Coupling Ontology with Reference Architectures to Facilitate the Instantiation Process of Software System Architectures

Zaid Jafer Fadil Al-Bayati

School of Computing, Science and Engineering

College of Science and Technology

University of Salford, Salford, UK

Submitted in Partial Fulfilment of the Requirements of the Degree of Doctor of Philosophy

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

5SUF	5-Scale User Feedback
DL	Description Logic
FaCT++	Fast Classification of Terminologies
GQM	Goal/Question/Metric
IEEE	Institute of Electrical and Electronics Engineers
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
No	Number
OWL	Web Ontology Language
PTDR	Participant’s Tasks Development Results
PuLSE-DSSA	Product Line Software Engineering – Domain Specific Software Architecture
RA	Reference Architecture
RACER	Renamed ABox and Concept Expression Reasoner
SA	Software Architecture
SPSS	Statistical Package for the Social Sciences
T	Task
TOVE	TOronto Virtual Enterprise
UML	Unified Modelling Language
UPON	Unified Process for Ontology
XML	Extensible Markup Language
W3C	World Wide Web Consortium

ACKNOWLEDGEMENTS

First and foremost, I would like to thank ALLAH ALMIGHTY for giving me the patience and strength to conduct and complete this thesis. I would like to express my special appreciation and thanks to my supervisors Dr Adil Al-Yasiri and Prof. Farid Meziane, for their support, encouragement, and guidance throughout this PhD research journey, without their help, I could not have finished my thesis successfully.

I pass all my success to my father's soul. My deepest gratitude and appreciation to my mother, wife and children, brother and my sisters for their help and support in difficult times. Without their constant support, this PhD research would have been much difficult to complete.

I gratefully acknowledge the help provided by the Iraqi government represented by the Ministry of Higher Education and Scientific Research in funding this research. I owe a debt of gratitude to the University of Mosul in Iraq for the scholarship which allowed me to achieve the PhD. Thanks are also due to the Iraqi Cultural Attaché in London, for their continuing help and support whilst undertaking this research. I would like to thank the University of Salford and their staff for all the help and the support provided during my PhD studies, as well as, all my colleagues in the College of Science and Technology at Salford University.

Finally, I would like to say thanks to everyone who helped me achieve success in this research.

Zaid Jafer Fadil Al-Bayati

PUBLICATIONS

The following are the list of publications during the studying years:

1- Zaid J. Fadil Al-Bayati, Adil Al-Yasiri. Coupling Ontology with Reference Architectures to Facilitate the Instantiation of Software System Architectures, University of Salford, College Dean’s Annual Research Showcase Event, May-2015.
2- Zaid J. Fadil Al-Bayati, Adil Al-Yasiri. Coupling Ontology with Reference Architectures to Facilitate the Instantiation of Software System Architectures, University of Salford, Postgraduate Annual Research Conference, June-2016.
3- Zaid J. Fadil Al-Bayati, Adil Al-Yasiri. Ontology-based Approach to Represent the Artefacts of Reference Architecture. University of Salford, Annual PGR Symposium, March-2017.

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Signature: ____ Date: ___/___/_______

Zaid Jafer Fadil Al-Bayati

ABSTRACT

A reference architecture can be defined as a generic architecture for a set of information systems that is used as a basis to a software system architecture. It provides the foundation for the design of concrete architectures in terms of architectural design guidelines and architectural elements. In addition, it can be used by many software developers and architects to design software system architectures' instances that best fit their customers' requirements.

Software system architectures play an essential role in defining the achievement of software systems. Therefore, it can be derived efficiently from a well-structured reference architecture. There is a lack of a well-defined methodology that instantiates the knowledge of the reference architecture to a clear and customised software system architecture. Consequently, the instantiation process of the software system architecture from the reference architecture is a difficult task because the reference architecture includes a huge amount of knowledge. This knowledge is not organised and not structured. In addition to that, there is no standard terminology used to describe the knowledge of the reference architecture.

To tackle this issue, a mixed method research approach has been adopted in this research. In order to achieve the aim and objectives of the research, a qualitative approach, utilising multiple case studies, has been adopted to collect the qualitative data, and a quantitative approach, utilising survey questionnaires, has been adopted to collect the quantitative data.

This thesis aims at facilitating the instantiation process of the software system architecture from the reference architecture by using an ontology. The ontology has been used as a tool to present the artefacts of a reference architecture in an organised and structured way.

General vocabularies have been defined based on understanding the domain and the literature and by using multiple case studies from the literature. These vocabularies have been utilised as a basis to construct an ontological model. The ontological model will be utilised to organise and structure the artefacts of the reference architecture. It aims at providing vocabularies to software architects and developers to reduce the misunderstanding between them. Furthermore, to enable clear communication between software architects and developers and to achieve the unique representation of concepts by avoiding redundancies.

User study has been adopted to evaluate the usability of the proposed methodology in term of the simplicity of the instantiation process of the software system architecture from the reference architecture. The results achieved in the evaluation study offered an evidence that the ontological model can positively affect the development of software system architectures. In addition to that, it can reduce the development time. Based on the final evaluation results, it can be concluded that our research has been successful in introducing the proposed methodology as a new idea to reduce complexity in the development process.

CHAPTER ONE: INTRODUCTION

1.1 Overview

The fact that each system can be shown to be composed of components and relationships between them mandates that there is an architecture for every system [1]. A software architecture is considered as the backbone for any successful software system [2], [3]. It performs an important function in determining the system quality [4],[5] and it plays a significant role in determining the success of the software system.

Software architecture (SA) is a significant step in the software development process. It represents “the structure(s) of the system, which includes software elements, the externally visible properties of these elements and the relationships among them” [6]. The software architecture is a set of explicit architectural design decisions made about the software system over time [7].

Nowadays, the complexity and size of information systems demand new software engineering methods to develop software system architectures [8]. There are different approaches to design software system architectures. They can be designed from scratch; however, this will take a long time while it is possible to invest this time in another part of the development process. Using a reference architecture (RA) is considered as another method of the design process, it allows knowledge and components to be systematically reused when developing a software system architecture [9], [10]. The reference architecture can be defined as a generic architecture for a set of information systems in the domain that is used as a basis to develop software system architectures.

There are many benefits when using the reference architecture to design the software system architecture; it increases the productivity of application builders, saves the costs of maintenance of the applications and decreases the development time. Furthermore, faster delivery of applications is another advantage of the reference architecture [11]. In this context, the reference architecture is defined as “a general architecture for a set of information systems that are used as a basis for the design of software architectures” [12]. It can guide the development, standardisation and evolution of systems’ architectures in a particular domain [13], [14].

The instantiation process of software system architectures from the reference architecture is not an easy task [12], [14], [15]. Furthermore, the inclusion of reference architectures in the current software processes of an organisation is also not a trivial task [14]. The reason behind that is that it encompasses a huge amount of knowledge. However and most of the times, this knowledge is almost non-structured and non-organised too [16], in addition to that, there is no standard terminology to describe the artefacts of reference architectures.

To tackle this issue, the researcher adopted a mixed method (Qualitative and Quantitative Approaches) to achieve the aim and objectives of the study. This study includes two phases: design and evaluation. In the design phase, a qualitative research approach has been conducted. A qualitative data was collected from multiple case studies and analysed by the researcher by reviewing it. In the evaluation phase, a quantitative research approach has been conducted. A quantitative data was collected by conducting a survey questionnaire and analysed by using SPSS software and Microsoft Excel. Hence, this research proposes a methodology to facilitate the instantiation process of the software system architecture from a reference architecture by using an ontology. The ontology was used as a tool to present the artefacts of the reference architecture. The reason behind that is the definition of the ontology itself.

According to Gruber [17], an ontology is defined as “a formal and explicit specification of a shared conceptualisation”; this definition shows that an ontology can play essential roles in solving many software engineering development problems. The structure of an ontology includes a set of classes in addition to the associated relationships between these classes.

General vocabularies were defined based on the literature and understanding the domain and by using multiple case studies from the literature. The vocabularies were used to construct a general ontological model. The ontological model was developed based on the general vocabularies to provide vocabularies of a reference architecture for software developers and architects to facilitate the instantiation process by tracking the relationships between the components to find another component.

The proposed methodology has been evaluated by using a user study experiment. The user study experiment was adopted to measure the usability of the proposed methodology in term of the simplicity of the instantiation process and the development time. The experiment was conducted at Salford University – School of Computing, Science, and Engineering. Two groups of developers were employed in the evaluation process.

The evaluation results found that the ontological model facilitated the instantiation process of a software system architecture from a reference architecture. All participants that used the ontology found out that it helps to design the software system architecture with reduced development time.

1.2 Research Problem

Software architecture plays an important role in the software development process and in determining the system quality [4], [5], [18]. Despite the impact of the architectures on the software development process and the system quality, there is not yet an agreement about a description method of these architectures [4].

The software system architecture design is very complex [19] because of the involvement of non-functional requirements in the development process [20], [21]. The current design methods lack specificity and preciseness. Consequently, it is extremely difficult to develop a comprehensive and proper software system architecture [21]–[23].

The reference architecture can be adopted as a way to reuse architectural knowledge when designing a new architecture for a software system. It contains the essence of the architecture of a set of similar systems [10]. Presently, most software system architectures are developed in a particular method without a well-organised approach to creating, preventing the creation and maintenance of the applications [24], [25].

According to [14], [15], [26], [27] the instantiation process of the software system architecture from the reference architecture is not an easy task and there is no straightforward method of the instantiation process; therefore, methods and techniques that systematise such task are important. Moreover, the inclusion of the reference architectures in the current software development processes of an organisation is not trivial. Thus, it is extremely difficult to develop a comprehensive and appropriate software system architecture even though it is recognised as primary artefacts [21].

Reference architectures have been developed for various domains. They encompass a huge amount of knowledge. However, and most of the times, this knowledge is almost non-structured and non-organised too [28], besides; it is presented as informal and semi-formal too [29]. In addition, since there is no standard terminology to describe the artefacts of reference architectures [5], that led software architects and engineers to use their vocabularies to describe the artefacts. This issue makes it not clear and not understandable by a variety of stakeholders [16].

1.3 Research Aim and Objectives

This study aims to propose a methodology to facilitate the instantiation process of a software system architecture from a reference architecture by using an ontology as a tool, to present the knowledge, relationships and attributes of the reference architecture differently. Eventually, the proposed methodology tries to decrease the complexity of the architectural development process. In addition to the aim of this research, some objectives have been highlighted.

1- Review the development approaches of a reference architecture to highlight the tools that are used to present the artefacts of a reference architecture.
2- Review the existing instantiation process of a software system architecture from a reference architecture to highlight the shortage of the current instantiation process.
3- Review an ontology principle.
4- Define general vocabularies to be used for constructing an ontological model.
5- Develop an ontological model for presenting knowledge about the reference architecture.
6- Develop a process in order to describe the artefacts of a reference architecture.
7- Evaluate the proposed methodology by conducting a user study experiment.

1.4 Research Contributions

During this research, the literature review has addressed the role of ontology in software engineering. However, it did not report its role in presenting the artefacts of reference architectures. Thus, we conduct a study to identify the major problems related to the reference architecture design, as a consequence of this study, we have identified that currently there are no existing methodologies that can help software architects to describe a reference architecture formally and we identified they do not have a standard vocabulary. Therefore, ontology has been utilised as a tool to present the artefacts of reference architectures. The reason behind that is the definition of ontology, which refers to a formal way of knowledge representation, and it encompasses concepts and relationships.

General vocabularies were defined based on understanding the domain and the literature and multiple case studies. These vocabularies present the general aspect of reference architectures. They were utilised to construct an ontological model. The ontological model includes general vocabularies with relationships between them. It has been used to organise and structure the artefacts of reference architectures.

The ontological model aims to provide vocabularies to software developers so as to facilitate the instantiation process of a software system architecture from a reference architecture. Furthermore, it has been used to guide the developer in the development process by tracing components in the domain then by determining the relationships among these components. As a result, the ontological model simplifies the instantiation process of the software system architecture and saves delivery time.

1.5 Research Process

As mentioned previously, the aim of this research is to propose a methodology to facilitate the instantiation process of a software system architecture by using an ontology as a tool to present the artefacts of a reference architecture.

To achieve the aim and objectives of this research, a mixed method approach was adopted to collect the data. Multiple case studies have been used to collect qualitative data and survey questionnaires were used to collect quantitative data. The researcher begins by selecting the research area. Next to that, many studies have been reviewed on software architecture, reference architecture, reference architecture development approaches, software architecture instantiation process, and ontology to gain a full understanding and to know the state of the art of the instantiation process, and also, to highlight the shortcomings of these processes. This was followed by formalising the research problem. After that, the researcher proposed a solution for the identified problem. The proposed solution was evaluated by conducting a user study experiment. Then, the proposed methodology was improved and modified according to the result of the evaluation process. The final step of the present research is the conclusions and recommendation. The main stages of the research process are illustrated in Figure 1-1.

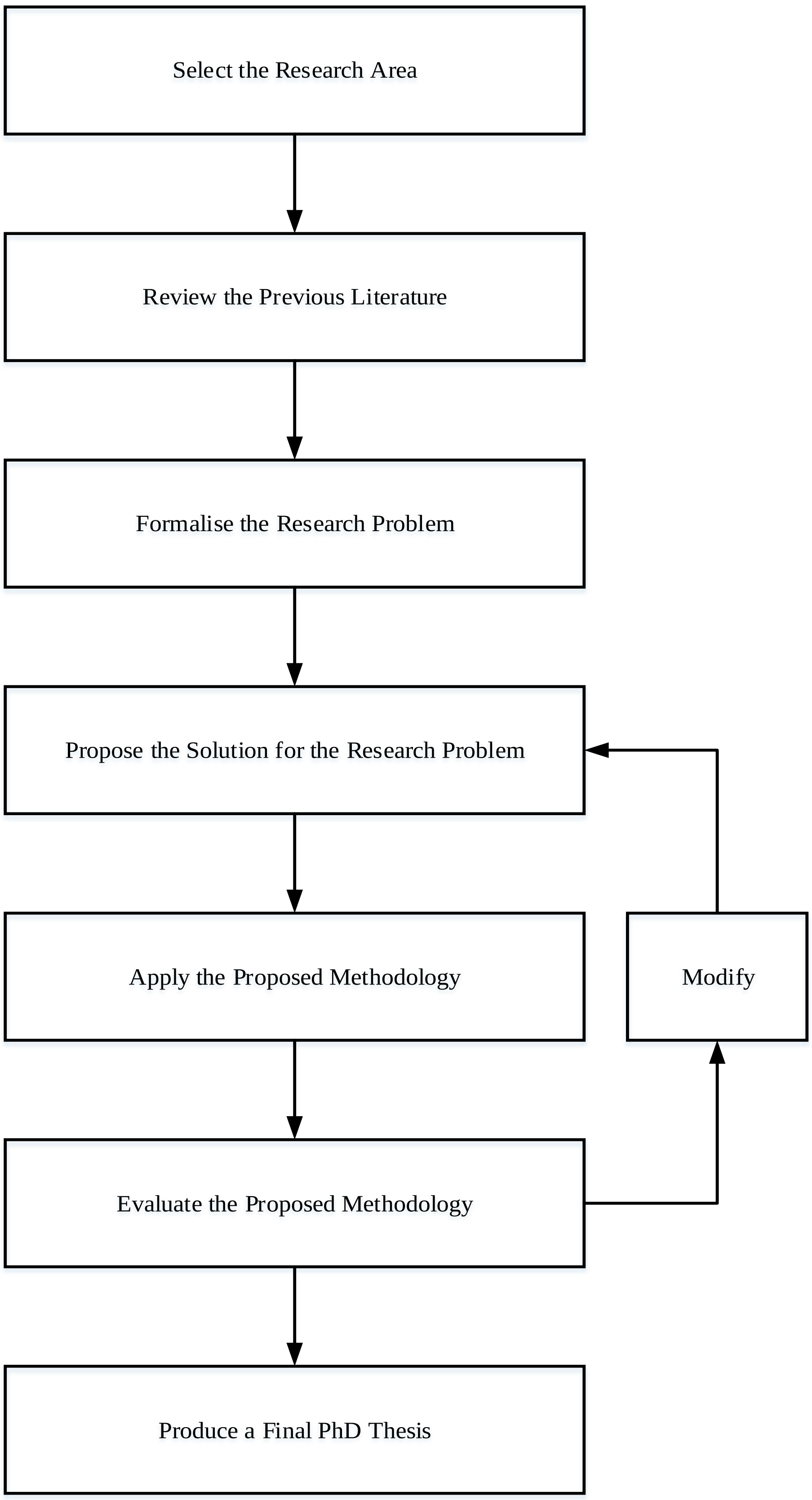

Select the Research Area
Review the Previous Literature
Formalise the Research Problem
Propose the Solution for the Research Problem
Apply the Proposed Methodology
Modify
Evaluate the Proposed Methodology
Produce a Final PhD Thesis

Figure 1-1: Research Process

1.6 Outlines of the Thesis

In order to achieve the aim and objectives of this research as described in Section 1.3, this thesis has been divided into seven chapters. Each chapter describes a major component of the research.

CHAPTER ONE: INTRODUCTION

The first chapter describes the problem and the aim and objectives of the research as well as the contributions of this study. Finally, it illustrates the research process followed to achieve the aim of the research.

CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW

This chapter provides a background and literature review that supports the topics investigated in this thesis. Initially, the definition of the software architecture is illustrated, then the description of the reference architecture is discussed, followed by clarifying the difference between the software architecture and reference architecture. After that, the instantiation process of software system architectures from a reference architecture is explained. The chapter also describes the ontology principle, which covers definition, components, representation languages, development tools, and design methodologies of ontology. Finally, it shows the role of ontology in software engineering.

CHAPTER THREE: RESEARCH METHODOLOGY

This chapter exhibits the research methodology which was adopted to achieve the aim and objectives of the research. It describes the type and model of the research followed by the ethical consideration.

CHAPTER FOUR: DEVELOPMENT OF GENERAL ONTOLOGICAL MODEL

This chapter demonstrates the overview of the development process of the ontological model. It explains the steps of the development process. Furthermore, it shows the case studies which are used to define the general vocabularies followed by the validation process of the defined general vocabularies. Finally, the defined general vocabularies have been used as a basis to construct the general ontological model.

CHAPTER FIVE: USING ONTOLOGY FOR PRESENTING THE ARTEFACTS OF REFERENCE ARCHITECTURE

This chapter describes the process of using the ontological model to develop an ArchiOntology. The ArchiOntology presents the artefacts of a reference architecture formally followed by the process of using the ArchiOntology. Next to that, the process has been illustrated within two examples.

CHAPTER SIX: EXPERIMENTAL EVALUATION OF PROPOSED METHODOLOGY

This chapter explains the evaluation process of the ontological model and the discussion. A user study experiment has been used to evaluate the ontological model. It also shows the criteria and metrics that are used in the evaluation process.

CHAPTER SEVEN: CONCLUSION AND FUTURE WORK

This chapter demonstrates the achieved conclusions of the study. It also discusses the research outcome and finally outlines the possible future work.

APPENDICES

The appendices are used to include extra data and detail which it is not possible to include in the body of the thesis.

CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW

2.1 Overview

This chapter presents an overview of the topics that underlie the research developed in this thesis. The organisation of the chapter is as follows. Section 2.2 shows the main concept of software architecture. Section 2.3 describes the term reference architecture in details. Section 2.4 presents the difference between software architecture and reference architecture. Particularly, Section 2.5 characterises the state-of-the-art of the instantiation process of a software system architecture. Section 2.6 shows the ontology principle. Section 2.7 demonstrates the role of ontology in representing the architectural knowledge.

2.2 Software Architecture

Software architecture is an essential step in the development process of software. It is the outcome of the architectural decisions made during the development process of architecture [30]. It is one of the many valuable artefacts in software development, no definition of software architecture is commonly accepted upon [20], [31]. However, it is accepted that the software architecture is concerned with elements of the system and their interactions and properties [30]. Most people agree that the primary concern of the software architecture is, the high-level structure [32].

According to the literature, there are various definitions of the software architecture (SA). A brief definition is given by Garlan and Shaw [33] that SA is an organisational structure of a system that includes components, connections, constraints and rationale. Bass et al. [30] described the SA as “the structure or structures of the system, including software components, the properties of the components that are externally visible and the relationship among them”. It helps to understand, reuse, construction, evaluation, analysis, and management of the systems [34], [35].

The Institute of Electrical and Electronics Engineers (IEEE) defines the SA in the “IEEE Std 1471-2000” standard, as follows: "software architecture is the essential organisation of a system represented in its components, their relationships, the environment and the principles guiding its design and evolution" [36]. Additionally, Jansen and Bosch [7] defined software architecture as “a collection of explicit architectural design decisions made over time”.

2.3 Reference Architecture

Reference architecture has developed as an important area of research in software architecture. It is considered a blueprint of software development since it guides the design of software system architectures for a given application domain [5], [37], [38]. RAs can directly impact on the quality and design of a range of concrete architectures and software systems developed from them [39]. Therefore, they must consider the best practices of software design, architectural styles, business rules, and software components that support the development of systems of the application domain. Furthermore, RAs must be supported by an unambiguous, unified, and widely understood domain terminology [40].

Different institutions in both industry and academia have already developed and used RAs in various application domains. There are examples of RAs developed for Situated Multiagent Systems [41], Mobile Learning Environments [42], Cloud Computing [43], Web Servers [44], Sensor Networks Integration and Management [45], Ubiquitous Computing

[46], Web Browsers [47], Robot Teleoperation [48], and so forth. However, software architects and engineers use their terminology to describe the artefacts of RAs because there is no standard vocabulary to describe the artefacts of reference architectures.

According to Muller [38], an RA can be used to facilitate the development of SAs or as a standardisation asset that supports interoperability between systems or elements of systems. Figure 2-1 shows the same RA can result in different concrete architectures, depending on the context and involved stakeholders.

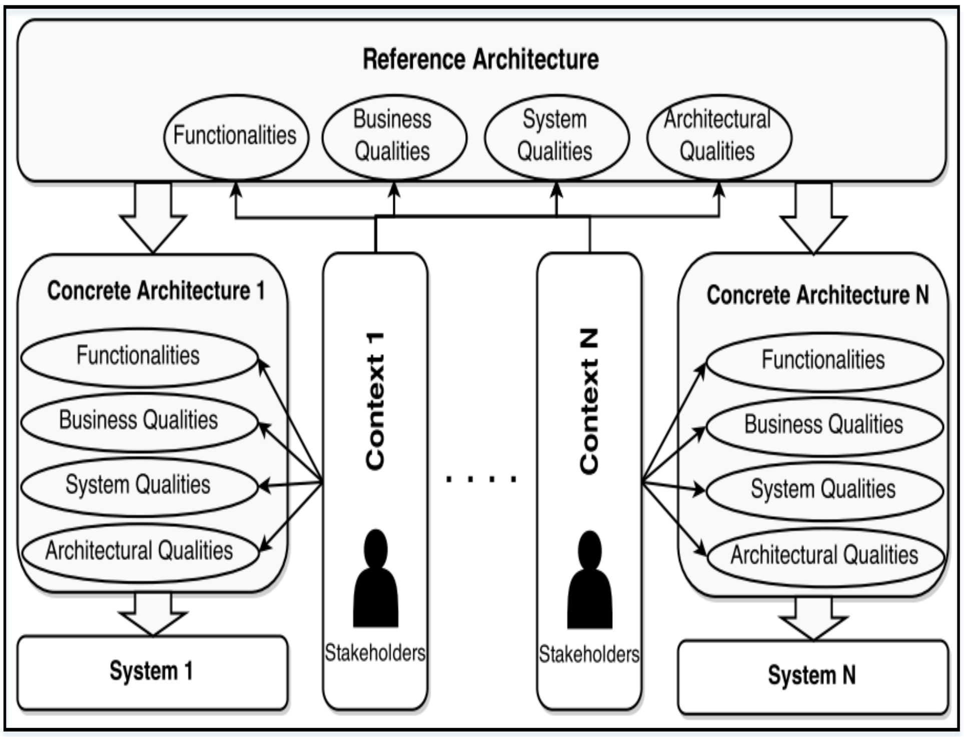


Figure 2-1: Role of Stakeholders and Contexts for RAs and Concrete Architectures [37]

Due to the variety of application domains and interests, RAs can be classified according to three dimensions as described below [12], [39].

- **Context dimension**: RAs can be developed in the context of a single organisation or multiple organisations that share a common characteristic, such as geographical location and market domain. Various types of organisations (e.g., software organisations, user organisations, research centres, and standardisation organisations) are usually included in the establishment of these architectures. Besides, such architectures can be developed before any existing systems or after accumulating the experience from the development of several systems.

- **Goal dimension**: RAs can be developed with two main goals: standardisation and facilitation. Reference architectures for standardisation aim at improving interoperability among systems by promoting a unified understanding of the domain at the architectural level. On the other hand, facilitation RAs aim at providing guidelines for the design of concrete architectures.
- **Design dimension**: RAs are represented by several types of elements, including software components, interfaces, protocols, algorithms, policies, and guidelines.

2.3.1 Review of Reference Architecture Development Approaches

In this section, we will review a number of approaches to developing reference architectures. There are different works describing the development process of reference architectures as shown in Table 2-1. The RAs involve software organisations, user organisations, standardisation organisations, and research centres [12], [38]. They can be the basis for several software systems, studies have concentrated on the development of this type of architectures.

DeBaud et al., [49] proposed a Product Line Software Engineering – Domain Specific Software Architecture (PuLSE-DSSA). PuLSE-DSSA constructs a reference architecture by: generate scenarios from requirements; categorise the scenarios based on variability, structure, and priority; develop initial architectures for the structure-based scenarios; rank the architectures based on coverage of scenarios; build architecture prototype; select best architecture; and evaluate the architecture.

Avgeriou [15] described an approach to document, apply, and evaluate a RA that is based on a combination of the IEEE 1471 standard for “Recommended Practice for Architectural Description of Software-Intensive Systems” [36], the Rational Unified Process [50], and the Unified Modelling Language (UML) [51], [52].

Dobrica and Niemela [53] presented a procedure for designing RAs. The approach encompasses two phases. These phases are design description and architectural development.

Galster and Avgeriou [10] suggested an approach to develop and evaluate RAs which includes six-step for designing RA, including the decision on what type of RA to develop, the selection of a development strategy, the empirical acquisition of data, the development of the RA, enriching the RA with variability, and the evaluation process.

Muller [38] illustrated a set of recommendations in order to design and maintain reference architectures; shortly, RA must be acceptable and understandable for a vast set of stakeholders, up-to-date, accessible and actually read by majority of the organisation, address the main issues of the specific domain, satisfactory quality, add value to the business, and maintainable.

Angelov et al. [12] proposed a framework for creating and analysing reference architectures. They suggested that a reference architecture documentation include information about the context, aims and development decisions. The context dimension covers the purpose, the organisation(s) that is (are) designing an RA and its maturity stage.

Nakagawa et al. [54] proposed a process called PROSA-RA to design, representation, and evaluation of RAs which includes four steps: investigate the information source, analyse the architecture, synthesis the architectural, and evaluate the architecture.

The consensus for presenting the reference architecture is through the use of standardised diagrams such as UML and other architecture description languages and describing the architecture through different viewpoints to cover the concerns of stakeholders in the system [6]. UML [51] diagrams for RAs abstract the implementation details of a system and show the relationships between the elements of a system [52].

All these studies have described valuable guidelines for the design of RAs. However, reference architectures for different domains represented by informal notation and semi-formal languages. For example, the reference architecture of the web browser [47] described by informal technique and only the main components and connections between them. Figure 2-2 shows the subsystems of the web browser with their connections, which are presented informally. Also, Arch-int et al. [55] used an informal notation to describe a reference architecture for interoperating existing e-Learning systems.

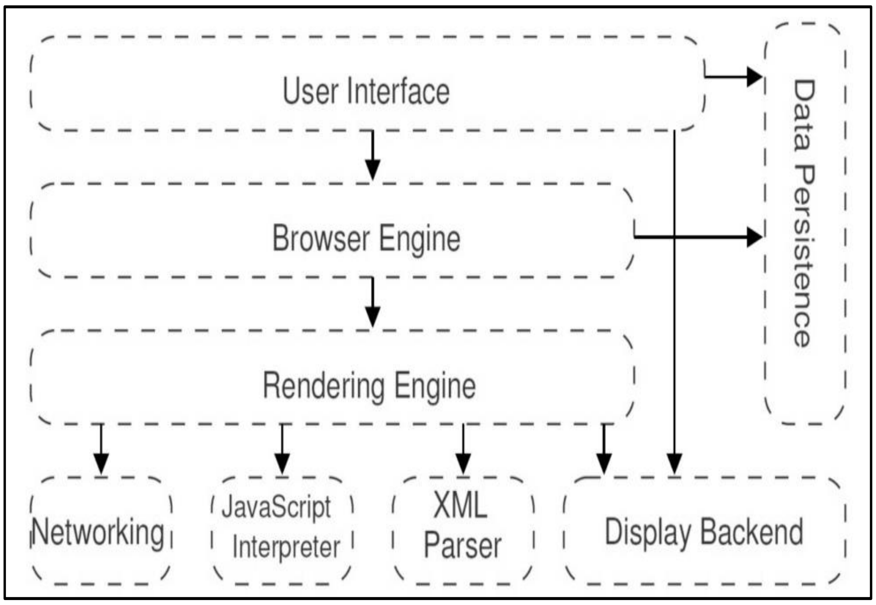


Figure 2-2: Reference Architecture of a Web Browser [47]

Nakagawa et al. [56] developed the reference architecture of the software engineering environments. They described the artefacts of the RA semi-formally by using UML to present the artefacts as shown in Figure 2-3.

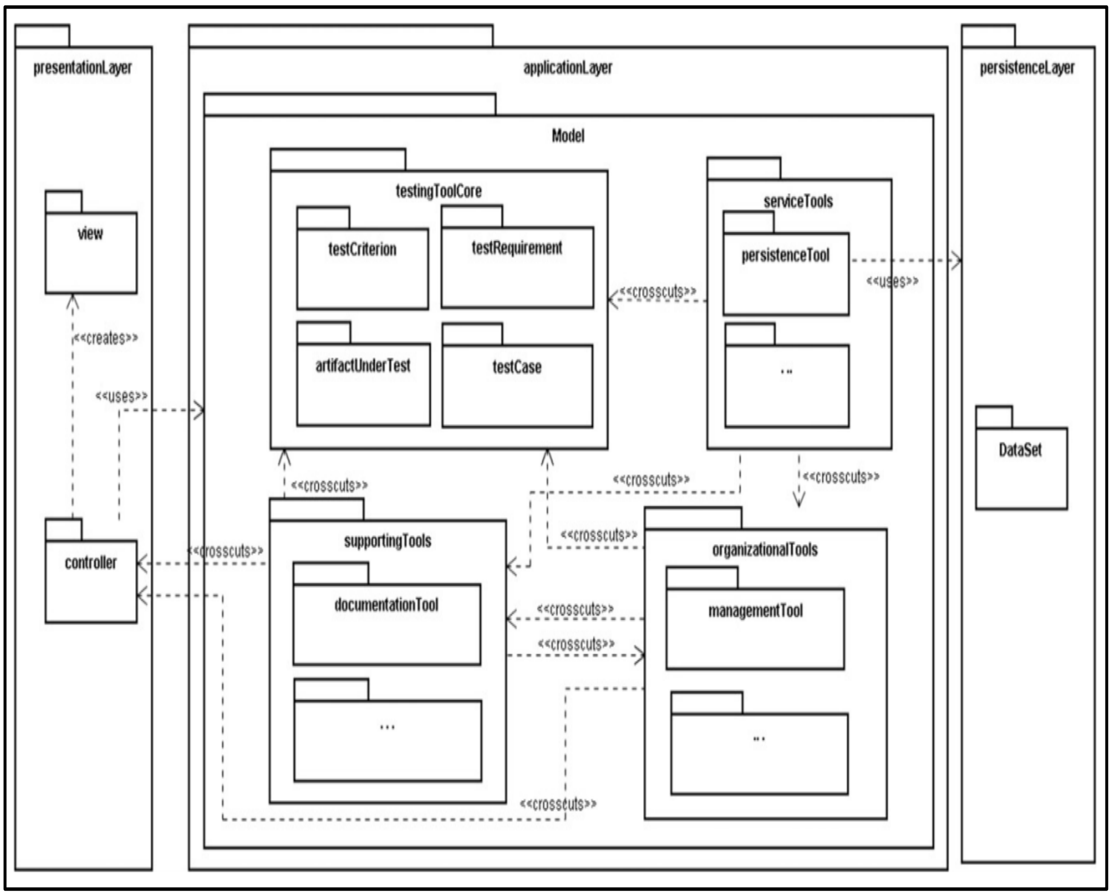


Figure 2-3: Reference Architecture of a Software Engineering Environments [56]

Table 2-1: Reference Architecture Development Approaches

Author(s)	**Year**	**Title**	**Description tool**
DeBaud, Oliver & Knauber	1998	PuLSE-DSSA—A Method for the Development of Software Reference Architectures	Informal description (shapes and arrows)
Avgeriou	2003	Describing, instantiating and evaluating a reference architecture: A case study	Semi-formal description (UML)
Dobrica & Niemela	2008	An Approach to Reference Architecture Design for Different Domains of Embedded Systems	Informal description (textual)
Galster & Avgeriou	2011	Empirically-grounded Reference Architectures: A Proposal	Informal and Semi-formal description (UML + textual)
Muller	2012	A reference architecture primer	Informal description (textual)
Angelov, Grefen & Greefhorst	2012	A Framework for Analysis and Design of Software Reference Architectures	Semi-formal description (UML)
Nakagawa, Guessi, Maldonado, Feitosa, & Oquendo	2014	Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures	Semi-formal description (UML)

2.3.2 Reference Architecture Artefacts

Reference architecture illustrates the infrastructures of the end systems. It is then refined to design an SA for a particular system [57]. The infrastructures of the RA have received little attention [12]. However, several works in the reference architecture literature illustrate the artefacts that could be used to create software systems based on the RA; these artefacts are also identified as an infrastructure [5].

