile Software Development', The International Conference on Informatics and Applications (ICIA2012), 255-264.

Research Design Now, [Online], Available: http://www.transtutors.com/homework-help/management/marketing/market-research-process/research-design/ [November 2016].

Riviere, J., Picard, D. and Coble, R. *Syllabus Construction*, [Online], Available: https://cft.vanderbilt.edu//cft/guides-sub-pages/syllabus-design/ [November 2017].

Robinssons, P. and Coultor, M.K. (1999) *Management.* Prentice Hall

Roussey, C., Pinet, F., Kang, A.M. and Corcho, O. (2011) 'An Introduction to Ontologies and Ontology Engineering', in *Ontologies in Urban Development Projects. Advanced Information and Knowledge Processing*, 1st edition, Londoan: Springer.

Saunders, M., Skinner, D., Dietz, G., Gillespie, N. and Lewicki, R. (2010) *Organizational Trust: A Cultural Perspective.* Newyork : Cambridge University Press

Schober, D., Malone, J. and Stevens, R. (2009) 'Practical experiences in concurrent, collaborative ontology building using Collaborative Protégé', International Conference on Biomedical Ontology, UK.

Schreiber, G., Wielinga, B. and Jansweijer, W. (1995) 'The KACTUS View on the 'O' Word', IJCAI Workshop on Basic Ontological Issues in Knowledge Shar, g, Montr.

Schreiber, T. and Akkermans, H. (1999) *Knowledge engineering and management:the CommonKADS methodology.* MIT Press Cambridge, MA, USA

Sean, L. and Heflin, J. (2000) *SHOE 1.01 Proposed Specification. Parallel Understanding Systems Group*, [Online], Available: http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.html [February 2016].

Selltiz , C. (1962) *Research methods in social relations.* Rev. one-vol. ed edition New York

Sivakumar, R. and Swaminathan, V. (2010) 'A Study on Ontology Learning Process frameworks', *College Sadhana- Journal for Bloomers of Research*, vol. 2, no. 2, February, pp. 120-126.

Six, F. and Sorge, A. (2008) 'Creating a High-Trust Organization: An Exploration into Organizational Policies that Stimulate Interpersonal Trust Building', *Journal of Management Studies*, vol. 45, no. 5, July, pp. 857-884.

Slimani, T. (2015) 'Ontology Development: A Comparing Study on Tools, Languages and Formalisms', *Indian Journal of Science and Technology*, vol. 8, no. 24, September.

Software Development Life Cycle, [Online], Available: https://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm [October 2016].

Sommerville, (2010) *Software Engineering.* 9th edition USA : Addison-Wesley Publishing Company

Soren, A. and Herre, (2006) 'Rapid OWL---An Agile Knowledge Engineering Methodology', Sixth International Andrei Ershov Memorial Conference - Perspectives of System Informatics (PSI'06), Russia.

Sosnovsky, S. and Gavrilova, T. (2006) 'Development of an Ontology for C programming', *International Journal "Information Theories & Applications*, vol. 13, pp. 303-308.

Staab, S., Schnurr, H.P., Studer, R. and Sure, Y. (2001) 'Knowledge Process and Ontology', *IEEE Intelligent Systems* , vol. 16, no. 1, pp. 72-79.

Staab, S. and Studer, R. (2003) *Handbook on Ontologies.* 2nd edition Springer-Verlag Berlin Heidelberg

Sua´rez-Figueroa, M.C., Go´mez-Pe´rez, A. and Ferna´ndez-Lo´pez, M. (2012) *The NeOn Methodology for Ontology engineering*, [Online], Available: http://www.springer.com/cda/content/document/cda_downloaddocument/9783642247934-c1.pdf?SGWID=0-0-45-1320441-p174265158 [October 2015].

Suárez-Figueroa, M.C., Gómez-Pérez, A. and Terrazas, V. (2008) *How to write and use the Ontology Requirements Specification Document*, Italy.

Sure, Y., Angele, J. and Staab, (2003) 'OntoEdit: Multifaceted Inferencing for Ontology Engineering', *Journal on Data Semantics*, vol. 1, no. 1, November, pp. 128-152, Available: https://pdfs.semanticscholar.org/5021/a50d86b2afc5bccebd3105e79362bd4f2580.pdf [October

2015].

Szturcova, D. and Rapant, P. (2013) 'Enhanced Methodology for Ontology Development', *Computing and Informatics*, vol. 32, p. 1038–1054.

Tetlow, P., Pan, J.Z., Wallace, E., Uschold, M. and Kendall, E. (2006) 'Ontology Driven Architecture and potential uses of the semantic web in systems and software engineering', *World Wide Constrotium Tech*, February.

Tudorache,T., Falconer,S., Noy, F., Nyulas, C., Üstün, B. and Storey, M.-A. (2010) 'Ontology Development for the Masses: Creating ICD-11 in WebProtégé', Knowledge Engineering and Management by the Masses, Lisbon, 74-89.

Uschold, M. and Gruninger, M. (1996) 'Ontologies: principles, methods and applications', vol. X1, no. 2, pp. 93-136.

Ushold, M. and King, M. (1995) 'Towards a Methodology for Building Ontologies', Basic Ontological Issues in Knowledge Sharing (IJCAL), 1-15.

Vilches-Blázquez, L.M., Bernabé-Poveda, M.-A., Suárez-Figueroa, M.C., Gómez-Pérez, A. and Rodríguez-Pascual, A.F. (2007) 'Towntology & hydrOntology: Relationship between Urban and Hydrographic Features in the Geographic Information Domain', in Teller, , Lee, J.R. and Roussey, C. *Ontologies for Urban Development*.

