Towards a Comprehensive Functional Layered Architecture for the Semantic Web

by

Aurona J. Gerber

Student no: 651-592-4

Thesis submitted in accordance with the requirements
for the degree of
DOCTOR OF PHILOSOPHY
in the subject
COMPUTER SCIENCE
at the
UNIVERSITY OF SOUTH AFRICA

PROMOTER: Prof. Andries Barnard
JOINT PROMOTER: Prof. Alta J. van der Merwe

November 2006

Dedicated to my family,
particularly Thinus,
my ever-supporting husband,
Mom with her lively interest in life,
and Dad, Dr Sybren Hiemstra, whose
enthusiasm for science and
thirst for knowledge
always remains an
inspiration.

ACKNOWLEDGEMENTS

In acknowledgement to my Father God,
Source of all Knowledge and Wisdom

I would sincerely like to thank both my supervisors. You were great, and worked extremely well together as a team to guide me through this study. And you are forever part of one of the most significant experiences in my life.

Prof. Andries Barnard: thank you for your inputs and insight given even though your personal circumstances were sometimes extremely difficult. Thank you for the very critical reading, which in the end, at last, had the desired result.

Prof. Alta van der Merwe: thank you for your support, motivation and friendship before, during and hopefully after, the writing of this thesis. Alta, jy was baie meer as 'n promotor, en ek waardeer dit opreg. 'Oh, the places you'll go!'

Dankie Thinus, sommer vir alles! Sonder jou was hierdie tesis nie moontlik nie.

Aurona en Ané, dankie vir julle begrip. Julle is twee wonderlike mensies. "Guilt is just part of the mothering deal, especially, it seems, for working moms." [Time, Jun.04, 2001]

I am very grateful towards the CSIR, and in particular, Hina Patel. Thank you Hina, you gave me the valuable time that was necessary to finish this thesis. Without your generosity, I would have been busy for at least another year. You spared many people a lot of agony.

The Bit2Phd group, you are all special friends and each of you motivated and inspired me when it was direly needed.

And last, but not least, Jackie Viljoen who was responsible for the language editing of this thesis, and who worked right through the Christmas season to finish in time despite several technical problems.

ABSTRACT

The Semantic Web, as the foreseen successor of the current Web, is envisioned to be a semantically enriched information space usable by machines or agents that perform sophisticated tasks on behalf of their users. The realisation of the Semantic Web prescribe the development of a comprehensive and functional layered architecture for the increasingly semantically expressive languages that it comprises of. A *functional architecture* is a model specified at an appropriate level of abstraction identifying system components based on required system functionality, whilst a *comprehensive architecture* is an architecture founded on established design principles within Software Engineering.

Within this study, an argument is formulated for the development of a comprehensive and functional layered architecture through the development of a Semantic Web status model, the extraction of the function of established Semantic Web technologies, as well as the development of an evaluation mechanism for layered architectures compiled from design principles as well as fundamental features of layered architectures. In addition, an initial version of such a comprehensive and functional layered architecture for the Semantic Web is constructed based on the building blocks described above, and this architecture is applied to several scenarios to establish the usefulness thereof.

In conclusion, based on the evidence collected as result of the research in this study, it is possible to justify the development of an architectural model, or more specifically, a comprehensive and functional layered architecture for the languages of the Semantic Web.

ABBREVIATIONS

Abbreviation	Description
ACM	Association for Computing Machinery
ANSI	American National Standards Institute
ARPANET	Advanced Research Projects Agency Network
CFL architecture	Comprehensive and Functional Layered architecture
CORBA	Common Object Request Broker Architecture
DARPA	Defence Advanced Research Projects Agency
DL	Description Logics
DLP	Description Logic Programs
DOM	Document Object Model
DTD	Document Type Declaration
ERCIM	European Research Consortium for Informatics and Mathematics
GUI	Graphical User Interface
HDLC	High-level Data Link Control
HTML	Hypertext Markup Language
ICCC	International Computer Communication Conference
ICT	Information and Communication technology
IEEE	Institute for Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IRI	Internationalised Resource Identifiers
IS	Information System
ISO	International Organisation for Standardisation
LDAP	Lightweight Directory Access Protocol
MIT	Massachusetts Institute of Technology
ODBC	Open Database Connectivity
OIL	Ontology Inference Layer

OSI	Open Systems Interconnect
OWL	Web Ontology Language
RDBMS	Relational Database Management System
RDF	Resource Descriptive Framework
RIF	Rule Interchange Format
SAWA	Situation Awareness Assistant
SDLC	Synchronous Data Link Control
SE	Software Engineering
SHOE	Simple HTML Ontology Extensions
SOAP	Simple Object Access Protocol
SW	Semantic Web
TAG	Technical Architecture Group
TCP/IP	Transmit Control Protocol/Internet Protocol
UCLA	University of California, Los Angeles
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	Extensible Markup Language
XMLDSig	XML Digital Signatures
XMLEnc	XML Encryption
XSL	Xtensible Stylesheet Language

THESIS CHAPTER LAYOUT

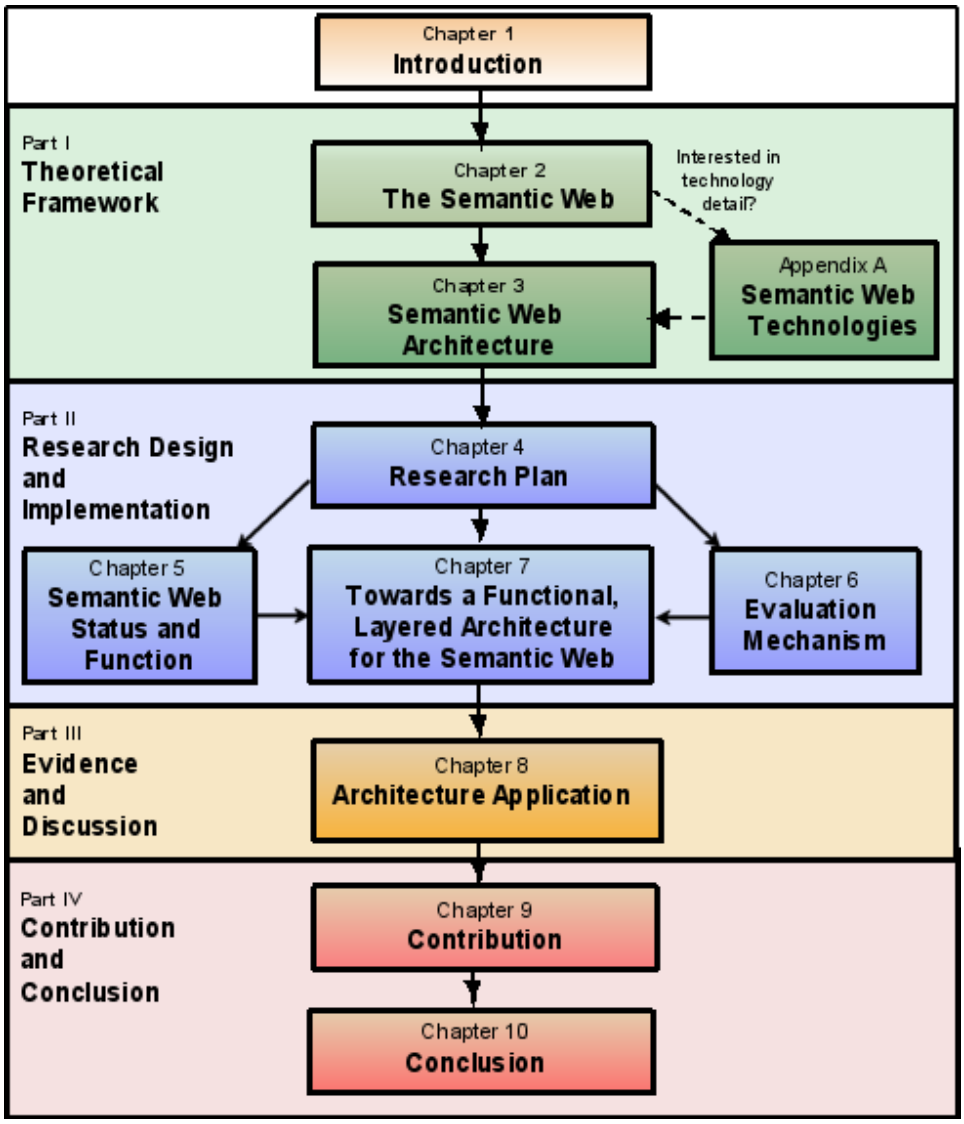


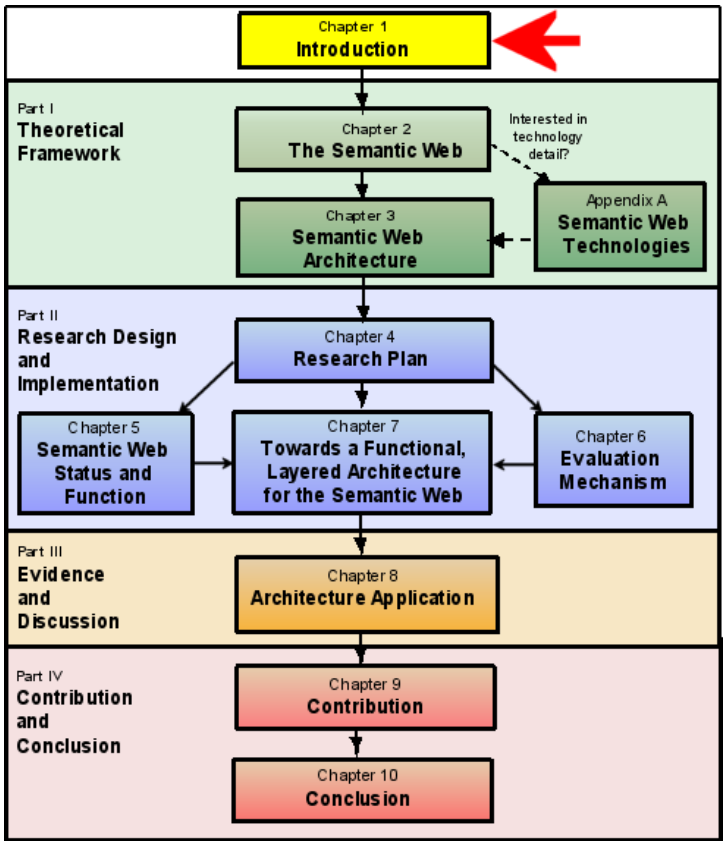
TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION



Thesis Chapter Layout.

The thesis is commenced with an introductory chapter, Chapter 1. This chapter layout is discussed further in Section 1.10 in Chapter 1.

Chapter Contents

Figures

Tables

1.1 INTRODUCTION

The Web revolutionised the way in which our generation collects, uses and disseminates information. However, the remarkable adoption and integration of the Web into society have resulted in several problems for its users, including aspects such as information overload, discrepancy and trustworthiness.

The information overload experienced by present-day users of the Web was already prevalent when the paper by Berners-Lee, Hendler and Lassila [38] containing a vision of the Semantic Web was published in the Scientific American in 2001. In this vision the Semantic Web is portrayed as an information space usable by sophisticated agents that act on behalf of their users to solve their information management problems. Thus, the Semantic Web that embodies substantially more computational and reasoning power for information management than the present Web, is regarded as the inevitable successor of the Web.

When the Semantic Web vision was envisaged, several Web technologies were already accepted as standards or recommendations by the W3C (World Wide Web Consortium). The protagonists of the Semantic Web vision foresaw the inclusion of several W3C technologies into the Semantic Web. In an attempt to depict and order these technologies, Berners-Lee in particular, presented several versions of a Semantic Web architecture where these technologies are layered into a so-called *stack* of increasingly expressive languages for meta-data specification [28, 31, 34, 35]. These Semantic Web architecture versions depict either W3C language specifications or functionality on different layers. However, the meaning and objectives of these versions of the Semantic Web architecture were not documented and upon scrutiny, it is possible to indicate that the versions currently proposed, depict inconsistencies and irregularities with regards to structure and intended meaning. This is argued in Chapter 3 of this thesis.

The purpose, intention and meaning of the currently proposed versions of the Semantic Web architecture are not clearly described and this gives rise to confusion and misunderstanding by users and developers who dis-

cuss, develop or adopt the Semantic Web with its associated technologies. The confusion is illustrated in literature for example, where several issues with regards to the layering of the technologies of the Semantic Web are published [59, 123, 131, 151, 191].

In order to address these problems, the research in this study proposes the development of a well documented and clear architecture for the Semantic Web that can serve as a framework for debate regarding language layering and inclusion, as well as technology specifications and implementations. This study proposes that a well documented and clear architecture will also be *comprehensive* and *functional*. A *functional* architecture depicts system components identified by their specific function within a system [18, 196], while a *comprehensive* architecture is an architecture reflecting design decisions based on established Software Engineering principles [13, 18].

1.2 MOTIVATION FOR THIS STUDY

Within Software Engineering an architecture is regarded as a model [10]. An architectural model depicts invisible aspects of software systems [155] such as the structure and design decisions about system composition [259].

A model is an abstraction of reality [10] and models are regarded as indispensable for the effective use of any knowledge within the Software Engineering domain [181]. Apart from conceptualising and categorising related system aspects, models are essential for successful communication between stakeholders [10].

The use of models has several advantages within Information System development, and the list below includes some of the benefits that are applicable to architectures:

- The abstraction depicted by a model simplifies the domain under discussion to reflect only the essential components [10].
- The conceptualisation and categorisation depicted by a model cre-

ates a common framework for discussion [10, 259].

- ▷ The conceptualisation of system aspects allows for comprehension and debate [259].
- ▷ A model that serves as a framework assists with development of specifications [265].
- ▷ Models are indispensable for communication between stakeholders about the invisible aspects of information systems [10, 196].

A Semantic Web architecture as a model would entail inclusion of the benefits of models into the Semantic Web discussion. According to Wentzel [259], a successful model should furthermore be useful, intuitive and predictive. Therefore, the Semantic Web architecture should comply with these characteristics in order to be regarded as a successful model.

In summation, the motivation for this study is the development of a comprehensive and functional layered architectural *model* for the Semantic Web.