Different authors stated significantly various views about the artefacts of the RA. They also explored artefacts of the reference architectures, to study when they are also presented in the RAs. Cloutier et al. [9] claimed that architectural information is the main part of the RAs. They mentioned as common components of the RAs: standards, implementing, business purpose, and guidance for a roadmap. Galster and Avgeriou [10] referred to that the basic structure of the RA consists of its common building blocks (model kinds, common stakeholders, views) according to ISO/IEC 42010 [58]. Angelov et al. [12] differentiated protocols, algorithms, components and connectors, interfaces, and policies and guidelines. Nakagawa et al. [5] mentioned that the RA infrastructure provides: software components that are used to design software systems, the general structure typically described by architectural styles, hardware components that host software systems based on the RA and guidelines, which show how to implement best practices. Herold et al. [59] recognised the following artefacts in the RA: reusable elements of software, operation platform, methodology, tools, and blue-line prints. Martinez-Fernández et al. [13] observed that the artefacts that constitute the reference architecture include software elements, guidelines, and documentation**.**

2.3.3 Benefits of Reference Architectures

According to Martinez-Fernández et al. [60] and Affonso et al. [61], the principal inducement behind Reference Architecture is, to facilitate reuse, reduce development cycle times, cost and risk. Furthermore, increase quality, as well as to assist in the development of a collection of systems that are designed from the same RA and to ensure standardisation and interoperability. Moreover, it provides guidance when designing systems for a particular application [62]. Furthermore, Angelov et al. [63] claimed that using the RA can assist organisations and researchers to:

A. Provide best practices.
B. Speed up design task.
C. Establish a standard architecture approach in the organisations.
D. Ensure reusability.
E. Ensure interoperability with another system.
F. Comply with standards.
G. Improve communication between different stakeholders.
H. Decrease development costs of new projects and provide an inspirational tool to designers.
I. Structure the task of architects.
J. Help developers to understand the systems.

Moreover, reference architectures provide a plan for building a system and reduce the cost of maintenance. It presents an overview description of the system. As well as, it allows software developers to view the main subsystems in the software system and the relations between them [44].

2.4 Differences between Reference Architecture and Software Architecture

There are some differences between reference architectures and software architectures as shown in Table 2-2. The RAs are developed to address the functionalities and qualities desired by all stakeholders in their particular contexts. There is not a distinct group of stakeholders for the RA. On the other hand, stakeholders of the SA are specific [15], [37].

The RAs are defined on a high level of abstraction due to their generic nature, while the SA has to address less architectural qualities than the RA. These additional architectural qualities are due to the generic nature of the RAs and their wider audience [37]. While the RAs cover all components of a domain, the SA includes only components for a particular application [10].

The requirements of the RAs and SAs are different. The requirements of the RAs must be obtained considering more diverse information sources. Furthermore, there is an inherent difficulty in capturing requirements that competently represent the entire domain. Consequently, methods and ways to capture requirements of the RA are also different, if matched with those used to extract requirements of the SA [64].

The RA is defined as the architecture for a set of application systems, whereas an application architecture is defined as the architecture for a single system [65]. The RA, on the other hand, describes a blueprint architecture that can be used to design software architectures in a particular domain. As a result, the RA cannot be evaluated in the same way as a software architecture [66].

Table 2-2: Differences between SA and RA

Reference Architecture	**Software Architecture**
Stakeholders of Reference Architectures are of generic nature.	Stakeholders of Software Architectures are specific.
Reference Architectures address wider architectural qualities due to their generic nature and wider audience.	Software Architectures have to address less architectural qualities than the reference architecture
Reference Architectures cover all components of a domain.	Software architectures include only components for a particular application.
Reference Architectures should have more diverse information sources; there is an inherent difficulty in capturing requirements that competently represent the whole domain and the methods and ways to capture requirements of a reference architecture are also different.	The requirements of Software Architectures are more specific, and it is easy to capture it.
Reference Architecture is defined as the architecture for a family of application systems.	Software architecture is defined as the architecture for a single system
There are no specific evaluation methods to evaluate Reference Architectures.	There are many evaluation methods to evaluate Software Architectures.

2.5 State-of-the-Art of the Instantiation Process of a Software System Architecture from a Reference Architecture

The instantiation process of a software system architecture from a reference architecture can be defined as an application engineering [65]. The application engineering is “a process of designing a particular application making use of the domain knowledge obtained during domain engineering” [67]. In this section, the researcher tried to show the processes that are used by other researchers to instantiate a software system architecture from a reference architecture.

In a different method, Avgeriou [15] claimed that the instantiating of a reference architecture was possible by using an integration of the IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems [36], and the widely adopted Rational Unified Process [50]. The instantiation process includes seven steps. Firstly, define the stakeholders of a system and any concerns that they may have in terms of any possible aspect of the system. Secondly, define the viewpoints that will explain the stakeholders’ concerns and describe the methodology used. Thirdly, define the views that are used to represent the components of the system. Fourthly, define the architectural patterns that describe components of the architecture. Fifthly, describe the quality attributes that are needed for the system. Sixthly, describe the implementation constraints. Finally, describe other issues that are necessary for the particular system being designed.

Weyns and Holvoet [68] used the Attribute-Driven Design [69] with the reference architecture to instantiate/refine a software architecture. The RA is used as a guideline in the decomposition process. The design process included several steps: identify the requirements of the system, order the requirements, design the software architecture, evaluate the software architecture, implement the application, and test to verify the fundamental system requirements.

Suganthy and Chithralekha [67] utilised a Domain Specific Software Architecture-based application engineering process for building an application. The process includes three steps. The first step is identifying application requirements. The second step is designing the application. The final step is implementing the application.

Pérez-Sorrosal et al. [26] described the instantiation process in three phases. Phase 1: Confront pattern assumptions with initial architecture. Phase 2: Pattern selection through trade-off analysis and Phase 3: Evaluation of quantitative requirements fulfilment.

According to Sala [70], the instantiation process can be done in four phases. Phase 1: Identifying specific users for the target software architecture. Phase 2: Identifying particular interconnection and interaction among the users in the target concrete architecture. Phase 3: Identifying the component model of the reference architecture by using this component model. Finally, the software architecture is implemented.

Nakagawa et al. [14] mentioned steps to instantiate a software system architecture from a reference architecture. Briefly, six stages need to be done to instantiate the software system architecture. Firstly, reading and understanding the reference architecture documents. Secondly, developing a software system by selecting either the entire architecture or those parts that are interesting and already present in the software system. Thirdly, instantiating the architecture or their parts, using the characteristics of a software system, including requirements, constraints, and the context of applications. Fourthly, documenting the architecture. Fifthly, evaluating the architecture. Finally, implementing the architecture.

Oliveira et al. [25] designed a software architecture for the Service Oriented Robotic System. They split the design method into five phases, which can be applied iteratively. Phase 1: Identifying the requirements and characterising the application. Phase 2: Recognising the skills that the system should present. Phase 3: Designing the architecture. Phase 4: Describing the functions. Phase 5: Evaluating the architecture.

All the processes mentioned above are considered general processes and software developers need to be very experienced in designing software architectures. These processes do not provide a concrete tool to show the artefacts of reference architectures. The instantiation process of software architecture is not a straightforward process [15] and not a trivial task [26], [14]. The effective instantiations are scarce [71].

2.6 Ontology Principle

2.6.1 Ontology Definition

Based on the literature, ontology can be defined as a formal means of knowledge representation. There are various definitions for the term ontology. Neches et al. [72] defined it as a set of basic relations and terms constituting the vocabulary of a topic field in addition to certain rules that combine those relations and terms to define extensions to the vocabulary. Gruber [17] provided one of the most cited definitions of an ontology as " An Ontology is an explicit specification of a conceptualisation"; this definition illustrates the role that ontologies can play to solve most software engineering problems. Such conceptualisation offers access to an abstract model pertaining to some phenomenon that could identify the pertinent notions of that very phenomenon.

Other definitions had also been proposed based on the definition of Gruber. For instance, Borst included two more requirements to the definition pertaining to ontology and those are: 1) Formal; which means that a machine is to process the ontology, as well as 2) Sharable; that means having a consensus on the knowledge acquired by the community of experts. According to Borst, ontology is defined as "a formal specification of a shared conceptualisation" (Borst 1997 cited in [73]).

A general definition (Uschold and Jasper 1999 cited in [73]) declares that an ontology may take a variety of forms but will necessarily include a vocabulary of terms and some specification of their meanings. This contains definitions and indications of how concepts are connected which collectively impose a structure on the domain and constrain the possible interpretations of terms.

As far as computer science is concerned, an ontology denotes resources pertaining to computer-science and ones which signify domain semantics that are agreed-upon. An ontology is comprised of relatively generic knowledge which an alternate task or type of application can further reuse [74].

From the definition and literature, ontology is a formal means of knowledge representation; it can also contribute to enhancing software development processes and modelling. It extremely affects all phases of software development such as analysing, designing and implementing. In this study, the researcher aims to use the ontology as a tool to present the artefacts of a reference architecture formally to facilitate the software development process.

2.6.2 Ontology Components

A number of knowledge representation languages exist for ontology implementation. Each of them gives various components that can be used in developing the ontology. However, the following minimal set of components is shared between ontology representation languages [75]. According to Calero, et al. [75], the main elements of an ontology are:

Classes: These describe concepts which are taken in a broad sense. Classes in ontology are usually organised in hierarchal taxonomies through which inheritance mechanism can be applied. Classes can include individuals, other classes (sub-classes) or a combination of both. Ontologies vary in whether they include a universal class (a class that contains everything) or not. In Web Ontology Language (OWL), the concept is represented as a class.

Relations: These represent a type of link between concepts of the domain. An ontology usually contains ordered binary relations where the domain of relations is represented by the first argument while the range is represented by the second argument. For example, the binary relation '*drives*' has the concept '*Person*' as a domain and the concept '*Car*' as the range.

Sometimes "Binary Relations" are utilised so as to refer to concept attributes; the latter usually have their own range as a datatype such as string, number, and so on. In OWL, relations are named Object Properties while attributes are named Datatype Properties. It is in order to describe the ontology's individuals or elements that "Instances" are used. Instances (or individuals) are the basic components "ground level" of the ontology. For example, 'Tom' is an instance of the class 'Person'.

Formal Axioms: These are model sentences that are always true. Formal axioms are used to infer new knowledge and to verify the conciseness of the ontology [17]. An axiom in the travelling domain could be that it is not possible to travel from North America to Europe by train.

2.6.3 Ontology Representation Languages

There are many languages available for ontology representation. In the 1990s, ontologies were constructed using mainly Artificial Intelligence modelling techniques. Such languages were based on:

- First order logic such as Knowledge Interchange Format [76].

- Frames combined with first-order logic such as Cyc ontology [77] and Ontolingua (Farquhar et al., 1997 cited in [73]).
- Description logic such as LOOM [78].

Well ahead, an ontology language was introduced due to the Internet and its revolutionary advancement. This language could take advantages of the features of Web-Based ontology or ontology markup language that is also termed as Web-based ontology languages [75]. The most important examples of these markup languages are: Resource Description Framework (RDF) [79], DAML+OIL [80] and OWL [81]. Out of all of them, RDF and OWL are the ones that are being actively supported now. Even though RDF is developed long before the Web, the serialised version of RDF(s) in Extensible Markup Language (XML) makes its way to the Web since the Web is based on XML. A detailed classification and review of ontology representation languages can be found in [73].

The current research opted to choose the OWL out of all known ontology representation languages. As much as W3C is concerned, this language is recently the primarily-recommended ontology language. The OWL knowledge representation can allow properties as either ObjectProperty (relation) or DatatypeProperty (attribute). It can also define objects as classes, and individuals (instances) of different classes. Additionally, it provides the chances to reason about individuals and classes. The OWL provides three sub-languages: OWL fully-ordered with increased expressiveness, OWL DL and OWL Lite.

2.6.4 Ontology Development Tools

Implementing ontologies directly in an ontology language, without a supporting tool, makes the ontology development process complex and time-consuming. To ease the task and help developers with some ontology development activities, the first ontology development environment was created in the 1990s. The number of ontology tools after that date increased greatly. According to Gómez-Pérez et al. [73], the following ontology tools have been of great importance: ontology evaluation tools, development tools, ontology learning tools, ontology merge and alignment tools, ontology querying, ontology-based annotation tools and inference engines. Analysis and overview of ontology techniques and learning tools are to be found in [75],[82].

The first ontology editing tool was the **Ontolingua Server** (Farquhar et al., 1997 cited in [73]) available as a World Wide Web service. This ontology editing tool had been developed by Knowledge Systems Laboratories in Stanford so as to ease the development of the ontologies pertaining to the Ontolingua. The Ontolingua supports collaborative and distributed editing of ontologies. Ontologies can be created from scratch or by extending existing ones.

The year 1997 witnessed the release of **WebOnto** (Domingue 1998 cited in [73]). Its considerable support for collaborative ontology edition represented its principal advantage indeed which facilitated both asynchronous and synchronous discussions regarding the ontologies which had been built by multiple users.

Another extensible tool is the **WebODE** (Arpirez et a1., 2001 cited in [73]). WebODE is actually based on HTML forms as well as Java applets. WebODE’s ontology access service represents its own core indeed. This is used by all applications and services that are plugged into the server.

Protégé tool [83] is a standalone application that is both an open and free source having an extensible architecture too. Its core is its own ontology editor. This editor may be further extended with the provision of plugins which can introduce more functions to the environment.

Based on a plugin architecture, the free, flexible and extensible environment **OntoEdit** (Sure et a1. 2002 cited in [73]) was created. OntoEdit provides a graphical interface that is both user-friendly and supportive of ontology maintenance and development. The ontology editor pertaining to OntoEdit is a stand-alone application which imports and exports ontologies in different formats {DAML+OIL XML, FLogic and RDF(S)}. OntoEdit has two editions with each version having its own group of functions: OntoEdit Professional and OntoEdit Free (with limited capabilities). Protégé was adopted in this research. It was selected due to the following reasons:

- Protégé is a free open source ontology-editing tool with a variety of widgets and plugins to support the system's capability and functionality.
- It has a user-friendly graphical interface with easy to use menu-command tool.
- It is supported with a clear user guide and supports the import and export of ontology to/from different ontology representation languages (such as OWL and RDF).
- Protégé has the ability to verify the ontology and to check consistency for conformance with the language rules.
- Moreover, the "Protégé-discussion" mailing list provides technical support for the users, which save time and efforts.

2.6.5 Review of Ontology Development Methodologies

In this section, we will review the most known ontology development methodologies. The research's literature states that many methodologies have been followed for developing ontologies. Somehow, no standard methodology for creating the ontology is available [84]–[86].

Uschold and King [84] defined the first methodology for developing an ontology. This ontology was further extended into Uschold and Gruninger [84]. There are four phases included within the latter: Identify the purpose of the ontology, construct the ontology (capture the knowledge, code it, and integrate existing ontology), evaluate the ontology and finally; document it.

In contrast with Uschold and King [84], Gruninger and Fox [87] relied on their experience in building the TOronto Virtual Enterprise (TOVE) project ontology and using first-order logic that they presented a more formal methodology for constructing ontology. The TOVE is a set of formal ontologies for various aspects of the business enterprise such as Time ontology, the Resource Ontology and so on. The methodology proposed the first use of the competency questions (a set of natural language questions used to determine the scope of the ontology) in building ontology. The following steps are included: identifying motivation scenarios, elaborating informal competency questions, specifying the terminology using first-order logic, formalising the competency questions, specifying axioms using first-order logic and specifying completeness conditions.

A step by step approach was proposed by Noy and McGuinness [83] intended for users to design ontology. The steps are as follows: determine the scope and domain of the ontology, consider reusing of the existing ontologies, enumerate the important terms in the ontology, define the class hierarchy and the classes themselves, define the properties of classes-slots, define the facets of the slots and finally create instances.

Nicola et al. [88] described a methodology for designing ontology. The methodology is called UPON (Unified Process for ONtology building). The development methodology closely follows the unified process. The following phases are included within the methodology: First is “Inception” phase including requirement capturing and modelling the use cases. Second is “Elaboration” phase including analysis of requirements and both identifying and capturing of fundamental concepts. Third is “Construction” phase, where a skeleton for the ontology may be designed based on the loosely identified concepts. Fourth is “Transition” phase where the ontology is subjected to rigorous testing, documentation and finally released for public use. Successive iterations of the first three phases will lead to refinement, and a more stable version of the ontology ultimately reached.

Fernandez et al. [89] developed a methodology called METHONTOLOGY in the Artificial Intelligence Laboratory at the Polytechnic University of Madrid. This methodology had been there for constructing ontology through re-engineering or reusing of existing ontology, starting from scratch or reusing existing ontology. METHONTOLOGY framework facilitates the construction of ontologies at the conceptual level and has its roots in software engineering and knowledge engineering methodologies. It consists of: (a) an ontology development process with the identification of the main activities, such as, conceptualisation, configuration, management, evaluation, integration implementation; (b) a life cycle based on evolving prototypes; and (c) a methodology, specifying the steps for performing the activities, the techniques used, the outcomes and their evaluation. A few of these methodologies are concerned with designing ontology from scratch while others reuse and integrate existing ontologies to design new ones [90].

2.6.6 Ontology Reasoning Techniques

Ontologies provide a formal meaning of concepts in a domain of knowledge leading to a shared and common understanding that improves communication between people and software agents. Using ontologies to represent domain knowledge allows not only the definition of concepts and their interrelationships but also inferring implicit relationships using reasoning techniques.

Reasoning is important to ensure the quality of an ontology, for example, to check concepts consistency and derive implied relations [91]. Ontology reasoning approaches support inference through various kinds of logic: description logic, first order logic, temporal logic to name a few [92]. There are various reasoners such as: FaCT++ (Fast Classification of Terminologies) [93], PELLET [94], and RACER (Renamed ABox and Concept Expression Reasoner) [95]. FaCT++ and RACER are the two most widely accepted OWL reasoners. They support automated class subsumption and consistency reasoning and some queries on OWL ontologies.

2.7 The Role of Ontology in Representing Architectural Knowledge

The ontology has a considerable role in representing architectural knowledge. Several works utilise ontology as a tool to represent the architectural knowledge such as Kruchten et al. [96] who presented an ontology to describe the Architectural Decisions. Akerman and Tyree [97] described an ontology-based approach that focused on architectural design decisions and included part of the ontology called “architecture assets”. Babu et al. [98] designed ArchVoc which is an ontology meant for representing the architectural terminology which is organised into three major types: architectural requirements, architectural design and architectural description. An ontology that focused on components and connectors as a general approach to describing architectural styles had been represented by Pahl et al. [99]. Nakagawa et al. [28] proposed to use an ontology to provide a mechanism to support the organisation, sharing, reuse and acquisition knowledge of the testing domain.

The technical standard of Service-oriented Architecture Ontology from [100] describes core concepts, terminology and semantics of a service-oriented architecture to improve the alignment between the business and Information Technology communities. Ameller and Franch [101] developed Arteon, an ontology for representing the architectural knowledge. It explains relationships existing between architectural styles and their variants too, frameworks, views, frameworks, architecture styles and their implementation in the context of a web-based application. Sun et al. [23] proposed to use an ontology to define ontological models that are specific to the design of software architectures. Through the OWL, the structures and restrictions in the relationships between the elements of the architecture are represented. Kruchten et al. [102] developed an ontology to describe architectural decisions and relationships between them, including aspects of reasoning. While pursuing exploitation of the knowledge of ontology, Kruchten et al. also proposed a tool that can preserve the graphs pertaining to design decisions as well as all interdependencies of theirs so as to support the systems' maintenance and evolution too. López and Colab [103] presented an approach driven by ontology to recover architectural reasoning from documents in plain text and to synthesise it in a repository of centralised knowledge. The proposed approach, called Toeska Rationale Extraction (TREx), also has two ontologies: one ontology to represent the software architecture of the system and another to describe the reasoning of the project.

Figueiredo et al. [104] illustrated a framework to enable the search for information on software architectures in documentation artefacts generated in virtual community environments such as emails, meeting minutes and Wikis. The approach consists of defining an ontology of software architectures altogether with ontologies of the application domain that model knowledge of the development domain of the system. Duran-Limon et al. [105] proposed an ontology-based product architecture derivation (OntoAD) model to automate the derivation of product-specific architectures from a software product line architecture.

Roldán et al. [106] presented an ontology-based approach to retrieve, integrate and share of knowledge from different sources of architecture knowledge.

There are other works that apply the use of ontologies to the representation of knowledge of the domain of software architectures such as [107], [108] which focused on the retrieval of knowledge in textual documents as well as searching (file-based documentation). These ontologies are used to index the artefacts as well as to visualise the results of the searches so as to help the users to explore, discover and analyse information through a mechanism of semantic search.

2.8 Summary

The background for the contributions described in the following chapters had been presented here in this chapter. Firstly, the concept of software architecture was discussed. Then, the fundamental concept of reference architecture was explained. Followed by the differences between the reference architecture and software architecture. Next, an overview of ontology was provided. Finally, the role of ontology in representing architectural knowledge was explained. Within the literature review of this chapter, development approaches of a reference architecture and the state-of-the-art of the instantiation process had been characterised too. This review allowed the researcher to recognise the limitations of the development processes. Despite the existence of various development approaches for reference architectures reported in the literature, these approaches have been used to design reference architectures for various domains. In particular, informal and semi-formal tools have been used to describe and present the artefacts of reference architectures in these approaches. This leads to making the instantiation process of a software architecture from a reference architecture very difficult task.

In this chapter, the most relevant definitions of the term ontology have also been illustrated; other definitions can be found in Artificial Intelligence and Information Technologies literature. However, it can be noted that with all these definitions there is almost always a consensus of the usage of the term ontology among ontology developers and users. It can be concluded that ontology is used to capture knowledge of a domain that can be shared and reused by a group of people of software agents.

The contributions we present in the next chapters aim to present the artefacts of reference architectures in a formal way by using ontology as a tool. The reason behind that is the definition of ontology, which refers to a formal way of knowledge representation, and it encompasses concepts and relationships. The next chapter will present the research methodology and the design of this study. In Chapter 4, we propose a methodology to define a general ontological model which will be used to present the artefacts of reference architecture in an organised and structural way. Chapter 5 describes a process of using the general ontological model.

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Overview

The methods and techniques that a research follows in order to systematically tackle the research's problems are termed as "Research Methodology" [109], [110]. Additionally and according to Collis and Hussey [111]; who have stated that selection of research methodology should reflect the assumptions of the research paradigm. Therefore, that means the methodology of research depends on the problem of research and the aim and objectives, which the researcher seeks to achieve in the study. Section 3.2 and Section 3.3 discuss the research type and model. The research model includes the philosophy of the study, research approaches, deductive and inductive approaches, qualitative and quantitative research approaches, research strategies, data collection methods and data analysis. Finally, Section 3.4 shows the ethical consideration.

3.2 Research Type

According to the literature, there are various classifications for various types of research. They can be classified into their purpose, process, logic and outcomes [111], [112] as shown in Figure 3-1. The following is a presentation of each type.

- ***The purpose of research:*** The answer(s) to the question of (why does the researcher conduct the research?) draws the features of the research purpose. The researcher is the one who can best investigate why in the first place he/she opted to wage through that very subject of research.

- ***The process of research:*** It is about the systematic and scientific way of collecting data and analysing them.
- ***The logic of the research:*** A research is either "Inductive" or "Deductive" based on whether it starts from specific observations moving on to the general ones or the other way round, respectively.
- ***The outcomes of the research:*** Whether the yield of the research is to serve as a contribution to the chain of scientific advances and knowledge or simply to offer a solution for an existing problem, the outcome of a research should conform to the anticipated yield in conjunction with the researcher's early vision and targets.

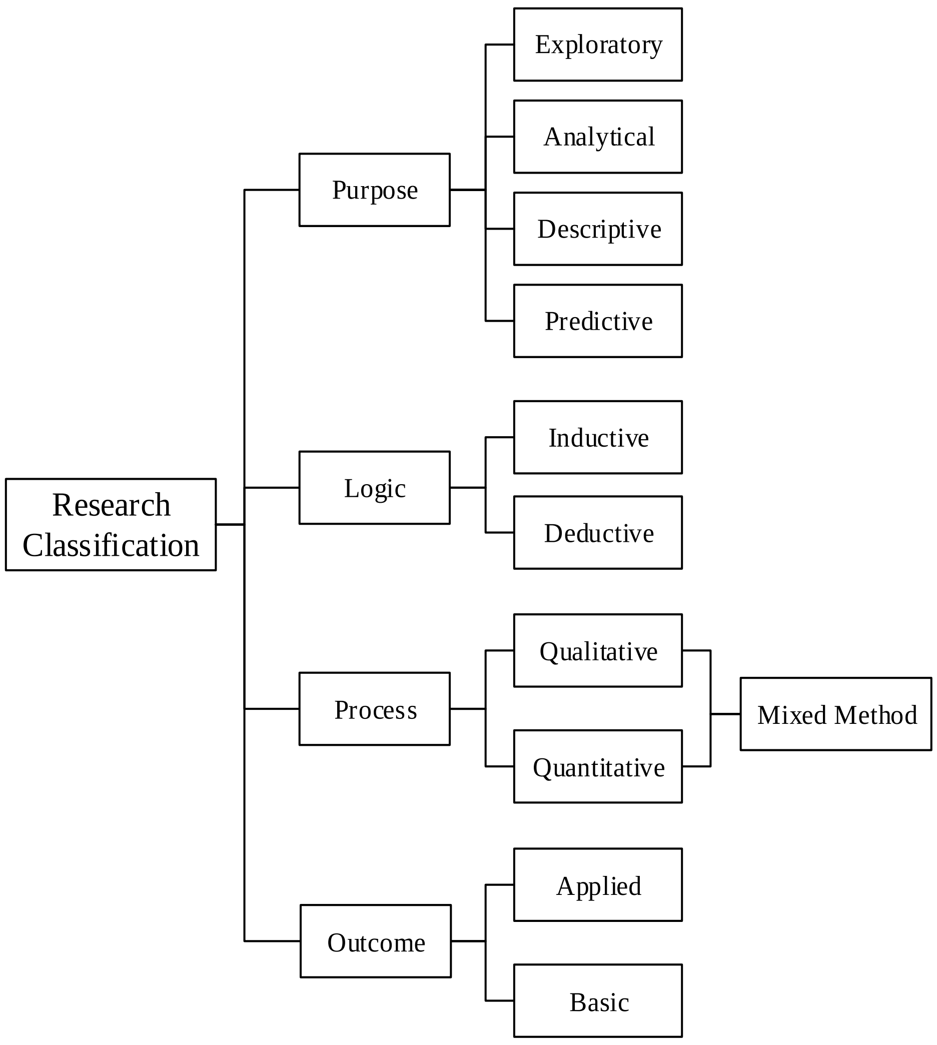


Figure 3-1: Classification of Research Types [111], [112]

According to Collis and Hussey [111], the research methods include one or more of the followings:

- Exploratory research: This type of researching is resorted to when the amount of information available from earlier studies for the intended research are not enough. Since there is no already available hypothesis that is prone to verification, confirmation or testing, an exploratory work targets hypothesizing, ideas and/or patterns, rather than any other of what is mentioned above.
- Descriptive research : When there is an already existing problem or phenomena, the approach is to follow “Descriptive “ mode of researching. This mode explains what is lying there for the researcher to deal with. For a relevant problem or phenomena, this type of researching seeks information pertaining to the relevant characteristics as well as identification too. Most of the times, the collectable data is more of a quantitative nature. Additionally, the researcher uses statistical techniques so as to integrate and summarise the collected information. In contrast to descriptive researching; the descriptive mode of researching extends further more than what an exploratory study does as per examining the relevant problem because it both explains and ascertains the characteristics of the researched problem.
- Analytical research comes next to descriptive studies. Rather than just describing those characteristics; it follows explanation and analysis regarding how and why it happens. Therefore, the aim of analytical research is measuring and probing the causal connections correlating them to each other.
- Predictive research goes further than the explanatory study. While an explanatory study cares for establishing an illustration of what occurs in a phenomenal or problematic context, a “ Predictive “ study prophecizes the chances of a situation occurring somewhere else. Predictive studies aim at a generalised perspective in

contrast with the analysis through prediction of definite phenomena from expansive and hypothesised relationships.

Additionally, there are also other steps that define the purpose of any research as stated by Cavana et al. [113], and Uma and Roger [114]. According to their recommendations, there should be an emphasis on the specific purpose of the study while picking the suitable design and framework of the research. The followings are the four steps pertaining to Cavana et al. [113] , and Uma and Roger[114]:

- Whenever the intended subject to be researched misses sufficient amount of information (or when the work itself is almost a pioneering one as to that very specialy) the research would be an “Explanation Study” one. Such researches help provide preliminary or introductory information regarding the phenomena itself and the situation too.
- This approach is sought to highlight the characteristics of the parameters pertaining to an already available case offering additional information and details.
- In case there is a group of variables with some relationships correlating them to each other, a “Hypothesis-Testing” study can elaborate more on such relationships as well as clarifying more on their nature.
- As the title implies, an elaborative work of data-collection is conducted as per a specific issue or phenomena that is underemphasis.

On the other hand, Ghauri and Gronhaug [115] considered that explanatory and descriptive researches are the most common ones while it is the nature of the problem that decides which type of research to follow.

In the present research, the aim is to propose a methodology to facilitate the instantiation process of a software's system architecture from a reference architecture by using ontology as a tool so as to present the artefacts of a reference architecture. This way, the current research is considered as an exploratory research because there is a lack of studies that address the problems of the software system architecture instantiation process [116]. Figure 3-2 illustrates the classification of the present research according to purpose, process, logic and outcomes.

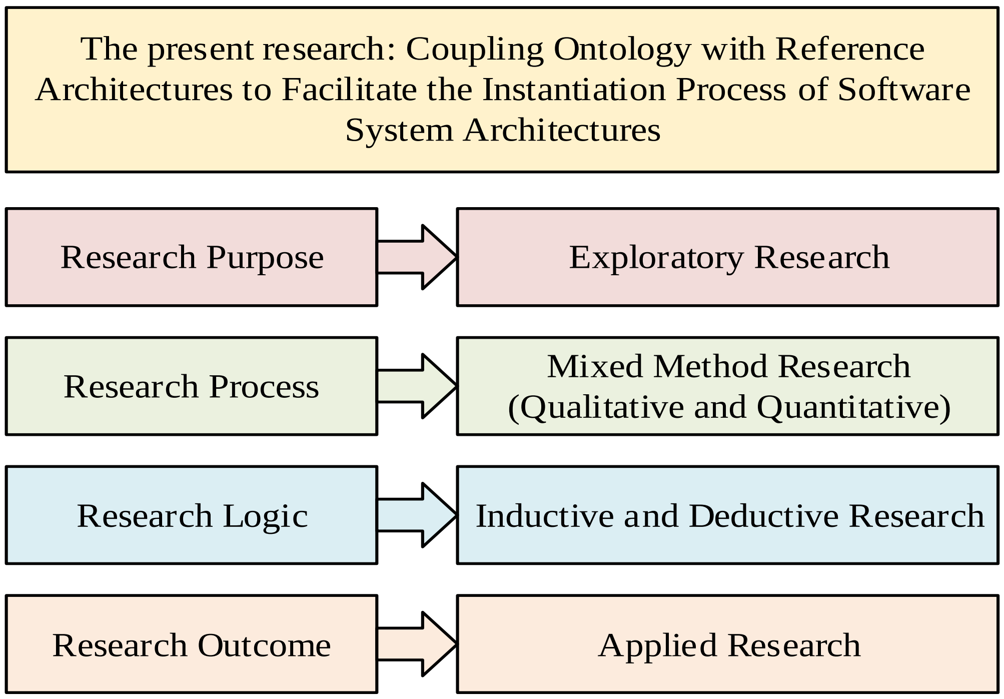

The present research: Coupling Ontology with Reference Architectures to Facilitate the Instantiation Process of Software System Architectures
Research Purpose
Exploratory Research
Research Process
Mixed Method Research (Qualitative and Quantitative)
Research Logic
Inductive and Deductive Research
Research Outcome
Applied Research

Figure 3-2: The Classification of the Present Research [111]

3.3 Research Model

The researcher should describe and apply the research model to achieve the aim and objectives of the research. Saunders et al. [110] proposed a research model that explains the process that the researcher should adopt in his research. Figure 3-3 shows the parts of the research model. The research model includes philosophies, approaches, strategies, time horizons, and data collection methods. This proposed model has been followed to achieve the aim and objectives of the research.

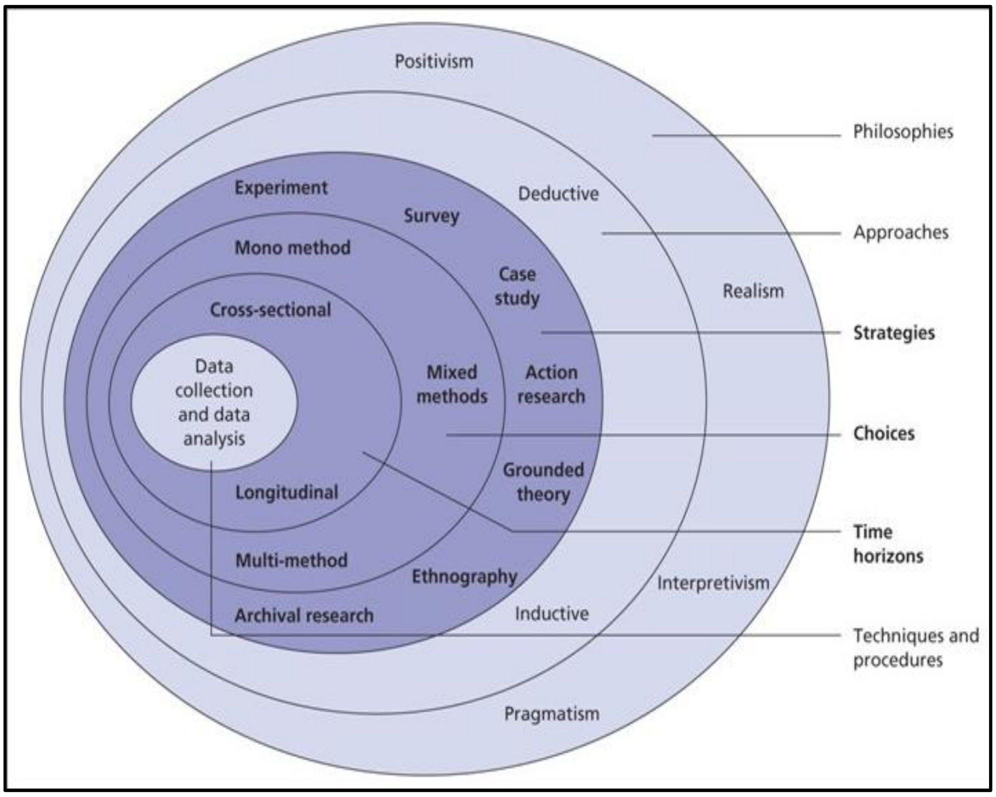


Figure 3-3: The Research Model [110]

The researcher followed a research model adapted from Saunders et al. [110] as shown in Figure 3-4. This model is comprised of research paradigm, approach, strategies, a method adopted to conduct research, data collection and data analysis. Figure 3-4 presents the research design applied to the current research. The research includes two phases. In the first phase, a qualitative data set has been collected with the help of documents (Multiple Case Studies). In the second phase, a quantitative data set has been collected from the participants in the user study by conducting survey questionnaires. The qualitative data has been analysed to design the ontological model and the quantitative data has been analysed to evaluate the proposed process. The research strategy is an exploratory strategy because there is a lack of studies addressing the problems of the instantiation process. "Mixed Method" is the research choice of this research study. The approach of the research focuses on both inductive and deductive methods. Finally, research philosophy has been decided upon based on the final development of the research which is interpretivism and positivism.