Vincent, J., Dombeu, F. and Huisan, M. (2011) 'Combining Ontology Development Methodologies and Semantic Web Platforms for E-goverment Domain Ontology Development', *International Journal of Web & Semantic Technology (IJWest)*, vol. 2, no. 2, April, pp. 12-25.

Weller, K. (2010) *Knowledge and Information : Knowledge Representation in the Social Semantic Web.* Berlin : DEU: Walter de Gruyter

Wieringa, J. (2014) *Design Science Methodology for Information Systems and Software Engineering.* Springer-Verlag

Yarandi, M., Jahankhani, H. and Tawil, A.-R.H. (2013) 'A personalized adaptive e-learning approach based on semantic web technology', *Webology*, vol. 10, no. 2, December, pp. 1-14.

ZWiga, G.L. (2001) *ontology its transformation from philosophy to information systems*, July,[Online],Available: http://pdf.aminer.org/000/212/672/ontology_its_transformation_from_philosophy_to_information_systems.pdf [July 2015].

APPENDICES

APPENDIX A: Progress of the Research Report

This Appendix of the document summarises the progress of the research project based on the work done by the researcher based on the valuable guidance of the Director of Studies, Supervisory team and the Doctoral College representatives during their visits to Middle East College throughout the period of research. Academic yearwise summary has been provided for the convenience of the reader.

Year	Milestone Achievements
2015-2016	The very first version of research proposal has been prepared and submitted to DoS. The initial proposal was submitted after preliminary literature review. The proposal has been reviewed and changed to ontology development methodology domain. This has been done based on the advice of Supervisory panel and a paper published in the ***International Journal of Information and Education Technology, Vol. 4, No. 4, August 2014*** *(Refer Appendix C-Publication1 for the published article)*
	Finalised the areas for literature review and the approach followed for the literature review. Eight weeks have been spent in CU and attended various research design workshops, classes and seminars.
	Extensive literature review has been done during the initial year. Research problem, research objectives, research questions and research plan have been finalised towards the end of initial year supported with literature review.
	Research idea has been presented as a paper to Research idea presented in International Conference (ICKE 2016-Los Angeles, USA). An extended version of the paper has been published in the ***International Journal of Knowledge Engineering vol. 2, no. 1, pp. 13-19, 2016***. www.ijke.org/list-41-1.html *(Refer Appendix C-Publication2 for published article)*
	Ethics Approval has been completed. First Progress Review Panel (PRP 1) has been cleared and enrolled successfully to Year 2.
	Detailed Literature review has been done to study the existing ontology development methodologies and their pros and cons. A detailed comparison among the well-known methodologies has been completed based on a set of parameters derived.
	Refined research idea and proposed methodology concepts has been published as a research paper in the ***International Journal of Information Technology Convergence and Services vol.6 no.1, pp 37-44,2016.*** http://aircconline.com/ijitcs/V6N1/6116ijitcs04.pdf *(Refer Appendix C-Publication3 for published article)*
	Working draft of the Research Methodology chapter has been completed. This draft has been reviewed by the DoS and the members of the supervisory panel.

2016-2017	Conceptual framework of the proposed novel methodology (Software Centric Methodology (SCIM) has been drafted and it has been reviewed by DoS and Supervisory panel members.
	Co-authored a research paper **"Study of Trust Models and Semantic Structural Relationship between the Concepts of Organizational Trust for building High Trust Organization"**and presented *in Fourth International Conference on Recent Trends in Communication and Computer Networks – Nov 12th ComNet 2016, Bangalore, India*.
	Completed the taught module **Writing for Computer Science and Engineering course (M31AAE)** with Merit grade.
	Review Panel (PRP 2) has been cleared and enrolled successfully to Year 3.
2017-2018	The core solution to the research problem addressed the development of a novel methodology for ontology development-Software Centric Innovative Methodology for Ontology development (SCIM) has been finalized in line with the PRP and post PRP comments.
	Literature review has been tuned with detailed review on the ontology application cases. Further to that, the domain has been chosen for the application (prototype development) of proposed methodology.
	The stages, workflows and activities of the new methodology have been tuned and finalized with DoS prior starting the application. CASE tool has been chosen for the prototype development.
	Information gathering techniques have been applied among the target group for the identification of the domain concepts for prototype ontology development (Java Learning Education Ontology JLEO).
	A novel methodology with defined Ontology Development Life Cycle with the employment of appropriate techniques aligned to the philosophy has been proposed.
	The details of the proposed methodology has been published as a research paper in the proceedings of the 9th International ACM Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2017) – vol.2: KEOD in Maderaia, Portugal Nov 1-3, 2017.pp139-146. SCITEPRESS. https://www.scitepress.org/papers/2017/64829/64829.pdf *(Refer Appendix C-Publication4 for published article)* Review Panel (PRP 3) has been cleared and enrolled successfully to Year 4.
	Literature review has been fine tuned. Necessary tuning/ amendments have been made to the conceptual elements of SCIM.
	JLEO prototype development with SCIM methodology has been completed. The components of SCIM have been applied to the chosen domain for the development of prototype ontology.

2018-2019	The newly proposed methodology has been validated by following a custom made GQM approach. Necessary instruments have been used among the target group of evaluators for the effective validation of proposed methodology.
	The application part of SCIM (JLEO development) has been published as a research paper in the proceedings of 15th IEEE International Conference on E Business Engineering – (ICEBE-2018), Xi'an, China, October 12-14, 2018 *(Refer Appendix C-Publication5 for published article)*
	Document and assemble thesis chapters, review for consistency, completeness and prepare presentation for defense.