1.3 PROBLEM STATEMENT AND PURPOSE OF THIS STUDY

The current Semantic Web domain is characterised by the following phenomena:

1.3.1 The technologies of the Semantic Web permeate a comprehensive domain

The Semantic Web with its associated technologies permeates various fields and application domains within ICT (Information and Communications Technology). In general, it is difficult to assimilate and understand the impact and role of these technologies within the context of the Semantic Web, specifically with regard to the proposed versions of the Semantic Web architecture [28, 31, 34, 35]. Therefore, it is imperative to investigate the purpose of these technologies, and to determine their function and present

status within the context of the Semantic Web. The documentation of such an investigation into the Semantic Web technologies provides, amongst other things, a starting point to assimilate Semantic Web terminology and associated concepts.

The determination of the *function* of technologies, specifically as regard Semantic Web meta-data specification, is a prerequisite for the development of a *functional* Semantic Web architecture.

1.3.2 Evaluation and design criteria for architectures are lacking

Within the Software Engineering specification of a system development life cycle, several essential artefacts for the successful design and development of information systems are specified. One such artefact is a well-defined functional architecture [13, 18, 105]. In addition to being *functional*, such an architecture should also be *comprehensive* to reflect design decisions according to fundamental Software Engineering design principles.

An architecture is regarded as essential for the successful implementation of any information system [18], and within literature generally there is consensus that an architecture at least depicts the structure and organisation of system components [18, 105]. However, authors emphasise different aspects of system development when referring to architectures. At present, literature offers no universally agreed-upon comprehensive definition for the term *architecture*. In addition, there is specifically no agreed-upon definition for the term *layered architecture*, even though it is a common best practice tactic used by system architects in system design. Although several design principles and best practises for system design and architecture development were documented [14], no evaluation mechanism for *architectures* and, in particular, *layered architectures*, could be found in literature. In order to achieve the development of a comprehensive and functional layered architecture for the Semantic Web, a mechanism for its evaluation is required. Such a mechanism would require the establishment of design and evaluation criteria for layered architectures.

Within Software Engineering, current research - which includes architectural discussions - focuses mainly on the role of an architecture as a design artefact within the system development life cycle [18, 24, 105, 193]. The research in this study contributes to the Software Engineering discipline in that it proposes, amongst other things, an evaluation mechanism for layered architectures.

1.3.3 A comprehensive and functional layered architecture for the Semantic Web is required

Tim Berners-Lee presented several versions of a layered architecture for the Semantic Web as an attempt to organise the existing W3C Semantic Web technologies and identified functionalities for meta-data languages [28, 31, 34, 35]. However, the meaning of these versions of the architecture has not been defined. It is possible to indicate discrepancies and irregularities within these proposed versions, and the models are therefore confusing. In addition, several issues regarding layering of the technologies are still unresolved (this argument is elaborated upon in Chapter 3).

There are a number of W3C initiatives with regard to architecture, for example the *W3C Architecture Domain* and the TAG (*Technical Architecture Group*). However, none of the present W3C architectural initiatives focus on the Semantic Web architecture specifically.

Since the proposed versions of the Semantic Web architecture is not sufficient to resolve current issues within the Semantic Web application domain, it is imperative that a comprehensive and functional layered architecture for the Semantic Web is developed. Such an architecture is required by several Semantic Web role-players, including:

- The W3C: One of the purposes of the W3C is the development of Web specifications, also referred to as W3C Recommendations. The lack of an agreed-upon and accepted architecture for the Semantic Web results in obstacles for the specification, acceptance and adoption of W3C Semantic Web Recommendations, since no predetermined architectural framework exists.

- Semantic Web researchers: Generally current Semantic Web research focuses on specific technological issues with regard to layering of Semantic Web technologies [100, 191], the knowledge representation or *ontology* function of the Semantic Web, as well as the integration and combination of multiple and diverse ontologies [102, 144, 153, 157, 185, 262]. Research specifically with regard to a Semantic Web *architecture* is lacking, and the specification of an architecture should assist with the resolution of several of the academic debates regarding the Semantic Web.
- Semantic Web developers: Developers experience problems to adopt the Semantic Web with its associated technologies since no standard architecture is defined.

Therefore, this study addresses a current shortcoming within literature as well as W3C initiatives with its primary focus the development of a *functional and comprehensive Semantic Web layered architecture*.

1.3.4 Purpose of the study

The purpose of this study is thus stated to be the development of a comprehensive and functional layered architecture for the Semantic Web, where:

- An *architecture* is a model that depicts the structure of components of which a system comprises within a specific context.
- A *layered architecture* is an architecture that organises the system components or groups of components into successive layers logically similar and of equal rank. Any layer uses functionality from its lower layers and isolates these lower layers from layers above.
- A *functional* architecture is an architecture that depicts components identified by their function within the system, whilst
- a *comprehensive* architecture is an architecture reflecting design decisions based on established Software Engineering principles.
- Finally, the *Semantic Web* is regarded as the inevitable successor of the present Web, providing an intelligent information space comprising of a layered architecture of increasing semantically expressive languages for agents to roam and perform sophisticated information

management tasks on behalf of their users.

1.4 RESEARCH QUESTIONS

The research questions defined for this study support the purpose of the research as indicated in Section 1.3.4 above. In order to develop a *functional* architecture, it is necessary to determine the status and function of the languages and technologies that comprise the architecture. In order to develop a *comprehensive* architecture, it is necessary to investigate architectural design principles. In conclusion, when using the existing versions of the Semantic Web layered architecture as departure point, it is necessary to establish the adaptations required to develop a singular comprehensive and functional layered architecture. The research questions are therefore formulated as:

	Research Questions
(1)	What is the function of each technology included in the present versions of the Semantic Web layered architecture? ▷ What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?
(2)	To which criteria should a layered architecture conform in order to adhere to system design principles? ▷ Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?

(3)	How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?

Table 1.1: Research questions

1.5 SCOPE AND CONTEXT OF THE STUDY

It is not possible to complete a study of this magnitude without stating the context and limiting the scope. In Section 1.5.1 this study is put into context. Section 1.5.2 includes a discussion on aspects that are included in this study, whilst omissions are listed in Section 1.5.3.

1.5.1 Context of the study

The research described in this thesis resides within the Computer Science and Information Systems discipline, and specifically within the field of Software Engineering. This provides the positioning and departure point for the research argument. Consequently, the research is executed within the *Semantic Web application domain*.

This study supports the argument that the Semantic Web is not yet established [191] but that recognisable progress has been made towards the specification and acceptance of technology standards for the management of the meta-data layer required to realise the Semantic Web [232]. The W3C [250], supported by the IETF (Internet Engineering Task Force) [137] plays the leading role with regard to the specification and management of such standards. This study accepts the authority of the W3C in this domain and any discussion of the Semantic Web and its supporting technologies will be done within the context provided by the W3C.

1.5.2 Scope of the study

This study includes an investigation into existing Semantic Web technologies, as well as the proposed versions of the Semantic Web architecture of Berners-Lee [28, 31, 34, 35]. As a point of departure, this investigation includes the determination of the functionality of existing technologies within the context of the Semantic Web, as well as the establishment of the current status and adoption of these technologies.

In addition, the study includes an investigation into system architectures, which includes related aspects such as the development of an architecture, the design and evaluation thereof, layering, architecture documentation, as well as the design and evaluation of layered architectures.

In particular, the study is aimed at the development of a comprehensive and functional layered architecture for the Semantic Web. Therefore, the study includes discussions on aspects related to the adoption of functional architectures, and the benefits that the adoption of a layered architecture will have in the application domain of the Semantic Web.

1.5.3 Limitations of scope

The following aspects are not dealt with in this study:

- Any discussion about the advantages and/or disadvantages of specific Semantic Web technologies,
- Resolution of the layering debate with regard to technologies of the Semantic Web,
- Aspects pertaining to ontology engineering,
- Aspects pertaining to Semantic Web technology implementation,
- The unresolved reasoning and rules issues concerning the Semantic Web logical framework; and
- Software agents that are envisioned to be the *users* of the Semantic Web.

1.6 RATIONALE BEHIND THIS STUDY

The rationale behind this study is discussed from a national and scientific perspective.

1.6.1 National rationale

The South African government identified research and development, particularly in the field of Science, as one of the national priorities in South Africa. This lead to the establishment of the DST (Department of Science and Technology) [78, 94, 95]. In addition, ICT was identified as one of the compulsory research and development priorities in order to assist with growth in Southern Africa [94]. To support ICT initiatives, the Meraka Institute, operating under the auspices of the South African CSIR (Council for Scientific and Industrial Research), was formed.

> *The Meraka Institute derives its mandate as a national strategic initiative from President Mbeki's 2002 State of the Nation Address. The major objective of the Meraka Institute is to facilitate national economic and social development through human capital development and needs-based research and innovation, leading to products and services based on Information and Communication Technology* [164].

Key strategic areas identified within ICT include affordable broadband connectivity as well as access to the Internet and Web. In addition, research and development in Web technologies are regarded as important activities and constitute building blocks for the implementation of national priorities [94, 164].

The Semantic Web is regarded as the inevitable successor of the Web, and research in the Semantic Web domain with its associated technologies thus supports the South African national priorities. In addition, the Semantic Web with its technologies might assist with the effective adoption and use of the Web in such a technologically neglected society as South Africa,

especially since the envisioned Semantic Web will assist with sophisticated initial information management tasks on behalf of its users.

1.6.2 Scientific rationale

The scientific rationale refers to current limitations of the theory that could be addressed by this study. At present, no comprehensive and functional layered architecture for the Semantic Web has been proposed. However, it is noteworthy that the W3C developed several Web technology specifications and released it as W3C Recommendations. It is foreseen that several of these technology specifications will be incorporated into the eventual Semantic Web. In an attempt to organise these technologies and to stimulate discussion about the Semantic Web, Berners-Lee presented several proposed versions of a Semantic Web architecture [28, 31, 34, 35]. These architectural models depict either technologies or functionalities in an ad hoc manner in different layers and it is possible to identify irregularities and inconsistencies in their layered structure.

The adoption of the Semantic Web, as well as its integration into the Web community prescribes a generally agreed-upon comprehensive and functional layered architecture for the Semantic Web. Such an architecture:

- is an architectural model that should assist with communication and the resolution of debate about the Semantic Web and its associated technologies,
- is an architectural model based on fundamental principles of software design and should therefore aid with Semantic Web design decisions according to these principles,
- is a framework that aids the development of Semantic Web standards for data interoperability in a way similar to the ISO/OSI (International Organisation for Standardisation / Open Systems Interconnect) layered architecture that assisted with the development of protocol standards for network interoperability; and
- is a mechanism that allows for one of its layers to be instantiated by more than one technology in the same way the ISO/OSI architecture assisted with the specification of several protocols developed and re-

siding in the same layer, thus allowing for inclusiveness of diverse technologies with the same functionality.

The Semantic Web, as the next-generation Web, is necessitated by the information overload and its related problems experienced by Web users and developers. Given the indispensable role any architecture plays in the development of information systems, it is plausible to speculate that a comprehensive and functional layered architecture for the Semantic Web is a prerequisite for its successful development.

1.7 RESEARCH METHOD

Research was done by way of a qualitative study using qualitative data such as documents for the logic and conclusion of an argument [168]. The epistemological stance of the study is identified to be interpretevist since qualitative data is analysed and interpreted in order to obtain the research results [57, 168, 174, 227].

1.8 RESEARCH APPROACH

A research approach defines the processes that are executed during the research. Applied research, where artefacts are built and tested, is generally the preferred methodology within Computer Science and Information Systems research. *Applied research* is classified as either empirical or non-empirical, and is expanded to describe several *research approaches*. These research approaches include *theory-building or model-building studies* and *methodological studies* relevant to the research described in this thesis [172].

A *model-building study* is applicable to answer Research Questions 1 and 3 concerned with the status and function of the Semantic Web technologies, as well as the development of a comprehensive and functional Semantic Web layered architecture. A *methodological study* is applicable to the research activities related to Question 2 concerned with the devel-

opment of an evaluation mechanism for layered architectures.

When performing qualitative research, qualitative data analysis activities are applicable. Miles and Huberman [168] define qualitative data analysis as consisting of three concurrent flows of activity namely *data reduction*, *data display* and *conclusion drawing and verification*. *Data reduction* is defined as the process of selecting, focusing, simplifying, abstracting, and transforming the qualitative data. *Data display* aim to assemble organised information into accessible, compact form in order to draw justified conclusions. During *conclusion drawing and verification* a researcher decides what the data mean.

The qualitative data used for the research described in this thesis are documents obtained from formal academic sources, supplemented with publications from verified, informal sources such as dedicated Web sites, as well as insights gained from experience and best practice descriptions. The *data set saturation* technique was used to ensure completeness of the data set [194].

The *qualitative data analysis activities* were integrated into the identified applicable *research approaches* as indicated in Table 1.2.

	Methodological study	**Model-building study**
Data reduction	Collection of architectural related publications until dataset saturation was achieved.	Collection of Semantic Web and related publications until data set saturation was achieved.
Data display	Determination of the criteria list for layered architectures.	Determination of function and status of specific Semantic Web technologies as well as the layers of the layered architecture.

Conclusion drawing and verification	Establishing an evaluation mechanism for layered architectures and the calibration thereof using an accepted and established existing layered architecture.	The compilation of a Semantic Web status model. The adaption of the proposed Semantic Web layered architecture and its evaluation and application in case studies to determine its usefulness.

Table 1.2: Research design

1.9 PUBLICATIONS RESULTING FROM THE STUDY

The following peer-reviewed publications were generated as result of the research described in this thesis:

1. GERBER A.J., BARNARD A. AND VAN DER MERWE A.J., **Design and Evaluation Criteria for Layered Architectures.** In *Proceedings of the MSVVEIS Workshop hosted at the 8th International Conference on Enterprise Information Systems*, Paphos, Cyprus, (May 2006). ISBN 972-8865-49-8, pp. 163-172. [107]
2. GERBER A.J., BARNARD A. AND VAN DER MERWE A.J., **A Semantic Web Status Model.** In *Proceedings of the 9th World Conference on Integrated Design & Process Technology*, San Diego, California. IEEE (Jun 2006). ISSN 1090-9389. [108].
3. GERBER A.J., VAN DER MERWE A.J. AND BARNARD A., **Towards a Semantic Web Layered Architecture.** In *Proceedings of the International Conference on Software Engineering (IASTED SE2007)*, Innsbruck, Austria, (February 2007). ISBN: 978-0-88986-641-6, pp. 353-362 [110].

The following technical report was generated as a result of the investigation into Semantic Web technologies, and is included as an appendix in this thesis:

1. GERBER A.J., BARNARD A. AND VAN DER MERWE A.J., **Semantic**

Web Technologies. *Tech. Rep. UNISA-TR-2006-02*, UNISA (University of South Africa), (2006). [109].