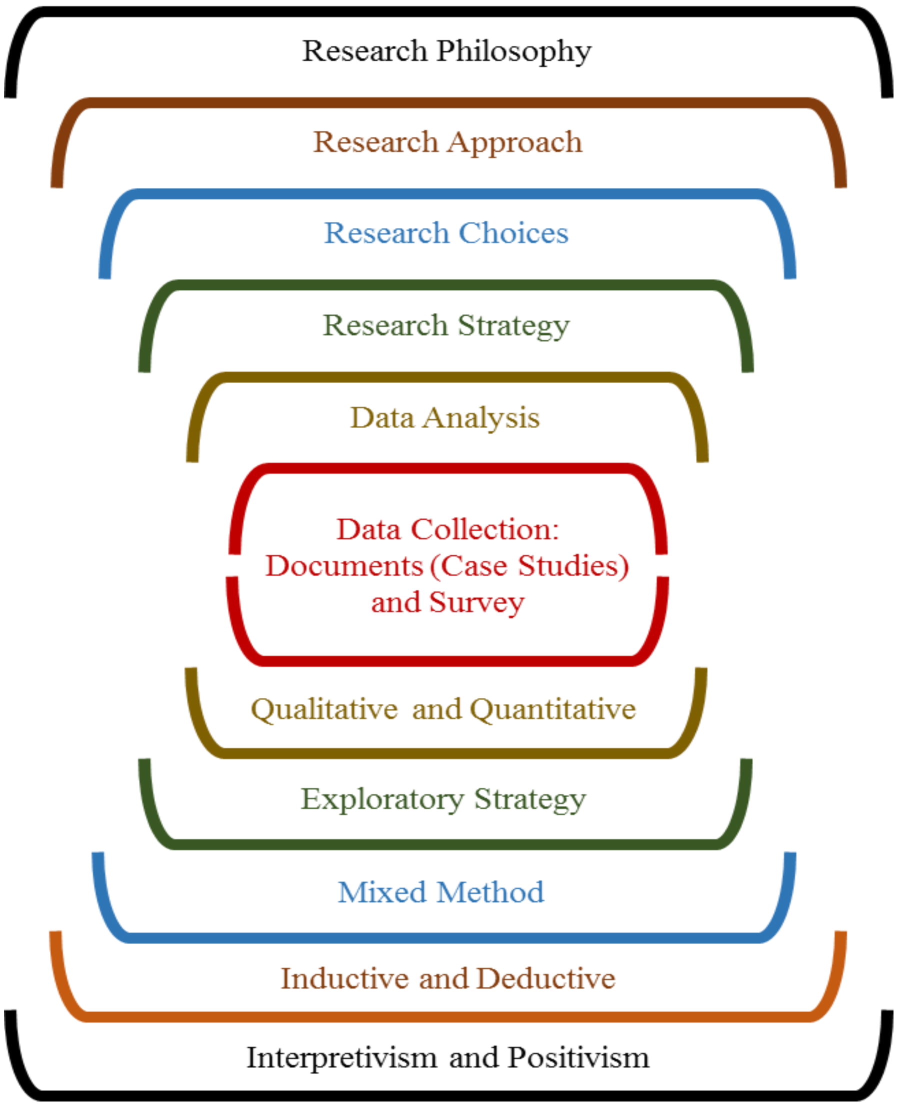


Figure 3-4: The Adopted Research Model [110]

3.3.1 Research Philosophy

The philosophical framework which dictates how to implement the research according to assumptions and philosophical conception regarding the nature of that knowledge is termed as " Research Paradigm" [116].

According to Oates [117] and Saunders et al. [118], researches can have one of three philosophical paradigms. These are Critical paradigms, Positivism and Interpretivism. IT artefacts are the main focus of Critical paradigms [117] . Weber [119] clarified that the research object with a positivist paradigm acquires inherent qualities which exist in an independent mode and irrespective of the researcher. Positivism is mainly after proving a concept and hypothesis through establishing a statistical or causal correlation. As for Interpretivism, this kind of paradigm is after identifying and exploring factors within a social setting or an organisation for the sake of comprehending the phenomena. Here, the meaning structure of the actual experience of the researcher interprets the object of research.

According to Collis and Hussey [111], there are two main types of paradigms: qualitative and quantitative , the first is phenomenological while the other is positivism. During the early stages of researching, the researcher has to adopt one of these paradigms. Phenomenological paradigm is about comprehending the human behaviour through the own frame reference of the participants while positivism paradigm pursues the roots of social phenomena with little consideration for the subjective state of the individual. The positivism paradigm pursues focusing on measurements while the phenomenological paradigm pursues focusing on the meaning. In addition to that and according to Saunders et al. [118], the intellectual traditions form the source of Interpretivism: phenomenology and symbolic interactions. Additionally and according to Collis and Hussey [111], and Saunders et al. [118], positivistic research is conducted in an artificial setting or a laboratory environment so as to control the variables of the researched case. Somehow, a research within a phenomenological paradigm is performed in real life inside a natural location that is the field of study. Here, the researcher has no control as per any of the phenomena’s aspects. The main features of the positivism and phenomenological paradigms (Interpretivism) are shown in Table 3-1.

In this study, a mixed method approach has been adopted to achieve the aim and objectives of the research. Thus, the researcher decided to adopt interpretivism and positivism philosophy, because the qualitative research approach is an interpretivism philosophy and the quantitative research approach is a positivism philosophy [118].

Table 3-1: Features of the Positivistic and phenomenological Paradigms [111]

Positivistic Paradigm	**Phenomenological Paradigm**
It produce quantitative data	It produce qualitative data
Reliability is high	Reliability is low
Validity is low	Validity is high
Concerned with hypothesis testing	Concerned with generating theories
Data is highly specific and precise	Data is rich and subjective
Has an artificial location	Has a natural location
Generalise from sample to population	Generalise from one setting to another

3.3.2 Research Approach

There are three main research approaches, namely quantitative, qualitative, and mixed method [120]–[122]. Quantitative research is mostly relevant to sample sizes of numerical data that can be generalised. Qualitative research, in contrast, is based on in-depth information [112], [122]. According to Cavana et al. [113], the methods of research are generally classified as qualitative and quantitative. In the qualitative method, the data that is based on words is usually collected via observations, documents, interviews and focus groups. Alternatively, the data in quantitative researches is rather based on numbers; it is collected through laboratory experimentations and questionnaires. Critical research and

Interpretivism are usually based on qualitative research methods while Positivist researches concentrate on quantitative research.

In addition to that and according to Collis and Hussey [111] and for the sake of gathering phenomena's data in depth, the main data collected in the phenomenological paradigm is qualitative. Alternately, the data collected is mainly quantitative in case of adopting the positivism paradigm within the research. This is due to the requirement for the data to be highly specific.

According to Silverman [123], in the case of qualitative data methods, the researcher collects data about the relevant phenomenon in depth. It is what the researcher is after achieving in the study that mandates whether to go qualitative or quantitative. The quantitative approach is the right choice in case the researcher is after making numerical comparisons between some phenomenons whereas the qualitative method suits a researcher seeking to comprehend the phenomena thoroughly.

On the other hand, "Mixed Method" research approach is a combination of qualitative and quantitative data collection and analysis such as documents, surveys, interviews, and action research used in the "Mixed Method" research [111], [124]. It involves logical conventions, and using the quantitative and qualitative approaches, and mixing both the approaches in a research study [124]. Saunders et al. [110] defined " Mixed Methods" as an approach to research in which both quantitative and qualitative data collection techniques and analysis procedures are used in research design, either at the same time (parallel) or one after the other (sequential).

In this research, a "Mixed Method" research approach has been adopted to study the problem and phenomenon in depth. Furthermore, the data has been collected from different documents by investigating multiple case studies and by conducting survey questionnaires to evaluate the proposed methodology. As a result, it has been concluded that the “Mixed Method” approach is more appropriate for this type of research.

3.3.3 Deductive and Inductive Approaches

It is a vital thing that researchers make up their minds as to which reasoning method they should follow. The available options are either inductive or deductive [110]. The main difference between the two reasoning methods is that “Deduction” includes subjecting an existing theory to a test via the designed research strategy. In contrast, “Induction” means establishing a theory via data sets that had been collected and analysed too, this eventually yields what is called as a “Theory” [110].

Deductive approach employs inferential reasoning to approach evaluating of the research aspects. In general, this is defined as a top-down approach since it primarily views the overall main image prior to narrowing the scope down to the more pinpointed details. Thus, in practice, there is already a general theory within the literature of the research. The researcher practices scrutiny on this theory to form a specific hypothesis. Observation and confirmation are the following steps of a deductive research. Furthermore, this process conducts testing of the hypothesis. Alternatively, “Inductive” reasoning views the subject from bottom to top starting from specific observation outwards to generalisations and further to theories. The researcher compiles all observation data so as to develop results in the form of findings or perhaps a theory [118], [125].

According to Bryman and Bell [121], a deductive approach “represents a view of the nature of the relationships between theory and research”. The process of a deductive approach is depicted in Figure 3-5. Alternatively, specific observations mark the beginning of inductive research while a theory defines its end. Eventually, theoretical generalisations form the results of inductive research. The process of induction is depicted in Figure 3-6.

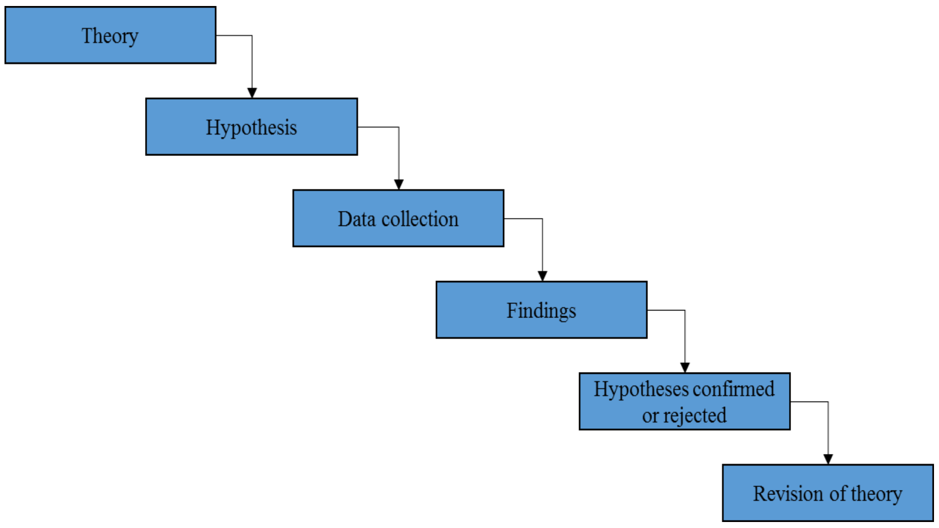


Figure 3-5: Deduction Process [121]

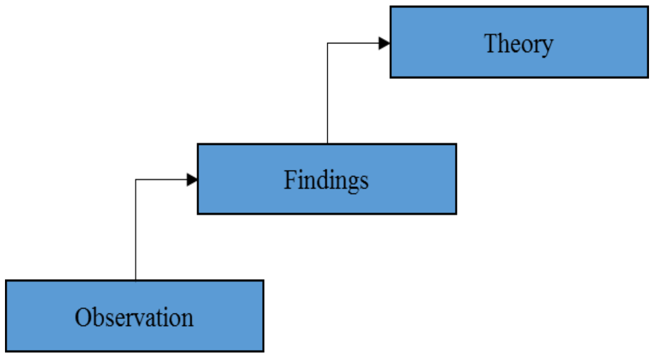


Figure 3-6: Induction Process [121]

Referring to Cavana et al. [113], a researcher who opts to adopt a deductive researching mode usually begins with a theoretical proposition, then he/she progresses into collecting data and analysing it either to accept or to reject the hypothesized vision. In Inductive researching, in contrast, the processes begin with specific phenomena and eventually land on a theory.

Alternatively and according to Saunders et al. [118], there are three main approaches for conducting a research: abductive, inductive and deductive. Saunders et al. [118] also argued that the research approach is defined by the beginning of the research. As for the case with abductive approaches, the first step to do is collecting data to explore the depths of a phenomena and then to identify themes that can modify or generate an existing theory; a theory that the researcher would then be testing while collecting more relevant data. The research should follow inductive approach in case it begins with collecting data to explore the problem (or phenomena) and it proceeds afterwards to shape a theory. Conversely, a deductive approach is recommended for the researcher in case the research starts with a theory developed from the literature and then moves on to put the theory to the test. Consequently, inductive and deductive approaches have been adopted in this research because the inductive approach is associated with a qualitative approach and the deductive approach is associated with the quantitative approach.

3.3.4 Research Strategies

There are various research strategies found in the literature. According to Saunders et al. [126], research strategies include six types: surveys, case studies, experiments, grounded theory, ethnography, and action research.

In this research, multiple case studies have been used to find and define general vocabularies, which are used to design an ontological model. In the evaluation process, survey questionnaires have been conducted to collect quantitative data from the participants.

3.3.5 Data Collection Methods

Clarifying the research problem and background is the key element in deciding what the most appropriate research methods for data collection are. Several methods for data collection can be used separately or all together to gain data and information.

Yin [127] suggests six sources of data used in collecting data: interviews, documentation, archival records, direct observation, participant observation and physical artefacts. Oates [117] and Patton [128] described four of these data generating methods as used in information system research: interviews, observation, questionnaires (Survey) and documents. It is important to choose the right data collection method(s) as this will allow collecting of data that will meet the objectives of the research.

Accordingly, the present research used two sources to collect data from. Firstly, documents have been used as a primary source of data. The data were derived from an in-depth review of the related literature to gain the necessary understanding of the topic under investigation. The data were then used for exploring the development process. Secondly, a survey has been used to collect data from a user study experiment to evaluate the proposed methodology.

- Documents provide basic information as a background for the subject under investigation or in making decisions and assisting the researcher in creating additional ideas to follow through more direct observation, interviewing or questionnaires. Documents are collected from publications, journals, books, program records, reports, personal diaries and internet websites [128].
- Survey is not only an instrument for collecting information; it is also a comprehensive research method for gathering data to describe, to compare and to explain knowledge, attitudes and behaviour [129]. The quantified survey is to produce statistics that are numerical explanations of some characteristics of the research studies [130]. The survey method of data collection is very efficient and effective in terms of time and cost [131].

3.3.6 Data Analysis

As stated earlier, the researcher used a qualitative and quantitative data generating method to collect data. The qualitative data has been analysed by the researcher through reviewing it to extract and define general vocabularies, while the quantitative data has been analysed by using SPSS software and Microsoft excel.

3.3.6.1 Justification of Using SPSS Software

The software termed as SPSS is a software package that had been designed for assisting in quantitative data analysis. Actually, it offers many tools that can aid the researcher to deal in extreme ease with the data that had been gathered from quantitative data sources.

The followings are the reasons why the researcher used SPSS software:

- In order to acquire free access to the SPSS program since the University of Salford provides a full licence of this software for such students.
- The researcher has already attended multiple training sessions regarding SPSS right at Salford University classes.

3.4 Ethical Consideration

According to Gray [122] and Sekaran and Bougie [132], it is important to acknowledge the participants of the purpose of the study as well as assuring them that data will be confidential, i.e. assuring them that no other party has the right or an access to preview or use the data. This is important since it can enhance the feelings of trust and comfort among the participants. The followings are some issues which the participants should have prior knowledge of:

- It is a voluntary participation for them to decide upon.
- No third party will ever be allowed to share the data gathered from respondents.
- It is the responsibility of the researcher to protect the data and privacy of respondents in case an organisation expresses its will to contribute to the study.
- Confidentiality and privacy are well- recognised and maintained for all participants.

As clarified in Appendix D.1 an ethical approval from Salford University had been obtained by the researcher as part of the commitment to achieving the ethical considerations in the present research.

3.5 Research Design

Creswell [124] considered a research design as an overarching composition that guides the researcher in all perspectives of research, from the philosophical theory behind the inquiry to the detailed data collection and analysis methods. The purpose of design is not only to lead the researcher but also to enable the audience to understand and evaluate the research and its results. Creswell [124] identified three factors that affect the choice of one research methodology over another, including the research problem, the personal experiences of the researcher, and the audience(s) to whom the report will be directed. However, in the Information Technology/Information System field, the choice of a research methodology is affected by several factors. For instance, Trauth [133] determined five factors influencing the selection of a research methodology in the Information Technology/Information System field. These factors are problem of the research, philosophical assumptions, the degree of uncertainty surrounding the phenomenon, the researchers' skills, and academic politics. Figure 3-7 shows the methodology phases which was adopted to achieve the research aim and objectives.

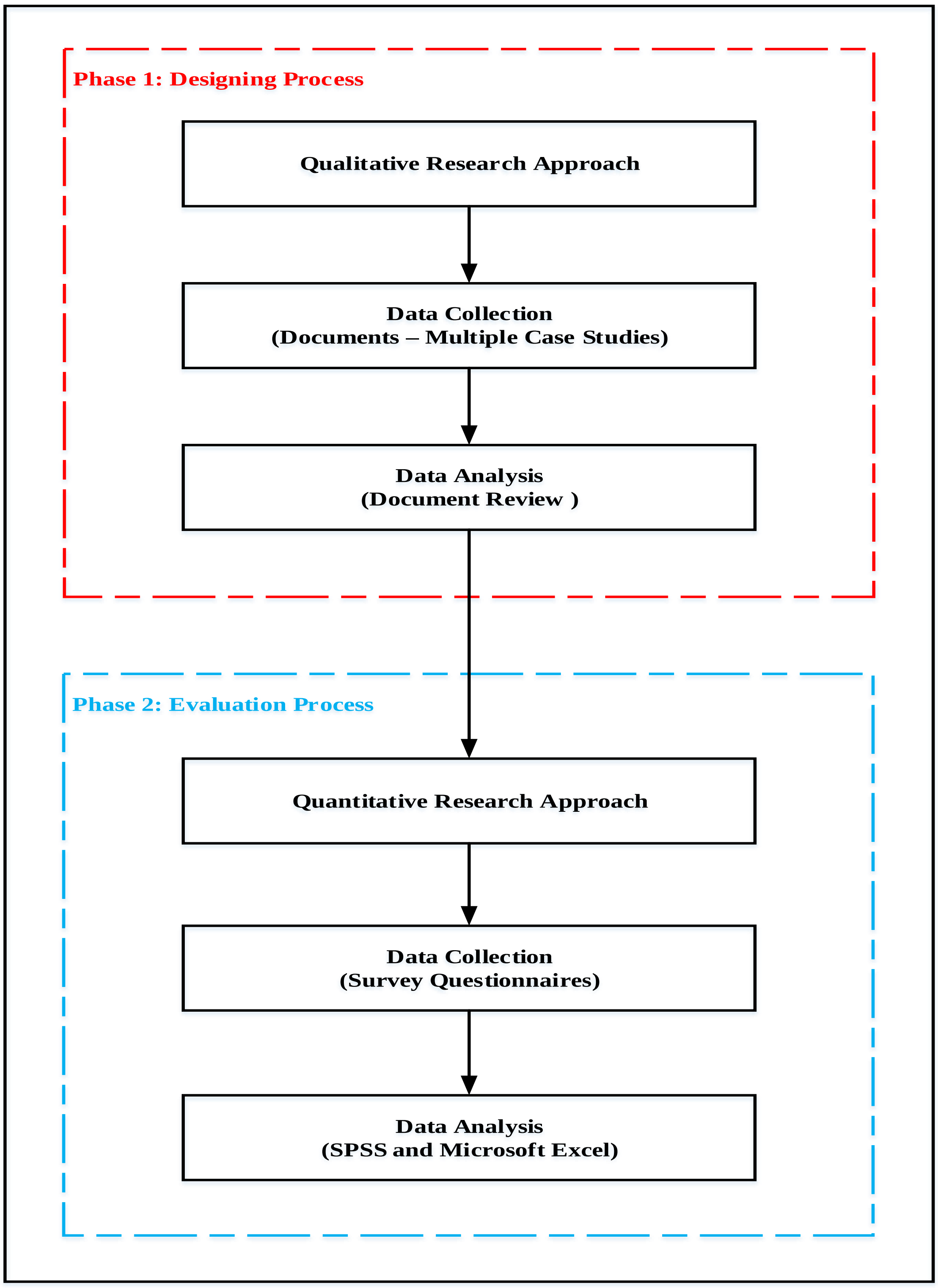

Phase 1: Designing Process
Qualitative Research Approach
Data Collection (Documents – Multiple Case Studies)
Data Analysis (Document Review)
Phase 2: Evaluation Process
Quantitative Research Approach
Data Collection (Survey Questionnaires)
Data Analysis (SPSS and Microsoft Excel)

Figure 3-7: The Research Design

In this research, a mixed method research methodology has been used to achieve the aim and objectives of the research. This research includes two phases: design and evaluation. In the design phase, a qualitative research design utilising a multi-case study approach has been adopted to extracting and defining general vocabularies which will be used to develop a general ontological model. A multi-case study approach also allowed to gather data from different documents (Yin 2003).

In the design phase, the process of developing a general ontological model has been represented. The researcher started doing a search by using the library to collect data such as google, google scholar, IEEE Xplore, ACM Library, Science Direct and Springer link. From these libraries, many documents related to the study have been collected. After that, all document that are collected to build a background about the topic have been reviewed. The knowledge that the researcher are acquired helped him to define general vocabularies. Furthermore, multiple case studies have been used to find and define more general vocabularies.

In the evaluation phase, a quantitative research approach has been adopted. Quantitative data are collected by conducting a survey questionnaires. Then, the collected data are analysed by using SPSS and Mocrosoft excel. Yin (2009) discusses the importance of using computer software packages in the analysis of data. SPSS and Mocrosoft excel are used because the University of Salford provides a full licence of these software for the postgraduate students, and this helps to access the program without any constraints.

3.6 Summary

The research methodology that was followed to attain both the aim and objectives targeted by the researcher is explained in this chapter. Additionally, chapter 3 deals with the rationale for adopting the positivism and interpretivism paradigms in addition to the approaches followed in the study. The sources of data included in the data collection are mentioned in this chapter as well as discussing data analysis too.

In the next chapter, the development process of the ontological model will be explained in details.

CHAPTER FOUR: DEVELOPMENT OF GENERAL ONTOLOGICAL MODEL

“*There is no one correct way to model a domain. There are always viable alternatives*…. *Ontology development is an iterative process*” [83].

4.1 Overview

According to the literature, there are many methodologies for building an ontology. However, there is no standard methodology for developing an ontology [90]–[92].

In this chapter, a proposed methodology for developing a general ontological model will be defined and explained in details in Section 4.24.2. The general vocabularies are defined and extracted based on understanding the domain and the literature and multiple case studies from the literature. The essential characteristic of these vocabularies presents the general aspect of reference architectures in an organised and structural way. The general vocabularies are validated as shown in Section 4.2.3. Next to that, these vocabularies are used as a basis to construct the general ontological model as described in Section 4.2.4.

4.2 Methodology for Constructing an Ontological Model

This section explains a proposed methodology for constructing a general ontological model as shown in Figure 4-1. The methodology includes the following:

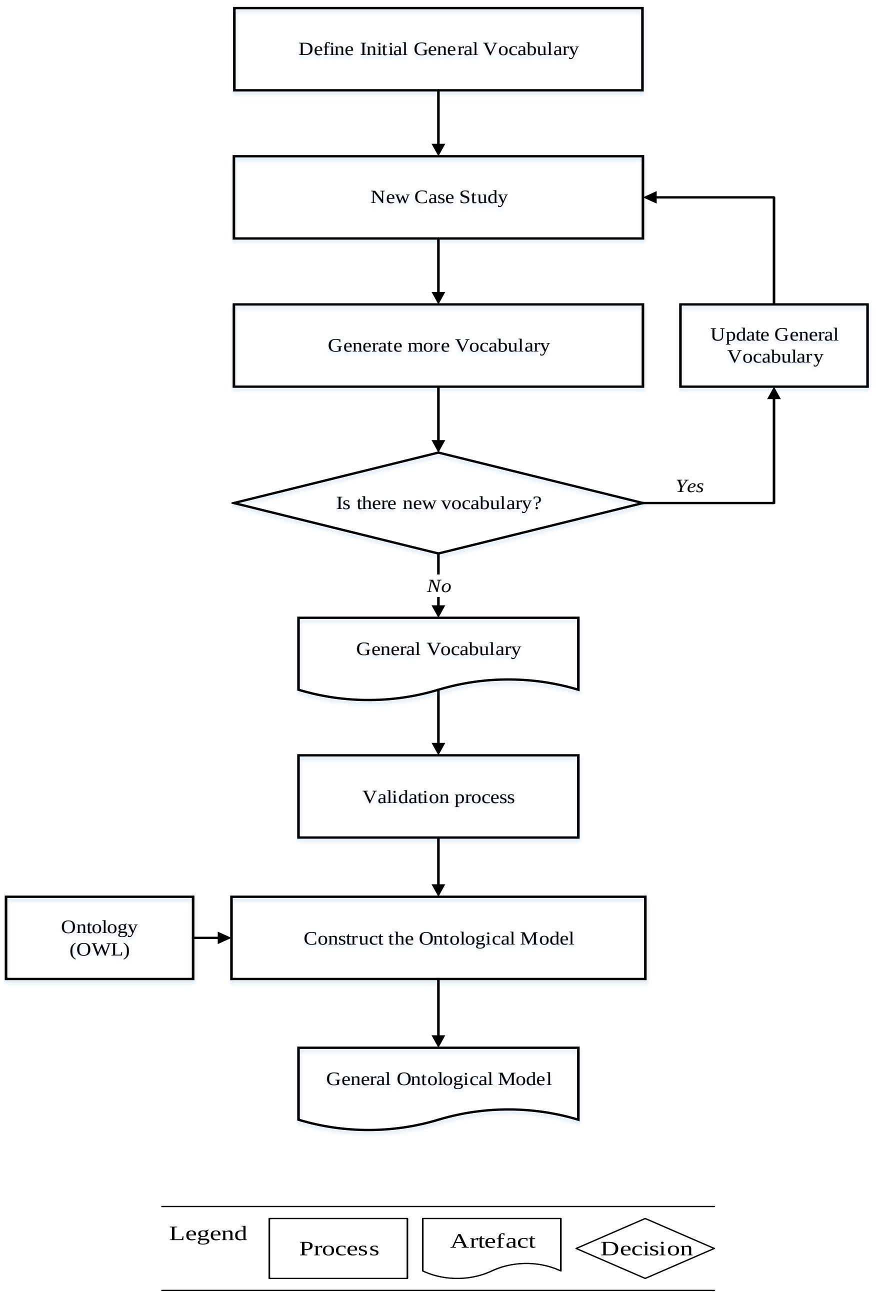


Figure 4-1: Methodology for Defining and Extracting General Ontological Vocabulary

4.2.1 Define Initial General Vocabulary

The initial general vocabularies are defined based on understanding of the domain and the literature. Appendix A shows all the studies are used to define the initial general vocabularies. The general vocabularies include two types: entities and relationships. An entity is used to describe a component and a relation is used to describe the connection between the components. Table 4-1 shows the initial general vocabularies with their descriptions.

Table 4-1: Initial General Vocabulary

Vocabulary	Description
Stakeholder	A stakeholder is a person who has some interest in the development of a new system and includes roles such as user, analyst, software architect, software developer, software tester, software engineer, team, software maintainer, agent, student, learner, teacher, lecturer, customer, manager and so forth.
System	A system is a set of components that work together to provide a function.
Subsystem	It aims to define a part of the whole system, which is working to provide a function.
Component	A component is an organisational unit of a reference architecture.
Interface	It aims to define a connection point between two subjects, components, systems, subsystems and so forth.
Architectural Style	It aims at defining a topology of architectural elements and their relationships which includes different types such as Pipe-Filter Architectural Style, Client-Server Architectural Style, Layered Architectural Style, Event-Based Architectural Style, Service-Oriented Architectural Style, Communicating Process Architectural Style, Peer to Peer Architectural Style, Blackboard Architectural Style and so forth.
Attribute	It defines a piece of information which determines the properties of a component of the architecture.
Concern	It defines a stakeholder’s need; each stakeholder has a different concern.
Function	It aims at defining the role of a component or stakeholder.
Task	It defines a piece of work to be done by a stakeholder.
Security	It defines a security requirement which is responsible for representing the security rules such as authentication and access restrictions.

View	It defines a whole architecture from the perspective of a related set of concerns such as logical view, module view, process view, physical view, deployment view, development view, component and connector view, conceptual view and so forth.
Service	It defines a logical representation of a repeatable activity which has a specified outcome.
Tool	It defines a set of software or hardware that will be used by system, subsystem, component, or stakeholder.
Protocol	It defines a series of steps in order to execute a function.
Process	It defines a series of steps taken in order to execute a task or activity.
Resource	It defines elements of hardware, software and human that support activity, task, process and so forth.

Relationship			
Include	Has a	Consist of	Is a
Describe	Apply to	Composed of	Produce
Execute	Require	Used by	Is part of
Consume	Define	Use a	

4.2.2 Generate More General Vocabulary from Multiple Case Studies

After defining the initial general vocabulary, the generation process started by applying the first case study. General vocabularies of this case study are extracted and compared with the initial general vocabularies. If there were new vocabularies, it would be added to the initial one. It is, however, important to mention that the synonyms will not be added in this process, the most generic concept is chosen to represent these synonyms. Again, this process is repeated with a new case study. After applying the fifth case study, no more general vocabularies were found. However, we applied one more case study to ensure the comprehensibility of our general vocabulary. At this stage, the process was finalised.

In this research, the researcher extracts the vocabularies from a reference architecture by reviewing it. The extracted vocabulary from the case studies was classified into two types: entity and relationship. The object and subject were considered as an entity, and the verb between them was considered as a relationship. For example, view describes a system. Therefore, ‘*view’* and ‘*system’* vocabularies are entities, and the verb ‘*describes’* is considered as a relationship

4.2.2.1 First Case Study - The Reference Architecture of the Situated Multi-Agent System

After defining initial general vocabularies, the process started with the first case study. The first case study is the Reference Architecture of the Situated Multi-Agent System [41]. In this case study, the authors used 72 vocabularies to describe the artefacts of the reference architecture (See Appendix B.1). Some of these vocabularies can be considered as general vocabularies and are compared with the general vocabulary as shown in Table 4-2. Table 4-3 illustrates the general vocabularies which are found in this case study.

The authors of this case study used different vocabularies to describe the artefacts of the reference architecture. Some of these vocabularies are not similar to the defined general vocabularies, but they give the same meaning such as (subsystem and module), (element, unit, and component) and (user and stakeholder). In this research, subsystem, component and stakeholder vocabularies are considered as general vocabulary. Table 4-4 shows the general vocabulary after update.

Table 4-2: Matching between the General Vocabulary and the Vocabulary of the First Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	System	✓	2.	Subsystem	✓	3.	Module	×
4.	Element	×	5.	Unit	×	6.	Component	✓
7.	View	✓	8.	Stakeholder	✓	9.	User	×
10.	Activity	×	11.	Mechanism	×	12.	Function	✓
13.	Data	×	14.	Repository	×	15.	Knowledge	×
16.	Process	✓	17.	Resource	✓	18.	Service	✓
19.	Interface	✓	20.	Responsibility	×	21.	Has a	✓
22.	Is part of	✓	23.	Enable	×	24.	Access to	×
25.	Is a	✓	26.	Describe	✓	27.	Decomposed into	×
28.	Use	✓	29.	Consist of	✓	30.	Execute	✓
31.	Include	✓	32.	Define	✓	33.	Provide	×

Legend:

- ✓ Refers to (the vocabulary has been defined).
- × Refers to (the vocabulary has not been defined).

Table 4-3: New General Vocabulary from the First Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Module	2.	Element	3.	Unit	4.	User
5.	Activity	6.	Mechanism	7.	Data	8.	Repository
9.	Knowledge	10.	Responsibility	11.	Access to	12.	Provide
13.	Enable	14.	Decomposed into				

Table 4-4: General Vocabulary after Update

Vocabulary					
Stakeholder	System	Subsystem	Component	Architectural Style	View
Function	Task	Process	Tool	Resource	Protocol
Interface	Attribute	Security	Data	Concern	Service
Repository	Knowledge	Responsibility	Activity	Mechanism	Require
Enable	Apply to	Include	Is a	Consist of	Use a
Describe	Produce	Composed of	Define	Execute	Consume
Has a	Used by	Is part of	Decomposed into	Access to	Provide

4.2.2.2 Second Case Study - The Reference Architecture of the Mobile Learning Environments

The second case study is the Reference Architecture of the Mobile Learning Environments [42]. The authors of this paper used 56 vocabularies to describe the artefacts of the reference architecture (See Appendix B.2). However, only 36 vocabularies can be considered as general terminologies. Table 4-5 shows the matching between the general vocabularies of the second case study and the general vocabulary.

In the second case study, 18 new vocabularies are found as shown in Table 4-6. However, the authors used different words to describe the artefacts of the reference architecture such as a database, element, user, and module. These vocabularies give same meaning of repository, component, stakeholder, and subsystem, respectively. Table 4-7 represents the general terminology after update.

Table 4-5: Matching between the General Vocabulary and the Vocabulary of the Second Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	View	✓	2.	Element	✗	3.	User	✗
4.	Information	✗	5.	Function	✓	6.	Role	✗
7.	Module	✗	8.	Mechanism	✓	9.	Feature	✗
10.	Service	✓	11.	Knowledge	✓	12.	Security	✓
13.	Activity	✓	14.	Database	✗	15.	Data	✓
16.	Request	✗	17.	Analyse	✗	18.	Perform	✗
19.	Change	✗	20.	Define	✓	21.	Located	✗
22.	Task	✓	23.	Use	✓	24.	Describe	✓
25.	Store	✗	26.	Retrieve	✗	27.	Return	✗
28.	Enable	✓	29.	Receive	✗	30.	Consume	✓
31.	Exchange	✗	32.	Produce	✓	33.	Control	✗
34.	Provide	✓	35.	Access to	✓	36.	Consist of	✓

Table 4-6: New General Vocabularies from the Second Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Element	2.	User	3.	Role	4.	Information
5.	Database	6.	Feature	7.	Module	8.	Analyse
9.	Perform	10.	Request	11.	Store	12.	Located
13.	Change	14.	Retrieve	15.	Return	16.	Exchange
17.	Receive	18.	Control				

Table 4-7: General Vocabulary after Update

Vocabulary					
View	System	Subsystem	Component	Architectural Style	Stakeholder
Function	Task	Process	Tool	Resource	Protocol
Interface	Attribute	Security	Data	Concern	Service
Repository	Knowledge	Responsibili ty	Activity	Mechanism	Role
Feature	Information	Require	Enable	Store	Change
Apply to	Include	Is a	Consist of	Use a	Describe
Produce	Composed of	Define	Execute	Consume	Has a
Used by	Is part of	Decomposed into	Access to	Provide	Analyse
Request	Exchange	Perform	Retrieve	Return	Located
Receive	Control				

4.2.2.3 Third Case Study - The Reference Architecture of the Cloud Computing

The third case study is the Reference Architecture of the Cloud Computing [43]. Accordingly, 44 vocabularies have been used to present the artefacts of the reference architecture (See Appendix B.3). Only 29 vocabularies can be considered as general terminologies as explained in Table 4-8.