APPENDIX B1: Filled Ontology Requirement Specification Document

Domain : ***Java Learning Educational Ontology (JLEO)***   **Date of Creation : February 2018**   **Version No : 1**   **Author(s) :** ***Santhosh John***   **Explicit Domain Assumptions if any : None**
Purpose :   The purpose of building the Java Learning Educational Ontology (JLEO) is to provide a structural knowledge representation model of introductory Java modules teaching and learning domain. This could be used by academic community.
Level of formality : The ontology has to be implemented in Web Ontology Language (OWL)
Scope: The scope of JLEO was on three J2SE based Introductory Java modules of Middle East College's curriculum of undergraduate computing course. The level of granularity is directly related to competency questions and terms identified.
Boundary: The boundary of JLEO has been defined to the core aspects of Java programming with respect to three different variations. The variations are Introduction to programming, object oriented programming and Internet programming.
Integration with any other existing ontology :
Intended Users: JLEO will be the core ontology to build a knowledge-based system based system based on the same to address its intended purpose. Domain expert and ontology development team had identified intended users based on the techniques such as JOD and interview. Besides that, motivating scenarios also have been used. The stated techniques allowed to identify the below mentioned intended users of the ontology.   User 1: Teacher/Faculty who is teaching three introductory java modules (Introduction to Programming, Object Oriented Programming and internet programming.   User 2: Student who is learning any of the modules mentioned above for searching conceptual information on java learning units.   User 3: Module leader/ Module reviewer who reviews the curriculum of introductory java modules stated above.
Intended Uses: Use 1: Publish the semantic structural model of the introductory Java courses in terms of the concepts and relationship among them belonging to the introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming)   Use 2: Search the various concepts of introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming)

Use 3: Can update the structural model as part of academic review of the of introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming)
Ontology Requirements : • *Functional requirements:* The specific set of requirements that the ontology should fulfill for its Intended users including optional priorities. In the context of JLEO, the knowledge model about java curriculum with language fundamentals (ITP), Object Oriented concepts (OOP) and java network programming (IP) have been considered. A use case diagram with intended uses of ontology as use cases and users as actors furnished. Competency Questions techniques has been applied among the users to know the concepts of the three variations of basic java courses. FR 1. Define a knowledge model of the concepts of three basic java courses • *Non-functional requirements* : The general requirements that the ontology should fulfill for its Intended users including optional priorities NFR 1: The developed ontology must be followed international naming convention.
List of sources: Module Guides of the modules Introduction to Programming, Object Oriented Programming. Text book, Java 2 Complete Reference and online java learning resources.
Techniques applied : JOD Session, Usage of multi-level Competency Questions, Informal Interviews, Domain Expert's Judgement, Formal Text Analysis, Validation, Usage of Protégé

APPENDIX B2: JLEO Ontology Requirement Specification Key Slots

1	**Purpose**
	The purpose of developing Java Learning Educational Ontology (JLEO) is to deliver a structural knowledge representation model for Java teaching and learning domain which can be used by teachers and learners.
2	**Scope**
	The scope of the ontology is three introductory java courses of Middle East College's undergraduate curriculum. The courses chosen are J2SE based Introduction to Programming, Object Oriented Programming and Java Internet Programming. The degree of granularity depended directly to the competency questions and the concepts extracted from the domain.
3	**Implementation Language**
	The formal language used for the implementation of ontology was Web Ontology Language (OWL)
4	**Intended Users**
	User 1: Teacher/Faculty who is teaching any of introductory java courses (Introduction to Programming, Object Oriented Programming and internet Programming). User 2: Student who is learning any of the introductory java courses (Introduction to Programming, Object Oriented Programming and internet Programming). User 3: Module leader/ Module reviewer who reviews the curriculum of introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming)
5	**Intended Uses**
	Use 1: Publish the semantic structural model of the introductory Java courses in terms of the concepts and relationship among them belonging to the introductory java courses. (Introduction to Programming,

	Object Oriented Programming and internet Programming) Use 2: Search the various concepts of introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming) Use 3: Can update the structural model as part of academic review of the of introductory java courses. (Introduction to Programming, Object Oriented Programming and internet Programming)

2017

APPENDIX B3: JOD Session

[Joint Ontology Development (JOD) for JLEO development]

Joint Ontology Development Session Overview

The Joint Ontology Development (JOD) session has been conducted on Sunday 13th August 2017 at Middle East College (MEC) computing department meeting room. The session has been conducted to facilitate the planning stage of proposed Software Centric Innovative Methodology (SCIM) for Ontology Development. The session conducted to jumpstart the requirement analysis workflow associated with planning stage of SCIM. The session assists to identify the purpose, scope, boundary and functional requirements of Java Learning Educational Ontology (JLEO) in a one meeting setting with Domain Expert, Intended users and representative of Ontology Development Team (ODT) Together, the group discussed the feasibility activity of JLEO and essential techniques being followed to finalize the purpose, scope, boundary, intended users and intended uses of JLEO. The quick benefits offered by the JOD session briefed below.

- It serves as a core platform/forum for JLEO detailed design stages and technical planning
- Launches strong communication and consensus among the parties involved
- Helps the team to frame the instruments recommended by methodology for Ontology Requirement Specification Document
- Ensures the incorporation of agile approaches in ontology development

Meeting Input

This meeting has been designed as a group brain storming session on the development of JLEO by applying the methodology SCIM. In order for this meeting to be as productive as possible, the following artefacts circulated to the members 48 hrs in advance.

- Draft framework of SCIM with stages, workflows, activities and techniques (diagram)
- Draft high-level Ontology Requirement Specification Document (ORSD) template

The above mentioned documents discussed in detail during the JOD session inline with the realization of ORSD

Meeting Deliverables

Following are the deliverables came from the successful JOD session in their draft form.