1.10 OUTLINE OF THE STUDY

In order to answer the identified research questions and to perform the research investigations, this study is divided into four parts comprising ten chapters.

In Chapter 1, this chapter, an introduction to this thesis and the research questions, as well as a brief summary of other aspects related to this study, is presented. In the remainder of this section, the different parts of the thesis with their associated chapters are discussed.

1.10.1 Part I: Theoretical framework

Part I comprises the theoretical framework and therefore the theoretical underpinning of the study, and includes Chapters 2 and 3. The inclusion of Appendix A is optional since it contains a discussion of Semantic Web technologies.

In Chapter 2 the Semantic Web and its background and history, as well as other related aspects such as the current adoption status of the Semantic Web, are discussed. In Chapter 3 the different versions of the presently proposed Semantic Web architecture, as well as current initiatives of the W3C with regard to architecture, are investigated. In addition, the presently proposed versions of the Semantic Web layered architecture are scrutinised in order to support the research problem statement of the thesis.

1.10.2 Part II: Research design and execution

Part II includes the chapters contributing towards the research design and its execution. Part II comprises Chapters 4, 5, 6 and 7.

In Chapter 4, the research plan, approach and design are presented.

Within this chapter, the discussion specifically addresses aspects pertaining to the processes and data analysis activities required to perform the research dictated by the identified research questions. The execution of the research plan results in the research execution as documented in Chapters 5, 6 and 7.

In Chapter 5, the status and function of the Semantic Web technologies are investigated. The findings of this investigation is presented in a Semantic Web status model.

In Chapter 6 an evaluation mechanism, consisting of a criteria matrix for layered architectures, is developed. The research discussed in this chapter commences with an investigation into layered architectures, as well as architectures in general. From this investigation the criteria for the evaluation of layered architectures are extracted and an evaluation mechanism for layered architectures is compiled.

In Chapter 7 the existing versions of the layered architecture of the Semantic Web are evaluated using the established evaluation mechanism for layered architectures. In addition, an adapted and newly proposed architecture referred to as the CFL architecture (**C**omprehensive, **F**unctional and **L**ayered) is developed. The CFL architecture for the Semantic Web is developed by using the Semantic Web status model, the evaluation mechanism for layered architectures as well as additional identified design principles.

1.10.3 Part III: Evidence and discussion

Part III comprises Chapter 8. In Chapter 8 the proposed CFL architecture for the Semantic Web is applied to four case studies in order to establish the usefulness and validity of the architecture.

1.10.4 Part IV: Contribution and conclusion

Part IV comprises Chapters 9 and 10, and summarises the research contribution, places it within the context of the Computer Science and Information

Systems discipline and concludes the study. In Chapter 9, a summary of the research results of the thesis with reference to the research questions is provided and the specific contributions of the research to the scientific knowledge base are presented. In Chapter 10 the thesis is concluded with a discussion that includes aspects such as methodological reflections and recommendations.

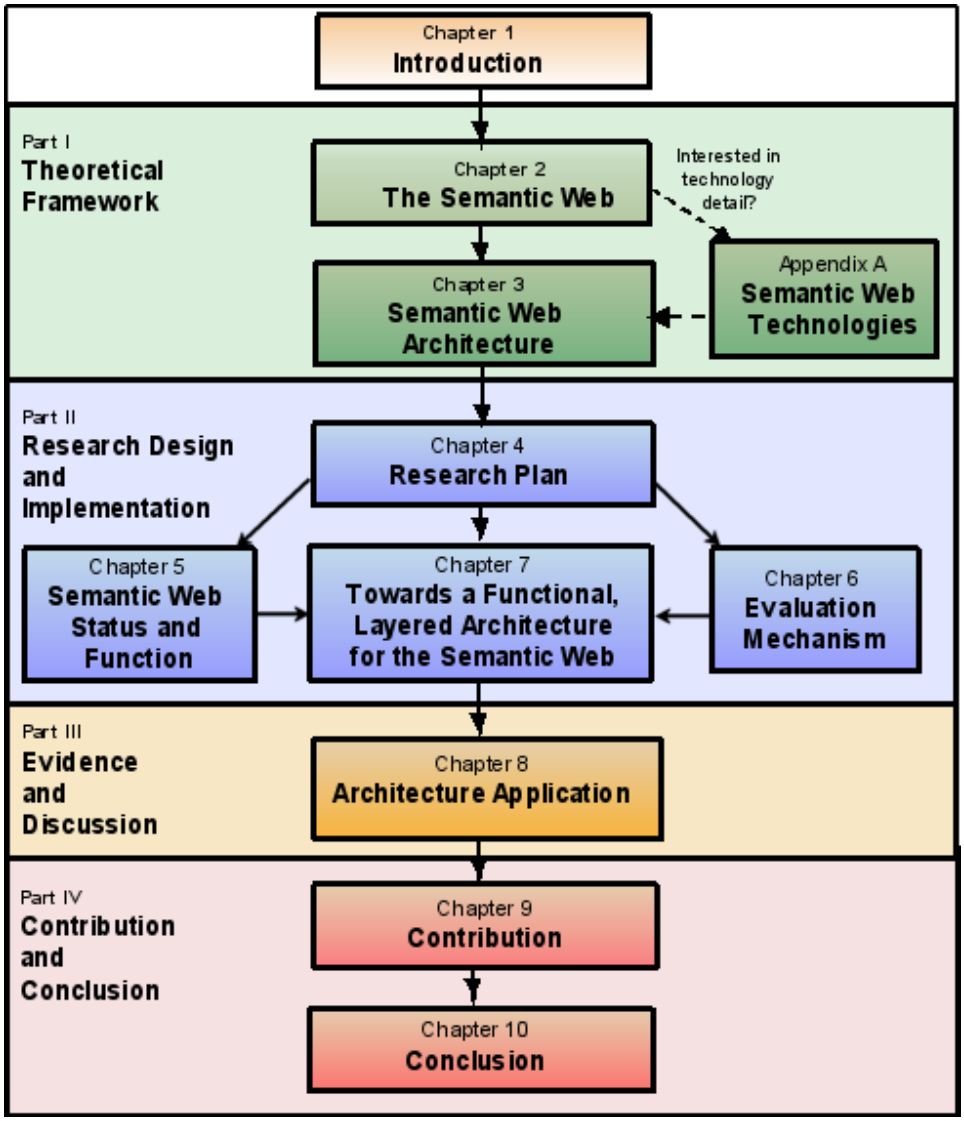


Figure 1.1: Thesis Chapter Layout

The structure of the study is graphically depicted as the thesis chapter

layout in Figure 1.1. The thesis layout is repeated throughout the thesis at the start of each chapter to assist the reader with orientation.

1.11 CONCLUSION

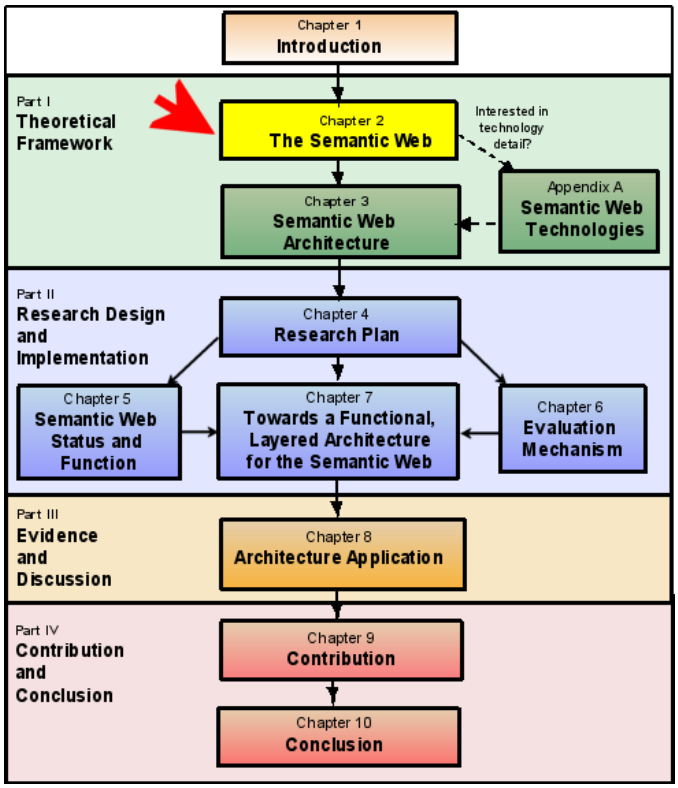
This chapter served as an introduction to this thesis. In particular it included a discussion of the background, the problem statement relating to and the purpose of the study. In addition, the research questions were proposed. Consequently, the rationale and scope of the study, potential contribution, research design, as well as a list of the research publications generated as result of the research contained in this thesis, were presented. In conclusion, a graphical representation of the thesis layout depicting the layout for the remainder of this thesis was presented.

Part I

THEORETICAL FRAMEWORK

CHAPTER 2

THE SEMANTIC WEB



Thesis Chapter Layout. Chapter 2 is the first chapter in Part 1:Theoretical Framework.

Chapter Contents

Figures

2.1 INTRODUCTION

The application domain for this study is the Semantic Web. The purpose of this chapter is to provide an introduction to the Semantic Web by addressing topics such as the history of the Semantic Web, the way the Semantic Web is defined and why the Semantic Web is important, both for present-day information systems and those of the future.

Semantic Web aspects addressed in this chapter include:

- Where did the Semantic Web originate?
- What is the Semantic Web?
- Why is the Semantic Web relevant?
- What is the current status of the Semantic Web?
- What is the current adoption status of the Semantic Web?
- What is the role of the W3C with regard to the Semantic Web?

In order to address these questions, the chapter commences with a discussion of the history of the Internet, World Wide Web (or Web for short) and Semantic Web in Section 2.2. Section 2.3 provides an overview of the Semantic Web as well as several definitions that serves as illustration of the diversity of technologies and applications embraced by the Semantic Web. This section also includes a definition for the Semantic Web compiled by making use of these existing definitions. For the purpose of this thesis, the definition for the Semantic Web as contained in Section 2.3.1, is adopted. The importance of the Semantic Web within current information systems is discussed in Section 2.4 by developing and discussing a behavioural model as well as a model for data exchange in information systems. Section 2.5 is devoted to a discussion of the status of the Semantic Web and the adoption of the Semantic Web within information systems.

The organisation that plays the most significant role towards the development of the Web and Semantic Web, is the W3C. An overview of the role and work of the W3C with regard to the Semantic Web is presented in Section 2.6. This chapter is concluded in Section 2.7.

2.2 HISTORY OF THE INTERNET, WEB AND SEMANTIC WEB

Research on packet switching at MIT (Massachusetts Institute of Technology) [170] and DARPA (Defence Advanced Research Projects Agency)[1] [80] in the early '70s is generally regarded as the inception of the Internet [118]. As a forerunner of the Internet, the initial successful demonstration of the ARPANET (Advanced Research Projects Agency Network) at the ICCC (International Computer Communication Conference) in October 1972 was presented in Washington [135, 162]. The ARPANET is generally regarded as the forerunner of the Internet. Established in 1969, ARPANET served as a test bed for new networking technologies, linking universities and research centres. The first two nodes that constituted the ARPANET were UCLA (University of California, Los Angeles) and the Stanford Research Institute, followed shortly thereafter by the University of Utah. Since this early demonstration the Internet and the Web evolved to become one of the primary communication mediums in the world today.

The Internet, together with various Web technologies, forms a widespread information infrastructure that is used by people across the world. The Web is considered an indispensable environment for any information or information technology worker, both to access or to publish information. According to latest survey of Lyman and Varian [152], it is estimated that 600 million people around the world have access to the Internet, and that by 2002 the Web contained approximately 170 terabytes of information on its surface [152, 189]. Surface information is web pages, generally available as `.html`-files (Hypertext Markup Language files). This does not include dynamic pages generated from databases or other systems. The information in 2002 on the Internet amounted to 532,897 terabytes, while print, film, magnetic, and optical storage media produced about 5 exabytes[2] [152]. Information flows through electronic channels

[1] DARPA is the central research and development organisation for the United States Department of Defence.

[2] If digitised with full formatting, the 17,000,000 books in the Library of Congress contain about 136 terabytes of information; five exabytes of information is equivalent in size to

(telephone, radio, TV, and the Internet) in 2002 amounted to almost 18 exabytes of information. During the same period, instant messaging generated 274 terabytes, and e-mail about 400,000 terabytes of information on the Internet [152].

In addition the Web is growing at an exponential rate as Web information is published on a daily base throughout the world, and the Web is regarded as the most popular information source and distribution medium available at present. It is plausible to state that, since its inception, the Internet and the associated Web technologies revolutionised the way in which our society generates, disseminates, accesses and uses information.

Berners-Lee [26] describes the difference between the Internet (he uses the form 'Net'), and the Web as follows:

> *The Web is an abstract (imaginary) space of information. On the Net, you find computers – on the Web, you find document [sic], sounds, videos,.... information. On the Net, the connections are cables between computers; on the Web, connections are hypertext links. The Web exists because of programs [sic] which communicate between computers on the Net. The Web could not be without the Net. The Web made the Net useful because people are really interested in information (not to mention knowledge and wisdom!) and don't really want to have [sic] know about computers and cables.*

The Internet was originally designed as an information space usable by both *humans and machines* [27]. However, the current *Web* is a publishing mechanism for information targeted mainly at *human* users.

As discussed, the popularity of the Web as a dissemination mechanism for human information resulted in its exponential growth [152, 191]. A consequence of this popularity is an overload of information available on almost any conceivable topic. This overload creates several new problems for users. For example, it is increasingly difficult to find applicable informa-

the information contained in 37,000 new libraries the size of the Library of Congress book collections.

tion or to verify the origin of information. In addition, it is difficult to verify that the information, or even the source of the information, is trustworthy [100, 157, 191].

In 2001, Berners-Lee, Hendler and Lassila [38] presented a vision of a Web that is an information space usable by, in particular, *machines*, thus coining the term *Semantic Web* for this envisioned *Web*. Instead of attempting to process and manipulate Web information, a user would have a personal *software agent* on his/her computer that would solve problems related to information overload, acquisition and discrepancy resolution [84]. The agent would execute the first level of information management and the user would only access or manipulate results.

The Semantic Web is at the time of writing mainly an international research effort with the goal to make Web content available for intelligent knowledge processing [59, 68, 83, 93, 98, 113, 156, 169, 179, 184, 204, 226]. The concept of the Semantic Web captured the imagination of Web users, as well as the interest of academia and industry even though several technological issues still have to be resolved.