Consequently, four new general vocabularies are found in this case study as demonstrated in Table 4-9. However, the author of the case study used '*actor*' and '*person*' terms, which give the same meaning to the term stakeholder that is defined in the general vocabularies. Table 4-10 represents the general vocabulary after adding the new terms.

Table 4-8: Matching between the General Vocabulary and the Vocabulary of the Third Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	Define	✓	2.	Describe	✓	3.	View	✓
4.	Actor	✗	5.	Activity	✓	6.	Function	✓
7.	Used by	✓	8.	Person	✗	9.	Resource	✓
10.	Tool	✓	11.	Security	✓	12.	Service	✓
13.	Process	✓	14.	Access to	✓	15.	Role	✓
16.	Instance of	✗	17.	Use	✓	18.	Attribute	✓
19.	Include	✓	20.	Consume	✓	21.	Manage	✗
22.	Is	✓	23.	Has	✓	24.	Produce	✓
25.	Execute	✓	26.	Require	✓	27.	Consist of	✓
28.	Apply to	✓	29.					

Table 4-9: New Vocabularies from the Third Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Actor	2.	Person	3.	Manage	4.	Instance of

Table 4-10: General Vocabulary after Update

Vocabulary					
View	System	Subsystem	Component	Architectural Style	Stakeholder
Function	Task	Process	Tool	Resource	Protocol
Interface	Attribute	Security	Data	Concern	Service
Repository	Knowledge	Responsibility	Activity	Mechanism	Role
Feature	Information	Require	Enable	Store	Change

Vocabulary					
Apply to	Include	Is a	Consist of	Use a	Describe
Produce	Composed of	Define	Execute	Consume	Has a
Used by	Is part of	Decomposed into	Access to	Provide	Analyse
Request	Exchange	Perform	Retrieve	Return	Located
Receive	Control	Manage	Instance of		

4.2.2.4 Fourth Case Study - The Reference Architecture of the Web Servers

The fourth case study is the Reference Architecture of the Web Servers [44]. From this case study, 47 vocabularies have been extracted (See Appendix B.4). Only 21 terms were considered as general as shown in Table 4-11. After matching them with the word list, only two vocabularies are new as shown in Table 4-12. The general vocabularies were updated as described in Table 4-13.

Table 4-11: Matching between the General Vocabulary and the Vocabulary of the Fourth Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	Component	✓	2.	Subsystem	✓	3.	Encompasses	✗
4.	System	✓	5.	Security	✓	6.	User	✗
7.	Resource	✓	8.	Define	✓	9.	Architectural Style	✓
10.	Service	✓	11.	Control	✓	12.	Protocol	✓
13.	Instance of	✓	14.	Include	✓	15.	Is a	✓
16.	Use	✓	17.	Is part of	✓	18.	Require	✓
19.	Consist of	✓	20.	Apply to	✓	21.	Describe	✓

Table 4-12: New General Vocabulary from the Fourth Case Study

No.	Vocabulary	No.	Vocabulary
1.	Encompasses	2.	User

The authors of this paper used *user* vocabulary that gives the same meaning of the *stakeholder* term, which is already defined in the general vocabulary.

Table 4-13: General Vocabulary after Update

Vocabulary					
View	System	Subsystem	Component	Architectural Style	Stakeholder
Function	Task	Process	Tool	Resource	Protocol
Interface	Attribute	Security	Data	Concern	Service
Repository	Knowledge	Responsibili ty	Activity	Mechanism	Role
Feature	Information	Require	Enable	Store	Change
Apply to	Include	Is a	Consist of	Use a	Describe
Produce	Composed of	Define	Execute	Consume	Has a
Used by	Is part of	Decomposed into	Access to	Provide	Analyse
Request	Exchange	Perform	Retrieve	Return	Located
Receive	Control	Manage	Instance of	Encompasses	

4.2.2.5 Fifth Case Study - The Reference Architecture of the Sensor Networks Integration and Management System

The fifth case study is the Reference Architecture of the Sensor Networks Integration and Management System [45]. In this case study, 43 vocabularies (See Appendix B.5) have been found which include 17 general vocabularies as explained in Table 4-14.

Table 4-14: Matching between the General Vocabulary and the Vocabulary of the Fifth Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	System	✓	2.	Component	✓	3.	Module	✗
4.	Architecture Style	✓	5.	Describe	✓	6.	Include	✓
7.	Responsibili ty	✓	8.	Protocol	✓	9.	Interface	✓
10.	Repository	✓	11.	Provide	✓	12.	Information	✓
13.	Use a	✓	14.	Access to	✓	15.	Used by	✓
16.	Has a	✓	17.	Is a	✓	18.		

After matching them with the general vocabulary list, there is only one new vocabulary found which gives the same meaning of the subsystem vocabulary. The new term is already defined in the general vocabulary.

4.2.2.6 Sixth Case Study - The reference architecture of the Ubiquitous Computing

The reference architecture of the Ubiquitous Computing has been taken as a sixth case study [46]. In this case study, the authors of the paper used 43 vocabularies to describe the artefacts of the reference architecture (See Appendix B.6). Some of these vocabularies can be considered as general vocabulary as explained in Table 4-15. However, three new vocabularies are found as shown in Table 4-16.

The new vocabularies are element, user, and module which are already defined in the general vocabularies list as a component, stakeholder, and subsystem, respectively. Therefore, no need to take a new case study. Table 4-17 illustrates the final general vocabulary.

Table 4-15: Matching between the General Vocabulary and the Vocabulary of the Sixth Case Study

No.	Vocabulary	Def.	No.	Vocabulary	Def.	No.	Vocabulary	Def.
1.	Element	✗	2.	Component	✓	3.	System	✓
4.	Task	✓	5.	Interface	✓	6.	User	✗
7.	Use	✓	8.	Activity	✓	9.	Process	✓
10.	Information	✓	11.	Data	✓	12.	View	✓
13.	Security	✓	14.	Module	✗	15.	Service	✓
16.	Is a	✓	17.	Responsibility	✓	18.	Has	✓
19.	Encompasses	✓	20.	Function	✓	21.	Include	✓
22.	Feature	✓	23.	Provide	✓	24.	Repository	✓
25.	Access to	✓	26.	Describe	✓			

Table 4-16: New General Vocabulary from the Sixth Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Element	2.	Module	3.	User

Table 4-17: Final General Vocabulary

Vocabulary				
View	System	Subsystem	Component	Architectural Style
Stakeholder	Function	Task	Process	Tool
Resource	Protocol	Interface	Attribute	Security
Data	Concern	Service	Repository	Knowledge
Responsibility	Activity	Mechanism	Role	Feature
Information	Require	Enable	Store	Change
Apply to	Include	Is a	Consist of	Use a
Describe	Produce	Composed of	Define	Execute
Consume	Has a	Used by	Is part of	Decomposed into
Access to	Provide	Analyse	Request	Exchange
Perform	Retrieve	Return	Located	Receive
Control	Manage	Instance of	Encompasses	

4.2.3 Validation Process of General Vocabulary

In this section, the general vocabulary will be validated. Two case studies were chosen to validate the general vocabulary. These case studies are the reference architecture of the web browser [47] and the reference architecture of the robots teleoperation system [48].

4.2.3.1 First Case Study - The Reference Architecture of the Web Browsers

A reference architecture of the web browser [47] has been taken as a case study to validate the general vocabulary. In this case study, 80 vocabularies are used by the authors of the paper to describe the artefacts of the reference architecture (See Appendix B.7).

However, only 23 terms can be considered as general vocabulary. Table 4-18 illustrates vocabularies that are used to describe the general aspect of the reference architecture. After comparing them with the general vocabulary, 22 vocabularies are matched and only one vocabulary (user vocabulary) is mismatched. In the general vocabulary, the stakeholder concept is defined as a general vocabulary instead of the user concept.

Table 4-18: General Vocabulary of the First Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Subsystem	3.	Component	4.	Security
5.	User	6.	Service	7.	Information	8.	Protocol
9.	Function	10.	Architectural Style	11.	Resource	12.	Data
13.	Feature	14.	Interface	15.	Describe	16.	Include
17.	Is a	18.	Use	19.	Store	20.	Has
21.	Receive	22.	Send	23.	Execute		

4.2.3.2 Second Case Study - The Reference Architecture of the Robot Teleoperation

The reference architecture of the Robot Teleoperation [48] has been taken as a second case study to validate the general vocabulary. In this case study, 49 vocabularies were used to describe the artefacts of the reference architecture (See Appendix B.8). However, only 32 terms can be considered as general vocabulary, as shown in Table 4-19.

Table 4-19: General Vocabulary of the Second Case Study

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Subsystem	3.	Module	4.	Component
5.	Element	6.	Architectural Style	7.	Function	8.	Activity
9.	User	10.	Describe	11.	Property	12.	Service
13.	Protocol	14.	Tool	15.	Mechanism	16.	Data
17.	Information	18.	Resource	19.	Send	20.	Include
21.	Develop	22.	Provide	23.	Require	24.	Consist of
25.	Exchange	26.	Request	27.	Has a	28.	Use a
29.	Receive	30.	Update	31.	Is a	32.	Execute

Some of these vocabularies are already defined in the general vocabularies. After comparing them with the general vocabulary, only four vocabularies are mismatched. These terms are module, element, develop and update. The (module and element) terms are already defined as subsystem and component vocabularies, and (develop and update) terms are used to define relationships between entities. In ontology, there is no limitation to describe various relationships among entities.

4.2.4 Construct Ontological Model

A complete ontological model has been created based on the defined general vocabularies (as explained in Figure 4-2). The general vocabulary includes two types: 1) vocabularies that describe the entity, and 2) vocabularies that describe the relationships between the entities. In general, there is no limitation to define relationships among entities in the ontology. The relationships could be infinite as per the situation. Various relationships are used to connect the general vocabulary to create the ontological model. Figure 4-2 shows the main concepts of the ontological model as OWL classes where the arrows represent relationships (OWL object properties) between domain classes (the head of the arrow) and range classes (the tail of the arrow) where the name on the line depicts the name of the relationship. The individuals will be modelled as 'objects' in the rectangular boxes. The ‘is-a’ property relates concepts to its instances (OWL individuals). In the model, Artefact is a concept (class) while Security, Feature, Architectural Style, View, Task, Service, Role, Responsibility, Knowledge, Information, and Data are all subclasses of the class Artefact. The subclasses of the artefact class were excluded from the model to make it simple and understandable by stakeholders.

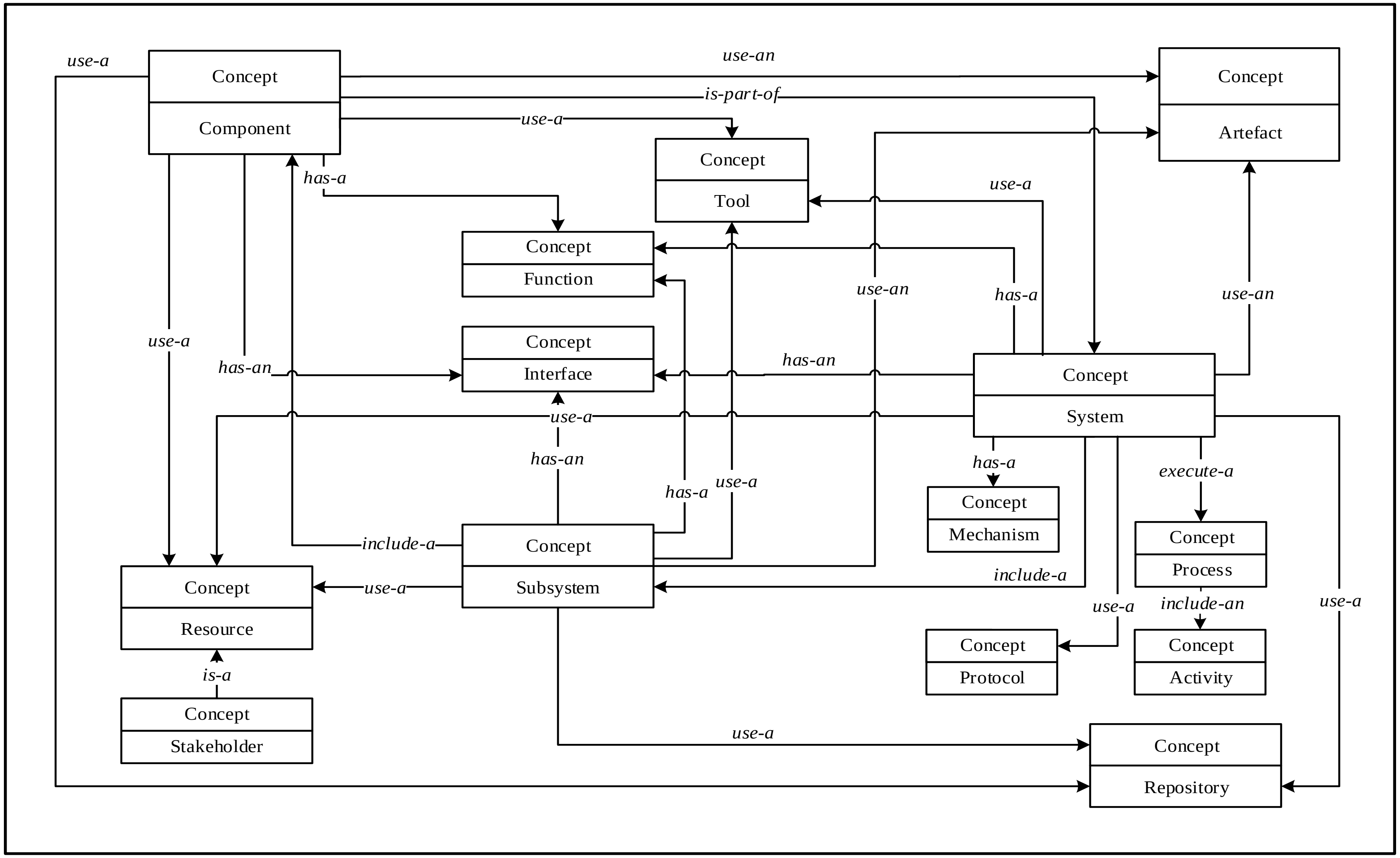


Figure 4-2: General Ontological Model

4.3 Justifications for Adopting These Case Studies

We have chosen these reference architectures for the following reasons:

1. These reference architectures are already published in different conferences and journals.
2. These reference architectures have many citations.
3. These reference architectures are already used to derive different architectures.
4. Nevertheless, the information about the domain in the papers is concise.

4.4 Summary

After a review of existing ontology development methodologies, we found there is no standard methodology to build the ontology [84]–[86]. A methodology to build the ontological model was presented in this chapter. The methodology started by defining an initial general vocabulary. These vocabularies are defined based on understanding the domain, and the literature. After that, more general vocabularies were defined based on multiple case studies for different domains. The general vocabulary is validated by using two case studies for different domains, Next to that, the general vocabularies are used as the basis to design the ontological model.

In the next chapter, the process of using the ontological model to present the artefacts of the reference architecture will be introduced and explained in details.

CHAPTER FIVE: USING ONTOLOGY FOR PRESENTING THE ARTEFACTS OF REFERENCE ARCHITECTURE

5.1 Overview

A domain ontology is an ontology that captures concepts, relationships and properties about a domain. The defined ontological model will be used to present the artefacts of a reference architecture. The output of coupling the ontological model with the artefacts of the reference architecture called ArchiOntology. The ArchiOntology will provide vocabularies to software developers and architects to facilitate the instantiation process of a software system architecture from a reference architecture. These vocabularies describe the components of the reference architecture. They help the software developers and architects to find the components of the reference architecture by tracking the relationships between them.

This chapter explains in details the development process of the ArchiOntology which has been described in Section 5.2, followed by the process of using the ArchiOntology which is outlined in Section 5.3. Then, two examples are described in Section 5.4 to show the development process of the ArchiOntology and how the ArchiOntology provides vocabularies. Finally, a summary of the chapter is illustrated in Section 5.5.

5.2 Development Process of ArchiOntology

A development process of ArchiOntology model will be explained in details. In chapter 4, the general vocabularies were defined and the ontological model is constructed based on those general vocabularies. The ArchiOntology will be constructed based on the general ontological model, the knowledge and experience of a domain engineer and the vocabularies of a reference architecture. The ArchiOntology represents the components, relationships, and the constraints of a reference architecture for a specific domain. It represents a conceptual model of the reference architecture in an organised and structural way. It will help the software architects and developers to track the relationships between the components. Figure 5-1 shows the steps of the development process of the ArchiOntology.

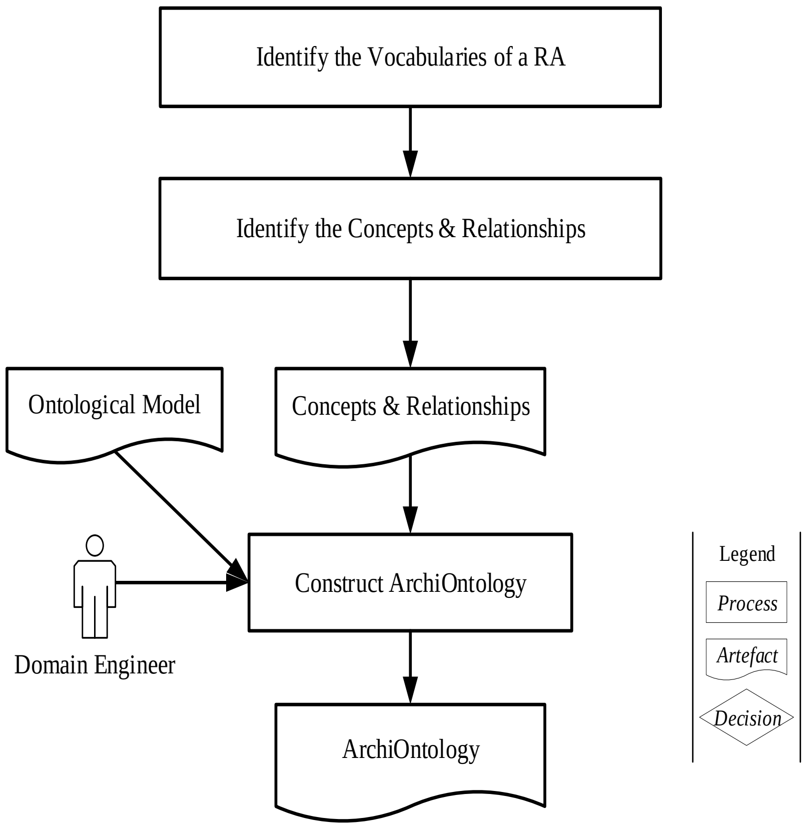


Figure 5-1: Development Process of ArchiOntology

The development process of the ArchiOntology model includes the following steps:

1- Identify the vocabularies of a reference architecture. The first step in the development process of the ArchiOntology for a specific domain is identifying vocabularies (Concepts) of a reference architecture. There are different methods that are used to extract knowledge from sources. These methods are classified into three types [134]: manual such as [135], semiautomatic such as [136], [137], and automatic such as [138], [139]. In this work, the concepts of a reference architecture are extracted manually by the researcher. The output of this step is a set of the vocabularies which are used to describe the artefacts of the reference architecture.
2- Identify the concepts which are used to describe the components of a reference architecture from the extracted vocabularies.
3- Identify the instances of the extracted concepts.
4- Identify the relationships between the concepts.
5- Identify the attributes of the concepts.
6- Identify the constraints that describe the conditions and rationales.
7- Construct ArchiOntology.

In this step, a construction process of the ArchiOntology is explained in details. The ArchiOntology resulted from coupling the general ontological model and the vocabularies of the reference architecture which are used to describe the artefacts of the reference architecture.

A domain engineer who designs the reference architecture plays a considerable role in identifying the artefacts of the reference architecture. The domain engineer has knowledge and experience that are used in the development process of the reference architecture.

However, these knowledge and experience are not documented in the reference architecture but are embedded in her/his mind.

Several tools can be used to present an ontology, such as **Ontolingua Server** (Farquhar et al., 1997 cited in [73]), **WebOnto** (Domingue 1998 cited in [73]), **WebODE** (Arpirez et a1., 2001 cited in [73]), **OntoEdit** (Sure et a1. 2002 cited in [73]) and **Protégé** tool [83]. The current research opted to choose the Protégé tool out of all known ontology tool. The following guidelines describe the construction process of the ArchiOntology:

1- Present the extracted concepts as subclasses from the classes of the ontological model. In Protégé will be presented as OWL subclasses [83].
2- Present the identified instances as OWL individuals [83].
3- Present the extracted relationships as OWL object properties [83].
4- Present the extracted attributes of the concepts and their instances as OWL data properties [83].
5- Present the constraints as OWL data and object properties [83] such as '*exclude*' and '*cannot be with*'. The object property determines the dependencies between the concepts.

The output of this step will be the ArchiOntology for a specific domain. The ArchiOntology presents the artefacts of the reference architecture in an organised and structural way.

5.3 Using Process of ArchiOntology

ArchiOntology presents the vocabularies which are used to describe the components, relationships and the constraints of a reference architecture in an organised and structured way. It provides vocabularies to software architects and developers. These vocabularies help the software architects and developers to find the concepts by tracking the relationships between them.

The ArchiOntology aims at facilitating the instantiation process of a software system architecture from a reference architecture. Figure 5-2 demonstrates the steps of the process of using the ArchiOntology.

The following steps describe the process of using the ArchiOntology.

Step 1: Identify the user of a system. It is the first step to determine the requirements of the desired system.

Step 2: Identify the requirements of the system from the user of the system.

Step 3: Identify possible concepts and relationships between them from the requirements of the system.

Step 4: Compare the extracted concepts with the concepts of the ArchiOntology.

A. If the concepts of the ArchiOntology fits the extracted concepts, then identify other concepts by tracing the relationships of the identified concepts. Ontology reasoning technique [92] is used to check the consistency between concepts. FaCT++ reasoner [140] is adopted in this research. Furthermore, DL Query plugin in Protégé tool [83] is used to find the concepts and individuals.
B. If the concepts of the ArchiOntology does not fit the extracted concept, then:
    a. Define a new concept.
    b. Define instances of the new concept, if required, which will be represented as individuals in the ontology.

c. Identify the attributes of the new concepts. The attributes will be represented as Data Properties in the ontology.
d. Define constraints for the new concepts.
e. Update the ArchiOntology by adding the new concepts to the ArchiOntology with its attributes, instances, relationships and constraints. Ontology developer will update the ArchiOntology by adding new concepts and defining relationships between them.

Step 5: Repeat Step 4 until completing all concepts.

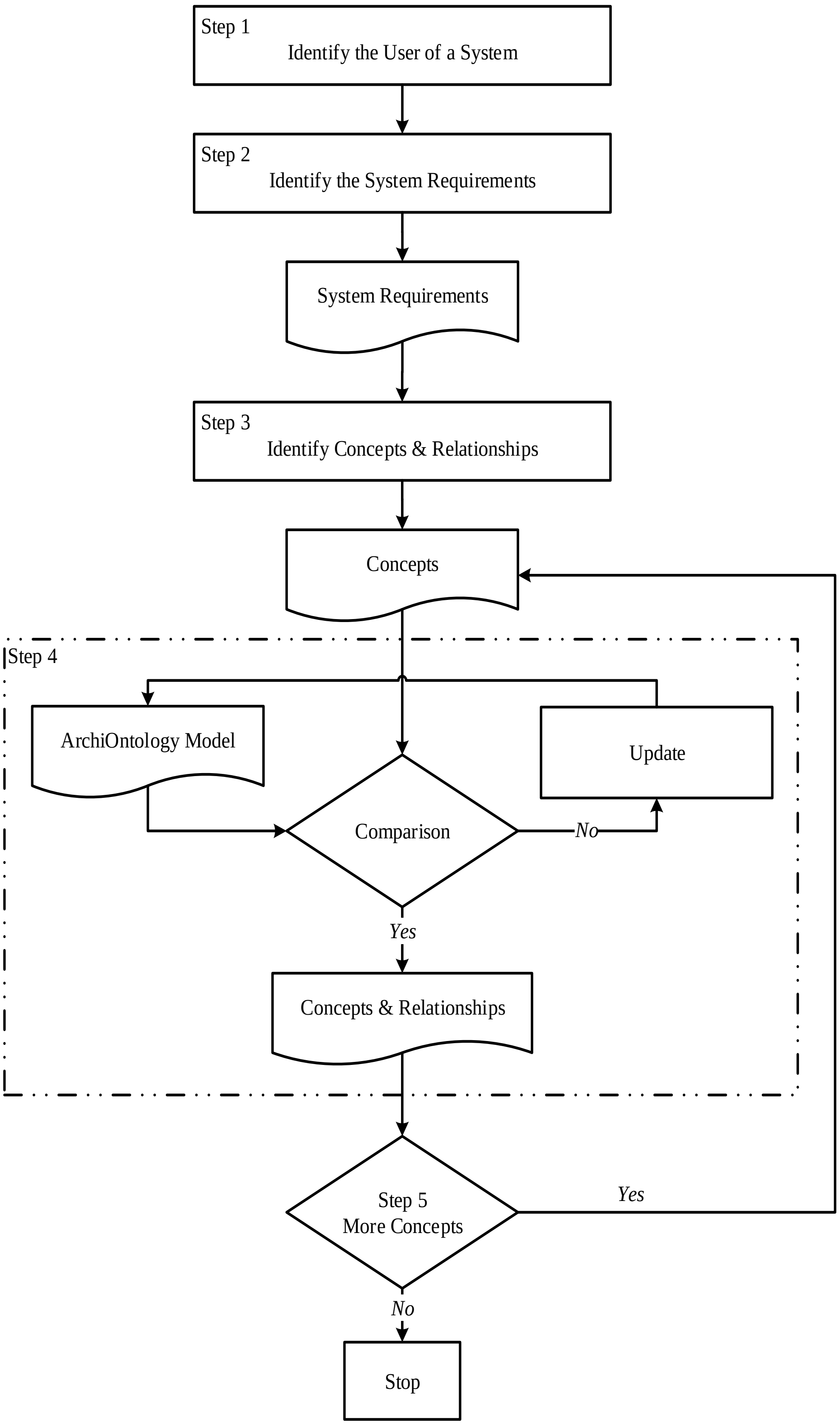


Figure 5-2: Process of Using ArchiOntology

5.4 Examples

In this section, we are going to apply the proposed methodology in two examples to show the workflow of the development process of ArchiOntology and how the ArchiOntology provides vocabularies to software developers and software architects. These examples have been selected for the following reasons:

5. These examples are already published in IEEE conference.
6. The first example has 135 citations and the second one has 74 citations.
7. These examples are already used to derive different architectures.
8. Nevertheless, the information about the domain in the papers is concise.

5.4.1 Example 1: The Reference Architecture of the Web Browsers

This example illustrates the workflow of the development process of ArchiOntology for a web browser and how the ArchiOntology provides vocabularies to the developers based on a paper written by Grosskurth and Godfrey [47].

A proposed process for designing ArchiOntology model was applied to design the ArchiOntology model for the web browser. The researcher analysed the reference architecture, by reviewing it, to extract the artefacts of the reference architecture. The artefacts of the reference architecture are explained as below:

- The reference architecture includes eight main subsystems with each of them having different functions. These subsystems are User Interface, Browser Engine, Rendering Engine, Data Persistence, Networking, JavaScript Interpreter, XML Parser and Display Backend. Figure 5-3 illustrates the subsystem of the web browser.
- The subsystems use different resources.

- The users execute web browsers on different hardware such as computers and cell phones.
- The web browsers use a hypertext transfer protocol to access to information in web servers.
- A layered architectural style is used to represent the architecture of the system.
- The web pages are written using the HyperText Markup Language and Cascading Style Sheets.
- The connection between subsystems are:

  A. The User Interface subsystem connects to the Data Persistent, Display Backend, and the Browser Engine subsystem.

  B. The Browser Engine subsystem connects to the Data Persistent, and Rendering Engine subsystem.

  C. The Rendering Engine subsystem connects to the Networking, JavaScript Interpreter, XML Parser and Display Backend subsystem.

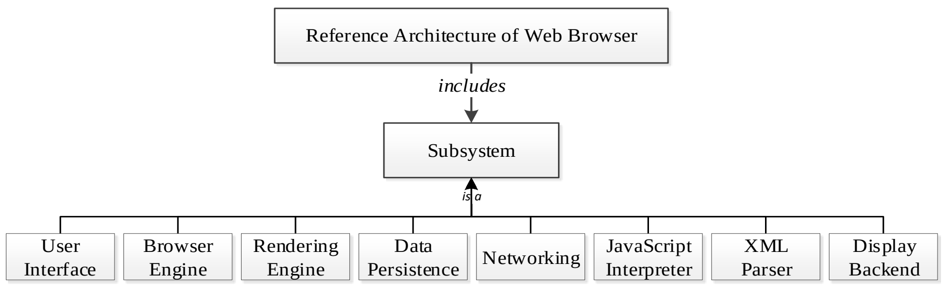


Figure 5-3: Subsystems of the Web Browser Reference Architecture [47]

- The functions of subsystems:

  A. The functions of the user interface subsystem (see Figure 5-4) are:

     1- Connect a user of the web browser to the browser engine subsystem.

     2- Provide features such as toolbars, visual page-load progress, smart download handling, preferences and printing.

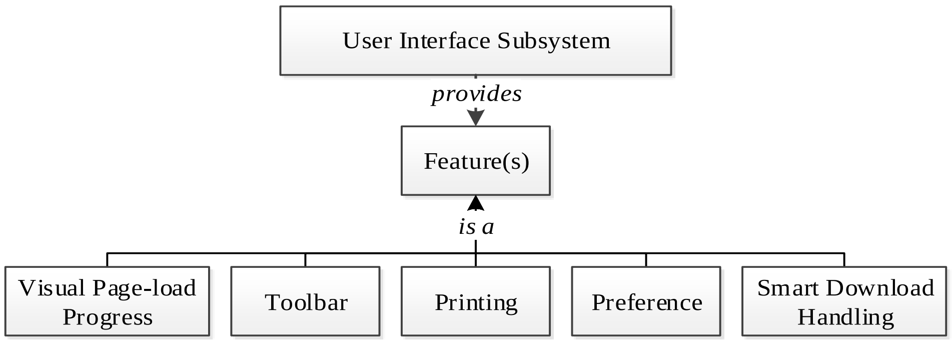


Figure 5-4: Features of the User Interface Subsystem [47]

B. The functions of the Browser Engine subsystem are:

1- Provide a high-level interface to the rendering engine subsystem.

2- Load a given URI.

3- Support primitive browsing actions.

4- Provide hooks for viewing the browsing session.

5- Allow the querying and manipulation of the rendering engine settings.

C. The functions of the Rendering Engine subsystem are:

1- Produce a visual representation for a given URI.

2- Display HTML and XML documents.

3- Calculate the exact web page layout.

4- Include the HTML parser.

D. The functions of the Networking subsystem are:

1- Implement file transfer protocols such as HTTP and FTP.

2- Resolve the MIME file.

3- Implement a cache of recently retrieved resources.

E. The function of the JavaScript Interpreter subsystem is:

- Evaluate JavaScript code.

F. The function of the XML Parser subsystem is:

        - Parse XML documents into a Document Object Model (DOM) tree.

G. The functions of the Display Backend subsystem are:

    1- Provide drawing and windowing primitives.

    2- Provide a set of interfaces.

    3- Provide a set of fonts.

H. The function of the Data Persistence subsystem is:

    - Store various data associated with the browsing session on disk.

- The subsystems of the web browser system include different components:

A. The user interface subsystem includes two components; user interface and UI Toolkit (XPEE). The user interface component provides features to the UI Toolkit (XPEE).

B. The networking subsystem includes three components; Necko, Wwwlib and Security.

C. The data persistence subsystem includes four components; User, Secure, Browser, and Persist. These components connect to each other and exchange features between them.

D. The JavaScript interpreter subsystem includes Spider−Monkey component.

E. The display backend subsystem includes three components; GTK+ Adapter, Curses and GTK+ / X11 Libraries.

F. The XML Parser Subsystem includes Expat component.

The constraint of the Lynx's architecture is that it does not include the JavaScript Interpreter and XML Parser subsystems. To represent this constraint, we defined an *exclude* relationship that describes this situation. The exclude relationship will be represented as an ObjectProperty in the ontology. The domain will be Lynx web browser, and the range will be JavaScript Interpreter and XML Parser subsystems.

The authors of the paper used various vocabularies to describe the artefacts of the reference architecture. These vocabularies describe the objects and the relationships between them as explained in Table 5-1 and Table 5-2.