- Purpose of JLEO
- Scope of JLEO
- Boundary of JLEO
- Intended Uses of JLEO
- Functional and Nonfunctional requirements of JLEO

JOD Session Roles and Description

Name(s)/Role	Description
Facilitator	▪ Committed to the JOD process. ▪ Not a part of the JLEO team ▪ Rich set of experience in facilitating meetings. ▪ Summarized the descriptions ▪ Facilitated the JOD session
ModuleLeader_ITP ModuleLeader__IP ModuleLeader__OOP, User Representative	▪ Module Instructors/Module Leaders of the modules ▪ Introduction to Programming ▪ Object Oriented Programming ▪ Internet programming. ▪ Student
Ontology Development Team Representative & DE	▪ Contributes technical information ▪ Domain Expert ▪ Ontology Designer

Planning the Meeting

JOD Location

The JOD session has been conducted at a room where got adequate space for all the participants to sit and discuss with the following facilities

- Whiteboard
- Flip Chart
- Projector

Meeting Check List

- ☐ Attendance of all members.
- ☐ Meeting room booking.
- ☐ Arrangement of presentation equipment (e.g., laptop, projector, flip chart paper, markers)
- ☐ Prepare agenda
- ☐ Send out email invitation to participants
- ☐ Distribute advance reading materials prior to the meeting (SCIM methodology framework).
- ☐ Review room arrangements and audio/visual tools prior to the session.

Joint Ontology Development (JOD) Session Agenda

Meeting Purpose: Finalize the feasibility of JLEO
Facilitator
Scribe: NA
Meeting Details:

Date: 13/08/2017
Time: 2 PM

Location: IBK 004, MEC

Attendees:

Module Instructor/ Introduction to Programming.

Module Instructor/ Internet Programming.

Module Instructor/ Object Oriented Programming.

Student representative

Member of Ontology Development Team & Domain Expert (MOD)

Advance Pre-reading Materials:

- Software Centric Innovative Methodology (SCIM)for Ontology development (publication)
- Proposed Framework of SCIM
- Ontology Requirement Specification Document (ORSD) template

Meeting Topics	Reference	Led By	Time
Introduction • Introductions of everyone in attendance • Introduction to the JOD Process		**Facilitator**	**2 PM**
Project Overview		**Facilitator**	
Planning stage Brainstorming: • Context of JLEO • Purpose of JLEO • Scope of JLEO • Boundary of JLEO • Intended Uses of JLEO • Intented Users of JLEO		**MOD** **All** **All** **All** **All** **All** **All**	
Review and recap issues and action items: • Assign action items as appropriate		**Facilitator**	
Summarize and Close	**All**		**3.30 PM**

Action Plans

Action Items	Person responsible
Informal Interview with intended users	MOD
Simple competency questions to intended users	MOD and Intended users
Development of motivational scenarios	MOD
Ontology Requirement Specification Document filling	MOD

Time Plan

Item #1 & #2 should be completed by August 25, 2017
Item # 3 should be completed by August 20, 2017
Item # 4 by September 5, 2017

Deliverables

Informal interview details
Completed Competency Questions
Motivational Scenarios
Filled template of ORSD of JLEO

APPENDIX B4: Informal Interviews

Dear ITP module leader

As part of my ongoing research, I am intending to design and develop an educational ontology-Java Learning Educational Ontology (JLEO) to apply a newly derived ontology development methodology called Software Centric Innovative Methodology (SCIM) for ontology development. The intended purpose of JLEO is organizing learning objects of Java Courses based on Java 2 Standard Edition (J2SE) to build knowledge based systems on top of the designed ontology. The purpose of this survey instrument is to identify the core concepts of teaching Java as a language with their properties and the relationships among concepts to build the ontology. Based on your teaching experience of the arena stated above, requesting to spend at least 15-30minutes to answer the following survey instrument. Your support is highly appreciated.

1. Could you let me know to what extent Java aspects are covered in MEC's Introduction to Programming?
   - ✓ *Introduction to programming elements: Character Set; Keywords and Identifiers; Constants, Variables - Data Types; Declaration of variables; Assigning values to variables. Input – Output constructs. Arithmetic Operators; Relational operators, Logical operators, Assignment operators, Increment and decrement operators. Arithmetic expressions; Evaluation of expressions, Precedence of arithmetic operators, computational problems, Type conversions in expressions.*
   - ✓ *Decision making, IF statement, IF ELSE statement, Nesting of IF ….ELSE statements. The ELSE IF ladder, the switch statement.* ***Repetition Structures: Looping:*** *For loop: working with for loop, nested for loops, While loop: Working for while loop, need for while loop, nested while loops, do-while loop: difference from while loop, working with do-while loop.*
   - ✓ *One-Dimensional, array initialization, accessing array elements , processing array contents, character arrays. Structure definition, initializing structures, accessing members of structures, unions, enumeration constants.*
   - ✓ *Functions: Defining and calling function, function prototype, parameter passing, returning value from function, Recursion, Various types of function arguments.*
   - ✓ *Files and streams, Creating a sequential access file, reading data from a sequential access file, random access files. #include preprocessor directive, symbolic constants.*

2. Do you think a conceptual model of java language learning concepts will help to design knowledge based systems?

   Yes

3. What are the key aspects of Programming language considered for teaching Introduction to Programming by using Java as the programming language?

   There are the 5 basic concepts of any programming language namely Variables Control Structures, Data Structures, Syntax and Tools

4. In your opinion, what are the essential concepts of Java language to be covered for delivering Introduction to Programming module by using Java as the tool?

   String Handling, Exception Handling

5. Could you let me know the properties associated with the concepts you identified for the previous question?

   No

6. What are the essential semantic relationships among the concepts you have identified? (Eg. data type is a part of variable declaration…)

   Base types:

   int, float, double, char, bool, etc. These are the primitive types provided directly by t he underlying hardware. There may be

   facility for user-defined variants on the base types

   Compound types: arrays, *pointers, classes and interfaces* .