2.3 WHAT IS THE SEMANTIC WEB?

As briefly discussed in the previous section, the Semantic Web is the name used to encapsulate the 2001 vision presented by Berners-Lee et al. [38] of a new Web as an information space usable by *machines* rather than *humans*. Web researchers realised that the rapid adoption of the Web and the associated information overload would necessitate alternative solutions and technologies where autonomous programs or *machines* assist humans to manage the information available on the Internet.

Several definitions and descriptions of the *Semantic* Web were published since its inception as practitioners and researchers adopted the notion of a Web semantically enriched [145]. Some of these definitions are provided below in order to highlight the diversity thereof. The list is by no means exhaustive.

Berners-Lee [27] originally described the Semantic Web as a mechanism that fulfils the goal of the original Web that was designed as an information space that is *useful, not only for human-human communication, but also for machines that would be able to participate and help their users*. The information available on the current Web is mostly designed for use by humans, even if this use is sometimes derived by means of a database with well-defined meanings and structure. Hence the available Web information is mostly not machine processable. In addition, Berners-Lee portrays the Semantic Web as an approach to develop languages for expressing information in a machine processable form so that migration is possible from the *Web of today to a Web in which machine reasoning will be ubiquitous and devastatingly powerful*. In addition, Berners-Lee describes the Semantic Web as a *Web of data*, similar to a *global database* [27].

Decker, Melnik, Van Harmelen, Fensel, Klein et al. [84] discuss the increased semantic interoperability that will be provided by the Semantic Web and that will enable intelligent services such as information brokers, search agents and information filters to execute complex tasks on behalf of their users.

Cherry [68] speculates means by which the Semantic Web makes the Web more homogeneous, more data-like and more amenable to computer understanding.

Grau [113] describes the Semantic Web as *an extension of the World Wide Web in which both data and its semantic definition can be processed by computer programs*. This author hypothesises that the next generation Web combines existing Web technologies with knowledge representation formalisms in order to provide an infrastructure allowing data to be processed, discovered and filtered more effectively. He portrays the Semantic Web as a set of languages organised as a layered architecture that allows users and applications to create and share information in a machine-readable manner.

Guha, McCool and Fikes [116] portray the central theme of the Semantic Web as software programs that are able to aggregate data easily from different sources, even though subtle differences in the *meaning* of the data

may exist.

Kalfoglou, Alani, Schorlemmer and Walton [145] identify semantics as the differentiating factor between the *Web* and the *Semantic Web*, and these semantics are provided by the W3C language specifications. In addition, the power of the Semantic Web will be realised when such software programs, also referred to as agents, collect Web content from diverse sources, process the information and exchange the results with other programs on behalf of its users. These software agents will become increasingly effective as more machine-readable Web content and automated services become available.

Baldoni, Baroglio and Henze [16] portrays the Semantic Web as a mechanism for content-aware navigation across different available resources in such a way that it is possible to identify those resources that best satisfy the user's requests. In addition, the Semantic Web is not seen only as an information provider but also as a service provider for sharing resources and services.

And recently, Berners-Lee [35] defined the Semantic Web as:

- a mechanism to assist with data interoperability across applications and organisations,
- a set of interoperable standards for knowledge exchange; and
- an architecture for interconnected communities and vocabularies.

From these different descriptions it is clear that there is a shift from a futuristic *new* Web with machine processable information, to more tangible portrayals such as a *set or architecture of standards* and a *mechanism for data interoperability*. In the most recent definition by Berners-Lee [35] the word 'Web' is not mentioned. It is plausible to speculate that this definition implies that the Semantic Web vision and the associated technologies have matured into more definitive concepts and applications. However, in an attempt to describe the Semantic Web, it is essential that the futuristic goals of the original vision should not be lost.

2.3.1 Semantic Web definition

From the definitions and discussions in the previous section, a definition for the Semantic Web for use in this thesis, is extracted.

The Semantic Web is -

1. a Web enriched with semantic meta-data that enables agents to execute complex information management tasks on behalf of its users,
2. a mechanism that contributes towards data, information and knowledge exchange and integration across communities and applications; and
3. a comprehensive architecture of meta-data language functionality that can be instantiated with different technology standards and specifications.

In (1) above, the founding vision of Berners-Lee, Hendler and Lassila [38] is acknowledged. In addition, it captures the departure point as well as the eventual goal of the Semantic Web endeavour.

In (2) the recent description of the Semantic Web by Berners-Lee [35] that focuses more on the present tangible applications of the Semantic Web, is acknowledged. With this second definition statement an attempt is made to capture the contemporary challenges with regard to the *information* in information systems, as well as the way the Semantic Web with its associated technologies is applied to solve present problems. The different technology solutions developed by Semantic Web researchers and practitioners (discussed in Chapters 3 and Appendix A), can assist on several levels with the challenges of data, information and knowledge exchange and integration.

In (3) the notion posed by Berners-Lee [35] that the Semantic Web is *an architecture for interconnected communities and vocabularies* is acknowledged and expanded. With this definition statement an attempt is made to capture the architectural structure of the Semantic Web. This definition supports the contribution of this thesis that develops a comprehensive and functional layered architecture for the Semantic Web. Such an architecture is essential and indispensable, and should be based on the fundamental

principles of architecture design in the Information Systems domain.

Software agents are often mentioned in connection with the Semantic Web [38, 66]. These agents are referred to as *intelligent programs* or the *machine processors* of an intelligent Semantic Web. In all cases, they are depicted as the eventual *users* of the Semantic Web. It is foreseen that agents will roam the Semantic Web, reason and make decisions on behalf of their users. This thesis excludes software agent technology from its scope as it is primarily concerned with the proposal of an architecture required to establish the meta-data functionality required for such agents to operate and function.

2.4 THE SIGNIFICANCE OF THE SEMANTIC WEB WITH ITS ASSOCIATED TECHNOLOGIES FOR MODERN INFORMATION SYSTEMS

In this section an argument is formulated that the Semantic Web is not only a futuristic vision, but is already prevalent in modern information systems. Semantic Web technologies could be significant for intelligent information management across diverse information sources as discussed in Section 2.4.1. In addition, modern Web applications require data exchange at different semantic levels. The Semantic Web with its associated technologies is already prevalent in these systems as will be discussed in Section 2.4.2.

2.4.1 The evolution of information systems

In this section the evolution in the behaviour of Information Systems is discussed. In addition, the discussion speculates on the role Semantic Web technologies could play towards integration across the heterogeneous information sources.

Figure 2.1 presents a model of an information system, which argues that an information system is not tangible. Any information system is only visible as a result of its behaviour. This behaviour can be detected in three

areas: (1) the *interface* the system displays towards targeted users, (2) the impact the system has on the *data* with which it interacts, and (3) the *physical* behaviour the system depicts when it consumes system resources during execution (such as memory or processing capacity).

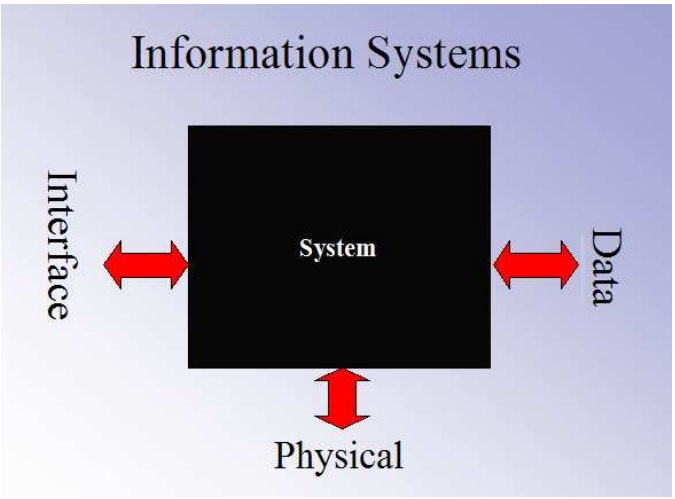


Figure 2.1: Information systems behaviour is detected in three areas: user interface, data manipulation and physical resource consumption.

On the physical dimension, modern information systems generally consume all the memory and processing power that is available from the hardware platform [196]. As systems evolve, they require resources and generally consume the resources that can be spared by the hardware platform. Therefore, when information system evolution is discussed, physical behaviour is summarised as *consuming the resources available from the platform* and *physical* behaviour depicted at the bottom of Figure 2.1 is hence omitted from Figure 2.2.

However, *interface* and *data* behaviour evolved by means of the inclusion of additional functionality and separation layers. In Figure 2.2, the model of an information system as presented in Figure 2.1 is expanded by means of three scenarios to depict an overview of the evolution of information systems with regard to *interface* and *data* behaviour.

Scenario 1 of Figure 2.2 depicts the original information system as de-

veloped by pioneers of programming such as Parnas and Dijkstra [89, 187]. In these types of information systems, data was usually integrated with the system, hard coded as part of the system, or the system used some kind of proprietary data storage mechanism. Interfaces were dedicated, simple, usually specialised towards the application and seldom even discussed. These information systems were often implemented to solve complex algorithms for dedicated applications.

Scenario 2 of Figure 2.2 presents the emergence of GUIs (Graphical User Interfaces) as the interface to information system functionality. These interfaces were standardised and included the specification of common mechanisms for access to object functionality such as *Windows* and *File Systems*. An example of such a mechanism is the closing of an interface *Window* with a click on the button with the cross in the top-right corner [173]. With regard to *data behaviour*, information systems evolved to incorporate databases. In these systems, the data is not integrated to be part of the system anymore, but it is separated from the system into a dedicated management environment such as an RDBMS (Relational Database Management System) [67]. Generally, access to both the GUI mechanisms and the databases was defined through standard *interface specifications*, which implies that other systems, for instance, can access the data in the database via the interface specification. An example of such an interface specification is the ODBC (*Open Database Connectivity*) specification [106].

Scenario 3 depicts modern information system behaviour. Standardised GUIs or other graphical interfaces are the norm, but various applications with multi-modal or ubiquitous computing interface requirements emerge. This is in part due to the prolific acceptance of cellular and mobile devices into society [149]. In addition, database systems are commonplace and because of the growth of the Web, diverse data sources such as HTML (Hypertext Markup Language) and XML (Extensible Markup Language) documents are used for data storage and retrieval. In addition, the emergence of service-base computing because of the interconnectivity provided by the Internet comes to the fore. These three sources of data are depicted on the right-hand side of Scenario 3 in Figure 2.2.

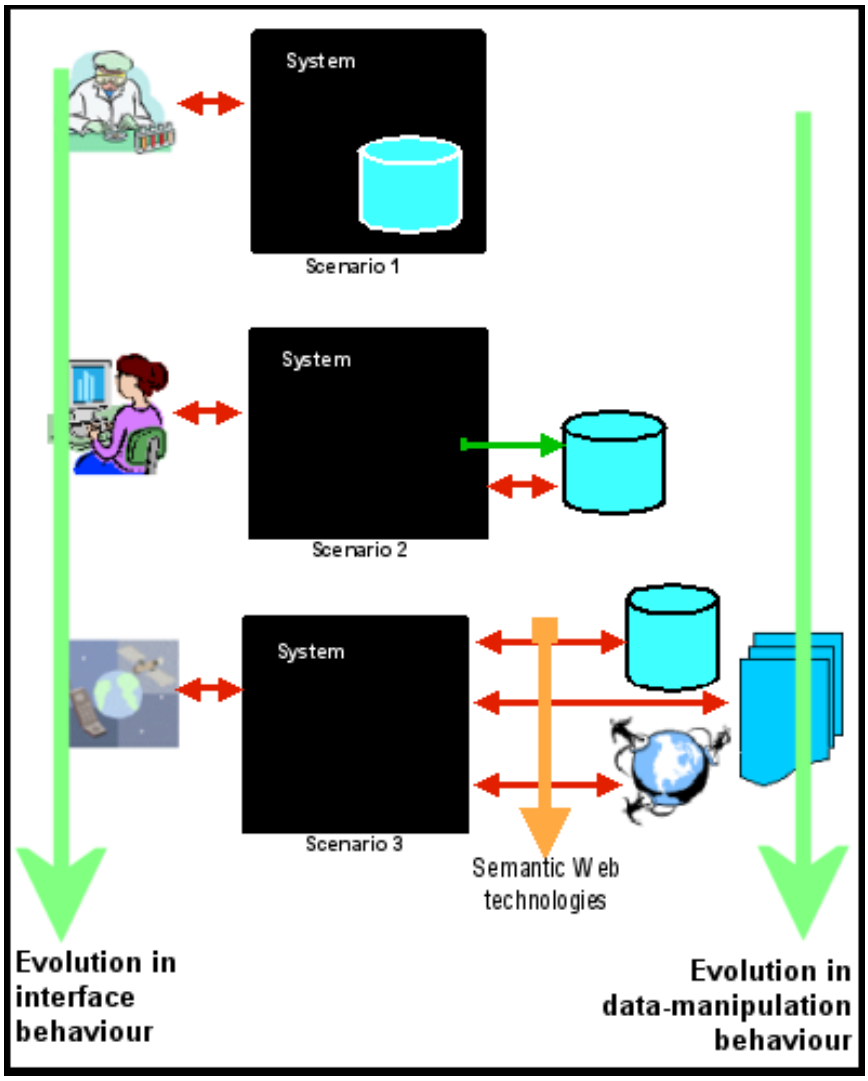


Figure 2.2: The evolution of information systems.

Such diverse data sources in modern information systems can possibly be integrated with the use of Semantic Web technologies as indicated by the orange arrow in Figure 2.2. Semantic Web technologies propose several means to integrate intelligently across diverse data sources. In addition, the Semantic Web can be applied to fulfil integration requirements at different semantic levels of *data*. More specifically, the different Semantic Web technologies can be used to integrate *data*, *information* or *knowledge*. The second definition statement of the Semantic Web definition of Section

2.3.1 on page 33 describes the Semantic Web as *a mechanism that contributes towards **data**, **information** and **knowledge** exchange and integration across communities and applications*. In order to expand this notion, it is necessary to distinguish between data, information and knowledge.

2.4.2 Data, information and knowledge exchange

2.4.2.1 Data, information and knowledge

Data is defined as structured or unstructured facts or symbols without any relations [23]. Data often occurs in the form of facts or figures obtained from experiments and stored in the form of numbers, text, images, and sounds suitable for processing [183].