Table 5-1: Vocabularies Describe the Objects of the Web Browser

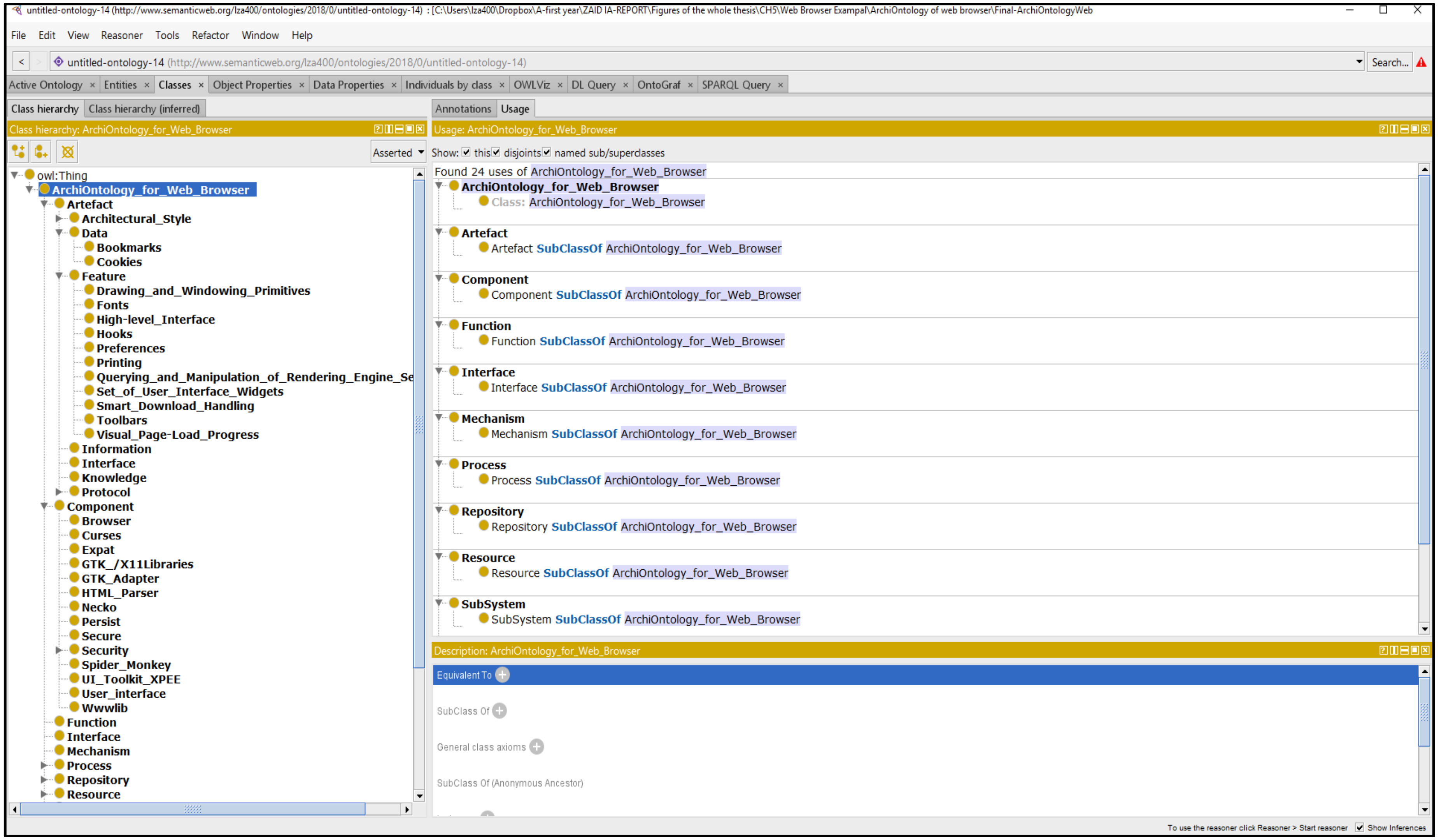
Vocabulary			
System	Web Browser System	Subsystem	Networking Subsystem
Function	Rendering Engine Subsystem	Browser Engine Subsystem	User Interface Subsystem
Resource	JavaScript Interpreter Subsystem	Data Persistence Subsystem	Display Backend Subsystem
Resource Hardware	Feature	Computer	XML Parser Subsystem
Web Server	Cell phones	HTML	HTTP
Toolbars Feature	Web Page	Smart Download Handling Feature	Preferences Feature
Printing Feature	Visual Page-load Progress Feature	Set of User Interface Widgets	Layered Architectural Style
CSS	Drawing and Windowing Primitives	User	Hooks
Necko Component	Component	wwwlib Component	UI Toolkit (XPEE) Component
Querying and Manipulation of the Rendering Engine	User interface Component	Browser Component	Security (Libgnutls) Component
Spider−Monkey Component	Security (NSS/PSM) Component	GTK+ / X11 Libraries Component	Persist Component
Fonts	Secure component	HTML Parser Component	Expat Component
Curses Component	GTK+ Adapter Component	Disk	Cookies Data
Bookmarks Data	XML	HTTP	FTP
JavaScript Code	Data	Visual Representation	

Table 5-2: Vocabularies Describe the Relationships

Vocabulary			
Access to	Allow a	Apply to	Calculate a
Connect to	Display a	Evaluate a	Execute on
Has a	Implement a	Include a	Load a
Parse a	Produce a	Provide a	Provide feature to
Represented as	Store in	Support a	Used by
Written by	Connect a	Execute a	Use a
Stored in			

The ArchiOntology for a web browser is constructed based on the general ontological model, general vocabulary, domain engineer and the extracted artefacts from the reference architecture. Protégé tool [83] is used to translate the artefacts of the reference architecture into a machine-processable ontology represented in OWL.

The extracted vocabularies are used to construct the ArchiOntology for the web browser as illustrated in Figure 5-5. Figure 5-6 shows the main concepts (classes) and subconcepts (subclasses) of the ArchiOntology of the web browser and Figure 5-7 illustrates the individuals of the ArchiOntology of the web browser. The object properties of the ArchiOntology are described in Figure 5-8.



untitled-ontology-14 (http://www.semanticweb.org/lza400/ontologies/2018/0/untitled-ontology-14) : [C:\Users\lza400\Dropbox\A-first year\ZAID IA-REPORT\Figures of the whole thesis\CH5\Web Browser Exampal\ArchiOntology of web browser\Final-ArchiOntologyWeb

File Edit View Reasoner Tools Refactor Window Help

untitled-ontology-14 (http://www.semanticweb.org/lza400/ontologies/2018/0/untitled-ontology-14)

Active Ontology × | Entities × | Classes × | Object Properties × | Data Properties × | Individuals by class × | OWLViz × | DL Query × | OntoGraf × | SPARQL Query ×

Class hierarchy | Class hierarchy (inferred)

Class hierarchy: ArchiOntology_for_Web_Browser

Asserted

```
owl:Thing
  ArchiOntology_for_Web_Browser
    Artefact
      Architectural_Style
      Data
        Bookmarks
        Cookies
      Feature
        Drawing_and_Windowing_Primitives
        Fonts
        High-level_Interface
        Hooks
        Preferences
        Printing
        Querying_and_Manipulation_of_Rendering_Engine_Se
        Set_of_User_Interface_Widgets
        Smart_Download_Handling
        Toolbars
        Visual_Page-Load_Progress
      Information
      Interface
      Knowledge
      Protocol
    Component
      Browser
      Curses
      Expat
      GTK_/X11Libraries
      GTK_Adapter
      HTML_Parser
      Necko
      Persist
      Secure
      Security
      Spider_Monkey
      UI_Toolkit_XPEE
      User_interface
      Wwwlib
    Function
    Interface
    Mechanism
    Process
    Repository
    Resource
```

Annotations | Usage

Usage: ArchiOntology_for_Web_Browser

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

Found 24 uses of ArchiOntology_for_Web_Browser

```
ArchiOntology_for_Web_Browser
    Class: ArchiOntology_for_Web_Browser
Artefact
    Artefact SubClassOf ArchiOntology_for_Web_Browser
Component
    Component SubClassOf ArchiOntology_for_Web_Browser
Function
    Function SubClassOf ArchiOntology_for_Web_Browser
Interface
    Interface SubClassOf ArchiOntology_for_Web_Browser
Mechanism
    Mechanism SubClassOf ArchiOntology_for_Web_Browser
Process
    Process SubClassOf ArchiOntology_for_Web_Browser
Repository
    Repository SubClassOf ArchiOntology_for_Web_Browser
Resource
    Resource SubClassOf ArchiOntology_for_Web_Browser
SubSystem
    SubSystem SubClassOf ArchiOntology_for_Web_Browser
```

Description: ArchiOntology_for_Web_Browser

Equivalent To

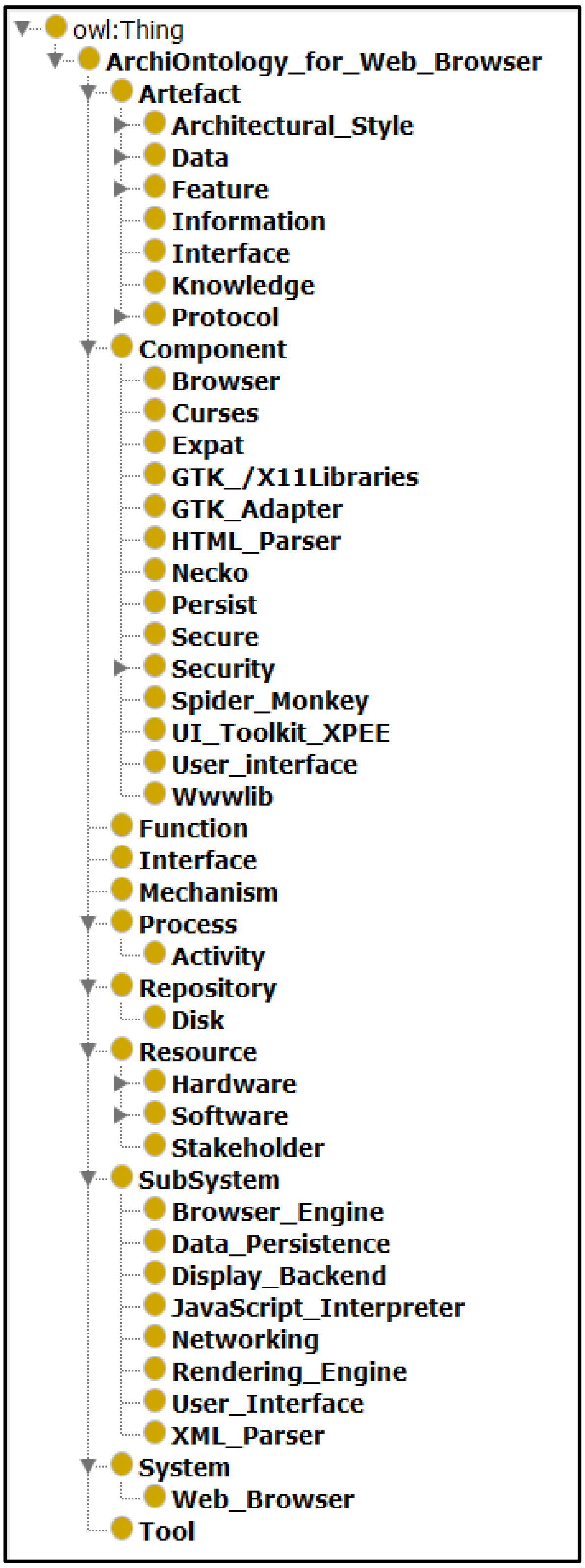
SubClass Of

General class axioms

SubClass Of (Anonymous Ancestor)

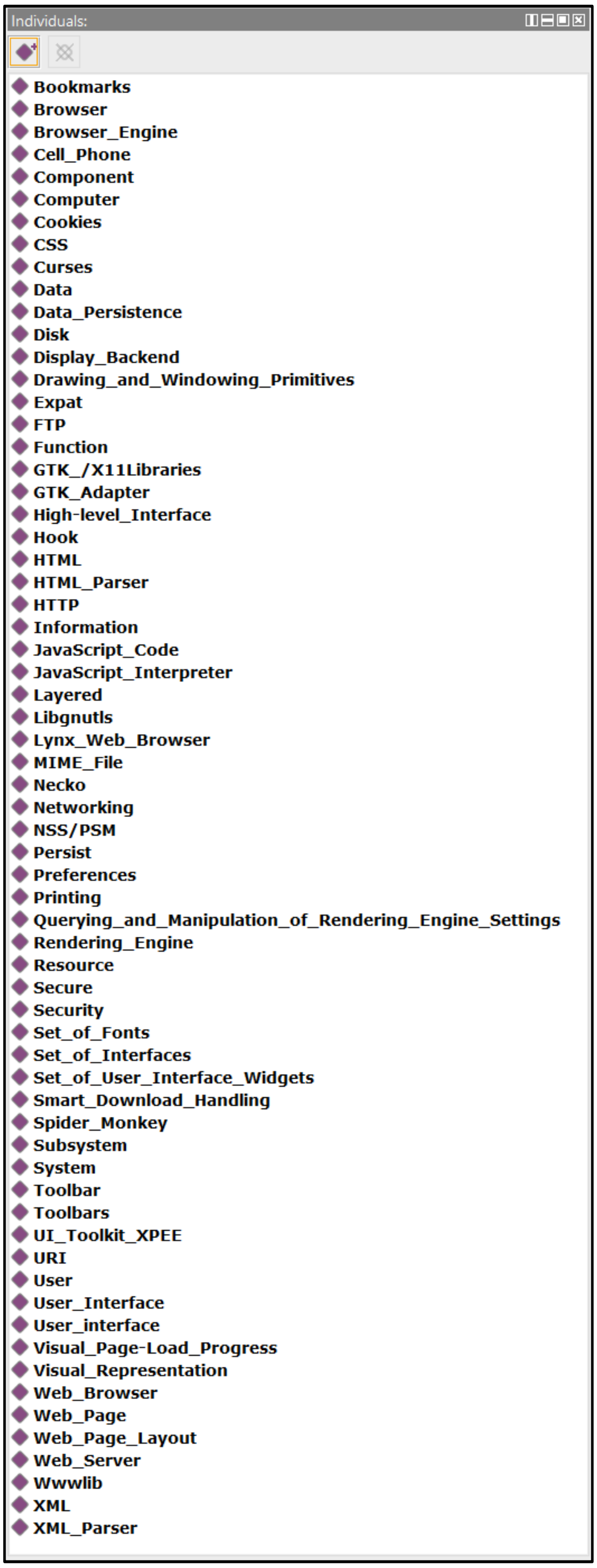
To use the reasoner click Reasoner > Start reasoner ☑ Show Inferences

Figure 5-5: ArchiOntology of the Web Browser in Protégé



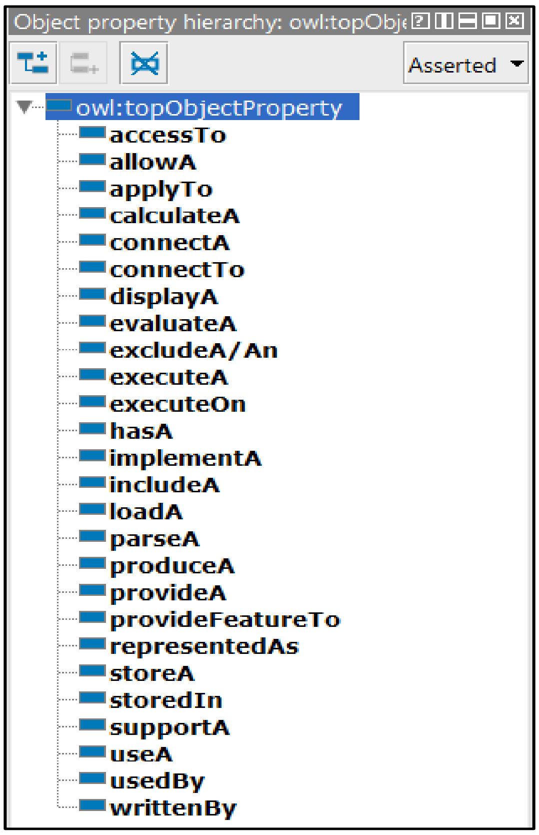
- owl:Thing
  - ArchiOntology_for_Web_Browser
    - Artefact
      - Architectural_Style
      - Data
      - Feature
      - Information
      - Interface
      - Knowledge
      - Protocol
    - Component
      - Browser
      - Curses
      - Expat
      - GTK_/X11Libraries
      - GTK_Adapter
      - HTML_Parser
      - Necko
      - Persist
      - Secure
      - Security
      - Spider_Monkey
      - UI_Toolkit_XPEE
      - User_interface
      - Wwwlib
    - Function
    - Interface
    - Mechanism
    - Process
      - Activity
    - Repository
      - Disk
    - Resource
      - Hardware
      - Software
      - Stakeholder
    - SubSystem
      - Browser_Engine
      - Data_Persistence
      - Display_Backend
      - JavaScript_Interpreter
      - Networking
      - Rendering_Engine
      - User_Interface
      - XML_Parser
    - System
      - Web_Browser
    - Tool

Figure 5-6: Main Concepts and Subconcepts of the ArchiOntology of the Web Browser


Individuals:

Bookmarks
Browser
Browser_Engine
Cell_Phone
Component
Computer
Cookies
CSS
Curses
Data
Data_Persistence
Disk
Display_Backend
Drawing_and_Windowing_Primitives
Expat
FTP
Function
GTK_/X11Libraries
GTK_Adapter
High-level_Interface
Hook
HTML
HTML_Parser
HTTP
Information
JavaScript_Code
JavaScript_Interpreter
Layered
Libgnutls
Lynx_Web_Browser
MIME_File
Necko
Networking
NSS/PSM
Persist
Preferences
Printing
Querying_and_Manipulation_of_Rendering_Engine_Settings
Rendering_Engine
Resource
Secure
Security
Set_of_Fonts
Set_of_Interfaces
Set_of_User_Interface_Widgets
Smart_Download_Handling
Spider_Monkey
Subsystem
System
Toolbar
Toolbars
UI_Toolkit_XPEE
URI
User
User_Interface
User_interface
Visual_Page-Load_Progress
Visual_Representation
Web_Browser
Web_Page
Web_Page_Layout
Web_Server
Wwwlib
XML
XML_Parser

Figure 5-7: Individuals of the ArchiOntology of the Web Browser in Protégé

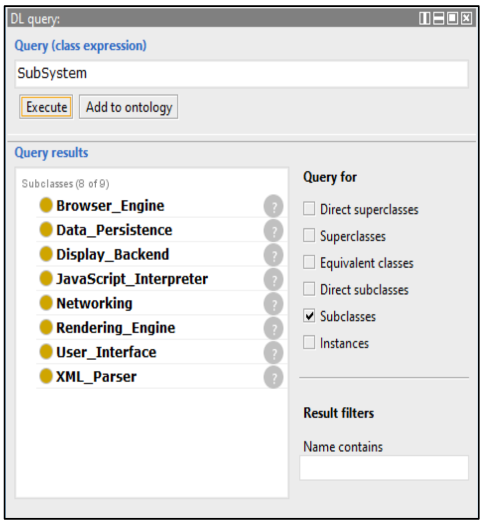

Object property hierarchy: owl:topObje

Asserted

owl:topObjectProperty
- accessTo
- allowA
- applyTo
- calculateA
- connectA
- connectTo
- displayA
- evaluateA
- excludeA/An
- executeA
- executeOn
- hasA
- implementA
- includeA
- loadA
- parseA
- produceA
- provideA
- provideFeatureTo
- representedAs
- storeA
- storedIn
- supportA
- useA
- usedBy
- writtenBy

Figure 5-8: Object Properties of the ArchiOntology of the Web Browser in Protégé

FaCT++ reasoner is used to check the consistency of the classes and DL Query plugin is used to execute enquiry such as enquiry about the subsystem as demonstrated in Figure 5-9. Figure 5-10 and Figure 5-11 show the features and the components of the ArchiOntology of the web browser, respectively.


DL query:

Query (class expression)

SubSystem

Execute | Add to ontology

Query results

Subclasses (8 of 9)
- Browser_Engine
- Data_Persistence
- Display_Backend
- JavaScript_Interpreter
- Networking
- Rendering_Engine
- User_Interface
- XML_Parser

Query for
- [ ] Direct superclasses
- [ ] Superclasses
- [ ] Equivalent classes
- [ ] Direct subclasses
- [✓] Subclasses
- [ ] Instances

Result filters

Name contains

Figure 5-9: Subsystem of the ArchiOntology

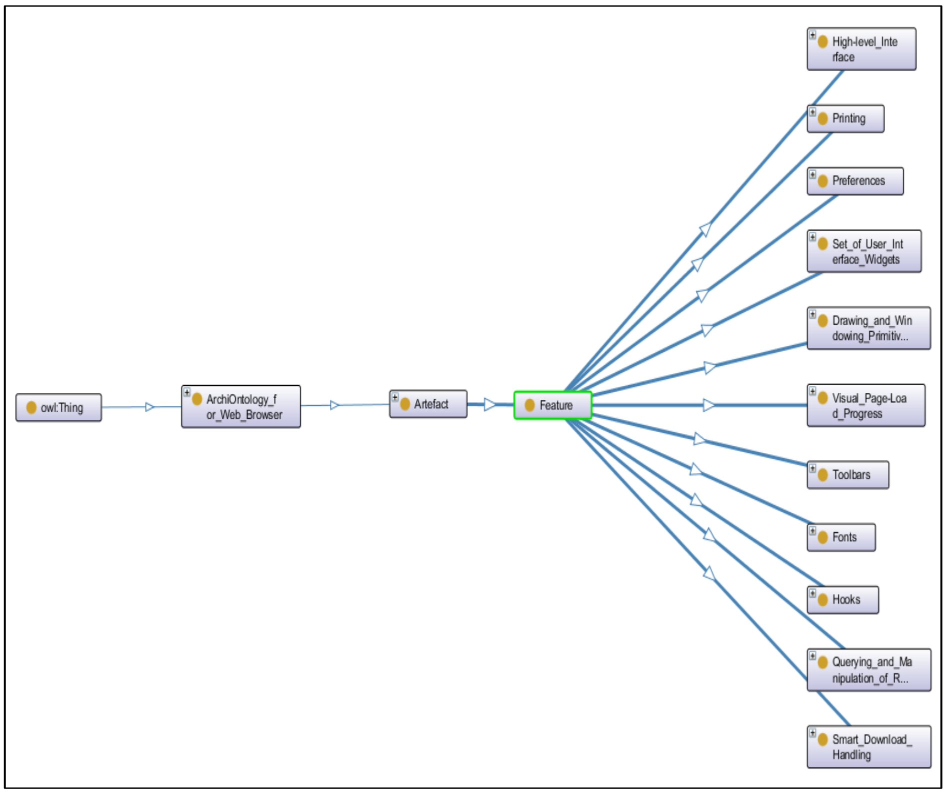

owl:Thing
ArchiOntology_for_Web_Browser
Artefact
Feature
High-level_Interface
Printing
Preferences
Set_of_User_Interface_Widgets
Drawing_and_Windowing_Primitiv...
Visual_Page-Load_Progress
Toolbars
Fonts
Hooks
Querying_and_Manipulation_of_R...
Smart_Download_Handling

Figure 5-10: Features of the Subsystem of the Web Browser in Protégé

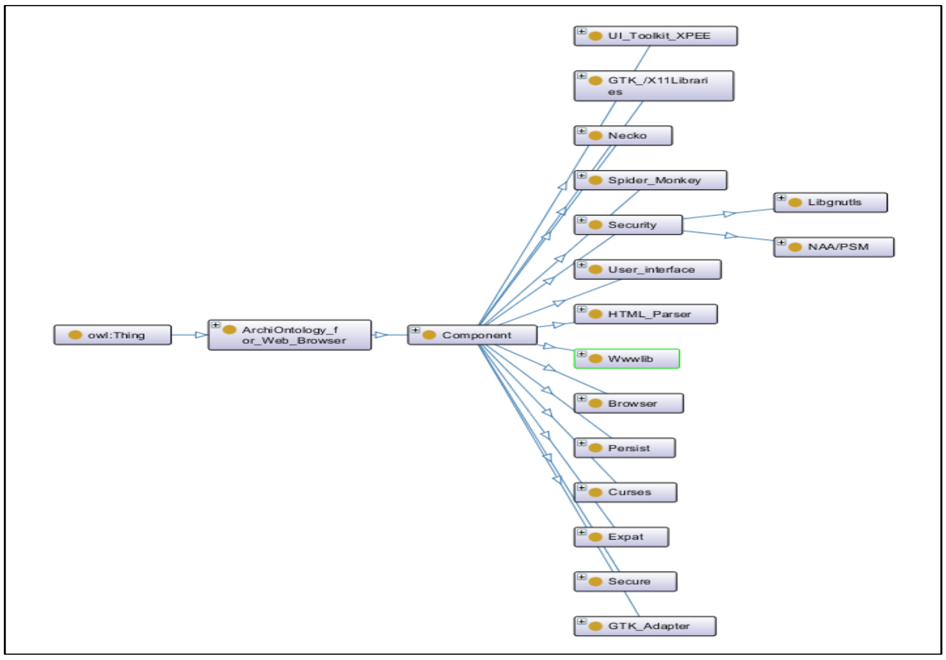


Figure 5-11: Components of the ArchiOntology of the Web Browser in Protégé

Figure 5-12 illustrates the hierarchy of the ArchiOntology for the web browser represented using OWLViz plugin in Protégé tool [83].

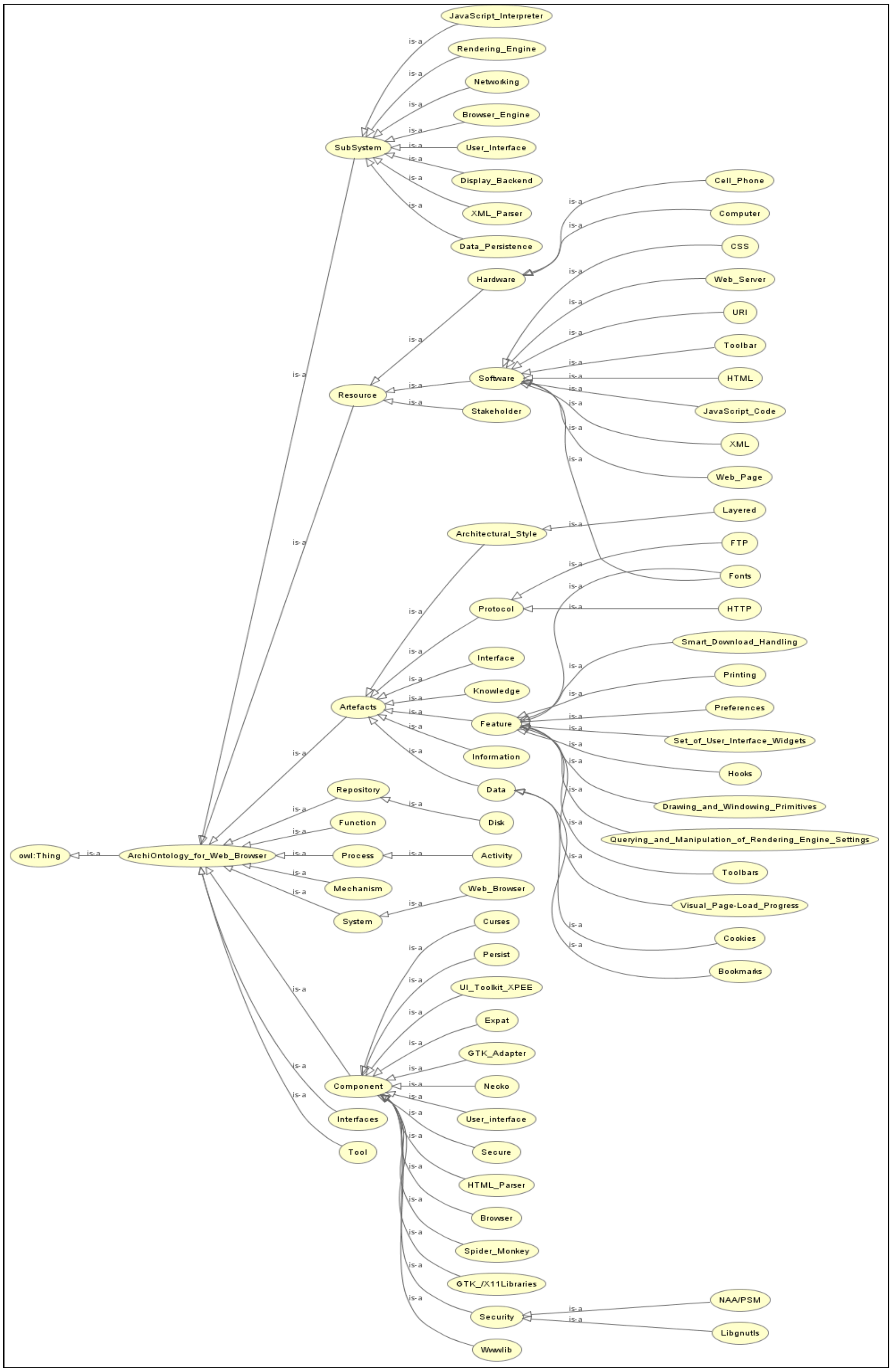


Figure 5-12: Hierarchy of the ArchiOntology of the Web Browser

5.4.2 Example 2: The Reference Architecture of the Web Servers

This example illustrates the workflow of the development process of ArchiOntology for web servers and how the ArchiOntology provides vocabularies to the developers based on a paper written by Hassan and Holt [44]. A proposed process was applied to develop the ArchiOntology model for the web servers. The researcher analysed the reference architecture, by reviewing it, to extract the artefacts of the reference architecture (the artefacts are represented in [44]). In this example, authors of the paper are used 47 vocabularies to describe the artefacts of the reference architecture. Table 5-3 and Table 5-4 show the vocabularies of the reference architecture of web servers and the relationships between them, respectively.

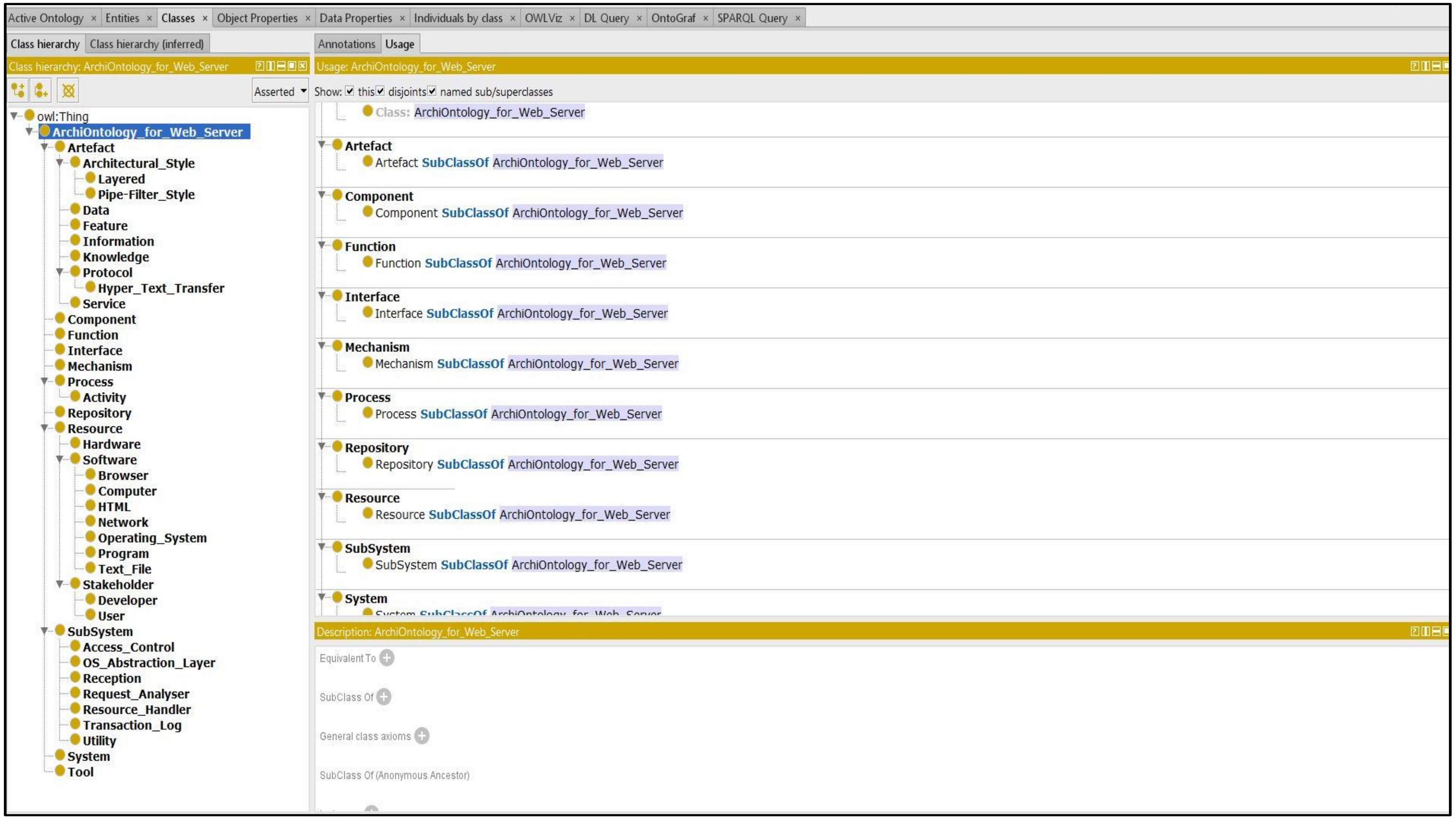
Table 5-3: Vocabulary of the Reference Architecture

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Subsystem	3.	Component
4.	Developer	5.	Security	6.	User
7.	Resource	8.	HTML	9.	Text File
10.	Service	11.	Daily News Service	12.	Email Service
13.	Architectural Style	14.	Pipe-Filter Style	15.	Layered Architectural Style
16.	Program	17.	Java Servlet Program	18.	Common Gateway Interface Program
19.	Protocol	20.	Hyper Text Transfer Protocol	21.	Operating System
22.	Network	23.	Computer	24.	Browser
25.	Netscape Navigator Browser	26.	Lynx Browser	27.	Utility Subsystem
28.	Internet Explorer Browser	29.	Transaction Log Subsystem	30.	Request Analyser Subsystem
31.	OS Abstraction Layer Subsystem	32.	Resource Handler Subsystem	33.	Reception Subsystem
34.	Access Control Subsystem	35.		36.	

Table 5-4: Relationships between the Vocabularies of the Reference Architecture

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Encompasses	2.	Use a	3.	Consist of
4.	Describe a	5.	Instance of	6.	Define a
7.	Is part of	8.	Is a	9.	Apply to
10.	Include a	11.	Control a	12.	Require a

The ArchiOntology for web servers is constructed which represents the artefacts of the reference architecture of the web servers. Protégé tool is used to present these artefacts. Figure 5-13 shows the ArchiOntology for the web servers. Figure 5-14 shows the main concepts (classes) and subconcepts (subclasses) of the ArchiOntology of the web servers and Figure 5-15 illustrates the individuals of the ArchiOntology of the web servers. Figure 5-16 shows the Classes, Subclasses and individual of the ArchiOntology of the web servers which are represented in OntoGraf plugin [83]. Figure 5-17 illustrates individuals of the Browser, Program and Service subclasses. Figure 5-18 illustrates the hierarchy concepts of the ArchiOntology of the web Servers which are represented in OWLViz plugin [83].