   These types are constructed as aggregations of the base types and simple compoun d types.

7. Could you recommend any learning resources/text book/course material for more module specific concepts and relationships?

   Java API, MEC module guide.

8. Is there anything else that you would like to share with me regarding the implementation of In particular, your experiences of working as Introduction to Programming module leader?

   It is a good approach for implementing java learning ontology. It could be better java ontology can be integrated with any E-learning platform for class room teaching purposes as well as Semantic Web Rule Language (SWRL), to infer more knowledge.

Dear OOP module leader

As part of my ongoing research, I am intending to design and develop an educational ontology (Java Learning Educational Ontology (JLEO)) by apply a newly proposed ontology development methodology called Software Centric Innovative Methodology (SCIM). The intended purpose of JLEO is organizing learning objects of Java Courses based on Java 2 Standard Edition (J2SE) to build knowledge based systems on top of the designed ontology. The purpose of this survey is to identify the core concepts involved in teaching of Java language along with their properties and relationships to build the ontology. Please provide your response to survey questions adequately based on your teaching experience of the Java language teaching. Your support is highly appreciated.

1. What extent Java aspects are covered in MEC's Object Oriented Programming?

   Basic concepts of JAVA programming required to build simple console applications to learn object oriented concepts are a covered. Java Language features, Java Architecture, Packages and interfaces in Java, Exception handling, I/O Stream -I/O Streams, Reading and writing to files, Byte Streams, Character Streams are also there. Applets, Servlets etc. are not included.

2. Do you think a conceptual model of java language based OOP learning concepts will help to design knowledge based systems?

   Yes very much

3. What are the key aspects of Programming language considered for teaching OOP by using Java as the programming language?

   Key words , Data types ,Variables declarations, scope and lifetime of variables ,operators, I/O statements and operational statements (arithmetic , logical etc.), conditional constructs, iterative constructs, arrays, functions, garbage collection , exception handling

4. In your opinion, what are the essential concepts of OOP to be covered for delivering object oriented programming module by using Java as the tool?

   Classes ,Objects, Constructors, Data Hiding, Encapsulation, Abstraction- Interfaces, Polymorphism-Constructor and method overloading, Inheritance , Overriding

5. Could you let me know the properties associated with the concepts you identified for the previous question?

   Yes

6. What are the essential semantic relationships among the concepts you have identified? (Eg, object is an instance of class, instance attributes are the attributes belongs to object…)

Variables declaration and method declaration are part of class declaration.

Object is created using a class.

Inheritance connects two classes in a hierarchical relationship

Interfaces contain abstract methods

Constructor is a class method

Constructor has the same name as the class.

Constructor is invoked automatically when object is created,

Overloaded methods have the same name

All overloaded constructors have the same name as the class

7. Could you recommend any learning resources/text book/course material for more module specific concepts and relationships?

   Schildt, Herbert. Java2: the complete reference. 5th rev ed. McGraw-Hill, 2002, ISBN: 0072224207

   Flanagan, D. Java in a nutshell, 5th ed, O'Reilly,2005.

   Horstmann, C.S. Big Java. 2nd ed. John Wiley ,2006.

8. Is there anything else that you would like to share with me regarding the implementation of java learning ontology? In particular, your experiences of working as Object Oriented Programming module leader?

   No

Dear Internet programming (IP) module leader

As part of my ongoing research, I am intending to design and develop an educational ontology (Java Learning Educational Ontology (JLEO)) by apply a newly proposed ontology development methodology called Software Centric Innovative Methodology (SCIM). The intended purpose of JLEO is organizing learning objects of Java Courses based on Java 2 Standard Edition (J2SE) to build knowledge based systems on top of the designed ontology. The purpose of this survey is to identify the core concepts involved in teaching of Java language along with their properties and relationships to build the ontology. Please provide your response to survey questions adequately based on your teaching experience of the Java language teaching. Your support is highly appreciated.

1. What extent Java aspects are covered in MEC's Network programming with Java (Internet Programming module)?

   Java Applets, Java Network Programming, Java Socket Programming, Distributed programming techniques, Servlets.

2. Do you think a conceptual model of java language networking concepts will help to design knowledge based systems?

   Yes, Knowledge-based systems are an important class of intelligent systems in the field of web technology and Java become a powerful tool to develop and deploy such secured systems.

3. What are the key aspects of Java network programming considered for teaching network programming by using Java as the programming language?

   The key aspect which is covering through internet programming is Java Applets with GUI programming, Exploring java.net packages. Different tiers of java client server architecture, Socket programming, UDP sockets, TCP client sockets, RMI for distributed programming.

4. In your opinion, what are the essential concepts of Java language to be covered for delivering network programming module by using Java as the tool?

   As a powerful tool for develop web services in Java, it is being used Applet GUI programing, socket programming in both ways (one / two), Serialization for writing the state of an object on the network. RMI, Servlets for extending the capabilities of server.

5. Could you let me know the properties associated with the concepts you identified for the previous question?

   They are the concepts not properties

6. What are the essential semantic relationships among the concepts you have identified? (Eg, DNS programming is related Internet addressing…)

   The basic network features by using java as a tools are for printing DNS record for an Internet address, distributing data over network , cookie Management ,URL connection caching ,HTTP authentication, connection persistence , Java networking and proxies , socket options, socket exceptions etc.

7. Could you recommend any learning resources/text book/course material for more module specific concepts and relationships?

   Java Network Programming, 4th Edition
   Developing Networked Applications
   By Elliotte Harold
   Publisher: O'Reilly Media
   Release Date: October 2013

   TCP/IP Sockets in Java, Second Edition: Practical Guide for Programmers (The Practical Guides) 2nd Edition. by Kenneth L. Calvert (Author), Michael J. Donahoo (Author)

8. Is there anything else that you would like to share with me regarding the implementation of java learning ontology? In particular, your experiences of working as Internet Programming module leader.