Information is defined as *interpreted data* [9] or data that has been given meaning by way of a relational connection such as contained in a relational database. *Information* embodies the understanding of a relationship of some sort, possibly cause and effect such as *the temperature dropped 15 degrees and the water froze* [23]. Information is therefore definite data and facts acquired or supplied about something or somebody such as computer data that has been organised and presented in a systematic fashion to clarify the underlying meaning [183].

Knowledge is the collection of information in such a manner that it is useful. Knowledge is gained through a deterministic process and it represents a pattern that connects and generally provides a high level of predictability as to what is described or what will happen next [23]. Bellinger, Castro and Mills [23] propose that it would be knowledge to state that *the atmosphere is often unlikely to be able to hold the moisture so it rains if the humidity is very high and the temperature drops substantially*. Knowledge is therefore the general awareness or possession of information, facts, ideas, truths, or principles regarding a particular subject or situation. Knowledge is also defined as familiarity or understanding gained through experience or study [183].

Figure 2.3 (obtained from Bellinger et al. [23]) depicts the increasing

understanding and connectedness when migrating from data to knowledge and wisdom.

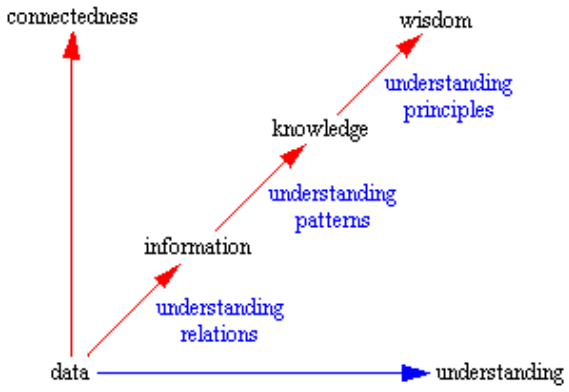


Figure 2.3: Data, information and knowledge [23].

Since the difference between the data exchange levels *data*, *information* and *knowledge* has been explained, the next section includes a discussion on how the Semantic Web can be used to facilitate exchange at the various levels.

2.4.2.2 The Semantic Web as facilitator for data, information and knowledge exchange

Figure 2.4 depicts three hypothetical applications that use the Semantic Web layered architecture of Berners-Lee [28] as a mechanism for **data**, **information** and **knowledge** exchange and integration.

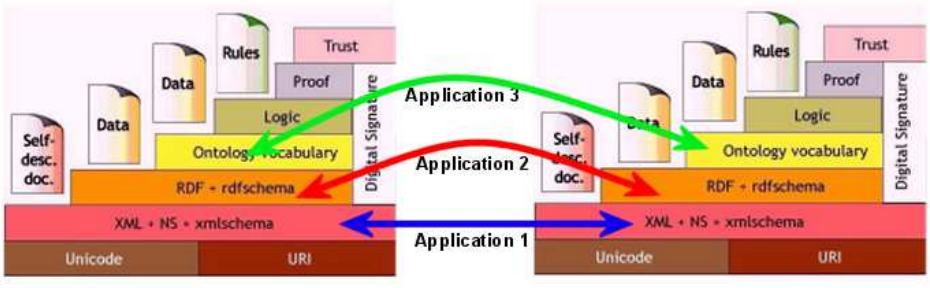


Figure 2.4: Integration across diverse data sources.

In Application 1 of Figure 2.4 two systems exchange data serialised with XML and thus contributes towards *data* exchange as per the Semantic Web definition[3]. RDF (Resource Descriptive Framework) provides a data model to depict the relations between resources[4], and in Application 2 in Figure 2.4 the two systems collaborate on the RDF level. These systems therefore exchange *information*. Application 3 operates on the ontology level. Ontologies capture the semantics of data or information and application 3 therefore exchanges *knowledge*.

The Semantic Web could thus assist with exchange of data, information and knowledge depending on the semantic requirements of the application by making use of various Semantic Web technologies.

2.4.3 Concluding remarks

In this section, mechanisms for the integration of the Semantic Web and its associated technologies in information systems were discussed. The Semantic Web is not only a futuristic vision as presented by Berners-Lee et al. [38], but in modern information systems the technologies can be used when any two systems exchange data, information or knowledge. As example, given the discussion about *data exchange*, the exchange of XML data that is commonplace in current Web system implementations, can be regarded as a simplistic Semantic Web implementation. In addition, certain Semantic Web technologies are thus already integrated to some extent into certain present-day information systems through the adoption of its technologies. It would benefit system developers to take cognisance of a Semantic Web architecture as it would assist with the adoption of additional technologies for intelligent information integration in modern information systems.

[3](Refer to Section A.4 for a discussion of XML and XML Schema)

[4](Refer to Section A.5 for a discussion of RDF and RDF Schema)

2.5 THE ADOPTION STATUS OF THE SEMANTIC WEB

The main aim of the Semantic Web is to extend the current human-readable Web in order to accommodate machine-processability [27]. This is done by encoding some of the semantics of resources using specified formalisms. Moving beyond syntax will allow more advanced applications and functionality to reside on the Web [68, 84].

Through mainly the endeavours of the W3C, the core technological building blocks are seen to be in place and available [167]. With the final acceptance of RDF and OWL (Web Ontology Language) as W3C Recommendations in February 2004, the W3C announced that the Semantic Web emerged as commercial-grade infrastructure for sharing data on the Web [244]. Before the release of these two technologies, the Semantic Web was regarded mainly as a research effort, moreover, an active effort.

The Semantic Web is an active discussion, research and development topic at present. An indication of this interest by researchers may be presented by conducting a search on Google Scholar [112] for 'Semantic Web', which returned approximately 204,000 hits. Google Scholar provides a search engine for scholarly literature. This is noteworthy since the vision of the Semantic Web was only conceptualised in 2000. A search on Google [111] for 'Semantic Web' returned 32,500,000 hits. This result indicates that there is a substantial general interest in the Semantic Web as well.

In addition to Web publications, several noteworthy dedicated conferences address the Semantic Web. The fifth International Semantic Web Conference (ISWC06) [141], as well as the third European Semantic Web Conference (ESWC06) [97] were held in 2006. The WWW2006 introduced a refereed Semantic Web track [260], SemTech2006 was the second Semantic Web Technology Conference [207], and the SWWS'06 (Semantic Web and Web Services 2006) international conference was held simultaneously with WORLDCOMP'06, which is considered one of the largest annual gatherings of computer scientists [219]. The growth in both the

number of recent dedicated Semantic Web conferences and attendance of these serve as an indication of academic interest.

It is plausible to state that, far from being just an academic research initiative, the Semantic Web has captured the interest and imagination of a significant number of Web users. Even though there is a substantial interest in the Semantic Web as indicated in the previous discussion, it is necessary to investigate the actual adoption of related Semantic Web technologies to determine the current status of the Semantic Web.

The adoption of Semantic Web technologies is diverse. Recent examples include SAWA (a Situation Awareness Assistant), described by Matheus, Kokar, Baclawski and Letkowski [154]. SAWA is based on Semantic Web technologies that facilitate the development of user-defined domain knowledge in the form of formal ontologies and rule sets that permits the application of relevant domain knowledge to the monitoring of relevant relations as they occur insitu. In another example, Oberle, Staab, Studer and Volz [180] developed an infrastructure that facilitates plug'n'play engineering of ontology-based modules and thus, the development and maintenance of comprehensive Semantic Web applications.

In order to support the development of Semantic Web applications, the Semantic Web Challenge [206] was initiated in 2003. The Challenge is regarded as a showcase of the most significant Semantic Web technology adoptions. The Challenge offers participants the opportunity to present relevant and significant Semantic Web applications. The following quote from the Semantic Web Challenge suffices [167]:

> *The overall objective of the challenge is to apply Semantic Web techniques in building online end-user applications that integrate, combine and deduce information needed to assist users in performing tasks. Intentionally, the challenge does not define specific task, data set, application domain or technology to be used because the potential applicability of the Semantic Web is very broad. Instead, a number of minimal criteria have been defined which allow people to submit a broad range of applications. In addition to the criteria, a number of specific de-*

> *sires have been formulated. The more desires are met by the application, the higher the score will be. The Semantic Web Challenge Advisory board also defines an additional goal every year based on the development of the Challenge.*

In order to illustrate the type of applications that have been implemented for the Challenge using Semantic Web technologies, the definition of a Semantic Web application for the Challenge is provided. An application has to meet the following minimal requirements [167]:

- The meaning of data has to play a central role.
  - Meaning must be represented using formal descriptions,
  - Data must be manipulated/processed to derive useful information; and
  - The semantic information processing has to play a central role in achieving things that alternative technologies cannot do as well.
- Information sources used by the application should:
  - have diverse ownerships (i.e. there is no control of evolution),
  - be heterogeneous (syntactically, structurally, and semantically); and
  - contain real world data, i.e.shoule be more than imaginary examples.
- The application must assume an open world, i.e. it should assume that the information is never complete.

The winner of Challenge 2003 was CS AKTive Space implemented by the University of Southampton in the UK [203]. CS AKTive Space is a Semantic Web application that manages content relating to Computer Science research in the UK. The content is distributed and semantically heterogeneous. The content is gathered on a continuous basis using a variety of methods. CS AKTive Space supports the exploration of patterns and implications inherent in the content and exploits a variety of visualisations and multi-dimensional representations.

In 2004, the winner was Flink [103] of the Vrije Universiteit Amsterdam in the Netherlands. Flink is a unique application as it is the result of the effort of a single Ph.D. student opposed to products of large and well-funded

EU or DARPA projects. The question Flink intends to answer is whether it is possible to develop, with minimal effort, an engaging and cutting-edge Semantic Web application from the different building blocks available as open source products [166]. Flink integrates the information sources of the traditional Web (HTML pages) with those of the Semantic Web FOAF (Friend of a Friend) profiles [167].

In 2005, the Challenge chose CONFOTO as winner [70]. CONFOTO was implemented by Appmosphere Web Applications in Germany and uses recent Web as well as Semantic Web technologies to create a browsing and annotation service for conference photos. In CONFOTO, recent Web trends such as tag-based categorisation, interactive user interfaces and syndication are combined with Semantic Web platforms and technologies. In addition, CONFOTO offers tools to annotate and browse pictures. CONFOTO also includes a tailored photo browser and gallery generator for pictures [167].

The applications chosen as winners of the different Semantic Web Challenges are integrated technology demonstrators depicting some maturity regarding the tools and technologies used. It is therefore plausible to speculate that, even though the Semantic Web is at present still mainly a research effort, it is on the brink of moving beyond the realm of research. The emergence of useful applications and technology demonstrators such as those developed by the contestants of the Semantic Web Challenge will assist with this progress.

2.6 THE W3C

Any discussion about the Semantic Web should include a description of the role and initiatives of the W3C. The W3C is dedicated to the futuristic vision of an interoperable Web that can accommodate the growing diversity of people, hardware, and software in the world [250]. The mission statement of the W3C is given as:

To lead the World Wide Web to its full potential by developing

> *protocols and guidelines that ensure long-term growth for the Web* [250].

The W3C views the establishment of Web standards and guidelines as the primary mechanism to pursue its mission. Standards allow accepted Web technologies to be compatible and to collaborate. The W3C refer to this as *Web interoperability*, a fundamental requirement for adoption of the Web. Since 1994, ninety W3C Recommendations (the W3C terminology used to indicate *standards*) were released by means of an established collaborative process. The W3C serves as open forum for discussions about all issues related to the Web and Semantic Web. In addition, the W3C supports several educational initiatives on the W3C mission, activities and Recommendations. W3C's global initiatives also include continuing liaisons with regional, national and international organisations in order to foster global participation in the development of the Web and its associated standards.

W3C operations are supported by a combination of membership dues, research grants, and other sources of public and private funding. W3C operations are jointly administered by MIT CSAIL (the MIT Computer Science and Artificial Intelligence Laboratory) in the USA, ERCIM (the European Research Consortium for Informatics and Mathematics) headquartered in France, as well as Keio University in Japan. W3C also maintains established World Offices around the world [250].

The W3C's work is arranged into domains, consisting of specific activities. Currently W3C defines four domains: (1) Architecture, (2) Interaction, (3) Technology and Society, and (4) Web Accessibility Initiative. Within these domains, there are twenty-three activities and fifty-three groups divided into (1) thirty-five working groups, (2) six coordination groups, and (3) twelve interest groups. The domains, activities and groups are supported by the *Technical Architecture Group* and the *Advisory Board* [250].

The Semantic Web Activity is part of the Technology and Society Domain. This activity focuses on developing enabling technologies, as well as exploring areas of advanced development to facilitate deployment and adoption of the Semantic Web [169].

W3C conducts its work mainly on its Web site[5] and the site is organised along three dimensions:

- **W3C Activities**: Each W3C technology belongs to an Activity, and each Activity has a home page with links to related specifications, tutorials, and news.
- **W3C Recommendations**: The core work of W3C appears in the index of specifications called technical reports, which are at various levels of development. Those specifications that passed through the W3C consensus process are elevated to the status of a *W3C Recommendation*, which are considered to be Web standards.
- **W3C Mailing lists**: The W3C Mailing lists are used to foster debate regarding technology issues between users, technology developers and implementers. Mailing lists often constitute a helpful platform that facilitates technical assistance acquisition and exchange.

As stated, the Semantic Web Activity are responsible for the management of groups and initiatives concerning the Semantic Web [232]. This Activity was launched in February 2001. Originally, the Semantic Web Activity was chartered to support the RDF and RDF Schema and to advance both these technology recommendations. In addition, the activity formed a Web Ontology Working Group in November 2001 that was responsible for an ontology language, the result of which was OWL.

The following groups are at present charted and part of the Semantic Web Activity:

- The **Semantic Web Coordination Group** is tasked to provide a forum for managing the interrelationships and interdependencies among groups focusing on standards and technologies that relate to the goals of the Semantic Web Activity. This group is designed to coordinate, facilitate and (where possible) to assist with shaping the efforts of other related groups to avoid duplication of effort and fragmentation of the Semantic Web by way of incompatible standards and technologies.