Active Ontology × | Entities × | Classes × | Object Properties × | Data Properties × | Individuals by class × | OWLViz × | DL Query × | OntoGraf × | SPARQL Query ×

Class hierarchy | Class hierarchy (inferred)

Class hierarchy: ArchiOntology_for_Web_Server

Asserted

- owl:Thing
  - ArchiOntology_for_Web_Server
    - Artefact
      - Architectural_Style
        - Layered
        - Pipe-Filter_Style
      - Data
      - Feature
      - Information
      - Knowledge
      - Protocol
        - Hyper_Text_Transfer
      - Service
    - Component
    - Function
    - Interface
    - Mechanism
    - Process
      - Activity
    - Repository
    - Resource
      - Hardware
      - Software
        - Browser
        - Computer
        - HTML
        - Network
        - Operating_System
        - Program
        - Text_File
      - Stakeholder
        - Developer
        - User
    - SubSystem
      - Access_Control
      - OS_Abstraction_Layer
      - Reception
      - Request_Analyser
      - Resource_Handler
      - Transaction_Log
      - Utility
    - System
    - Tool

Annotations | Usage

Usage: ArchiOntology_for_Web_Server

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

Class: ArchiOntology_for_Web_Server

- Artefact
  - Artefact SubClassOf ArchiOntology_for_Web_Server
- Component
  - Component SubClassOf ArchiOntology_for_Web_Server
- Function
  - Function SubClassOf ArchiOntology_for_Web_Server
- Interface
  - Interface SubClassOf ArchiOntology_for_Web_Server
- Mechanism
  - Mechanism SubClassOf ArchiOntology_for_Web_Server
- Process
  - Process SubClassOf ArchiOntology_for_Web_Server
- Repository
  - Repository SubClassOf ArchiOntology_for_Web_Server
- Resource
  - Resource SubClassOf ArchiOntology_for_Web_Server
- SubSystem
  - SubSystem SubClassOf ArchiOntology_for_Web_Server
- System
  - System SubClassOf ArchiOntology_for_Web_Server

Description: ArchiOntology_for_Web_Server

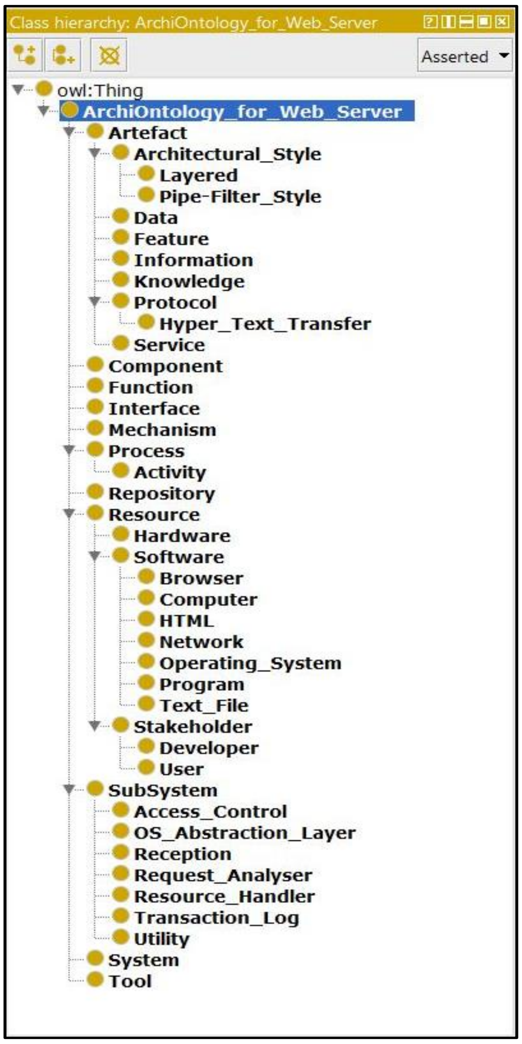
Equivalent To

SubClass Of

General class axioms

SubClass Of (Anonymous Ancestor)

Figure 5-13: ArchiOntology of the Web Servers

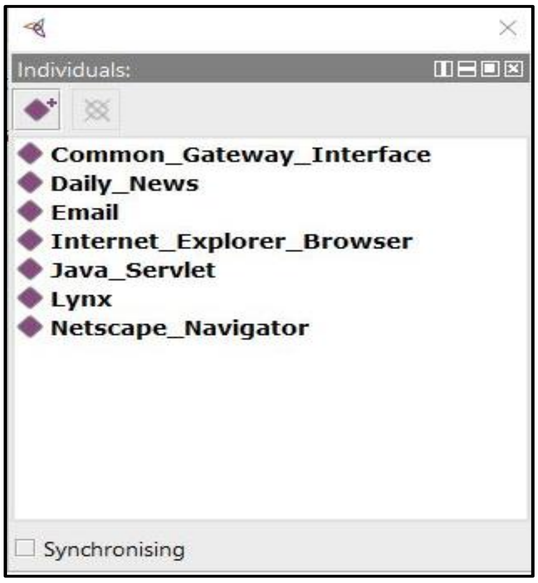


Class hierarchy: ArchiOntology_for_Web_Server

Asserted

```
owl:Thing
  ArchiOntology_for_Web_Server
    Artefact
      Architectural_Style
        Layered
        Pipe-Filter_Style
      Data
      Feature
      Information
      Knowledge
      Protocol
        Hyper_Text_Transfer
      Service
    Component
    Function
    Interface
    Mechanism
    Process
      Activity
    Repository
    Resource
      Hardware
      Software
        Browser
        Computer
        HTML
        Network
        Operating_System
        Program
        Text_File
      Stakeholder
        Developer
        User
    SubSystem
      Access_Control
      OS_Abstraction_Layer
      Reception
      Request_Analyser
      Resource_Handler
      Transaction_Log
      Utility
    System
    Tool
```

Figure 5-14: Main Concepts and Subconcepts of the ArchiOntology of the Web Servers



Individuals:

- Common_Gateway_Interface
- Daily_News
- Email
- Internet_Explorer_Browser
- Java_Servlet
- Lynx
- Netscape_Navigator

Synchronising

Figure 5-15: Individuals of the ArchiOntology of the Web Servers

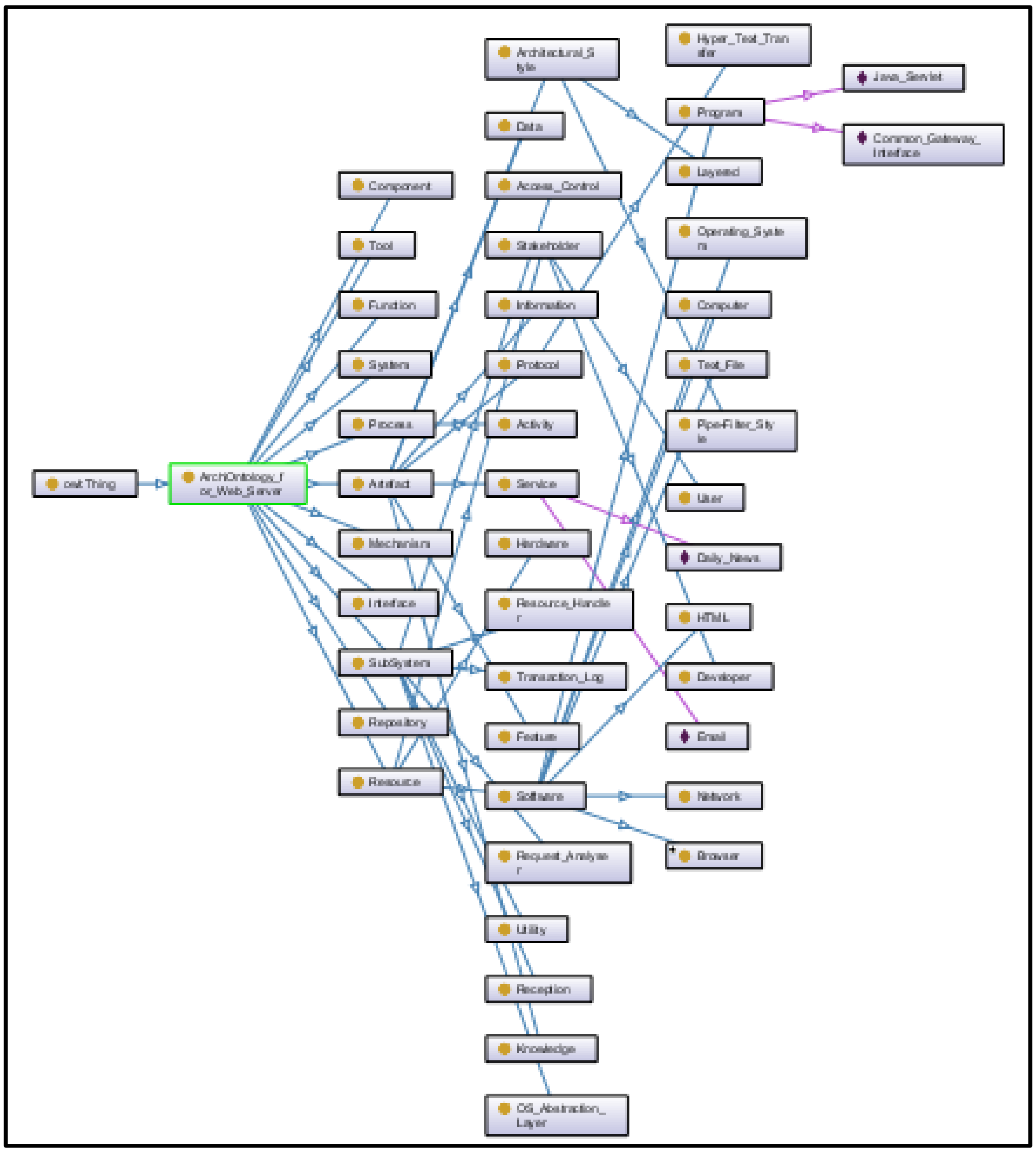
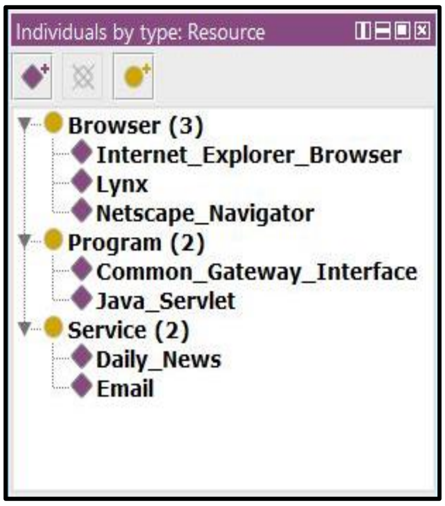


Figure 5-16: Classes, Subclasses and individual of the ArchiOntology



Individuals by type: Resource

- Browser (3)
  - Internet_Explorer_Browser
  - Lynx
  - Netscape_Navigator
- Program (2)
  - Common_Gateway_Interface
  - Java_Servlet
- Service (2)
  - Daily_News
  - Email

Figure 5-17: Individuals of the Browser, Program and Service subclasses

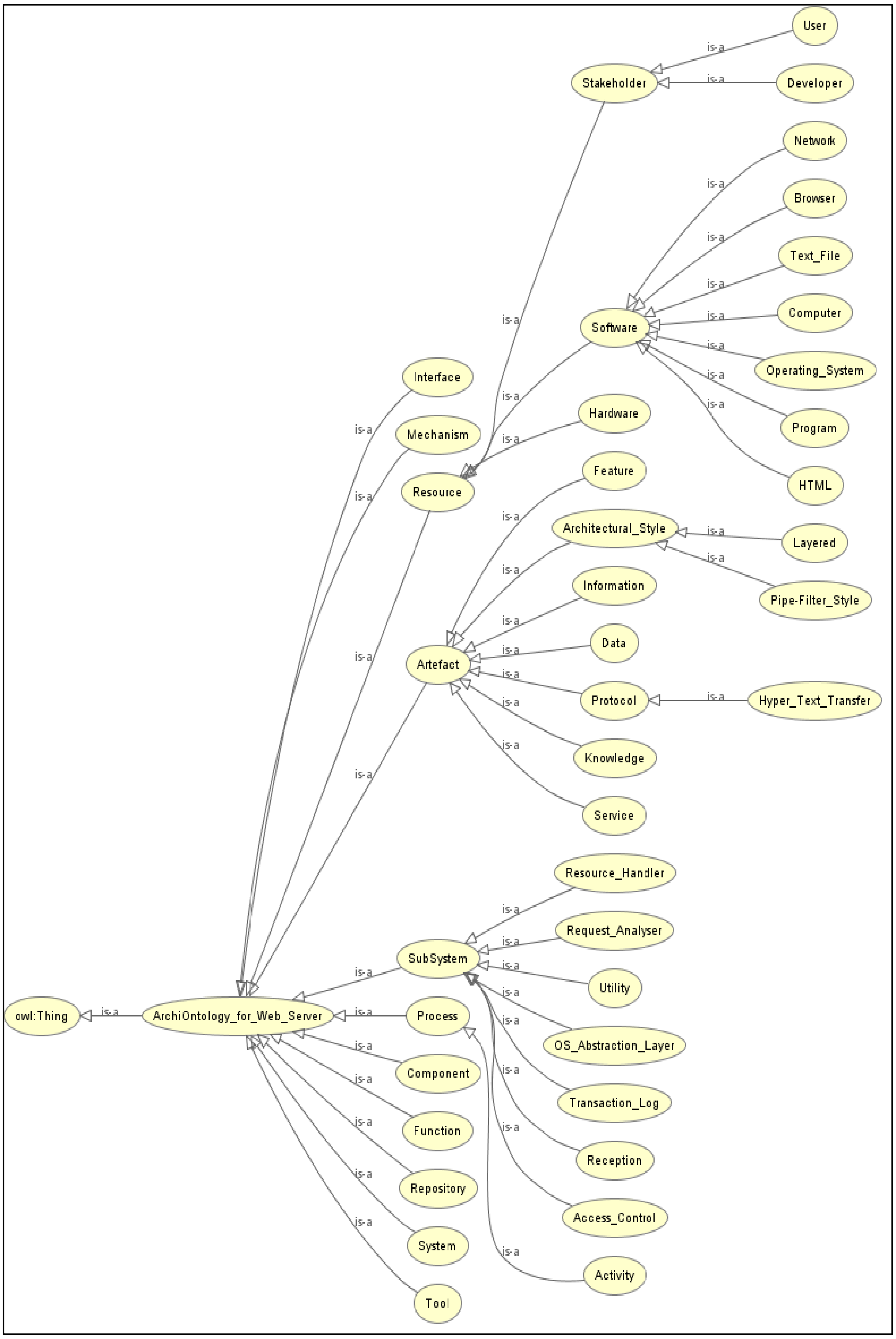


Figure 5-18: Hierarchy of the ArchiOntology of the Web Servers

5.5 Summary

This chapter presented a detailed description of the development process of an ArchiOntology. The ArchiOntology is the output of using the ontological model as a basis to present the artefacts of a reference architecture. This was followed by process of using the ArchiOntology. Next to that, two examples are used to illustrate the processes of developing and using the ArchiOntology.

The next chapter will present the experimental evaluation of the proposed methodology.

CHAPTER SIX: EXPERIMENTAL EVALUATION OF PROPOSED METHODOLOGY

6.1 Overview

In this chapter, the evaluation process pertaining to the proposed methodology is to be dealt with thoroughly. Such evaluation aids with comprehending all limitations that exist within the proposed methodology and accordingly assists in developing a better solution.

First, an evaluation plan is outlined using user study experiment [141] and the Goal/Question/Metric (GQM) approach [142], [143]. In this approach, a group of questions is identified depending on the metrics that are used for evaluating the goal of usability of the proposed methodology according to the research objectives. The evaluation phase is supposed to offer answers for these questions. Section 6.3.1.3 presents the metrics as well as the derived questions that have been used for evaluating the research goal. Finally, the summary in Section 6.4 shapes the eventual results that led to the final answers demanded by the proposed questions.

6.2 Plan of the Evaluation

Every architectural development methodology has its strengths and weaknesses. Basili et al. [143] described and classified several methods for software evaluation. Such evaluation includes a thesis and test process. Since no general evaluation method can be applied for any purpose, software, project and so forth, one has to choose which method best fits the purpose of and resources for evaluation. Based on what is mentioned above, we have tried to evaluate the proposed methodology in term of support for software engineering that focuses on the developer's perception. Therefore, an evaluation plan has been prepared for that purpose. The evaluation plan that was utilised to assess the proposed methodology is illustrated in Figure 6-1 and Table 6-1.

In this research, a user study experiment is applied to evaluate the proposed methodology. A Goal-Question-Metric (GQM) approach [142], [143] is used to ensure the integrity and validity of the results. The goal of the evaluation process is to evaluate the proposed methodology in term of usability in the following aspect:

Can the ontological model facilitate the instantiation process of a software system architecture and minimise the development time?

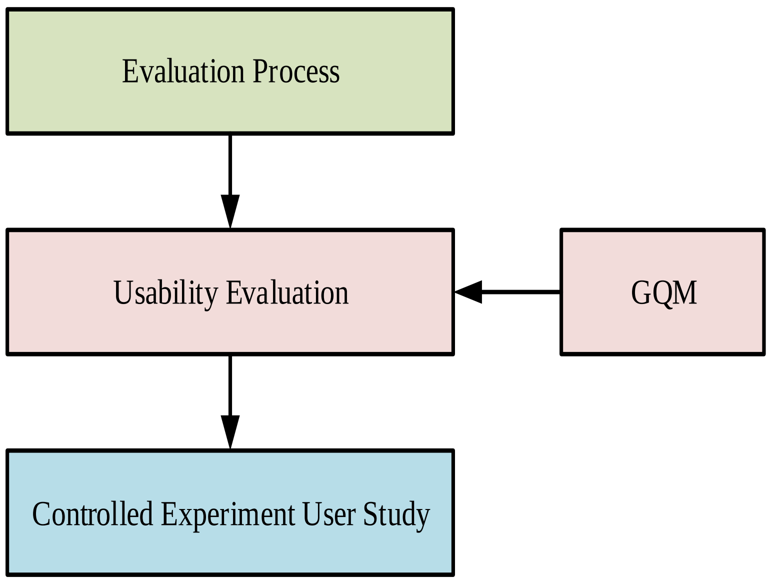


Figure 6-1: Plan of the Evaluation

Table 6-1: Plan of the Evaluation

Evaluation Goal	Evaluation Question	Evaluation Method	Criteria
Usability	Can the ontological model facilitate the instantiation process of a software system architecture and minimise the development time?	Controlled Experiment User Study	• Complexity • Traceability • Understandability • Clarity • Time

6.3 User Study Experiment

For the sake of evaluation of the proposed methodology, a controlled experiment was implemented employing students from the School of Computing, Science and Engineering, Salford University.

This experimental study aimed at checking whether the methodology could simplify the instantiation process of a software system architecture from a reference architecture. Some tasks controlled experiments were performed to assess the proposed methodology regarding the usability. Guidelines for the user study experiment described by Jedlitschka et al. [141] and Goal, Question, Metric (GQM) approach offered by Basili [143] were followed so as to ensure the integrity and correctness of the results of the experiment. Ethical approval has been obtained for this study (See Appendix D.1).

6.3.1 Goal/Question/Metric

This section explains how the goal, question and metrics are identified for this research as per the GQM approach. The guidelines of Jedlitschka et al. [141] were taken into consideration.

6.3.1.1 Goal

The experiment goal was defined with the help of a pre-defined format that Jedlitschka et al. [141] did provide and according to the aim of the research. In view of that, the experiment's goal can be defined as follows:

To analyse the aspects of usability when using the proposed methodology for the purpose of the software system architectures development process.

The usability of the proposed methodology is evaluated according to the following metrics: Complexity [9], [144], [145], Traceability [145], [146], Understandability [29], and Clarity [144], [7], [147]. Also, the development time will be evaluated implicitly at the end of the development process by using Equation 6-1 and Equation 6-2.

6.3.1.2 Question

A number of questions have been formulated according to the GQM approach [148]. The questions of the experiment are divided into two parts:

1. **Pre-experiment questions:** this group includes seven questions that are meant to generate a participant's profile such as developing skill and study background, yet no personal information is included therein.
2. **Post-experiment questions:** this group includes eleven questions focusing on the description and presentation of the artefacts of a specific reference architecture**.** These questions can be further divided into three subgroups as follows:
   A. The first group includes introductory questions, Q1 and Q2 that focus on the presentation. These questions were given to the participants in order to assess the clarity of the presentation.

B. The second group includes questions Q3-Q10. These questions are both related to the goal of the experiment and the target to measure the usability of the proposed methodology which implicitly measures Complexity, Traceability, Understandability, and Clarity.

C. The third group includes a single question (Q11). This is an open question offering an opportunity for participants to add their own comments in case they need to do so.

Appendix D.3 and D.4 offer a complete list of both pre-experiment and post-experiment questions.

6.3.1.3 Metric

The metrics identified for this user-study experiment are directly measured from participants' feedback based on specific questions. Furthermore, time is measured throughout the experiment, such as how much time the two groups needed to complete the tasks. Table 6-2 and Figure 6-2 both illustrate the relationships between the Goal/Question/Metric for the user-study experiment. The following is a summary for the metrics identified for the user-study experiment:

- **M1.** This metric has been defined for evaluating the **Complexity** of the development process, finding the components and the relationships between the elements [9], [144], [145]. The purpose of **M1** is to measure whether or not the ontological model reduces the complexity. Questions 3, 4, & 5 are identified for measuring this very metric.
- **M2.** This metric is defined to evaluate the **Traceability** between the components of the reference architecture [145], [146]. The purpose of **M2** is to measure whether or not the

ontological model helps participants to trace the components. Question 6 is identified for measuring of this very metric.

- **M3.** This metric has been defined for evaluating the **Understandability** of the description of the components and relationships between them [29]. The purpose of **M3** is to measure whether or not the ontological model helps the developers to understand the description of the artefacts of a reference architecture. Questions 7 & 8 are identified for the measuring of this very metric.
- **M4.** This metric has been defined for evaluating the **Clarity** of the structure and organisation of the components and relationships between the components [144], [146], [147]. The purpose of **M4** is to measure whether or not the ontological model presents the artefacts of a reference architecture in an organised and well-structured way. Questions 9 & 10 are identified for measuring this metric.
- **M5.** This metric has been defined for evaluating the development **Time** which includes:
  - **M51.** Time-Saving for Task Accomplishment. This metric calculates the percentage of time saved through using the ontological model to accomplish a task as compared to ad hoc manner. Equation 6-1 is used here and this measurement is calculated per group for each task.

$$\frac{Ad\ hoc\ manner\ Time - Ontological\ Model\ Time}{Ad\ hoc\ manner\ Time} * 100$$

Equation 6-1: Percentage of Ontological Model Time Saving for Each Task

  - **M52.** Total Time Saved for Tasks Accomplishment: This metric calculates the percentage of total time saved via using the ontological model to accomplish all tasks as compared with ad hoc manner. Equation 6-2 is employed here and this measurement is calculated per group for the set of tasks.

$$\frac{Ad\ hoc\ manner\ Total\ Time - \ Ontological\ Model\ Total\ Time}{Ad\ hoc\ manner\ Total\ Time} * 100$$

Equation 6-2: Percentage of Ontological Model Time Saving for All Tasks

Table 6-2 and Figure 6-2 show the GQM mapping for user study experiment.

Table 6-2: GQM Mapping for User Study Experiment

Goal	**Questions**	**Metrics**	
------	Q1-Q2	Introductory Questions (IQ)	
Usability	Q3-Q5	M1 Complexity	
	Q6	M2 Traceability	
	Q7-Q8	M3 Understandability	
	Q9-Q10	M4 Clarity	
	------	M51 Time-Saving for Task Accomplishment	M5 Time
		M52 Total Time Saved for Tasks Accomplishment	

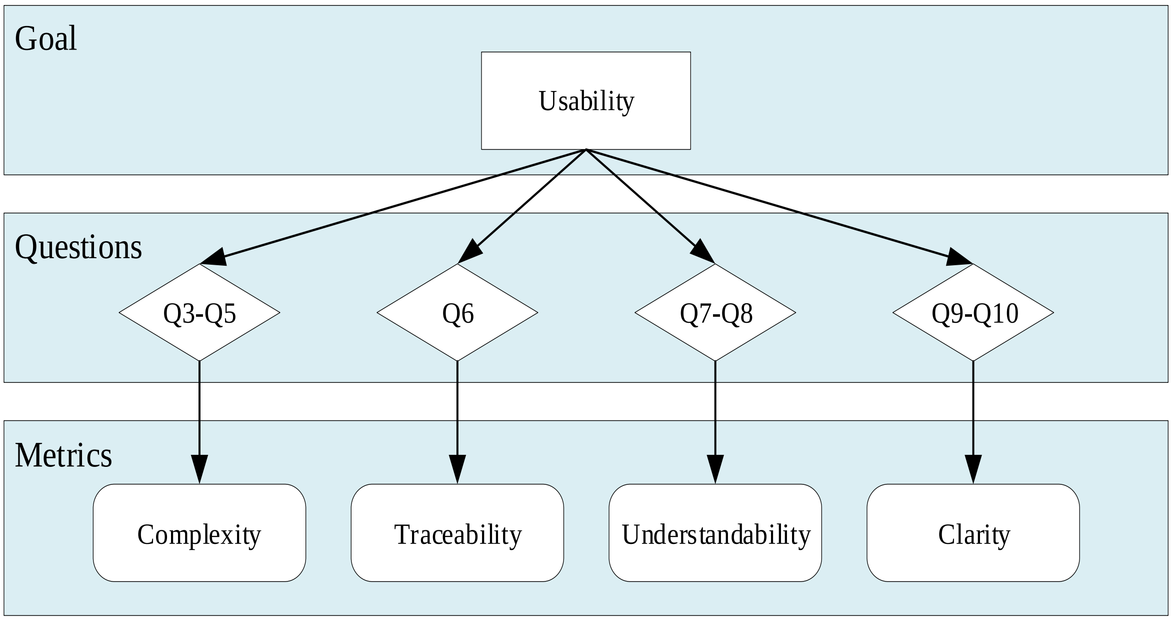


Figure 6-2: GQM Mapping for the User Study

6.3.2 Metrics Benchmarking and Question Rating Scales

Based on the literature review with the purpose of devising questions rating, user-study experience can be utilised with many scaling rates such as (3, 4, 5 and 7). Adopting an even or an odd number of values is another issue of debate and controversy by many researchers. The following guidelines proposed by Tullis & Albert [146] are of considerable attraction here regarding rating scale:

1. Far more reliable data from the user can be ensured through using multiple scales.
2. In order to enable the user to be neutral, use an odd number of values. This is considered a natural behaviour in real-world situations.
3. As for total number of points: there are some researchers who approve of always using more points. Somehow, and according to [149], using more than nine points will seldom provide any additional information that are of any use. In addition to that, five and seven points are the highest number of scaling values used in real-world user experience questionnaires. Also, Finstad [150] established an interesting study which compared five and seven versions of the same set of rating scales. According to that study and in contrast with seven-points five-point scales are more likely.

Having considered the above-mentioned guidelines, the rating for the questions feedback had been designed according to the nature of the questions and the purpose behind them. In this study, the Likert scale has been adopted [151]. Five-scale Likert feedback has been used for questions to measure the satisfaction of the participants. The Likert scale ranged from (Strongly Agree) to (Strongly Disagree) and from (Very Easy) to (Very Difficult). Samples 5-scale statements and the value assigned to each scale are shown in Table 6-3 .

Table 6-3: Samples 5-scale Statements with Assigned Value

Question/Statement	Likert Scale	Value
How do you find the development of software system architecture?	☐ Very Easy	5
	☐ Easy	4
	☐ Neutral	3
	☐ Difficult	2
	☐ Very Difficult	1
The relationships of a reference architecture are presented clearly.	☐ Strongly Agree	5
	☐ Agree	4
	☐ Neutral	3
	☐ Disagree	2
	☐ Strongly Disagree	1

All metrics have been measured based on the feedback from the participants after understanding the presentations and executing the required tasks . This does not include M51 (Time-Saving for Task Accomplishment) and M52 (Total Time Saved for Tasks Accomplishment) which are measured implicitly based on the task time of the participant.

Table 6-4 shows the available metrics and their measurement methods, the metrics required and the results range and the minimum required results. As for the measurement methods, two methods to measure a metric are available indeed. The first method is called 5-Scale User Feedback (5SUF) and includes calculation of the mean value of the participants' feedback using 5-scale rating feedback. The second method is called Participant's Tasks Development Results (PTDR) and is used to measure the participant's tasks development statistically using both Equation 6-1 and Equation 6-2.

Table 6-4: Metrics Measurement Methods and Possible Values

Goal	Metrics		Measurement Method	Range	Baseline Value
Usability	M1 Complexity		5SUF	1-5	3
	M2 Traceability		5SUF	1-5	3
	M3 Understandability		5SUF	1-5	3
	M4 Clarity		5SUF	1-5	3
	M5 Time	M51 Time-Saving for Task Accomplishment	PTDR	0% - 100%	50%
		M52 Total Time Saved for Tasks Accomplishment	PTDR	0% - 100%	50%

6.3.3 Tasks of the Experiment

Studies related to web browser have been implemented previously in [47] and [152] who both designed a software architecture for Mozilla, Konqueror, Epiphany, Lynx, and Safari web browser systems. Three tasks have been selected for that purpose and have been used in our experiment user study. The following description refers to tasks (T1, T2, and T3) that had been offered to the participants:

T1. Identify and name the subsystem of the web browser system.

T2. For each subsystem in the web browser system, identify and name the relationships between them.

T3. For each subsystem in the web browser system, identify the components for each subsystem; then identify the features of each component.

The participants were given different times for each task as shown in Table 6-5. The participants in each group used these tasks to measure the metrics by answering the questions.

Table 6-5: Given Time for Each Task

Task	Time
T1	15 minutes
T2	20 minutes
T3	30 minutes

6.3.4 Participants of the Experiment

The experimental work of the study employed a user study which included inviting twelve participants who were already on the course of a PhD program at the University of Salford. All of the participants in this study were volunteers and no compensation in any form was offered or paid. The respondents were also acknowledged that they had the option of leaving any statement blank in case they did not wish to answer it (See Appendix D.2). The participants were divided into two syndicates (Group A and Group B), each group included six participants. The number of participants being twelve can be regarded rational considering the fact that earlier user studies implemented by [153]–[157] opted to use 4, 6, 10, 12, and 26 participants, respectively. The answers provided by the participants for the questions that initiated the experiment led to the construction of the participants' comprehensive profile:

- **Study Background of the Participants**

  The academic background of the invited participants varied as to the following fields: Computer Science, Computer Engineering, Software Engineering and

Information System. Four participants have Computer Science background, five participants have Software Engineering background, one participant has Information System background, and two participants have Computer Engineering background, as illustrated in Figure 6-3. Some also had an experience in software development process because they worked in a private company.

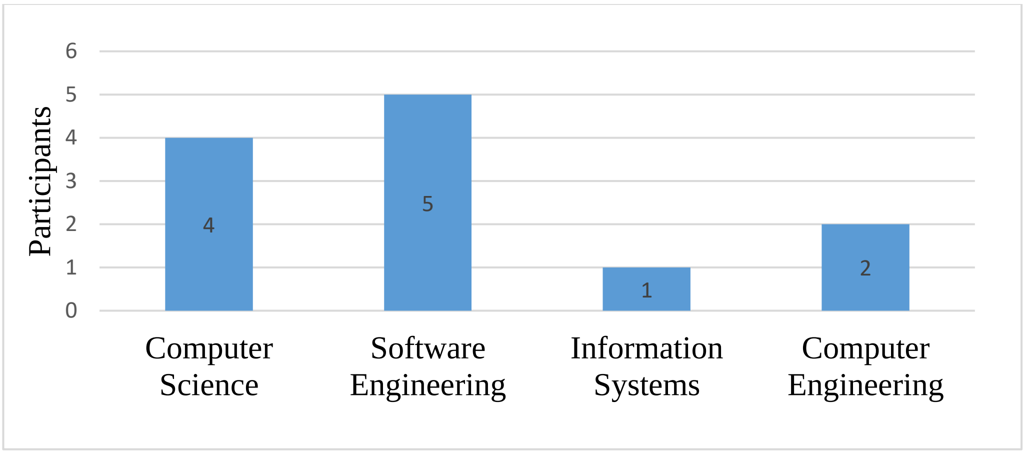


Figure 6-3: Participants' Study Background

- **Knowledge of Architectural Development**

As shown in Table 6-6, using participants with variant levels of knowledge in development background led to a suitable way of assessing the proposed methodology since the employed participants had various knowledge levels of software development. The percentages in Table 6-6 have been calculated based on the pre-experiment questions which are answered by the participants.

Table 6-6: Participants' Knowledge of Architectural Development

Scale	**Knowledge of Architectural Development**
No Information	0%
Beginner	65%
Intermediate	25%
Expert	10%

- **Software Development Experience**

  A versatile level of experience in software development was available within all participants.

6.3.5 Materials, Tools and Equipment of the Experiment

Participants had been divided into two groups: group A and group B. Each group included six participants. All participants were provided with a short presentation about the tasks. The presentation included explaining the tasks. Group A received the descriptions of tasks with a reference architecture of the web browser. On the other hand, group B received the ArchiOntology model for the web browser and they attended a presentation about it, in addition to the same tasks with a reference architecture of the web browser. Also, they received a PC, which includes Protégé software. The Protégé software is an ontology development tool (see Section 2.1.4). Figure 6-4 shows the two groups with materials, tools and equipment. In addition, the two groups received questions about the design process for which the participants had to answer. The experiment questions were constructed from the literature.

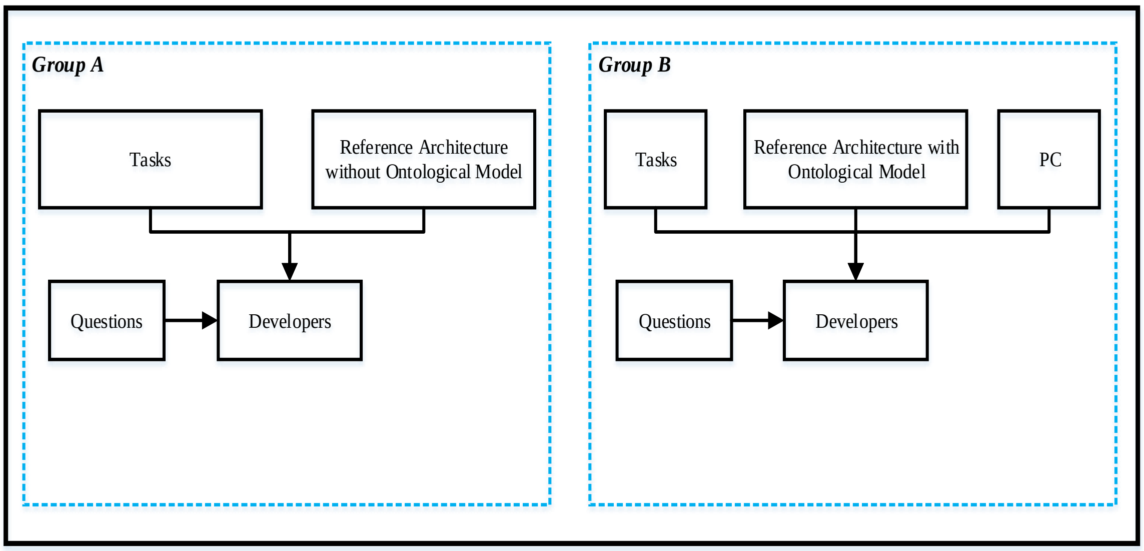

Group A
Tasks
Reference Architecture without Ontological Model
Questions
Developers
Group B
Tasks
Reference Architecture with Ontological Model
PC
Questions
Developers

Figure 6-4: Group A and B with Materials, Tools, Equipment, and Questions

6.3.6 Protocol of the Experiment

Table 6-7 explains the general experiment agenda. It took a total time of approximately 02:00 hours for the user study controlled experiment to be implemented. The duration was separated into three sessions, the first of which was about a welcoming speech for identifying the purpose behind this user study as well as filling in the pre-experiment survey which took 10 minutes. A presentation part of 20 minutes followed for elaboration on reference architecture and software architecture. The first session was ended with 5 minutes of discussion period which was about explaining any rising issues that needed clarifications for the participants.