   No

   Java could help for mapping the object oriented model with sematic web languages for electrophysiological metadata. By exploring the package 'java.net' we can communicate with any server as well as construct your own server.

   Based on the internet programming, the socket programming could help for developing an ontology for showing the properties of a domain and the relations between them. A socket is one of the most fundamental technologies of computer network programming. Sockets allow network software applications to communicate using standard mechanisms built into network hardware and operating systems
   Although it might sound like just another feature of Internet software development, socket technology existed long before the Web. And, many of today's most popular network software applications rely on sockets.
   A socket represents a single connection between exactly two pieces of software (a so-called point-to-point connection). More than two pieces of software can communicate with client/server or distributed systems by using multiple sockets.

Socket-based software usually runs on two separate computers on the network, but sockets can also be used to communicate locally (inter process) on a single computer. Sockets are bidirectional, meaning that either side of the connection is capable of both sending and receiving data. Sometimes the one application that initiates communication is termed the "client" and the other application the "server," but this terminology leads to confusion in peer to peer networking and should generally be avoided.

APPENDIX B5: Introduction to programming Domain Vocabulary

Ontology Title	Java Learning Educational Ontology (JLEO)
Methodology	SCIM
Domain	Three Introductory Java Modules of MEC's UG Curriculum Programme Curriculum
Methodology Stage	Conceptualisation (Stage 2)
Workflow	Domain Analysis Workflow
Activity	Domain Vocabulary Acquisition (Activity 1)
Techniques applied	Informal Interview with instructors/Competency Questions, Domain Expert's Judgement, and Text Analysis
Text books referred	Scheldt, Herbert. Java2: the complete reference. 5th rev ed. McGraw-Hill, 2002, ISBN: 0072224207, Flanagan, D. Java in a nutshell, 5th ed, O'Reilly,2005. Horstmann, C.S. Big Java. 2nd ed. John Wiley, 2006.
Web resources referred:	https://docs.oracle.com/javase/7/docs/api/ https://www.javatpoint.com/java-tutorial; https://www.tutorialspoint.com/java/java_variable_types.htm; https://www.w3schools.in/java-tutorial/ http://projectsgeek.com
Other resources used:	Module Descriptors (ITP, OOP & IP), module guide/ lecture slides

Key Concept	Sub Concepts	Remarks (If any)
Datatype	Primitive Non primitive	Language Specific and User defined data types respectively
Variables	Class variables Local variables Instance variables	
Type Conversion	Widening Conversion Narrowing Conversion	Typecasting techniques (Conversion of one data type to another one based on the compatibility)
Statement	Control Statement Expression Statement Labelled Statement Selection Statement Iteration Statement Jump Statement Synchrinization Statement Guarding Statement	Combination of variables and operators.
Expression	Arithmetic Expression Assignment Expression Conditional Expression Logical Expression Relational Expression MemberAccess Expression Arrayelement access expression Member Call Expression	Programming constructs
Decision Making	Multiple Selection Structure Single Selection Structure Double Selection Structure	Execution of a block of statements based on a Boolean value
Looping	Entry Controlled Exit Controlled	Two types of looping constructs in terms of its operating principle.
Complex Structure	Nested looping structure Nested selection structure	
Operators	Bitwise Operators Increment/decrement operators Relational operators Special Operators Conditional Operators	Java Language Operators

	Assignment Operators Arithmetic Operators Logical Operators	
Literals	Character literals Boolean Literals String Literals Floating point Literals Integer Literals	
Character set	Unicode	
Keyword		Language specific terms
Seperators	Brackets Paranthesis Braces Semicolumn Comma Period	Part of java syntax
Development	IDE	Integrated environment for coding, debugging and execution
Coding Style	Naming Convention Indentation	Standard naming practices
Debugging	Error Exception	
Testing Process	Testing Strategies Testing Techniques	

APPENDIX B6: Interview with Domain Expert

For the completion of the domain vocabulary acquisition activity belonged to Conceptualization stage, one of the techniques applied as suggested by SCIM is Domain Expert's Judgement. This technique has been applied to identify the concepts if any missed out after the application of competency questions and formal text analysis. Besides, this technique also has been used for the refinement of identified concepts.

Brief Research and Expertise profile with respect to Java Domain and Ontology

Domain Expert (DE) has completed the Bachelor project with title "Industrial Automation System" using the software Java in the year 2001. Since then, the DE is using Java as a programming language for the software consultancy activities and teaching purpose extensively till date. DE's expertise in Java is of very advanced status. For domain expertise, the DE has been working as an academic staff starting from 2001-2006 and 2016 to till date. During this period, DE has taught various Java based courses in the Departments of Information and Computer Science and Engineering of higher education institutions.

The DE started exploring Ontology related topics during the Master's in Engineering (ME) study period. DE's Master's thesis entitled "A Framework of Ontology Based Ranking, Classification and Clustering of Documents" was completed in the year 2010. Since then, DE has been continuing the research on "Ontology based Text Processing". DE has developed tourism domain ontology and is described briefly in one of the publications. DE has published 9 International publications pertaining to ontology design and development. The research perspective has been carried out with various aspects of incorporating ontology in several modules of Information Retrieval and Extraction. One of the research works is in the automatic enrichment of domain ontology with concepts and relations. In addition to this, DE has worked with few modules in CLIA Project – UNL based search engine.