[5] W3C maintains more than one million Web pages in the www.w3.org domain plus more than another million pages of mailing list archives at lists.w3.org

- The focus of the **RDF Data Access Working Group** is to evaluate the requirements for a query language and network protocol for RDF and already defined formal specifications and test cases for supporting such requirements.
- The **Rules Interchange Working Group** is chartered to produce a core rule language together with extensions that collectively allow rules to be translated between rule languages, thus facilitating transfer between rule systems.
- The mission of the **Gleaning Resource Descriptions from Dialects of Languages Working Group** is to complement the concrete RDF/XML syntax with a mechanism to relate other XML syntaxes to the RDF abstract syntax via transformations identified by URIs (Uniform Resource Identifiers).
- The **Semantic Web Deployment Working Group** aims to provide guidance in the form of W3C Technical Reports on issues of practical RDF development and deployment practices in the areas of publishing vocabularies, OWL usage, and integrating RDF with HTML documents.
- The **Semantic Web Interest Group** is a forum for W3C members and non-members to discuss innovative applications of the Semantic Web. The interest group also initiates discussion on potential future work items related to enabling technologies that support the Semantic Web, and the relationship of that work to other activities of W3C and to the broader social and legal context within which the Web is situated.
- The **Semantic Web Health Care and Life Sciences Interest Group** is designed to improve collaboration, research and development, and innovation adoption in the health care and life science industries. Aiding decision-making in clinical research, Semantic Web technologies will bridge many forms of biological and medical information across institutions.
- The **Semantic Web Education and Outreach Interest Group (SWEO)** is chartered to collect proof-of-concept business cases, demonstration prototypes, etc, based on successful implementations of Semantic Web technologies, to collect user experiences, to de-

velop and facilitate community outreach strategies, as well as to maintain training and educational resources. The goal includes bringing together leaders in organisations using and/or interested in applying Semantic Web technologies to the enterprise. The focus of the group is informational by nature, rather than providing technical specifications.

Standards that promote the Semantic Web are released through the W3C and this section provided the reader with a short overview of the W3C mission, goals and structure. The W3C formally manages Semantic Web technology specifications and promotes the Semantic Web with its associated technologies through the various initiatives as listed.

2.7 CONCLUSION

This chapter comprised an overview of the Semantic Web in general, including its history, various definitions, present status, adoption status and the Semantic Web initiatives of the W3C. A detailed discussion of technological specificities, detail information and status of the different technologies comprising the Semantic Web is presented in Appendix A.

In addition to the overview, a definition of the Semantic Web for the purpose of this thesis was compiled. The Semantic Web is:

1. a Web enriched with semantic meta-data that enable agents to execute complex information management tasks on behalf of their users,
2. a mechanism that contributes towards data, information and knowledge exchange and integration across communities and applications; and
3. a comprehensive architecture of meta-data language functionality that can be instantiated with different technology standards and specifications.

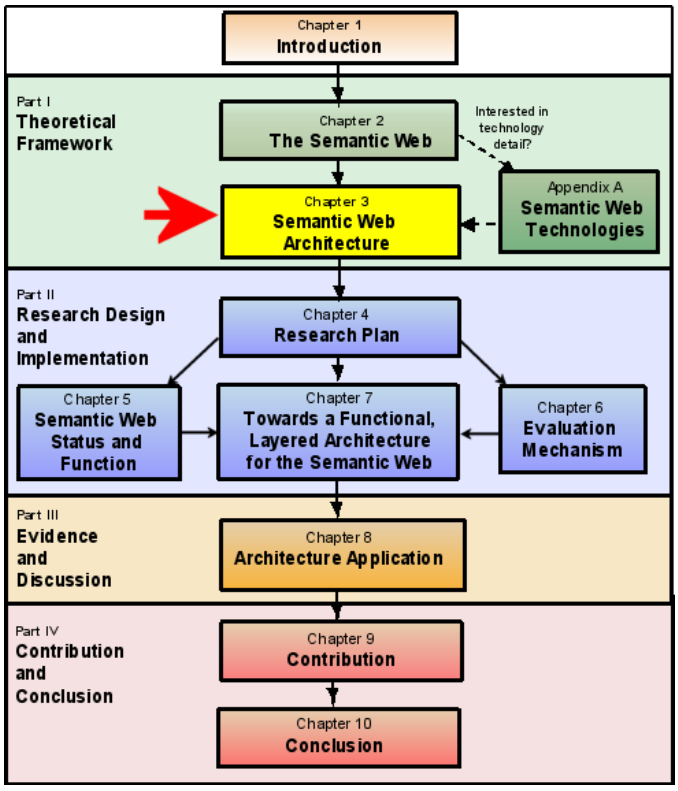
The remainder of this chapter expanded the various points of the definition, including a discussion on information systems evolution and the differences between data, information and knowledge. In addition, the current

interest and adoption of Semantic Web technologies were discussed. The initiatives of the W3C with regard to the Semantic Web were summarised in a subsequent section.

In the next chapter, the different versions of the Semantic Web architecture are discussed.

CHAPTER 3

SEMANTIC WEB ARCHITECTURE



Thesis Chapter Layout

Chapter Contents

Figures

3.1 INTRODUCTION

The purpose of this study is the development of a comprehensive, functional layered architecture for the Semantic Web. This chapter aims to provide context for the study by discussing presently proposed versions of the Semantic Web architecture as well as formal architectural initiatives. The questions that are of concern in this chapter, are:

- ▷ Which versions of a Semantic Web architecture were released?
- ▷ What is the adoption status of the Semantic Web architecture?
- ▷ Are there architectural initiatives that might have an impact on the Semantic Web architecture?

Since the publication of the original Semantic Web vision of Berners-Lee, Hendler and Lassila [38], Berners-Lee proposed several versions of a Semantic Web architecture that are discussed in Section 3.2 in this chapter. In order to support the research reported on in this study, Section 3.3 reflects on the meaning and structure of these proposed versions of the Semantic Web architecture.

In addition, in Section 3.4 an investigation is conducted into formal architectural initiatives that might address the issues associated with an architecture for the Semantic Web. These initiatives include the W3C TAG (Technical Architecture Group) within the W3C Architecture Domain and the development of Web architecture strategy as formulated in the W3C *Architecture of the World Wide Web, Volume 1* Recommendation.

As result of the discussions contained in Sections 3.3 and 3.4, Section 3.5 presents an argument supporting the research problem of this thesis, namely that any discussion or adoption of the Semantic Web or its associated technologies is precluded by a well-defined, comprehensive and functional layered architecture for the Semantic Web that is specified in an unambiguous manner. This chapter is concluded in Section 3.6.

3.2 SEMANTIC WEB ARCHITECTURE MODELS

Tim Berners-Lee proposed four versions of the Semantic Web architecture over the past six years. The first version was released in 2000 [28] whilst the most recent version was released in July 2006 [35]. In these versions of the Semantic Web architecture, semantic language functionalities and technologies are layered into an increasingly expressive stack [28, 31, 34, 35]. In Section 3.2.1 an argument is formulated that the proposed Semantic Web architecture may indeed be considered as an *architecture*. In addition, the four proposed versions of the Semantic Web architecture are discussed in sections 3.2.2 to 3.2.5.

3.2.1 The Semantic Web architecture as *architecture*

Often some researchers point out that the existing versions of the Semantic Web architecture cannot be described as architectures since Semantic Web *languages* or *W3C technologies* are depicted and that this is not the nature of an architecture. They argue that the term *architecture* is generally used to depict system functionality at different conceptual levels, and that the proposed versions of the Semantic Web architecture does not depict functionality and should be described as a *stack* or even *layered cake* [6, 123].

In order to respond to this argument, a definition for the term *architecture* is compiled. There is general consensus in literature that an architecture *at least* depicts the structure of a system within a specific context [18, 105]. This structure should portray the components that a system comprise of, as well as the relationships between components. System architects depicting the organisation of system components often make use of several identified architectural patterns, one of which is the *layered architecture* [18].

In the context of the *languages required for meta-data specification*, the proposed versions of the architecture of Berners-Lee [28, 31, 34, 35] depict the organisation of the *language* components for the Semantic Web. The layering of the languages provides the *structure*. Thus, the general

definition of an *architecture* is adhered to. The fact that *functionality* is not always depicted on layers within the current versions should be regarded as an omission. This omission provides one of the motivations for the development of a *functional* architecture for the Semantic Web in this study.

In conclusion of the proposition that the presently proposed Semantic Web layered architecture versions are regarded as architectures, an example of a similar, widely disseminated architecture is presented. The ISO/OSI (International Standards Organisation / Open Systems Interconnect) layered architecture specifies the functionality required to define *protocols* necessary for *network interoperability* between applications [265]. The Semantic Web layered architecture has purpose similar to that of the the ISO/OSI layered architecture in that it aims to depict the *languages* necessary for *data interoperability* between applications. The ISO/OSI model is regarded as an *architecture*, and thus it is proposed that the Semantic Web model should be regarded in the same way as an *architecture* for the purpose of this study.

3.2.2 V1: The first version of the layered architecture

In support of the founding vision of the Semantic Web by Berners-Lee et al. [38], Berners-Lee introduced a first version of a layered architecture for the Semantic Web [28]. For the purposes of the discussions in this study, this will be labelled **V1**. This version, or the *V1 Semantic Web layered architecture*, is depicted in Figure 3.1.

In order to understand what is depicted by this architecture, the circumstances and status of the technologies at the time of its release are presented. During the time of the inception of this version of the architecture, the W3C[1] initiated the Semantic Web Activity with two working groups for RDF and an ontology vocabulary respectively[2]. XML with its associated Semantic Web technologies (XML Schema and Namespaces)[3] as indicated by the V1 architecture, were adopted as recommendations by the

[1] Section 2.6, p.44

[2] Sections A.5.1, p.316 and A.6.1, p327.

[3] Section A.4, p.292.

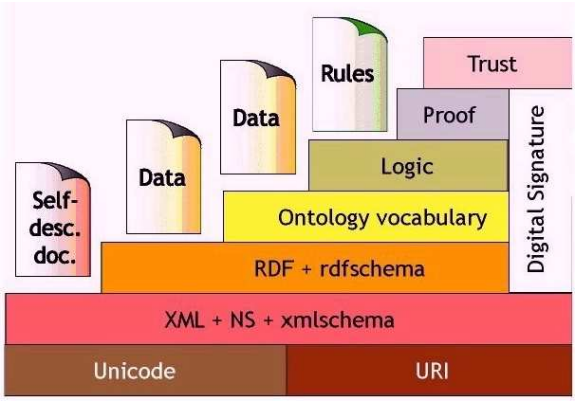


Figure 3.1: The Semantic Web architecture (V1) proposed by Berners-Lee in December 2000 [28].

W3C. Similarly, URI as RFC2396 [225][4] was adopted in 1998 by the IETF [137]. In addition, several releases of Unicode Version 3[5] were released in 2000. For the purpose of this discussion, these W3C technologies will be classified as *established* technologies.

In V1 in Figure 3.1, Berners-Lee [28] depicts all these established technologies on different layers. In addition, Berners-Lee denotes *Logic*, *Proof* and *Trust* as separate layers building on top of the established technology layers. *Digital Signatures* is depicted as a vertical layer starting on top of *XML+NS+XML Schema*.

Furthermore, in Figure 3.1, Berners-Lee [28] adds descriptive notes labelling *XML+ NS+ XML Schema* as *Self describing documents*, *RDF+rdfschema* and *Ontology vocabulary* as *Data*, and *Logic* as *Rules*.

3.2.2.1 Adoption status of the V1 version of the Semantic Web layered architecture

The V1 version of the Semantic Web architecture was adopted by several Semantic Web authors as a relevant Semantic Web architecture. Shortly after the release of this architecture it was used by Fensel [99] in 2000

[4] Section A.3.2, p.290.
[5] Section A.3.1, p.288.

and by Fensel, van Harmelen, Horrocks, McGuinness and Patel-Schneider [102] in 2001 to describe the relationship of DAML-ONT[6] and OIL[7] with supporting lower layers, specifically RDF Schema. With the development of OWL[8] and the identification of several layering issues, Fensel [100] subsequently in 2002 adapted the architecture to reflect the status quo with regard to W3C Recommendations. This adapted architecture is depicted in Figure 3.2.

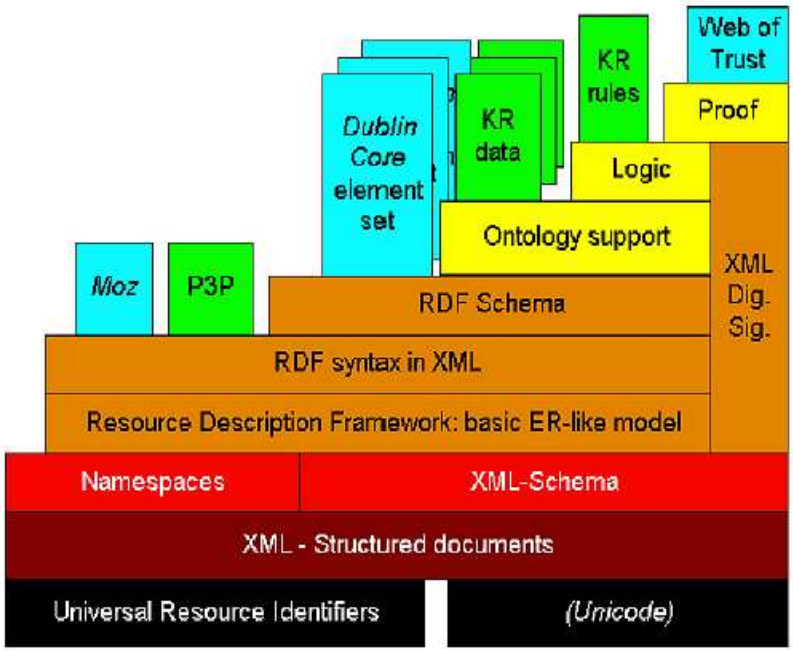


Figure 3.2: The adapted layer language model of Fensel [100].

In addition to these initial adoptions, Cost, Finin, Joshi, Peng, Nicholas et al. [74] explain the relationship of DAML+OIL[9] with the Semantic Web using the V1 architecture, and Hendler [123] describes the interaction of

[6]DAML-ONT is an ontology language that was developed by DARPA and released in 2000.

[7]OIL was developed as an ontology language in 2000 by On-to-Knowledge, an European IST project

[8]Section A.6.1.1, p.329.

[9]DAML-ONT, OIL and DAML+OIL are all ontology languages that were eventually integrated into the OWL W3C specifications. Refer to Section A.6.1, p.327.

agents with the Semantic Web in the context of the V1 architecture. Horrocks and Patel-Schneider [131] refers to the V1 architecture when they introduce their *three theses of presentation in the Semantic Web*. They define *presentation* as the relationship between constructs in a language and entities in the world. In addition, Patel-Schneider and Fensel [191] highlight several issues with regard to the layering of the Semantic Web architecture, especially with regard to the ontology vocabulary, by using the V1 architecture as departure point. Shah, Finin, Joshi, Cost and Mayfield [211] use the layered architecture to describe the background in their paper discussing information retrieval on the Semantic Web.