Table 6-7: Time Table of Experiment

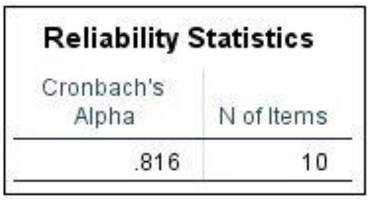
Session	Sub-session Activities	Time	Total Session Time
1.	Welcoming and participants complete a pre-experiment survey.	10 min	30 min
	Presentation about the reference architecture and software architecture for both groups.	20 min	
2.	Explain the required tasks to the participants and provide the needed materials.	15 min	80 min
	Tasks Execution (T1, T2 and T3).	65 min	
3.	Participants fill the post-experiment questions	10 min	10 min

As for the next session, the required tasks to be done by the participants consumed the first 15 minutes, participants executing the assigned tasks consumed the next (65 min) with time duration pertaining to each individual task being recorded too. The participants implemented those tasks in sequence (T1, T2, and T3). In conclusion, the participants were instructed to fill in the final post-experiment questions in the third session which took (10 min). The participants in Group B received an extra session (20 min) to give them an opportunity to familiarise themselves with the ontological model.

6.3.7 Results and Discussion of the Experiment

Microsoft Excel and SPSS software were used in this section for processing the results of the tasks as well as the feedback of recipients too. This section discusses the results begotten through the user study experiment by observing the time durations required by participants in order to for accomplish the tasks allotted for them and also by evaluating the participants' feedback and observing the time needed to accomplish the set of tasks given to them.

To check the reliability of the results as well as validating the results' integrity, Cronbach's alpha index [158] was used for examining their internal consistency. This measurement is already widely-used to analyse and verify the reliability of Likert-type question results. The preferred alpha index value is > 0.7 [124], [159]. Figure 6-5 shows that the results of this very research had calculated Cronbach's alpha index as 0.816 which can be considered acceptable and reflect highly inter-correlated results.


Reliability Statistics

Cronbach's Alpha	N of Items
.816	10

Figure 6-5: Reliability Statistics

Both Table 6-8 and Table 6-9 illustrate the feedback of the participants as per the clarity of the provided and presented materials as well as the task descriptions too. Most participants (83%) found the tutorials and presentations easy to understand and all participants (100%) found the tasks description clear and informative. A mean value for the introductory questions was 4.58.

Table 6-8: First Introductory Statement Results

Statements	Likert Scale				
	Very Easy	Easy	Neutral	Difficult	Very Difficult
The given tutorials and presentation were easy to understand.	58%	25%	17%	0%	0%
Average	58%	25%	17%	0%	0%

Table 6-9: Second Introductory Statement Results

Statements	Likert Scale				
	Strongly Agree	Agree	Neutral	Disagree	Strongly disagree
The descriptions of the tasks were clear.	75%	25%	0%	0 %	0%
Average	75%	25%	0%	0%	0%

6.3.7.1 Results of the Metrics

The results pertaining to the metrics already identified are discussed here.

- **M1 (Complexity),** three statements were used to measure this metric. Results in Table 6-10 show that 39% and 50% of the participants in Group A found that the development process pertaining to an architecture of a software system, and the finding process of the components and the relationships between them were difficult and very difficult, respectively. On the other hand, Table 6-11 shows 22% and 78% of the participants who used the ontological model and found the development process pertaining to a software system architecture, and the finding process of the components and the relationships between them easy and very easy, respectively, with a mean value = 4.77.

Table 6-10: A Response from Group A to the Statements 3, 4, and 5

Statements	**Likert Scale**				
	Very Easy	**Easy**	**Neutral**	**Difficult**	**Very Difficult**
How do you find the development of software system architecture?	0%	0%	0%	50%	50%
How do you evaluate the difficulty in finding the components of the reference architecture?	0%	0%	33%	50%	17%
How do you evaluate the difficulty in finding the relationships between the components of the reference architecture?	0%	0%	0%	17%	83%
Average	0%	0%	11%	39%	50%

Table 6-11: A Response from Group B to the Statements 3, 4 and 5

Statements	**Likert Scale**				
	Very Easy	**Easy**	**Neutral**	**Difficult**	**Very Difficult**
How do you find the development of software system architecture?	50%	50%	0%	0%	0%
How do you evaluate the difficulty in finding the components of the reference architecture?	100%	0%	0%	0%	0%
How do you evaluate the difficulty in finding the relationships between the components of the reference architecture?	83%	17%	0%	0%	0%
Average	78%	22%	0%	0%	0%

- **M2** (Traceability), one statement was used to measure this metric. Table 6-12 shows that 67% and 33% of the participants in Group A are (strongly disagreed) and (disagreed), respectively. On the other hand, Table 6-13 displays that 83% of the participants in Group B are (strongly agreed) with statement 6, and 17% of the participants have agreed that the traceability between the components was easy, with the mean value = 4.83.

Table 6-12: A Response from Group A to the Statement 6

Statements	**Likert Scale**				
	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
The components of a reference architecture are easily traceable by developers.	0%	0%	0%	33%	67%
Average	0%	0%	0%	33%	67%

Table 6-13: A Response from Group B to the Statement 6

Statements	**Likert Scale**				
	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
The components of a reference architecture are easily traceable by developers.	83%	17%	0%	0%	0%
Average	83%	17%	0%	0%	0%

- **M3** (Understandability). Two statements were used to measure this metric. Table 6-14 explains that 33% of the participants in Group A said the description of the components and relationships between them was very difficult to understand and 58% of the participants said the description was difficult to understand. However, 41.5%

and 58.5% of the participants in Group B said that the description of the components and the relationships between them was very easy and easy to understand, respectively, as shown in Table 6-15, with a mean value = 4.42.

Table 6-14: A Response from Group A to the Statements 7 and 8

Statements	Likert Scale				
	Very Easy	Easy	Neutral	Difficult	Very Difficult
The description of components is easy to understand by developers.	0%	0%	17%	50%	33%
The description of relationships between the components is easy to understand by developers.	0%	0%	0%	67%	33%
Average	0%	0%	8.5%	58.5%	33%

Table 6-15: A Response from Group B to the Statements 7 and 8

Statements	Likert Scale				
	Very Easy	Easy	Neutral	Difficult	Very Difficult
The description of components is easy to understand by developers.	33%	67%	0%	0%	0%
The description of relationships between the components is easy to understand by developers.	50%	50%	0%	0%	0%
Average	41.5%	58.5	0%	0%	0%

- **M4** (Clarity), statements 9 and 10 are used to measure this metric. All participants (100%) in Group A strongly disagreed about these two statements as shown in Table 6-16. On the contrary, all participants (100%) in Group B strongly agreed that the components of the reference architecture with their relationships are represented in an organised and structured way as illustrated in Table 6-17, with a mean value = 4.91.

Table 6-16: A Response from Group A to the Statements 9 and 10

Statements	Likert Scale				
	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The components of a reference architecture presented in an organised and structural way.	0%	0%	0%	0%	100%
The relationships of a reference architecture are presented clearly.	0%	0%	0%	0%	100%
Average	0%	0%	0%	0%	100%

Table 6-17: A Response from Group B to the Statements 9 and 10

Statements	Likert Scale				
	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The components of a reference architecture presented in an organised and structural way.	100%	0%	0%	0%	0%
The relationships of a reference architecture are presented clearly.	83%	17%	0%	0%	0%
Average	91.5%	8.5%	0%	0%	0%

- **M5** (Time).
  - **M51** (Time Saving for Task Accomplishment). The detailed time needed for every task using an ad hoc manner and ontological model too are displayed in Table 6-18.

The average times required to complete T1, T2 and T3 using the ad hoc manner were 00:13:00, 00:17:00 and 00:25:00 minutes, respectively, whereas the participants in Group B implemented these tasks using the ontological model in 00:05:0, 00:08:00 and 00:11:00 minutes, respectively. Using Equation 6-1 for calculating the Time-Saving for Task Accomplishment shows that the average percentages of time saved for T1, T2 and T3 using the ontological model were 62%, 53% and 56%.

- **M52** (Total Time Saving for Tasks Accomplishment) can be calculated using Equation 6-2. Ontological model saved 57% of the time needed to complete the set of tasks as compared with ad hoc manner. This illustrates the potential power of the ontological model in matters of saving development effort and time.

Table 6-18: Ontological Model and Ad hoc Manner Task Completion Times

Group A			**Group B**			**Time-Saving for Task Accomplishm ent**			**Total Time Saving for Tasks Accomplishment**
Ad hoc Manner			**Ontological Model**						
T1	**T2**	**T3**	**T1**	**T2**	**T3**	**T1**	**T2**	**T3**	
00:13:00	00:17:00	00:25:00	00:05:00	00:08:00	00:11:00	62%	53%	56%	57%

In view of the main metrics for the user study experiment and their relevant baseline values that are laid out in Table 6-4, Table 6-19 illustrates the final results of the experiment which shows higher values for all metrics compared to baseline values. It is evident from such results that the ontological model enables the instantiation process of a software system architecture from a reference architecture.

Table 6-19: User Study Experiment Metrics Results

Goal	Metrics		Measurement Method	Range	Baseline	Value
Usability	M1 Complexity		5SUF	1-5	3	**4.76**
	M2 Traceability		5SUF	1-5	3	**4.83**
	M3 Understandability		5SUF	1-5	3	**4.42**
	M4 Clarity		5SUF	1-5	3	**4.91**
	M5 Time	M51 Time-Saving for Task Accomplishment	PTDR	0% - 100%	50%	**62%** **53%** **56%**
		M52 Total Time Saved for Tasks Accomplishment	PTDR	0% - 100%	50%	**57%**

6.3.7.2 Final Experiment Results

As for usability and in terms of M1, M2, M3, and M4, the final conclusion is that all participants in group B found that the ontological model was usable and that it facilitated the instantiation process of a software system architecture from a reference architecture by tracking the relationships between the components and also reduced the development time. Furthermore, the ontological model saves 57% of the time that is required to implement a set of tasks in contrast with an ad hoc manner.

6.4 Summary

The ontological model could facilitate the instantiation process pertaining to a software system architecture from a reference architecture. This has been demonstrated in this chapter. Next to that, the chapter discussed the method followed to assess and evaluate the proposed methodology. The study adopted this method in pursuit of an answer for the question that was proposed at the beginning of this chapter and which was:

Can the ontological model facilitate the instantiation process of a software system architecture and minimise the development time?

Regarding the above-mentioned question as to whether (or not) the ontological model can facilitate the instantiation process, the answer has been made answered through conducting a controlled experiment that was conducted through utilising twelve participants to assess the proposed methodology. The results of the experiment indicated that the proposed methodology facilitated the development process of the software system architecture from a reference architecture.

The conclusion, contributions of this research and future work are summarised and presented in the next chapter.

CHAPTER SEVEN: CONCLUSION AND FUTURE WORK

7.1 Overview

This final chapter closes the thesis by presenting a summary of the work and describing the main contributions. It also illustrates the possible areas for future work. This thesis contributed in this sense of facilitating the instantiation process of a software system architecture from a reference architecture.

Achievements of this work include the definition and validation of general vocabularies which are used to describe the artefacts of reference architectures, creation of the general ontological model and the proposal of a process for presenting the artefacts of the reference architecture (constructing ArchiOntology).

In this chapter, the objectives of the thesis will be revised and the means of realising them will be illustrated in Section 7.2. Section 7.4 will present the ideas and suggestions for the future development.

7.2 Significant Contribution

This thesis provides a number of contributions that are described in the following.

1- Define general vocabularies to describe the general aspect of reference architectures as explained in Chapter 4.

2- One of the main contributions of this thesis is to construct a general ontological model which can be used as a foundation for presenting the artefacts of a reference architecture as a conceptual model concepts as described in chapter 4.

3- Another contribution is using an ontology as a tool to describe the knowledge about reference architecture formally.

7.3 Review of the Research Objectives

This section introduces the research objectives and also reviews the means of achieving them.

7.3.1 Objective 1

- **Review the development approaches of a reference architecture**.

In order to achieve this objective, the development approaches of reference architecture have been reviewed in Section 2.2.1 of Chapter 02 to show what type of tools have been used in these approaches so as to present and describe the artefacts of a reference architecture. The review has demonstrated that there are no standard vocabularies that are used in the development process, and all of these approaches used informal and semi-formal tools to present and describe the artefacts of reference architectures.

7.3.2 Objective 2

- **Review the existing instantiation process of a software system architecture from a reference architecture.**

This objective aims at highlighting the shortcoming of the instantiation process of a software system architecture from a reference architecture. This objective has been achieved in Section 2.5 of Chapter 2. The review showed that all the processes provide general guidelines for the instantiation process and there is no concrete tool that has been used in the instantiation process.

7.3.3 Objective 3

- **Review an ontology principle**.

This objective aims at reviewing an ontology definition, components, representation language and development tools. This objective has been achieved in Section 2.6 of Chapter 2. The review has demonstrated the characteristics of the ontology. These characteristics have been utilised in presenting the artefacts of a reference architecture.

7.3.4 Objective 4

- **Definition general vocabularies to be used for constructing an ontological model.**

To achieve this objective, we defined general vocabularies based on understanding the domain and the literature and also from multiple case studies. Initial general vocabularies are defined. Next to that, more general vocabularies were extracted and defined from six case studies for various domains from the literature. This is all explained in details in Chapter 4.

7.3.5 Objective 5

- **Development of an ontological model for presenting knowledge about the reference architecture.**

To achieve this objective, we developed a general ontological model to present the artefacts of reference architectures. The ontological model is constructed based on the defined general vocabulary. The ontological model aims at providing vocabularies to software architects and developers. This helps the software architects and developers to find the components of a reference architecture by tracking the relation between them. The development process of the ontological model is explained in details in Section 4.2.4 of Chapter 4.

7.3.6 Objective 6

- **Develop a process to describe the artefacts of a reference architecture.**

To achieve this objective, we proposed two processes. The first process shows how the ontological model will be coupled with a reference architecture to produce an ArchiOntology. The second process shows how the ArchiOntology will be used to provide vocabularies. The processes are explained in details in Chapter 5.

7.3.7 Objective 7

- **Evaluate the proposed methodology by conducting a user study experiment.**

To achieve this objective, we evaluated the proposed methodology in an experimental study which was illustrated in Chapter 6. The user study experiment compared the current ad hoc approach used to instantiate software architectures and the development using ontology. Results gave evidence that the ontology can facilitate the instantiation process of software system architectures from a reference architecture.

7.4 Future Work

Many opportunities of research emerged during the development of this thesis. They represent perspectives of future research that can contribute to the areas of a reference architecture. In the future, we plan to:

1. Develop a tool that aims at extracting the artefacts of reference architecture. The tool should help software engineers and architects to extract the concepts of architecture architectures automatically.
2. Develop visual tool support that aims at assisting the design and verification of architecture models based on the proposed ontological model. The tool should allow the developers to design their architecture models graphically.
3. Conduct the proposed methodology to design a reference architecture for a specific domain based on the proposed ontological model.
4. Conduct a real case study within an industrial context in order to provide additional evidence that increases the confidence towards the adoption of this methodology.

REFERENCES

[1] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005.

[2] P. Kruchten, H. Obbink, and J. Stafford, "The Past, Present, and Future for Software Architecture," *IEEE Softw.*, vol. 23, no. 2, pp. 22–30, Mar. 2006.

[3] M. Shaw and P. Clements, "The Golden Age of Software Architecture," *IEEE Softw.*, vol. 23, no. 2, pp. 31–39, Mar. 2006.

[4] M. Guessi, L. B. R. Oliveira, and E. Y. Nakagawa, "Representation of Reference Architectures: A Systematic Review," *23rd Int. Conf. Softw. Eng. Knowl. Eng.*, 2011.

[5] E. Y. Nakagawa, F. Oquendo, and M. Becker, "RAModel: A Reference Model for Reference Architectures," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European*, 2012, pp. 297–301.

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Second Edi. USA: Addison-Wesley, 2003.

[7] a. J. a. Jansen and J. B. J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," *5th Work. IEEE/IFIP Conf. Softw. Archit.*, 2005.

[8] E. Y. Nakagawa, P. Oliveira Antonino, and M. Becker, "Reference Architecture and Product Line Architecture: A Subtle But Critical Difference," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6903 LNCS, I. Crnkovic, V. Gruhn, and M. Book, Eds. Essen, German: Springer Berlin Heidelberg, 2011, pp. 207–211.

[9] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, "The Concept of Reference Architectures," *Syst. Eng.*, vol. 14, no. 3, 2009.

[10] M. Galster and P. Avgeriou, "Empirically-grounded Reference Architectures: A Proposal," in *Proceedings of the joint ACM SIGSOFT conference*, 2011, p. 153.

[11] S. Martínez-Fernández, C. P. Ayala, X. Franch, H. Marques, and D. Ameller, “Towards Guidelines for Building a Business Case and Gathering Evidence of Software Reference Architectures in Industry,” *J. Softw. Eng. Res. Dev.*, vol. 2, no. 1, p. 7, 2014.

[12] S. Angelov, P. Grefen, and D. Greefhorst, “A Framework for Analysis and Design of Software Reference Architectures,” *Inf. Softw. Technol.*, vol. 54, no. 4, pp. 417–431, Apr. 2012.

[13] S. Martínez-Fernández, C. Ayala, X. Franch, and H. M. Marques, “Artifacts of Software Reference Architectures: A Case Study,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE ’14*, 2014, pp. 1–10.

[14] E. Y. Nakagawa, F. Oquendo, and J. C. Maldonado, “Reference Architectures,” in *Software Architecture 1*, M. Oussalah, Ed. London: Wiley, 2014, pp. 55–82.

[15] P. Avgeriou, “Describing, Instantiating and Evaluating a Reference Architecture: A Case Study,” *Can. Med. Assoc. J.*, 2003.

[16] G. Muller and P. van de Laar, “Researching Reference Architectures and Their Relationship with Frameworks, Methods, Techniques, and Tools,” *INSIGHT*, vol. 13, no. 2, pp. 24–30, 2010.

[17] T. R. Gruber, “A Translation Approach to Portable Ontology Specifications,” *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[18] E. Y. Nakagawa, M. Gonçalves, M. Guessi, L. B. R. Oliveira, and F. Oquendo, “The State of the Art and Future Perspectives in Systems of Systems Software Architectures,” in *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems - SESoS ’13*, 2013, pp. 13–20.

[19] J. Gonzalez-Huerta, E. Insfran, and S. Abrah, “Model-Driven Engineering and Software Development,” vol. 580, pp. 12–31, 2015.

[20] L. Liao, “From Requirements to Architecture: The State of the Art in Software Architecture Design,” *Citeseer*, pp. 1–13, 2002.

[21] D. Perovich, M. C. Bastarrica, and C. Rojas, “Model-Driven Approach to Software Architecture Design,” in *2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, 2009, pp. 1–8.

[22] T. Srikanth, D. R. Kumar, and M. N. Kumar, “Model Driven Design Method for Software Architecture,” vol. 2, no. 6, pp. 2816–2821, 2011.

[23] J. Sun, H. H. Wang, and T. Hu, “Design Software Architecture Models using Ontology,” *SEKE 2011 - Proc. 23rd Int. Conf. Softw. Eng. Knowl. Eng.*, pp. 191–196, 2011.

[24] L. B. R. Oliveira, E. Leroux, K. R. Felizardo, F. Oquendo, and E. Y. Nakagawa, “Towards a Process to Design Architectures of Service-Oriented Robotic Systems,” in *Software Architecture, Ecsa 2014*, vol. 8627, P. Avgeriou and Z. Uwe, Eds. Vienna, Austria: Springer, 2014, pp. 218–225.

[25] L. B. R. Oliveira, E. Leroux, K. R. Felizardo, F. Oquendo, and E. Y. Nakagawa, “ArchSORS: A Software Process for Designing Software Architectures of Service-Oriented Robotic Systems,” *Comput. J.*, vol. 60, no. 9, pp. 1363–1381, Sep. 2017.

[26] F. Pérez-Sorrosal, R. Jiménez-Péris, and M. Patiño-Martínez, “Methodology to Write Instantiation Guidelines for the NEXOF Reference Architecture,” 2010.

[27] L. R. D. B. Oliveira, “Architectural Design of Service-oriented Robotic Systems,” Université de Bretagne Sud, 2015.

[28] E. Y. Nakagawa, E. F. Barbosa, and J. C. Maldonado, “Exploring ontologies to support the establishment of reference architectures: An example on software testing,” in *2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA 2009*, 2009, pp. 249–252.

[29] M. Guessi, S. Carlos, L. B. R. Oliveira, S. Carlos, and S. Carlos, “Towards a Formal Description of Reference Architectures for Embedded Systems,” in *Exploring Component-based Techniques for Constructing Reference Architectures (CobRA), 2015 1st International Workshop on*, 2015, pp. 1–4.

[30] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Second. USA: Addison-Wesley, 2003.

[31] R. Kazman, M. Klein, and P. Clements, "ATAM : Method for Architecture Evaluation," 2000.

[32] C. Hofmeister, N. Robert, and S. Dilip, *Applied Software Architecture*. Addison-Wesley, 2000.

[33] D. GARLAN and M. SHAW, "An Introduction to Software Architecture," in *In Advances in software engineering and knowledge engineering*, vol. 1, no. January, WORLD SCIENTIFIC, 1993, pp. 1–39.

[34] D. Garlan, "Software Architecture: a Roadmap," in *Proceedings of the conference on The future of Software engineering - ICSE '00*, 2000, pp. 91–101.

[35] M. H. Valipour, B. Amirzafari, K. N. Maleki, and N. Daneshpour, "A Brief Survey of Software Architecture Concepts and Service Oriented Architecture," *2009 2nd IEEE Int. Conf. Comput. Sci. Inf. Technol.*, pp. 34–38, 2009.

[36] S. Committee, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," vol. 1471–2000, no. 42010, p. i-23, 2000.

[37] S. Angelov, J. J. M. Trienekens, and P. Grefen, "Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case," in *Software Architecture*, vol. 5292 LNCS, R. Morrison, D. Balasubramaniam, and K. Falkner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 225–240.

[38] G. Muller, "A Reference Architecture Primer," *Eindhoven Univ. Techn., Eindhoven, White Pap.*, pp. 0–20, 2012.

[39] S. Angelov, P. Grefen, and D. Greefhorst, "A Classification of Software Reference Architectures: Analyzing their Success and Effectiveness," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 2009, pp. 141–150.

[40] E. Y. Nakagawa, P. Oliveira Antonino, and M. Becker, "Reference Architecture and Product Line Architecture: A Subtle But Critical Difference," in *Software*

Architecture: 5th European Conference, ECSA 2011., I. Crnkovic, V. Gruhn, and M. Book, Eds. Springer, 2011, pp. 207–211.

[41] D. Weyns and T. Holvoet, "A Reference Architecture for Situated Multi-Agent Systems," in *Environments for Multi-Agent Systems III,* vol. 4389, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–40.

[42] N. F. D. Filho and E. F. Barbosa, "A Contribution to the Establishment of Reference Architectures for Mobile Learning Environments," *IEEE Rev. Iberoam. Tecnol. del Aprendiz.*, vol. 10, no. 4, pp. 234–241, Nov. 2015.

[43] W. Bumpus, "NIST Cloud Computing Standards Roadmap," Gaithersburg, MD, Jul. 2013.

[44] A. E. Hassan and R. C. Holt, "A Reference Architecture for Web Servers," in *Proceedings Seventh Working Conference on Reverse Engineering*, 2000, pp. 150–159.

[45] V. Casola, A. Gaglione, and A. Mazzeo, "A Reference Architecture for Sensor Networks Integration and Management," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* vol. 5659 LNCS, 2009, pp. 158–168.

[46] C. A. Machado, E. Silva, T. Batista, J. Leite, and E. Nakagawa, "RA-Ubi: A Reference Architecture for Ubiquitous Computing," pp. 98–105, 2014.

[47] A. Grosskurth and M. W. Godfrey, "A Reference Architecture for Web Browsers," in *21st IEEE International Conference on Software Maintenance (ICSM'05),* 2005, pp. 661–664.

[48] B. Álvarez, A. Iborra, A. Alonso, and J. A. de la Puente, "Reference Architecture for Robot Teleoperation," *Control Eng. Pract.*, vol. 9, no. 4, pp. 395–402, Apr. 2001.

[49] J.-M. DeBaud, O. Flege, and P. Knauber, "PuLSE-DSSA — A Method for the Development of Software Reference Architectures," *Work. Softw. Archit.*, pp. 25–28, 1998.

[50] P. Kruchten, *The Rational Unified Process: An Introduction,* Second. Addison-

Wesley, 2000.

[51] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 2nd ed. Addison-Wesley, 2004.

[52] Y. Zhou, Y. Chen, and H. Lu, "UML-based Systems Integration Modeling Technique for the Design and Development of Intelligent Transportation Management System," *2004 IEEE Int. Conf. Syst. Man Cybern. (IEEE Cat. No.04CH37583)*, vol. 7, pp. 6061–6066, 2004.

[53] L. Dobrica and E. Niemelä, "An Approach to Reference Architecture Design for Different Domains of Embedded Systems," *Proc. 2008 Int. Conf. Softw. Eng. Res. Pract. SERP 2008*, pp. 287–293, 2008.

[54] E. Y. Nakagawa, M. Guessi, J. C. Maldonado, D. Feitosa, and F. Oquendo, "Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures," *Proc. - Work. IEEE/IFIP Conf. Softw. Archit. 2014, WICSA 2014*, pp. 143–152, 2014.

[55] N. Arch-int, C. Lursinsup, and P. Sophatsathit, "A Reference Architecture for Interoperating Existing e-Learning Systems using Metadata and Web Services Model," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, 2005, vol. 2, pp. 891–896.

[56] E. Y. Nakagawa, F. C. Ferrari, M. M. F. Sasaki, and J. C. Maldonado, "An Aspect-oriented Reference Architecture for Software Engineering Environments," *J. Syst. Softw.*, vol. 84, no. 10, pp. 1670–1684, 2011.

[57] B. P. Gallagher, "Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study," 2000.

[58] ISO/IEC/ IEEE 42010, "Systems and Software Engineering — Architecture Description," 2011.

[59] S. Herold, M. Mair, A. Rausch, and I. Schindler, "Checking Conformance with

Reference Architectures: A Case Study," in *2013 17th IEEE International Enterprise Distributed Object Computing Conference*, 2013, pp. 71–80.

[60] S. . MartÃ-nez-FernÃ¡ndez, C. Ayala, X. . X. Franch, H. M. H. M. . Marques, D. D. . Ameller, and S. Martinez-Fernandez, "A framework for software reference architecture analysis and review," *CIbSE 2013 16th Ibero-American Conf. Softw. Eng. - Memorias del 10th Work. Latinoam. Ing. Softw. Exp. ESELAW 2013*, pp. 89–102, 2013.

[61] F. J. Affonso, K. R. F. Scannavino, L. B. R. Oliveira, and E. Y. Nakagawa, "Reference Architectures for Self-Managed Software Systems: A Systematic Literature Review," in *2014 Eighth Brazilian Symposium on Software Components, Architectures and Reuse*, 2014, pp. 21–31.

[62] M. Galster, "Software Reference Architectures," in *Proceedings of the 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures - CobRA '15*, 2015, pp. 5–8.

[63] S. Angelov, J. Trienekens, and R. Kusters, "Software Reference Architectures - Exploring Their Usage and Design in Practice," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, 7th Europe., vol. 7957 LNCS, no. July, pp. 17–24.

[64] E. Y. Nakagawa and L. B. R. Oliveira, "Using Systematic Review to Elicit Requirements of Reference Architectures.," 2011.

[65] M. Akşit, *Software Architectures and Component Technology*. Boston, MA: Springer US, 2002.

[66] D. Weyns, "An Architecture-Centric Approach for Software Engineering with Situated Multiagent Systems," Katholieke Universiteit Leuven, 2006.

[67] A. Suganthy and T. Chithralekha, "Domain-Specific Architecture for Software Agents," *The Journal of Object Technology*, vol. 7, no. 6. p. 77, 2008.

[68] D. Weyns and T. Holvoet, "Architecture-Centric Software Development of Situated Multiagent Systems," in *Engineering Societies in the Agents World VII*, vol. 4457

LNAI, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 62–85.

[69] F. Bachmann and L. Bass, "Introduction to the Attribute Driven Design Method," *Proc. 23rd Int. Conf.*, pp. 745–746, 2001.

[70] P. Sala, "Part III: The universAAL Reference Architecture for Ambient Assisted Living," 2013.

[71] G. Muller, "How Reference Architectures Support the Evolution of Product Families," *Econ. Aff.*, 2008.

[72] R. Neches *et al.*, "Enabling Technology for Knowledge Sharing," *Ai Mag.*, vol. 12, no. 3, pp. 36–57, 1991.

[73] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. London: Springer-Verlag, 2004.

[74] P. Spyns, R. Meersman, and M. Jarrar, "Data Modelling versus Ontology Engineering," *ACM SIGMOD Rec.*, vol. 31, no. 4, p. 12, Dec. 2002.

[75] A. A. Storey *et al.*, "Ontologies for Software Engineering and Software Technology," *Proc. Natl. Acad. Sci.*, vol. 104, no. 25, Jun. 2007.

[76] M. R. Genesereth and R. E. Fikes, "Knowledge Interchange Format," California, 1992.

[77] D. B. Lenat and R. V Guha, *Building Large Knowledge-based Systems*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[78] R. M. MacGregor, "Inside the LOOM Description Classifier," *ACM SIGART Bull.*, vol. 2, no. 3, pp. 88–92, Jun. 1991.

[79] O. Lassila and R. R. Swick, "Resource Description Framework(RDF) Model and Syntax Specification.," *Miscellaneous*, no. October, 1999.

[80] F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks, "Reference Description of the DAML+ OIL Ontology Markup Language," *Contrib. T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D.*

McGuinness, LA Stein, 2001.

[81] G. K. Saha, “Web Ontology Language: OWL,” *Ubiquity*, vol. 2007, no. September, pp. 1–1, Sep. 2007.

[82] F. López, “Overview Of Methodologies For Building Ontologies,” *Proc. IJCAI99 Work. Ontol. Probl. Methods Lessons Learn. Futur. Trends CEUR Publ.*, vol. 1999, no. 2, pp. 1–13, 1999.

[83] N. F. Noy and D. L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” *Artif. Intell. Med.*, Sep. 2001.

[84] M. Uschold and M. Gruninger, “Ontologies: Principles, Methods and Applications,” *Knowl. Eng. Rev.*, vol. 11, no. 02, p. 93, 1996.

[85] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, “The NeOn Methodology Framework: A Scenario-based Methodology for Ontology Development,” *Appl. Ontol.*, vol. 10, no. 2, pp. 107–145, 2015.

[86] D. E. Forbes, P. Wongthongtham, C. Terblanche, and U. Pakdeetrakulwong, “Ontology Engineering,” in *Ontology Engineering Applications in Healthcare and Workforce Management Systems*, vol. 123, Cham: Springer International Publishing, 2018, pp. 27–40.

[87] M. Gruninger, M. S. Fox, and others, “Methodology for the Design and Evaluation of Ontologies,” *Proc. Work. Basic Ontol. Issues Knowl. Sharing, IJCAI*, vol. 95, pp. 1–10, 1995.

[88] A. De Nicola, M. Missikoff, and R. Navigli, “A Proposal for a Unified Process for Ontology Building: UPON,” in *DEXA 2005: Proceedings of the 16th International Conference on Database and Expert Systems Applications*, vol. LNCS 3588, 2005, pp. 655–664.

[89] M. Ferndndez, A. Gomez-Perez, and N. Juristo, “METHONTOLOGY: From Ontological Art Towards Ontological Engineering Mariano,” in *Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS’07)*, 2007, vol. SS-97-06, pp. 115–122.

[90] N. O. Bajnaid, "An Ontological Approach to Model Software Quality Assurance Knowledge Domain," 2013.

[91] F. Baader, I. Horrocks, and U. Sattler, "Description Logics as Ontology Languages for the Semantic Web," pp. 228–248, 2005.

[92] A. Shehzad, H. Ngo, K. Pham, and S. Lee, "Formal Modeling in Context Aware Systems," *... Model. Retr. ...*, 2004.

[93] D. Tsarkov and I. Horrocks, "FaCT++ Description Logic Reasoner: System Description," pp. 292–297, 2006.

[94] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semant.*, vol. 5, no. 2, pp. 51–53, 2007.

[95] V. Haarslev and R. Möller, "RACER System Description," pp. 701–705, 2001.

[96] P. Kruchten, P. Lago, and H. van Vliet, "Building Up and Reasoning About Architectural Knowledge," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4214 LNCS, 2006, pp. 43–58.

[97] A. Akerman and J. Tyree, "Using Ontology to Support Development of Software Architectures," *IBM Syst. J.*, vol. 45, no. 4, pp. 813–825, 2006.

[98] L. Babu T., M. Seetha Ramaiah, T. V. Prabhakar, and D. Rambabu, "ArchVoc-Towards an Ontology for Software Architecture," in *Second Workshop on Sharing and Reusing Architectural Knowledge - Architecture, Rationale, and Design Intent (SHARK/ADI'07: ICSE Workshops 2007)*, 2007, pp. 5–5.

[99] C. Pahl, S. Giesecke, and W. Hasselbring, "Ontology-Based Modelling of Architectural Styles," *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1739–1749, 2009.

[100] Technical Standard, *Service-Oriented Architecture Ontology*. The Open Group, 2010.

[101] D. Ameller and X. Franch, "Ontology-Based Architectural Knowledge Representation: Structural Elements Module," in *Lecture Notes in Business Information Processing*, vol. 83 LNBIP, C. Salinesi and O. Pastor, Eds. Springer

Berlin Heidelberg, 2011, pp. 296–301.

[102] P. Kruchten, P. Lago, H. Van Vliet, and T. Wolf, "Building up and Exploiting Architectural Knowledge," *Proc. - 5th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2005*, vol. 2005, pp. 291–292, 2005.

[103] C. López, V. Codocedo, H. Astudillo, and L. M. Cysneiros, "Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach," *Sci. Comput. Program.*, vol. 77, no. 1, pp. 66–80, 2012.

[104] A. Figueiredo, J. Dos Reis, and M. Rodrigues, "Improving Access to Software Architecture Knowledge An Ontology-based Search Approach," *Int. J. Multimed. Image Process.*, vol. 2, no. 1/2, pp. 143–149, 2012.

[105] H. A. Duran-Limon, C. A. Garcia-Rios, F. E. Castillo-Barrera, and R. Capilla, "An Ontology-Based Product Architecture Derivation Approach," *IEEE Trans. Softw. Eng.*, vol. 41, no. 12, pp. 1153–1168, Dec. 2015.

[106] M. L. Roldán, S. Gonnet, and H. Leone, "An Ontology-based Approach for Sharing, Integrating, and Retrieving Architectural Knowledge," *CLEI 2017 - 43rd Lat. Am. Comput. Conf.*, 2017.

[107] N. Choobdaran, S. M. Sharfi, and M. R. Khayyambashi, "An Ontology-Based Approach For Software Architectural Knowledge Management," vol. 11, pp. 93–104, 2014.