Description of Informal interview with Domain Expert

An informal interview session was conducted with the Domain Expert on Sunday 8th December 2017 at the office of DE in India. The session was conducted to ensure/refine the concepts identified for JLEO development by following SCIM methodology suggested activities and techniques. The concepts identified have been supplied to the domain expert in advance and has taken her judgement in this regard by asking related questions. Commentary of the interview sessions with questions and answers are given below.

Question 1

In your opinion, the areas/modules that have been identified/chosen out by the Knowledge Engineer and Ontology Developer for the development of JLEO prototype are appropriate?

Answer: *Java Programming is a widely used robust technology with multi-platform environment. The important variations of Java programming that have been considered for the concept development of JLEO such as Introduction to Java Programming, Object oriented Programming and Network programming are appropriate to develop a Java Learning Educational Ontology (JLEO) prototype. Though the areas chosen are vast, I suggest limiting the attempt to core concepts for the prototype development.*

Question 2

How can the three areas chosen (mentioned above) play a major role in JLEO prototype development as you are saying the selection is appropriate?

Answer: *The basic concepts of core Java Programming is covered in the module descriptor of Introduction to Programming. Similarly, the core Object Oriented Concepts and their Java implementation along with other leading aspects such as packages, streams, strings and vectors have been well covered in the Object Oriented Programming Module. The Java Network Programming module covers the aspects ranging from DNS programming to distributed programming with Java using RMI. Apart from that, Java GUI related concepts also have been accommodated in the Network Programming module as per the module descriptor. Since the chosen modules covers all the core aspects of Introductory Java Programming, the choice of modules are appropriate for JLEO prototype development as I said earlier.*

Question 3

Are the inclusion of the main concept control structure and the associated concepts significant in JLEO prototype? How?

Answer: *Yes, the control structure and associated concepts are significant in JLEO prototype development. In any programming, the three kind of statements such as sequential, selection and iterative are fundamental ones for programme development. These statements and their related concepts have been identified with control structure concept. However, I suggest splitting the iterative concepts to entry controlled and exit controlled categories for building a more reasonable semantic hierarchy. In the case of Java Programming, it supports both of these categories with different variations of looping statements.*

Question 4

Whether the categories of data model concepts identified are relevant for JLEO? Any judgement on the identified?

Answer: *In any form of software application development, one of the building blocks of programming is the usage of data models. It plays a significant role in any programming language inclusive of Java. The usage of available data types in Java such as user defined and the built in types are well identified by the JLEO development team. I suggest to put a proper hierarchy of both primitive and non-primitive data types. The user defined data type(non-primitive)are arrays, classes and interfaces. These structures have the ability to store similar data types, enclosing values as instances to the variables and also links to the other module. The values used in the built in data types varies to be non-numeric and numeric. The non-numeric ones are boolean and character and the numeric concepts are integer and floating ones. The other building blocks identified under data model such as expression, statements and variable are fine. Need to ensure that all the different variations of such building blocks are to be a part of JLEO with necessary members. Besides the identified ones, it will be great if the concept of Type Conversion and relevant sub concepts can be added.*

Question 5

How to accommodate the aspects of testing related concepts in JLEO? Where do you prefer to pitch those aspects?

Answer: *I recommend to put the Testing as a separate main concept under the Development_and_Implementation main class. Testing should accommodate both the aspects of testing strategies and testing techniques. The testing strategies involved in each of the software process would seek for code inspection, code walkthrough, desk checking and the value tracing. The testing strategy makes the debugging of the code in the software. These strategies that exist are essential to any of the application development process. The testing strategies concept tends to be the additional concept in the development of Introduction to programing module.*

Question 6

In your view, are the Token class and associated concepts essential in JLEO? Please provide your judgement on this.

The concepts put under Token are essential for any programming language ontology development. The tokens would normally comprise of keywords, literals, separators,

charactersets and operators. The keywords which prefer to have specific functionalities of their own are being the basic elements in programming language. The values which are found to be constants are stored as literals of different formats in programming language. The separators are comprised of brackets, parenthesis, braces, semicolon comma and period are auxiliary features of programming language. The unique features that are represented as charactersets lay to be unicode representation in java. The different operators used forms the basis for the computation such as arithmetic operators, assignment operators, logical operators, conditional operators, special operators, relational operators, increment/decrement operators and the bitwise operators. Thus each component of the token is very essential concept in describing in Java programming. Therefore I strongly recommend incorporating all the concepts stated are to be in JLEO knowledge representation.

Question 7

Can we accommodate operators under tocken as they are the part of expressions in Java?

Answer: *Java Programming language has got a rich set of operators which are extensively using for writing programming statements/expressions as expression is a combination of operators and variables. Since these two are well connected in the frame of Java language, I suggest accommodating the aspects under both expressions and operator classes. Further to that, use the features of ontology editing platform for applying the semantic connectivity among the concepts of these two classes. This helps the Knowledge Base System developers to distinguish while building systems on top of JLEO.*

Question 8

While coming to the networking implementation aspects with regards to Java Programming, do you think InetAddress class is a significant concept for JLEO?

Answer: *Yes. In Java, DNS implementation is through the usage of InetAddress class. Therefore, it is a significant concept in Java Networking. The factory methods of InetAdress class allows to create the instances of IPs. It is recommended to incorporate the Address types, Host Resolution, caching etc in the concept taxonomy of JLEO. However, may be the full details can be reserved for the full version of ontology if time constraint is an issue for prototype development.*

Question 9

Java GUI aspects have been incorporated under Java _Network _Programming in JLEO prototype. Will it be fine if covered the concepts completely?

Answer: *In Ontology development, the order of placing the concepts doesn't make much value as ontology is used to design for building KBS. I understood that the GUI aspects are a part of COMP 0331.2 module in the curriculum referred for building JLEO. The important thing is the inclusion of concepts and relationships in the proposed prototype.*

Question 10

In your opinion, the concept of Applet needs to be separated or be a part of AWT?