To introduce the Semantic Web, Antoniou and von Harmelen [6] adopts the V1 architecture for a discussion on the Semantic Web in their *Semantic Web Primer*. In addition, Huang and Webster [134] adapt the V1 architecture in order to create a context-aware approach to enable agents to understand semantic resources. Baldoni, Baroglio and Henze [16] discuss personalisation of the Semantic Web by referring to the layers of this version of the architecture. Oberle, Staab, Studer and Volz [180] introduce the development of an application server for the Semantic Web, particularly the KAON SERVER. They use this V1 version of the architecture to indicate the *static* parts of the Semantic Web.

Although the V1 Semantic Web architecture was adopted by noteworthy Semantic Web authors such as Hendler, Horrocks, Patel-Schneider and Fensel [123, 131, 191], it is plausible to argue that its acceptance as a reference architecture for the Semantic Web is not as widespread as expected, given the number of academic and popular publications in the domain. One may speculate that this is due to the fact that the architecture was never presented formally in literature or as part of a W3C Recommendation. All of the Semantic Web architecture versions were presented by Berners-Lee in presentations. However, in spite of this, no other architecture could be found in literature that can may currently serve as a comprehensive and functional model for the language specifications of the Semantic Web. It remains problematic that neither a description, nor unambiguous meaning is associated with the V1 version graphical representation of the Semantic Web architecture.

3.2.3 V2: The second version of the layered architecture

As participant in the ongoing activities of the W3C, Berners-Lee proposed a second version of the Semantic Web architecture in 2003 as part of a presentation at the SIIA (Software & Information Industry) Summit [212]. For the purpose of this study, this will be labelled the V2 version of the Semantic Web layered architecture [32]. This V2 version of the architecture was furthermore presented as part of two presentations in 2003 [31, 33] and is depicted in Figure 3.3.

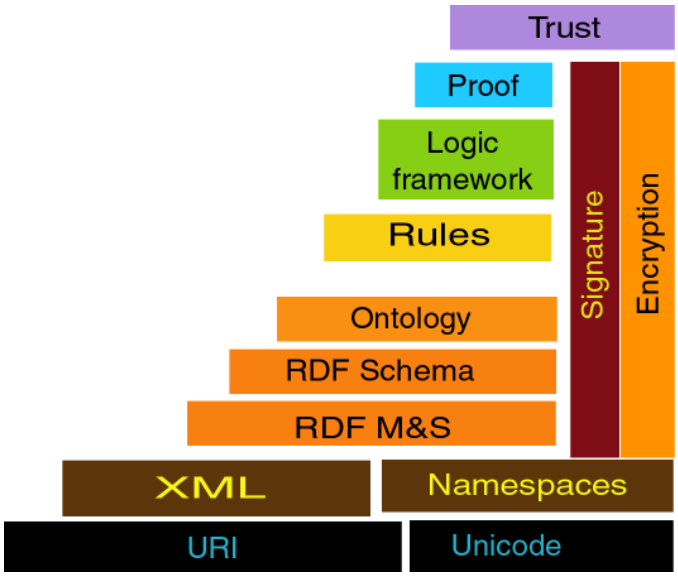


Figure 3.3: The subsequent Semantic Web architecture (V2) [31–33].

The V2 version of the architecture (Figure 3.3) is an adaptation of the V1 version as presented in Figure 3.1. URI and Unicode are still depicted on the bottom layer. However, *XML* and *Namespaces* are presented separately as layer 2 rather than as the combined *XML+NS+XML Schema* layer in the original version. The *RDF+rdfschema* layer in the original version is divided into two layers, *RDF M&S* and *RDF Schema* that build on one another. *Ontology* replaces *Ontology vocabulary* and *Logic framework* replaces *Logic*. Furthermore, *Logic framework*, *Proof* and *Trust* are still depicted as the upper layers. Another difference is the absence of the descriptive notes of the first architecture. *Rules* depicted as a note previously,

is now part of the architecture and resides as a layer above *Ontology*.

Furthermore, in Figure 3.1, Berners-Lee depicts *Digital Signatures* as a vertical layer on top of the *XML+NS+XML Schema* layer. In contrast, Figure 3.3 depicts two vertical layers, *Signature* and *Encryption*, which reside above the *Namespaces* layer.

Neither the precise meaning of the V2 architecture, nor the implications of the adaptations are discussed by Berners-Lee in his presentations [31–33]. A discussion could also not be found in other literature.

3.2.3.1 Adoption status of the V2 version of the Semantic Web layered architecture

After the release of the V2 version in 2003, a tendency remained to refer to the first version [6, 16, 130, 134, 180]. Patel-Schneider [190] refer to both versions of the architecture when he proposes an adapted architecture less tied to RDF. In Patel-Schneider's revised architecture different Semantic Web languages can have different syntaxes but use the same models [190].

Horrocks, Parsia, Patel-Schneider and Hendler [130] adopt both the second (V2) and subsequent third (V3) versions when they present different proposals for extending the Semantic Web architecture with a rules component. Similarly, Kifer, Bruijn, Boley and Fensel [147] refer to the V2 version of the architecture when they argue that a realistic architecture for the Semantic Web must be based on multiple independent, but interoperable, stacks of languages. In particular, they incorporate certain rule-based languages, which cannot be layered on top of OWL. These authors therefore adapt the V2 version of the architecture to include these languages in the Semantic Web architecture alongside with the stack of OWL-based languages [147].

However, from the preceding discussion, one can deduce that the V2 version of the Semantic Web architecture was not adopted by Semantic Web authors to the same extent as the V1 version.

3.2.4 V3: The third version of the layered architecture

Berners-Lee proposed a third version of the Semantic Web architecture in his keynote presentation at the WWW'2005 (World Wide Web Conference) [34]. In this study, this version depicted in Figure 3.4, is referred to as the V3 version of the Semantic Web layered architecture.

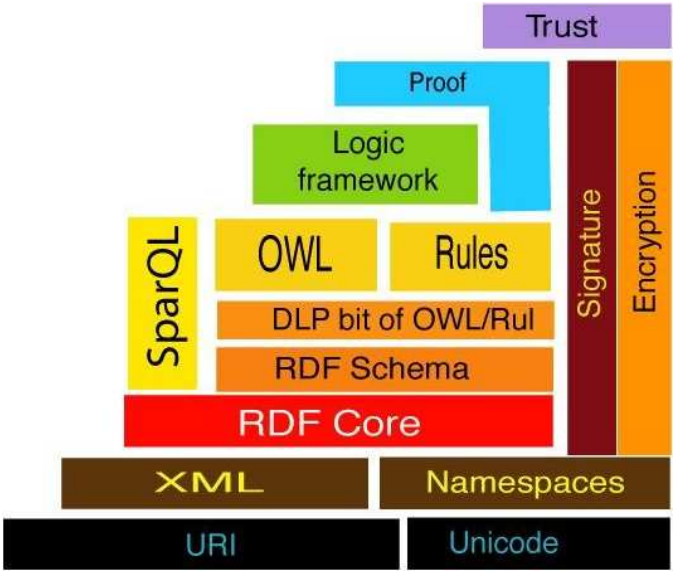


Figure 3.4: The V3 version of the Semantic Web architecture [34].

The V3 version of the architecture (Figure 3.4) extends the V2 version of Figure 3.3. *URI* and *Unicode*, as well as *XML* and *Namespaces* are presented similarly to version 2. *RDF M&S* are labelled *RDF Core* and a layer *DLP bit of OWL/Rul* is added above *RDF Schema*. The *Ontology* and *Rules* layers in V2 are presented on one layer in a side-by-side manner as *OWL* and *Rules* in V3. The *Proof* layer is also extended down to reside next to *Logic Framework*. *Trust* is still depicted as the upper layer. The vertical layers remain unchanged from those in V2.

As is the case with the prior versions of the Semantic Web architecture, neither the meaning nor the implications of the adaptations depicted in the V3 Semantic Web architecture, are discussed in literature.

3.2.4.1 Adoption status of the V3 version of the Semantic Web layered architecture

The only adoption of the V3 version of the Semantic Web architecture that could be found in literature, is the adoption by Horrocks et al. [130] of both V2 and V3 (discussed previously in Section 3.2.3.1) where the authors propose extensions to the architecture that includes a rules component.

3.2.5 V4: The latest version of a layered architecture for the Semantic Web

In his keynote address at the AAAI 2006 Conference (Twenty-First National Conference on Artificial Intelligence) in July 2006, Berners-Lee introduced the latest version of the Semantic Web architecture [35]. For the purposes of this thesis this version is referred to as the V4 version of the Semantic Web layered architecture. The V4 version is depicted in Figure 3.5.

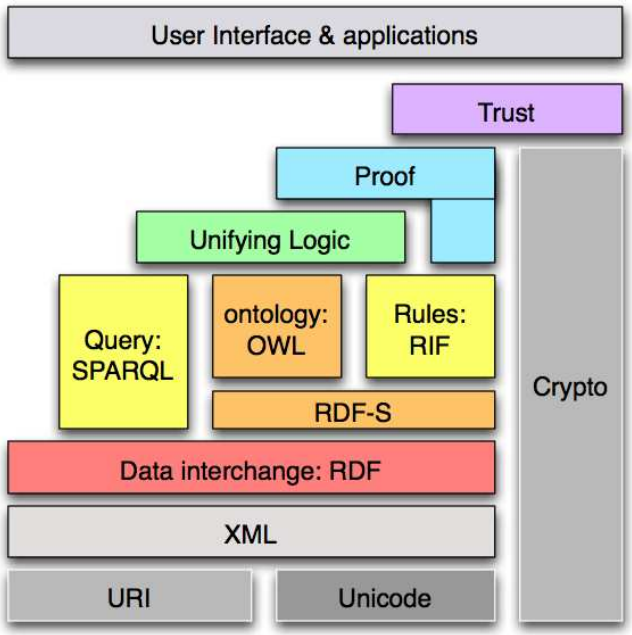


Figure 3.5: The latest V4 version of the Semantic Web architecture [35].

In this V4 version (Figure 3.5), the bottom layer still depicts URI and Uni-

code as in the previous V3 version (Figure 3.4). On the *XML* layer, *Namespaces* is omitted from the model and only *XML* now resides on this layer. An additional description, *Data interchange*, is included on the *RDF* layer. *RDF-S* still resides above RDF. *SPARQL*[10] that provides a mechanism to query RDF data, still resides above the *RDF* layer. On the *Ontology* layer, the caption *ontology: OWL* replaces the caption *Ontology* and *Rules:RIF* is introduced. *Rules* are therefore moved to the same level as OWL. RIF (Rule Interchange Format) is at present a Working Group of the W3C[11]. On the *Logic framework* layer, *Unifying Logic* replaces *Logic Framework* and *Proof* is altered to be above both *Rules: RIF* as well as above *Unifying Logic*. *Trust* still resides above the *Proof* layer. In addition, a layer is added above the *Trust* layer depicting *User Interface and applications* that seem to represent the notion that all applications and user interfaces of the Semantic Web will reside above the *Trust* layer.

The vertical layers in V3 in Figure 3.4 (*Digital Signatures* and *Encryption*) are replaced with a single vertical layer referred to as *Crypto* in V4, Figure 3.5. Unlike previously, the *Crypto* vertical layer no longer starts on top of the *XML* layer, but resides alongside all the layers except *Trust*. It is however not clear what the meaning of *Crypto* is, and it is possible to speculate that it is a combined layer reflecting the security needs of the Semantic Web architecture.

3.2.5.1 Adoption status of the V4 version of the Semantic Web layered architecture

An adoption of the V4 version of the Semantic Web architecture by researchers could not be found in literature as yet. As is the case with the previous versions, no description of this architecture exists within literature either.

[10] Section A.11.2.1, p.346
[11] Section A.11.2.3, p.348

3.2.6 Concluding remarks: The four present versions of the Semantic Web layered architecture

The discussion in this section summarised the different versions of the Semantic Web layered architecture as proposed by Berners-Lee [28, 31, 34, 35]. These four versions of the architecture were labelled V1, V2, V3 and V4 for the purposes of the discussion in this thesis. Along with the characteristics of each of these versions of the architecture, the adoption status of each version was also discussed.

As noted in the preceding discussions, Berners-Lee proposed all the versions of the architecture as part of presentations [28, 31, 34, 35] and no discussion of the intended meaning of the different versions of the architecture could be obtained from literature. However, despite this lack of meaning descriptions, it is plausible to argue that Berners-Lee made a significant contribution towards the establishment of the Semantic Web through his presentation of the different versions of the Semantic Web architecture, as these versions represent a conceptualisation of the languages required to establish the envisioned meta-data functionality of the eventual Semantic Web.

3.3 SEMANTIC WEB ARCHITECTURAL DISCUSSION

In this section, the four versions of the Semantic Web layered architecture are examined in terms of structure and intended meaning, and as a result, some observations are put forth.

Arguably, layering is a common best practice pattern used by software architects to decompose complex systems [12, 56, 178]. In a layered architecture the principal elements or components are arranged in the form of a stack where each layer resides above a lower layer. Generally, a layer represents a grouping of elements that provides related services. A higher layer may use either only various services defined by the immediate lower layer (closed architecture) or services by all of the lower layers

(open architecture). However, the lower layers are unaware of higher layers [12, 56, 104, 178], and are not allowed to access functionality provided by upper layers. This implies a strict ordering of access to the functionality provided by components in a layered architecture in one direction only [13].

The graphical representations of the four versions of the Semantic Web architecture depict a *layered architecture*. However, a concern about these versions of the Semantic Web architecture is that no concise description of the intended meaning of any of the versions was published so far. Furthermore, according to the characteristics of layered architectures, there seem to be inconsistencies on different levels. For example, as both functionality and technology is portrayed by the architectural layers, it is not clear what specifically is depicted by the versions (Figures 3.1, 3.3, 3.4 and 3.5). In particular, certain layers specify a functionality required at that layer in the stack (such as *Ontology*), whilst other layers depict a technology (such as *XML*) without an explanation of the functionality that the specific technology embodies.

It is also not clear what is intended by either the stacking of the layers, or the triangular structure. In addition, there appear more than one technology on certain layers and all of the versions of the architecture depict vertical layers. The intention behind this deviation to the structure of a layered architecture is not reflected in literature.