[108] K. A. de Graaf, P. Liang, A. Tang, W. R. van Hage, and H. van Vliet, "An exploratory study on ontology engineering for software architecture documentation," *Comput. Ind.*, vol. 65, no. 7, pp. 1053–1064, Sep. 2014.

[109] C. R. Kothari, *Research Methodology. Methods and Techniques*. New Age International, 2004.

[110] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, 4th ed. Harlow: Pearson Education Limited, 2007.

[111] J. Collis and R. Hussey, *Business Research: A Practical Guide for Undergraduate*

and Postgraduate Students, 2nd ed. Houndmills, Basingstoke, Hampshire, Palgrave-Macmillan, 2003.

[112] J. Hussey and R. Hussey, *Business Research: A Practical Guide for Undergraduate and Postgraduate Students*. London: Macmillan Press Ltd, 1997.

[113] R. Cavana, B. L. Delahaye, and U. Sekaran, *Applied Business Research: Qualitative and Quantitative Methods*. John Wiley & Sons Australia, Milton, Queensland, 2001.

[114] S. Uma and B. Roger, *Research Methods for Business: A Skill Building Approach*, 4th ed. John Wiley & Sons, 2013.

[115] P. Ghauri and K. Gronhaug, *Research Methods in Business Studies*, Third. Prentice Hall, 2005.

[116] J. Collis and R. Hussey, *Business Research*, Third. Palgrave-Macmillan, 2009.

[117] B. J. Oates, *Researching Information Systems and Computing*. Sage, 2005.

[118] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, 6th ed. Pearson, 2012.

[119] R. Weber, "The Rhetoric of Positivism Versus Interpretivism: A Personal View," *MIS Q.*, vol. 28, no. 1, pp. iii–xii, 2004.

[120] R. K. Yin, *Case Study Research: Design and Methods*, Second. Thousand Oaks: Sage Publications, 1994.

[121] A. Bryman and E. Bell, *Business Research Methods*, 3rd ed. Oxford, 2011.

[122] D. Gray, *Doing Research in the Real World*, 3rd ed. Sage Publications, 2014.

[123] D. Silverman, *Qualitative Research*, 3rd ed. London: Sage Publications, 2010.

[124] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, Third. SAGA, 2013.

[125] J. W. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*, 3d ed. Thousand Oaks: Sage Publications, 2012.

[126] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*. Prentice Hall, 2009.

[127] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed. California: Sage Publications, 2003.

[128] M. Q. Patton, *Qualitative Evaluation and Research Methods*. SAGE Publications, inc, 1990.

[129] A. Fink, *How to Analyze Survey Data*. SAGE, 1995.

[130] B. A. Kitchenham and S. L. Pfleeger, "Personal Opinion Surveys," in *Guide to Advanced Empirical Software Engineering*, London: Springer London, 2008, pp. 63–92.

[131] M. Van Der Velde, P. Jansen, and N. Anderson, *Guide to Management Research Methods*. 2004.

[132] U. Sekaran and R. Bougie, *Research Methods for Business: A Skill Building Approach*, 4th ed. John Wiley & Sons, 2010.

[133] E. M. Trauth, *Qualitative Research in IS*. IGI Global, 2001.

[134] J. Kim and J. F. Courtney, "A Survey of Knowledge Acquisition Techniques and Their Relevance to Managerial Problem Domains," *Decis. Support Syst.*, vol. 4, no. 3, pp. 269–284, 1988.

[135] D. I. Moldovan, R. Girju, and V. Rus, "Domain-Specific Knowledge Acquisition from Text," *Proc. 6th Conf. Appl. Nat. Lang. Process.*, no. 1, pp. 268–275, 2000.

[136] V. C. Storey, V. Sugumaran, and Y. Ding, "A Semi-automatic Approach to Extracting Common Sense Knowledge from Knowledge Sources," *Nat. Lang. Process. Inf. Syst.*, pp. 322–332, 2005.

[137] S. P. Overmyer, L. Benoit, and R. Owen, "Conceptual Modeling through Linguistic Analysis Using LIDA," *Proc. 23rd Int. Conf. Softw. Eng. ICSE 2001*, pp. 401–410, 2001.

[138] B. Gelfand, M. Wulfekuler, and W. Punch, "Automated Concept Extraction from

Plain Text," *AAAI 1998 Work. Text …*, pp. 1–7, 1998.

[139] H. M. Harmain and R. Gaizauskas, "CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis," *Autom. Softw. Eng.*, vol. 10, no. 2, pp. 157–181, 2003.

[140] D. Tsarkov and I. Horrocks, "FaCT++ Description Logic Reasoner: System Description," pp. 292–297, 2006.

[141] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, "Reporting Experiments in Software Engineering," *Guid. to Adv. Empir. Softw. Eng.*, pp. 201–228, 2008.

[142] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm," *Quality*. p. 24, 1992.

[143] V. R. Basili, G. Caldiera, and H. D. Rombach, "The Goal, Metric, and Question Approach," in *In Encyclopedia of Software Engineering*, Wiley, 1994.

[144] U. Frank, "Evaluation of Reference Models," *Ref. Model. Bus. Syst. Anal.*, no. 6, pp. 118–140, 2007.

[145] ISO, "Systems and Software Engineering - Architecture Description," *ISO/IEC/IEEE 42010*, pp. 1–46, 2011.

[146] I. Groher and R. Weinreich, "Integrating Variability Management and Software Architecture," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, 2012, pp. 262–266.

[147] E. Y. Nakagawa and J. C. Maldonado, "Reference Architecture Knowledge Representation: An Experience," in *Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge - SHARK '08*, 2008, p. 51.

[148] H. Koziolek, "Goal, Question, Metric," in *Dependability Metrics*, vol. 4909 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 39–42.

[149] E. P. Cox, "The Optimal Number of Response Alternatives for a Scale: A Review," *J. Mark. Res.*, vol. 17, no. 4, p. 407, 1980.

[150] K. Finstad, "Response Interpolation and Scale Sensitivity: Evidence Against 5-Point

Scales," *J. Usability Stud.*, vol. 5, no. 3, pp. 104–110, 2010.

[151] R. Likert, "A Technique for the Measurement of Attitudes," The Science Press, New York, 1932.

[152] A. Grosskurth and M. Godfrey, "Architecture and Evolution of the Modern Web Browser," *Prepr. Submitt. to Elsevier Sci.*, no. June, pp. 1–24, 2006.

[153] K. Tei, R. Shimizu, Y. Fukazawa, and S. Honiden, "Model-Driven-Development-Based Stepwise Software Development Process," *Syst. Man, Cybern. Syst. IEEE Trans.*, vol. 45, no. 4, pp. 675–687, 2015.

[154] M. A. Chauhan, "Foundations for Tools as a Service Workspace: A Reference Architecture," IT University of Copenhagen, Cambridge, 2015.

[155] D. Dermeval, T. Tenório, I. I. Bittencourt, A. Silva, S. Isotani, and M. Ribeiro, "Ontology-based feature modeling: An empirical study in changing scenarios," *Expert Syst. Appl.*, vol. 42, no. 11, pp. 4950–4964, 2015.

[156] A. Elsts, J. Judvaitis, and L. Selavo, "SEAL: A Domain-Specific Language for Novice Wireless Sensor Network Programmers," *Proc. - 39th Euromicro Conf. Ser. Softw. Eng. Adv. Appl. SEAA 2013*, pp. 220–227, 2013.

[157] K. A. De Graaf, A. Tang, P. Liang, and H. Van Vliet, "Ontology-based software architecture documentation," *Proc. 2012 Jt. Work. Conf. Softw. Archit. 6th Eur. Conf. Softw. Archit. WICSA/ECSA 2012*, pp. 121–130, 2012.

[158] J. R. A. Santos, "Cronbach's Alpha: A Tool for Assessing the Reliability of Scales," *J. Ext.*, vol. 37, no. 2, pp. 1–5, 1999.

[159] F. Shull, J. Singer, and D. I. K. Sjøberg, *Guide to Advanced Empirical Software Engineering*. 2008.

[160] L. B. R. de Oliveira, K. R. Felizardo, D. Feitosa, and E. Y. Nakagawa, "Reference Models and Reference Architectures Based on Service-Oriented Architecture: A Systematic Review," *Softw. Archit.*, no. 6285, pp. 360–367, 2010.

[161] E. Y. Nakagawa, "Reference Architectures and Variability: Current Status and Future

Perspectives," in *RProceedings of the WICSA/ECSA 2012*, 2012, pp. 159–162.

[162] M. Guessi, L. B. R. de Oliveira, and E. Y. Nakagawa, "Current State on Representation of Reference Architectures," no. 363, pp. 1–29, 2011.

[163] E. Y. Nakagawa, E. F. Barbosa, and J. C. Maldonado, "Exploring Ontologies to Support the Establishment of Reference Architectures: An Example on Software Testing," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European*, 2009, pp. 249–252.

[164] P. Reed, *Reference architecture: The best of best practices*. The Rational Edge, 2002.

[165] N. F. D. Filho and E. F. Barbosa, "A Service-Oriented Reference Architecture for Mobile Learning Environments," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, no. Icmc, pp. 1–8.

[166] L. T. C. E. Mettala and M. H. Graham, "The Domain Specific Software Architecture Program," *Proc. DARPA Softw. Technol. Conf.*, no. June, 1992.

[167] F. J. Ortiz, J. A. Pastor, D. Alonso, F. Losilla, and E. de J\' odar, "A Reference Architecture for Managing Variability among Teleoperated Service Robots," *2nd Int. Conf. Informatics Control. Autom. Robot.*, pp. 322–328, 2005.

[168] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, E. Y. Nakagawa, and F. Oquendo, "Self-Adaptive middleware for wireless sensor networks: A reference architecture," *ACM Int. Conf. Proceeding Ser.*, vol. 07–11–Sept, 2015.

APPENDICES

Appendix A: List of Included Studies

This appendix shows the studies that are used to define the initial general vocabularies.

No.	Study title	Ref.
1.	The Past, Present, and Future for Software Architecture	[2]
2.	Representation of Reference Architectures: A Systematic Review	[4]
3.	The Concept of Reference Architectures	[9]
4.	Empirically-grounded Reference Architectures: A Proposal	[10]
5.	Researching Reference Architectures and Their Relationship with Frameworks, Methods, Techniques, and Tools	[16]
6.	Towards a Formal Description of Reference Architectures for Embedded Systems	[29]
7.	An Introduction to Software Architecture	[58]
8.	IEEE Recommended Practice for Architectural Description of Software-Intensive Systems	[36]
9.	An Approach to Reference Architecture Design for Different Domains of Embedded Systems	[53]
10.	An Aspect-oriented Reference Architecture for Software Engineering Environments	[56]
11.	Reference Architecture Knowledge Representation: An Experience	[147]
12.	Reference Models and Reference Architectures Based on Service-Oriented Architecture: A Systematic Review	[160]
13.	Reference Architectures and Variability: Current Status and Future Perspectives	[161]
14.	Current State on Representation of Reference Architectures	[162]
15.	Exploring Ontologies to Support the Establishment of Reference Architectures: An Example on Software Testing	[163]

16.	Reference architecture: The best of best practices	[164]
17.	A Service-Oriented Reference Architecture for Mobile Learning Environments	[165]
18.	The Domain Specific Software Architecture Program	[166]
19.	A Reference Architecture for Managing Variability among Teleoperated Service Robots	[167]
20.	Self-adaptive middleware for wireless sensor networks: A reference architecture	[168]
21.	A Reference Architecture for Managing Variability among Teleoperated Service Robots	[167]

Appendix B: Vocabulary of the Case Studies

This appendix shows vocabularies which were used in each case study.

B.1: Vocabulary of the First Case Study - The Reference Architecture of the Situated Multi-Agent System

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Subsystem	3.	Module
4.	Element	5.	Unit	6.	Component
7.	View	8.	Module Decomposition View	9.	Component and Connector View
10.	Component and Connector Shared Data View	11.	Component and Connector Communicating Processes View	12.	Stakeholder
13.	Maintenance Engineer	14.	Project Manager	15.	Architect
16.	User	17.	Developer	18.	Function
19.	Activity	20.	Mechanism	21.	Constraint
22.	Data	23.	Repository	24.	Knowledge
25.	Process	26.	Resource	27.	Hardware Resource
28.	Software Resource	29.	Responsibility	30.	Service
31.	Property	32.	Interface	33.	Observe Interface
34.	Send Interface	35.	Deliver Interface	36.	Perceive Interface
37.	Request Interface	38.	Read-write Interface	39.	Act Interface
40.	Transmit Interface	41.	Receive Interface	42.	Update Interface
43.	Influence Interface	44.	Sense Interface	45.	Connector Component
46.	Data Accessor Component	47.	Representation Generator Component	48.	Observation & Data Processing Component
49.	Communication Mediation Component	50.	Concurrent Unit Component	51.	Synchronization & Data Processing Component
52.	Communication Service Component	53.	Low-Level Control Component	54.	Interaction Component

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
55.	Communication Subsystem	56.	Perception Subsystem	57.	Agent Subsystem
58.	Decision-making Subsystem	59.	Application Environment Subsystem	60.	Has a
61.	Is part of	62.	Enable	63.	Access to
64.	Is a	65.	Describe	66.	Decomposed into
67.	Use	68.	Consist of	69.	Execute
70.	Include	71.	Define	72.	Provide

B.2: Vocabulary of the Second Case Study - The Reference Architecture of the Mobile Learning Environments

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	View	2.	General View	3.	Module View
4.	Runtime View	5.	Deployment View	6.	Element
7.	Information	8.	Function	9.	Role
10.	XML Protocol	11.	SOAP Protocol	12.	User
13.	Mobile Device	14.	Web Server	15.	Browser
16.	Module	17.	Controller Module	18.	Services Engine Module
19.	Teaching Module	20.	Administration Module	21.	Personalization Module
22.	Access Module	23.	Communication Module	24.	Documentation Module
25.	Authoring Module	26.	Mechanism	27.	Feature
28.	Service	29.	Knowledge	30.	Security
31.	Activity	32.	Database	33.	Data
34.	Request	35.	Analyse	36.	Perform
37.	Change	38.	Define	39.	Located
40.	Task	41.	Use	42.	Describe
43.	Store	44.	Retrieve	45.	Return
46.	Enable	47.	Receive	48.	Consume
49.	Exchange	50.	Produce	51.	Control
52.	Represent	53.	Consist of	54.	Establish
55.	Provide	56.	Access to	57.	

B.3: Vocabulary of the Third Case Study - The Reference Architecture of the Cloud Computing

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Define	2.	Describe	3.	View
4.	Actor	5.	Activity	6.	Function
7.	Constraint	8.	Person	9.	Resource
10.	Hardware Resource	11.	Software Resource	12.	Service
13.	SaaS	14.	PaaS	15.	IaaS
16.	Tool	17.	Security	18.	Role
19.	Process	20.	Developing Application	21.	Managing Application
22.	Deploying Application	23.	Testing Application	24.	Monitoring Application
25.	Instance of	26.	Use	27.	Attribute
28.	Include	29.	Consume	30.	Manage
31.	Is	32.	Has	33.	Produce
34.	Execute	35.	Require	36.	Consist of
37.	Apply to	38.	Used by	39.	Access to
40.	Cloud Provider Actor	41.	Cloud Auditor Actor	42.	Cloud Consumer Actor
43.	Cloud Carrier Actor	44.	Cloud Broker Actor	45.	

B.4: Vocabulary of the Forth Case Study - The Reference Architecture of the Web Servers

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Component	2.	Constraint	3.	Encompasses
4.	System	5.	Subsystem	6.	User
7.	Developer	8.	Security	9.	Is
10.	Resource	11.	HTML	12.	Text File
13.	Service	14.	Daily News Service	15.	Email Service
16.	Architectural Style	17.	Pipe-Filter Style	18.	Layered Architectural Style
19.	Program	20.	Java Servlet Program	21.	Common Gateway Interface Program
22.	Protocol	23.	Hyper Text Transfer Protocol	24.	Consist of
25.	Network	26.	Computer	27.	Describe
28.	Instance of	29.	Define	30.	Is part of
31.	Use	32.	Control	33.	Apply to
34.	Operating System	35.	Include	36.	Require
37.	Browser	38.	Netscape Navigator Browser	39.	Lynx Browser
40.	Internet Explorer Browser	41.	Transaction Log Subsystem	42.	Request Analyser Subsystem
43.	OS Abstraction Layer Subsystem	44.	Resource Handler Subsystem	45.	Reception Subsystem
46.	Access Control Subsystem	47.	Utility Subsystem	48.	

B.5: Vocabulary of the Fifth Case Study - The Reference Architecture of the Sensor Networks Integration and Management System

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Component	3.	Wrapper Component
4.	Mediator Component	5.	Describe	6.	Include
7.	Module	8.	Mediator Communications Module	9.	Network Classification Module
10.	Architecture Style	11.	Layered Architecture Style	12.	Query Processing Module
13.	Responsibility	14.	Interface	15.	Mediator Interface
16.	Network Interface	17.	Wrapper Communications Module	18.	User Communications Module
19.	Protocol	20.	Provide	21.	Sensor Component
22.	Communication Manager Component	23.	Network Component	24.	Access Manager Component
25.	Query Builder Component	26.	Result Viewer Component	27.	User Interface
28.	Wrapper Interface	29.	Information	30.	Use a/an
31.	Repository	32.	Network Changes Handler Component	33.	Network Discovery Component
34.	Query Analyser Component	35.	Query Engine Component	36.	Results Handler Component
37.	Request Interpreter Component	38.	Access to	39.	Is a/an
40.	Has a/an	41.	Used by	42.	

B.6: Vocabulary of the Sixth Case Study - The reference architecture of the Ubiquitous Computing

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	View	2.	Component View	3.	Deployment View
4.	Process View	5.	Layered Architectural Style	6.	System
7.	Element	8.	Component	9.	Is a
10.	Task	11.	Interface	12.	User
13.	Use	14.	Activity	15.	Process
16.	Information	17.	Data	18.	Sensor Component
19.	Actuator Component	20.	Context Service Component	21.	Actuation Service Component
22.	Context Repository Component	23.	Context Component	24.	Reasoning Component
25.	Adaptation Component	26.	Coupling and Mobility Mechanism Component	27.	Aggregation Component
28.	Security	29.	Module	30.	Service
31.	Constraint	32.	Responsibility	33.	Has
34.	Encompasses	35.	Function	36.	Include
37.	Attribute	38.	Provide	39.	Repository
40.	Feature	41.	Describe	42.	Access to
43.	Collect	44.	Property	45.	

B.7: Vocabulary of the Reference Architecture of the Web Browsers

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Web Browser System	3.	Subsystem
4.	Networking Subsystem	5.	Rendering Engine Subsystem	6.	Browser Engine Subsystem
7.	User Interface Subsystem	8.	JavaScript Interpreter Subsystem	9.	Data Persistence Subsystem
10.	Display Backend Subsystem	11.	XML Parser Subsystem	12.	Resource
13.	Function	14.	Resource Hardware	15.	Computer
16.	Cell Phones	17.	Web Page	18.	Web Server
19.	HTML	20.	HTTP	21.	Feature
22.	Toolbars Feature	23.	Smart Download Handling Feature	24.	Preferences Feature
25.	Printing Feature	26.	Visual Page-load Progress Feature	27.	Hooks Feature
28.	Set of User Interface Widgets Feature	29.	Drawing and Windowing Primitives Feature	30.	Fonts Feature
31.	Layered Architectural Style	32.	Cascading Style Sheets	33.	User
34.	Component	35.	Curses Component	36.	GTK+ Adapter Component
37.	wwwlib Component	38.	UI Toolkit (XPEE) Component	39.	Browser Component
40.	Querying and Manipulation of the Rendering Engine	41.	User interface Component	42.	GTK+ / X11 Libraries Component
43.	Security (Libgnutls) Component	44.	Necko Component	45.	Secure component
46.	Persist Component	47.	Spider−Monkey Component	48.	Security (NSS/PSM) Component
49.	Expat Component	50.	HTML Parser Component	51.	Disk
52.	Data	53.	Cookies Data	54.	Bookmarks Data
55.	JavaScript Code	56.	FTP	57.	HTTP
58.	XML	59.	Visual Representation	60.	Calculate a
61.	Access to	62.	Allow a	63.	Apply to

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
64.	Connect to/ Connect a	65.	Display a	66.	Evaluate a
67.	Has a	68.	Implement a	69.	Include a
70.	Parse a	71.	Produce a	72.	Provide a
73.	Represented as	74.	Store in/ Stored in	75.	Support a
76.	Written by	77.	Execute on/ Execute a	78.	Provide feature to
79.	Used by/Use a	80.	Load a	81.	

B.8: Vocabulary of the Reference Architecture of the Robot Teleoperation

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	System	2.	Subsystem	3.	Module
4.	Component	5.	Element	6.	Architectural Style
7.	Client-Server Architectural Styles	8.	Communicating Process Architectural Style	9.	Layered Architectural Styles
10.	Function	11.	Activity	12.	User
13.	Describe	14.	Property	15.	Service
16.	Is a	17.	Network	18.	Mechanism
19.	Data	20.	Information	21.	Resource
22.	Hardware Resource	23.	Software Resource	24.	Operating system
25.	Communication Subsystem	26.	Protocol	27.	Programming Language
28.	Collisions Detection Subsystem	29.	User Interface Subsystem	30.	Controller Subsystem
31.	Graphical Representation Subsystem	32.	Tool	33.	Camera
34.	Sensor	35.	Computer	36.	Send
37.	Light	38.	Include	39.	Develop
40.	Provide	41.	Require	42.	Consist of
43.	Exchange	44.	Request	45.	Has a
46.	Use a	47.	Receive	48.	Update
49.	Execute				

Appendix C: General Vocabulary

This appendix illustrates the general vocabulary that describes the entities and relationships.

C.1: General Vocabulary that Describes Entities

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	View	2.	System	3.	Subsystem
4.	Stakeholder	5.	Function	6.	Task
7.	Resource	8.	Protocol	9.	Interface
10.	Data	11.	Concern	12.	Service
13.	Responsibility	14.	Activity	15.	Mechanism
16.	Information	17.	Component	18.	Architectural Style
19.	Security	20.	Process	21.	Tool
22.	Attribute	23.	Repository	24.	Knowledge
25.	Feature	26.	Role	27.	

C.2: General Vocabulary that Describes the Relationships

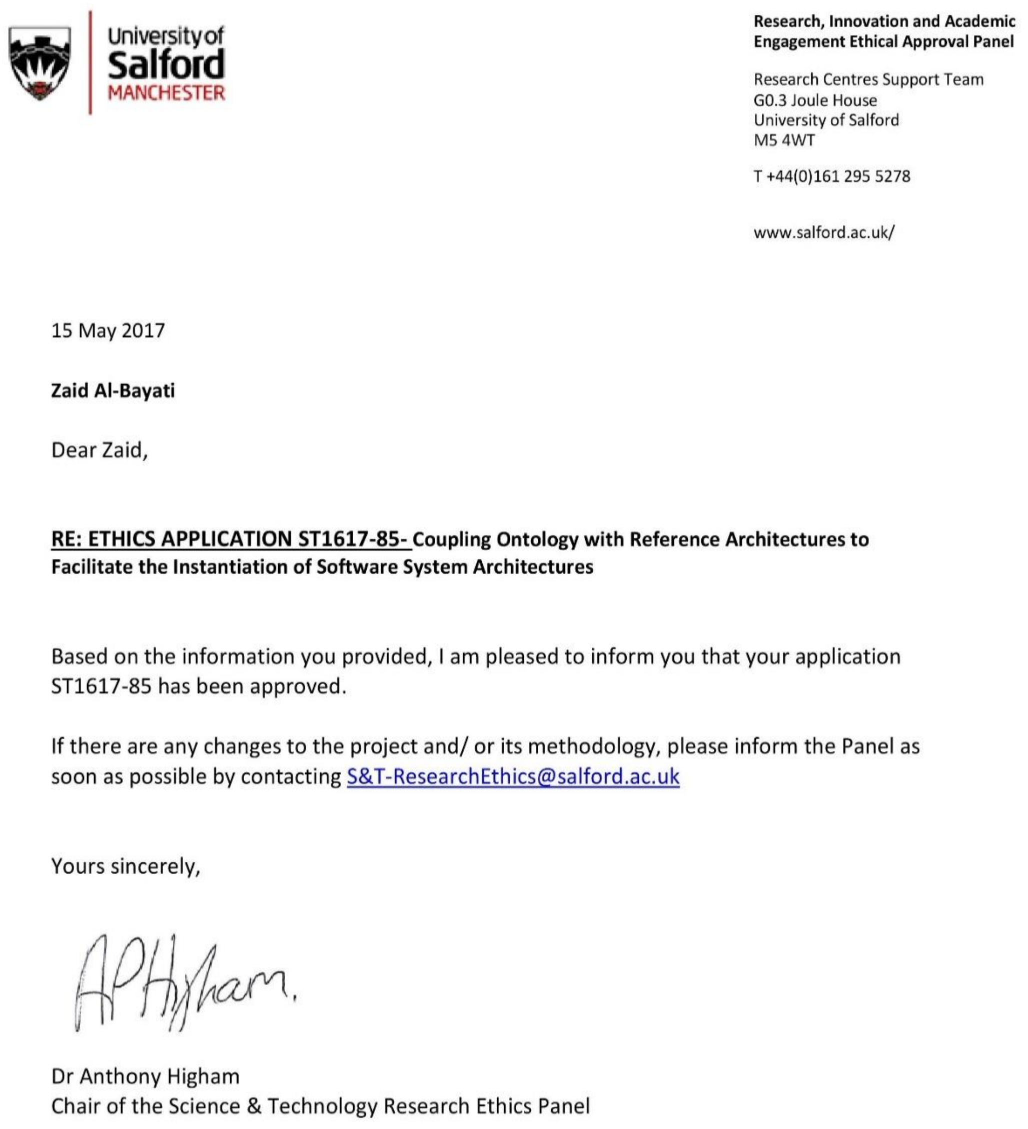
No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
1.	Require	2.	Enable	3.	Store
4.	Apply to	5.	Include	6.	Is a
7.	Describe	8.	Produce	9.	Composed of
10.	Consume	11.	Has a	12.	Used by
13.	Access to	14.	Provide	15.	Analyse
16.	Perform	17.	Retrieve	18.	Return
19.	Control	20.	Manage	21.	Instance of
22.	Change	23.	Encompasses	24.	Decomposed into

No.	Vocabulary	No.	Vocabulary	No.	Vocabulary
25.	Consist of	26.	Use a	27.	Exchange
28.	Define	29.	Execute	30.	Receive
31.	Is part of	32.	Request	33.	

Appendix D: User Study Experiment

D.1: Ethical Approval Letter

This appendix shows the ethical approval letter.



University of Salford MANCHESTER

Research, Innovation and Academic Engagement Ethical Approval Panel

Research Centres Support Team
G0.3 Joule House
University of Salford
M5 4WT

T +44(0)161 295 5278

www.salford.ac.uk/

15 May 2017

Zaid Al-Bayati

Dear Zaid,

RE: ETHICS APPLICATION ST1617-85- Coupling Ontology with Reference Architectures to Facilitate the Instantiation of Software System Architectures

Based on the information you provided, I am pleased to inform you that your application ST1617-85 has been approved.

If there are any changes to the project and/ or its methodology, please inform the Panel as soon as possible by contacting S&T-ResearchEthics@salford.ac.uk

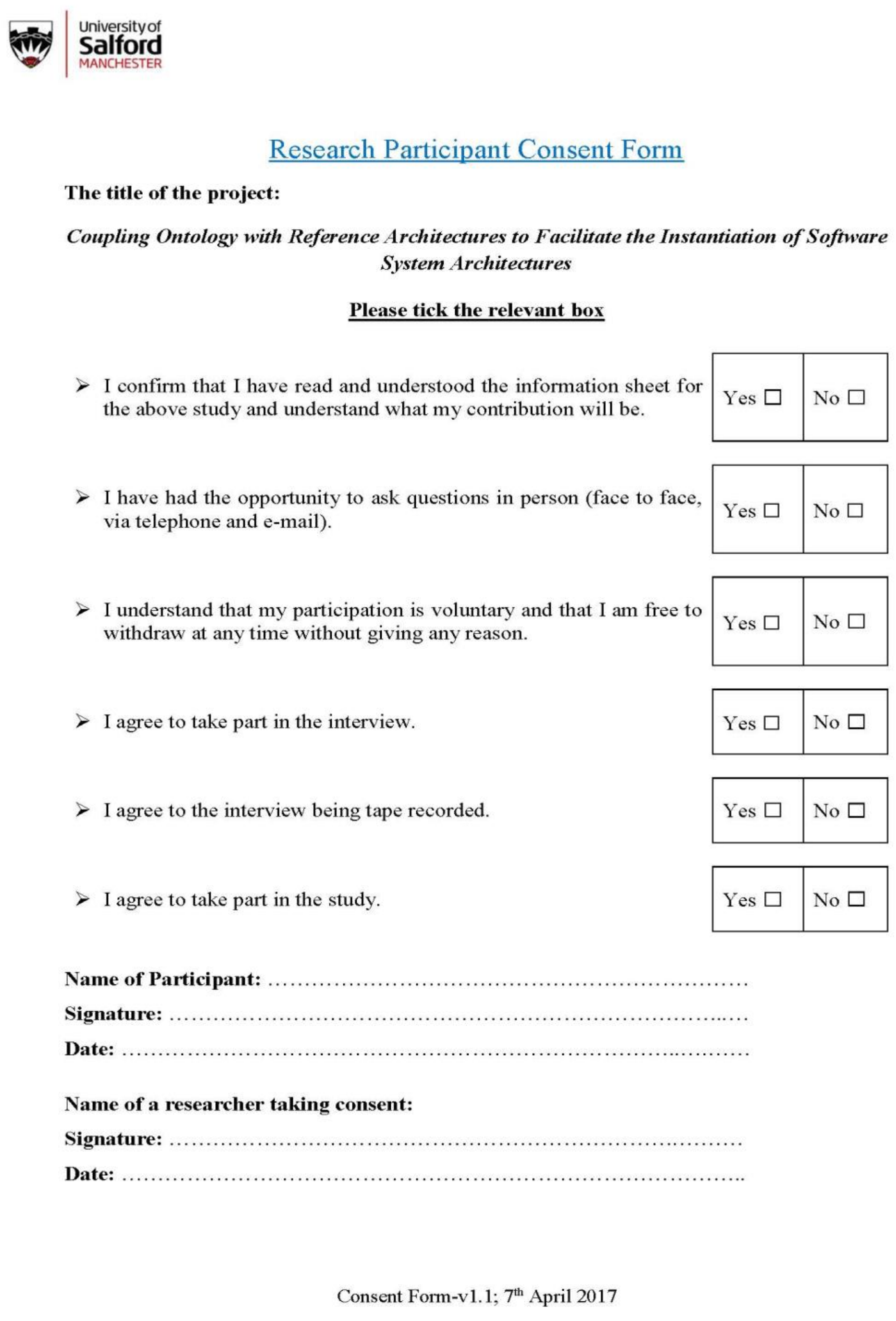
Yours sincerely,

APHigham.

Dr Anthony Higham
Chair of the Science & Technology Research Ethics Panel

D.2: Research Participant’s Consent Form

This appendix shows the research participant consent form.



University of Salford MANCHESTER

Research Participant Consent Form

The title of the project:

Coupling Ontology with Reference Architectures to Facilitate the Instantiation of Software System Architectures

Please tick the relevant box

➢ I confirm that I have read and understood the information sheet for the above study and understand what my contribution will be.	Yes ☐	No ☐
➢ I have had the opportunity to ask questions in person (face to face, via telephone and e-mail).	Yes ☐	No ☐
➢ I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason.	Yes ☐	No ☐
➢ I agree to take part in the interview.	Yes ☐	No ☐
➢ I agree to the interview being tape recorded.	Yes ☐	No ☐
➢ I agree to take part in the study.	Yes ☐	No ☐

Name of Participant: …………………………………………………………………

Signature: ………………………………………………………………………..…

Date: …………………………………………………………………………..…..……

Name of a researcher taking consent:

Signature: ………………………………………………………………………..………

Date: ……………………………………………………………………………..

Consent Form-v1.1; 7th April 2017

D.3: Pre-experiment Questions

No.	Question	Answer
1.	Study Background (Example: software engineering)	
2.	What is your current education level?	☐ Bachelor's Degree ☐ Master ☐ Doctorate ☐ Post-Doctorate ☐ Other
3.	What is the area related to your current education level?	☐ Software Engineering ☐ Information System ☐ Computer Science ☐ Other
4.	How would you rate your knowledge in software system architectures development?	☐ Beginner ☐ Intermediate ☐ Expert ☐ No information
5.	How do you rate your development skills proficiency?	☐ Advance ☐ Intermediate ☐ Basic ☐ No skill
6.	Have you developed a software system architecture?	☐ Yes ☐ No
7.	Have you used a reference architecture before?	☐ Yes ☐ No

D.4: Post-experiment Questions

Goal	Metrics	No.	Statement	Scale	
Introductory Questions		1.	The given tutorials and presentation were easy to understand.	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3
				☐ Difficult	2
				☐ Very Difficult	1
		2.	The descriptions of the tasks were clear.	☐ Strongly Agree	5
				☐ Agree	4
				☐ Neutral	3
				☐ Disagree	2
				☐ Strongly Disagree	1
Usability	Complexity	3.	How do you find the development of software system architecture?	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3
				☐ Difficult	2
				☐ Very Difficult	1
		4.	How do you evaluate the difficulty in finding the components of the reference architecture?	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3
				☐ Difficult	2
				☐ Very Difficult	1
		5.	How do you evaluate the difficulty in finding the relationships between the components of the reference architecture?	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3
				☐ Difficult	2
				☐ Very Difficult	1
	Traceability	6.	The components of a reference architecture are easily traceable by developers.	☐ Strongly Agree	5
				☐ Agree	4
				☐ Neutral	3
				☐ Disagree	2
				☐ Strongly Disagree	1
	Understandability	7.	The description of components is easy to	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3

			understand by developers.	☐ Difficult	2
				☐ Very Difficult	1
		8.	The description of relationships between the components is easy to understand by developers.	☐ Very Easy	5
				☐ Easy	4
				☐ Neutral	3
				☐ Difficult	2
				☐ Very Difficult	1
	Clarity	9.	The components of a reference architecture are presented in an organised and structural way.	☐ Strongly Agree	5
				☐ Agree	4
				☐ Neutral	3
				☐ Disagree	2
				☐ Strongly Disagree	1
		10.	The relationships of a reference architecture are presented clearly.	☐ Strongly Agree	5
				☐ Agree	4
				☐ Neutral	3
				☐ Disagree	2
				☐ Strongly Disagree	1
Open Question		11.	Do you have any recommendations to improve the proposed process? If yes, please write them here	☐ Yes	
				☐ No	
			If yes, please write your comments…….		