Answer: *It is better to put Applet and its related aspects as a separate class as AWT is applicable to both Applet and Application.*

Question 11

In what ways are the AWT (Abstract Window Toolkit) and Event Handling related to each other in JLEO.

Answer: *The AWT characterize the windows based applications where these applications have to be controlled by a suitable method. These controls are being described to be the part of events that can have their own specific functions over the system call. Thus the AWT and Event Handling are related concepts in JLEO. However, recommending representing the core concepts of AWT and Event_Handling as separate classes in JLEO.*

Question 12

The way of separating Socket Programming and RMI for knowledge representation in JLEO is advisable?

Answer: *Yes, It has to be like that. Socket programming and RMI are two different levels of network programming. Socket programming is low level whereas RMI is distributed programming. In the later case, objects are send through the network. I strongly recommend incorporating ObjectSerialization concept in JLEO without fail.*

Question 13

In which aspect, object serialization has a major impact in JLEO?

Answer: *The extended feature of object oriented programming concepts illustrates the remote procedure call in order to establish an application to be executed in a distributed environment. This application writes the instances into a byte stream. Hence the objects that are considered forms the substantial aspect in serializable which has the impact over JLEO concepts.*

Question 14

With the impact of RMI concept is there is any reflection of Client-server model?

Answer: *The impact of RMI concept provides the execution of applications in a distributed environment. The remote object of stub representation resides in client system and also acts as an access point in client side. The request has been passed by the remote object using the concept of skeleton. The communication is made between the remote object through the remote reference layer. The concept of stub and skeleton summaries the substantial element in client and server implications in JLEO.*

Question 15

Are the URL Processing concepts crucial in JLEO?

Answer: *The URL processing identifies the resources on internet with the specified address and their communication in order to retrieve a particular set of information from it. Thus the URL processing invokes the major elements like protocol, host name which determines the existence of resource along with the file name. These concepts frame the major criteria in URL processing in java. Hence the URL processing forms the crucial concept of JLEO.*

Question 16

In your view, does the networking module in JLEO cover all the required concepts?

Answer: *Yes. The major elements in the network module are InetAddress, URL processing, Socket programming, RMI, AWT, Event Handling etc. On top of these, I recommend to incorporate Object serialization as well.*

Question 17

Do you think any concept/s identified under Object_Oriented_programming overlap with the concepts identified for Java_Language category?

Answer: *Obviously overlapping is there. For instance, the concept of class is required to be there in the Java_Language category and without which, Java fundamentals cannot be well presented. Therefore, I suggest incorporating the concept of class in category 1 and focusing on the essential OOP aspects under Object_Oriented_programming category.*

Question 18

What about the concept of method? Would it be appropriate to incorporate under OOP category in JLEO?

Answer: *The concept of method is one of the class variables in its minimal model. But, the types of methods such as constructor and its different variations, abstract methods, finalizer methods are to be incorporated in the OOP category of JLEO.*

Question 19

What about multitasking concept? Does it have the major role in JLEO?

Answer: *The concept of multitasking describes an operating level feature. However, the unique aspect of Multi-threading and its mode of realization in Java should be incorporated in OOP part of JLEO as it is indicated in the MID of OOP module. Hence, it is better to put multitasking first in JLEO for establishing semantic link with multi-threading.*

Question 20

Are the concept of package and its sub concepts appropriate to JLEO prototype?

Answer: *The package contains collection of classes and interfaces for re-usability. Two types of packages exist in Java language. Hence packages concepts empower the ways of organizing the set of related classes and interfaces in Java. Therefore, the concepts identified for package aspect in JLEO are appropriate.*

Question 21

What is your judgement to accommodate the identified stream and string concepts in JLEO?

Answer: *The stream concept is an aided feature used in input/output of java programming. This feature illustrates that the intermediate operations can be pipelined under the collections, arrays and the I/O channels. It is required to incorporate all I/O streams, readers and writers in JLEO. Same way, the concept of String and related sub concepts should be represented in JLEO.*

Question 22

How are the overloading, overriding methods likely to be equivalent to the static and dynamic properties in java?

Answer: *Generally, Overloading is related to compile time (or static) polymorphism. Hence the overloading method is more likely to be equivalent to static properties of polymorphism in java and these concepts are equated to similar concepts in JLEO. Similarly the Overriding method is related to the dynamic method dispatch wherein the call to the overridden method is resolved at runtime. This emphasis the equivalency property between overriding method and the dynamic property concepts in JLEO.*

Question 23

Do you think extra packages like Swing, JDBC should be unavoidable for the first version of JLEO prototype?

Answer: *The extra packages like Swing, JDBC are generally meant to be the advanced topics in terms of AWT and database connectivity. This relates to the enhanced version of java concepts. Since the current version of JLEO focusses on introductory concepts of Java, next version can think of incorporating advanced topics.*

Question 24

I have chosen Protégé as the platform for ontology design. Could that be a right choice in your experience?

Answer: *Yes… It is. Probably your chosen version supports OWL code generation.*

Question 25

Any other valuable comments from your end?

Answer: *All the very best for the JLEO development with your innovative software centric methodology.*

APPENDIX C Publications

1. *Development of an Educational Ontology for Java Programming (JLEO) with a Hybrid Methodology Derived from Conventional Software Engineering Process Models*

2. *Incremental and Iterative Agile Methodology (IIAM): Hybrid Approach for Ontology Design towards Semantic Web Based Educational Systems Development*

3. *Proposal of an Hybrid Methodology For Ontology Development By Extending the Process Models of Software Engineering*

4. *Software Centric Innovative Methodology for Ontology Development*

5. *Towards a Software Centric Approach for Ontology Development: Novel Methodology and its Application*