In the remainder of this section issues identified with regard to the structure and possible meaning of the architecture are discussed in more detail.

3.3.1 Side-by-side layers

In all four versions depicted in Figure 3.6 *URI* and *Unicode* are depicted on the bottom layer as two separate blocks or side-by-side layers (refer to (1) in Figure 3.6). It is safe to assume that the intended implication is that the two technologies both reside on the bottom layer. However, this is contentious and inconsistent when referring to the presentation of other layers containing more than one technology. In V1 *XML + NS + xmlschema* are depicted as a single layer even though they all represent

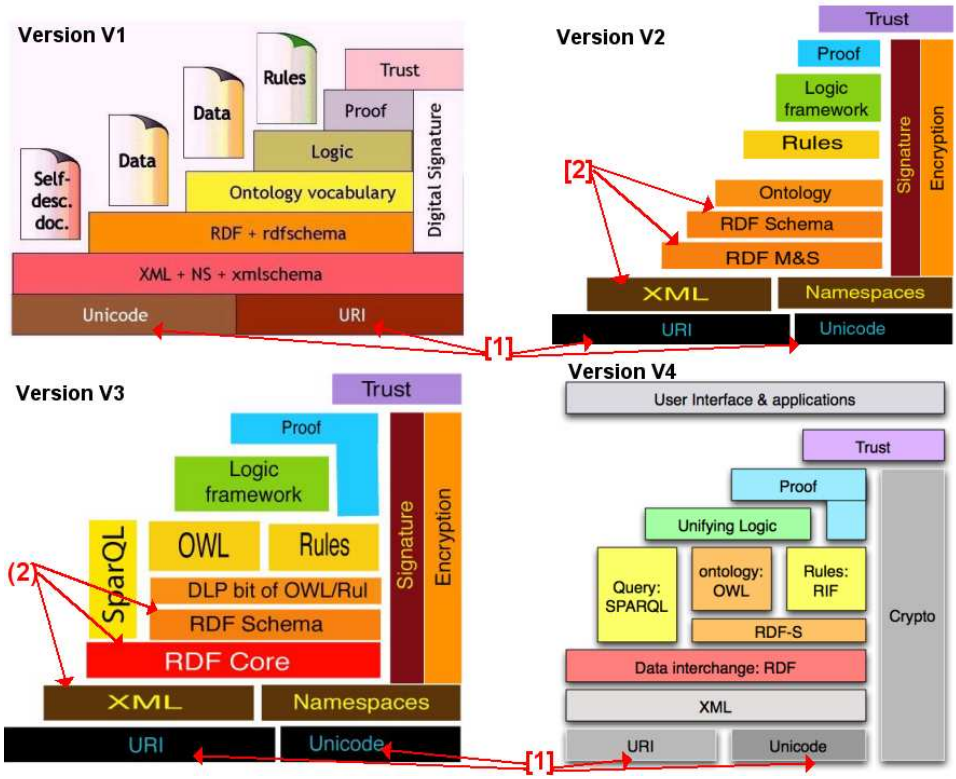


Figure 3.6: The four versions of the architecture: side-by-side layers (1) and the triangular structure of the architecture (2).

separate technologies. Similarly, V1 depict the *RDF* and *RDF Schema* technologies in one layer (*RDF + rdfschema*).

In the V2 version of the architecture in Figure 3.6, *XML* and *Namespaces* are also depicted as separate blocks or side-by-side layers. This is similar to *URI* and *Unicode* on the bottom layer. However, to be consistent with the assumptions of layering, the layering of this version should imply that *XML* only uses *URI*, whereas *Namepaces* uses only *Unicode* as a preceding layer. Based on our knowledge of XML and Namespaces[12], this is not the case even though it is depicted as such in version V2 of the architecture. In addition, one can speculate that URI *includes* Unicode and that *Unicode* should therefore be the bottom layer with *URI* the layer above it.

[12]Section A.4. p.292.

Furthermore, according to in Figure 3.6, V3 adds side-by-side layers with *SPARQL*, *RDF-S*, *ontology:OWL* and *Rules:RIF* and the precise meaning thereof is not clear. The *Proof* layer is also extended to be part of the *Unifying Logic* layer even though it resides above this layer. This may mean that *Proof* resides either as a layer above *Rules:RIF* or above *Unifying Logic*, or it may mean that *Proof* is a layer that requires functionality of both *Rules:RIF* and *Unifying Logic*.

3.3.2 Triangular structure of the layered architecture

In all four versions depicted in Figure 3.6 the layers are staggered into a triangular structure with the lower layers wider than upper layers (refer to (2) in Figure 3.6). It is not clear whether this means that the upper layers use only *part* of what is provided by lower layers, or whether a lower layer specifies additional functionality that is not used for the purpose of the Semantic Web. Nevertheless, the side-by-side layering of the bottom layer(s) is problematic in terms of meaning. It is also significant that both V1 and V4 depict the bottom layers to be the same width. In V1 the *Unicode* and *URI* layer in Figure 3.6 is of the same width as the *XML+NS+xmlschema* layer. V3 depicts *RDF Schema*, *DLP bit of OWL/Rul* and the side-by-side layers *OWL* and *Rules* to be of the same width. In V4 the *Unicode* and *URI*, the *XML* and the *Data Interchange* layers are of the same width. This is not the case in V2 where all layers are staggered in a triangular fashion.

3.3.3 Mixing technologies and functionality descriptions in the naming of layers

It is not clear what the layers in the four versions represent since certain layers are labelled using *technologies* whilst others are labelled using *functionality descriptions* (refer to (3) in Figure 3.7). According to V1, the bottom three layers in Figure 3.7 represent technologies, and all higher layers are labelled with functionality descriptions. Similarly, in V2, the bottom four layers represent technologies and the remaining layers functionalities. In both V3 and V4 (excluding the *User Interface and applications* layer) only

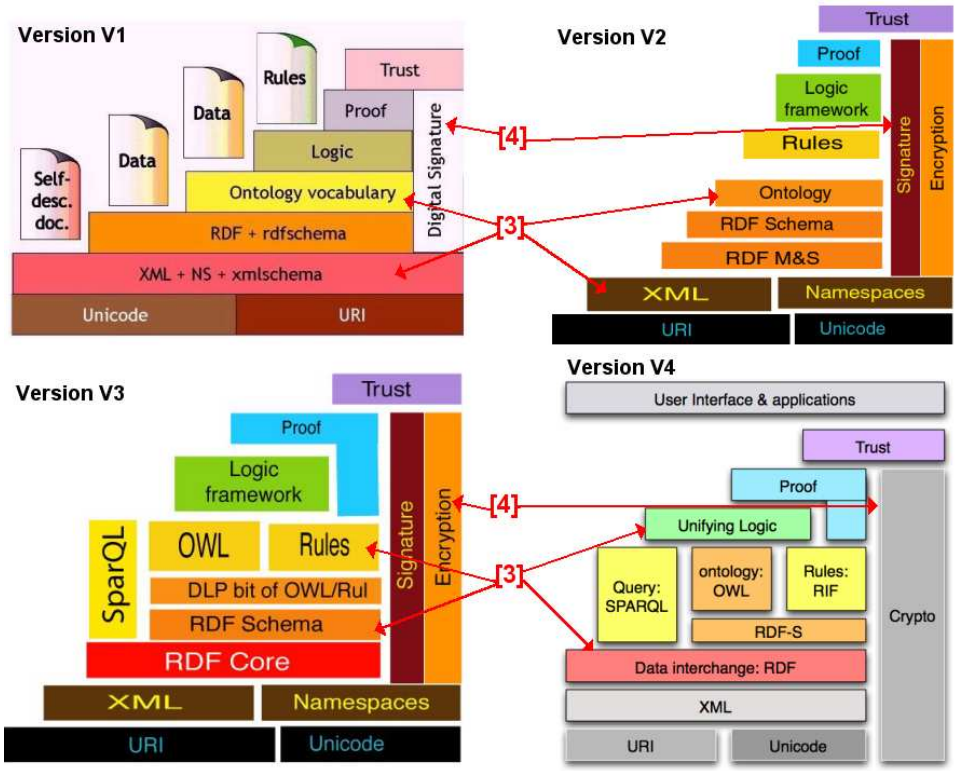


Figure 3.7: The four versions of the architecture: functionality and technology depictions (3) and vertical layers (4).

the top three layers represent functionalities. In both these versions several technology layers were added so that all the lower layers represent technologies.

3.3.4 Vertical layers

All versions of the architecture in Figure 3.7 depict vertical layers such as *Digital Signatures* in V1, *Signature* and *Encryption* in V2 and V3, *SparQL* in V3 and V4, and *Crypto* in V4 (refer to (4) in Figure 3.7). The precise meaning of any of these vertical layers is not specified in literature. It is possible to speculate that these layers are included in all the other layers, or that these layers reside alongside the other layers, or even that these layers only depict technologies building on top of their lower layers and

excluding the upper layers.

3.3.5 Concluding remarks: Semantic Web architectural discussion

The discussion of the different versions of the architecture identified several unresolved and contradictory issues with regard to its meaning. One possible consequence of these issues is that it may result in confusion and uncertainty as regards the adoption of these architecture versions by developers and researchers.

In addition, no agreed-upon conceptual framework has been specified in literature for the required languages to specify the meta-data layers of the envisioned Semantic Web.

In the next section, the architectural initiatives of the W3C are investigated in order to determine whether there is a formal activity that has an intention to address the identified issues of the Semantic Web layered architecture.

3.4 SEMANTIC WEB ARCHITECTURAL INITIATIVES

The specific purpose of this section is to scrutinise the activities of Berners-Lee (Section 3.4.1) as member of the W3C and presenter of the proposed versions of the Semantic Web architecture [28, 31, 34, 35], as well as W3C initiatives (Section 3.4.2) in order to determine their specific focus and whether their activities in any way address the *architecture of the Semantic Web*.

3.4.1 Activities of Sir Tim Berners-Lee

Tim Berners-Lee, one of the initiators of the W3C architecture initiatives, supports thorough design to underpin the standards released by the W3C. In *Axioms of Web Architecture: Principles of design*, Berners-Lee [29] describes the *design principles for the Web* as simplicity, modularity, decen-

tralisation and tolerance. In this section these *design principles* underlying all the W3C architectural initiatives are discussed.

Simplicity is supported by the *Keep It Simple, Stupid* or *KISS* axiom. Simplicity is often ignored because it is subjective. Something that is presented in a simple or even simplistic way should however never be confused with something that is *easy to understand*. Simplicity strives to use fewer basic elements to achieve the required results.

Modular design means that a system is divided into *loosely bound groups* of *tightly bound features*. This axiom is related to the Software Engineering principle of the *loose coupling* and *tight cohesion* of system elements [14].

Berners-Lee summarises *modular design* as follows [29]:

> *Modular design hinges on the simplicity and abstract nature of the interface definition between the modules. A design in which the insides of each module need to know all about each other is not a modular design but an arbitrary partitioning of the bits.*

Modularity also implies *evolution* in that the design of a system should not just be modular in itself, but any system could be a part of an as-yet unspecified larger system.

In addition, Berners-Lee [29] considers *tolerance* and *decentralisation* as founding principles to which Internet and Web standards should adhere. *Tolerance* is described as *be liberal in what you require but conservative in what you do* [29]. The principle of tolerance does not diminish the need for an unambiguous protocol specification that draws a precise distinction between a conformance and non-conformance. The principle of tolerance does not provide an excuse for a product that contravenes a standard.

Decentralisation is a principle for the design of distributed systems and societies. Any single common point within any operation tends to limit the way the system scales, and produces a single point of complete failure [29]. Conceptually, the Semantic Web must avoid centralisation in the definition of concepts, such as a central definition of the concept *automobile* as the term `http://www.kr.org/stds/industry/automobile`. This would restrict

users to be only those systems for whom this particular formulation of an automobile is valid.

The W3C aims, notably through TAG, to ensure that the W3C architectural and standardisation endeavours adhere to these design principles. However, at present there does not seem to be a specific activity formulated by the W3C or Berners-Lee that address the integration of these principles into a *Semantic Web architecture*.

In addition, these design principles are of significance for the development of a functional and comprehensive layered architecture for the Semantic Web, which is the purpose of the research in this thesis.

3.4.2 W3C architectural initiatives

Whilst the previous section highlighted underlying design principles for Web and Internet standardisation efforts spearheaded by Berners-Lee, the specific W3C initiatives pertaining to the architectures required for the implementation of the Internet, Web and Semantic Web, are discussed in this section.

The W3C[13] is regarded as an international, vendor-neutral consortium dedicated to the establishment of an interoperable Web that can accommodate the growing diversity of people, hardware, and software in the world [250]. At present the W3C operates in four domains, namely 1) Architecture, 2) Interaction, 3) Technology and Society, and 4) Web Accessibility Initiative. Within these domains, activities and groups (Working Groups, Coordination Groups and Interest Groups) are active. All the domains, activities and groups are supported by the TAG (*Technical Architecture Group*) and the *Advisory Board* [250].

The two W3C initiatives identified to be of importance for this study are the *W3C Architecture Domain* and the *Technical Architecture Group*.

[13] Section 2.6, p.44.

3.4.2.1 W3C Architecture Domain

The W3C Architecture Domain is an overarching, top-level organisation of certain related activities [126]. The purpose of the W3C Architecture Domain is to enhance the infrastructure of the Web and increase its automation, as described by the following quotation:

> *W3C leads the evolution of the web, empowering individuals, increasing social and economic efficiency, and exploiting the power of computing in our everyday lives. Exploiting that power is the mission of the W3C Architecture Domain* [126].

Within the Architecture Domain, five activities are presently defined [126]:

- XML (Extensible Markup Language),
- Web Services,
- Internationalisation,
- URI/IRI; and
- DOM (Document Object Model).

The mission of the Architecture Domain is to maintain and extend these *core* technologies of the Web [126]. The W3C architecture technologies have as specific focus to enable users of the Web to exchange data on the Web by means of technology specifications from the protocol level that includes HTTP (Hypertext Transfer Protocol) and SOAP (Simple Object Access Protocol), to the application level with technologies such as XML, XML Schema, and WSDL (Web Services Description Language). The Architecture Domain also includes the development of technologies for data manipulation such as XSL (eXtensible Stylesheet Language) Transformations, DOM (Document Object Model), and XML Query [126].

At present, the organisation of the W3C Domains does not include the *Semantic Web Activity* into the *W3C Architecture Domain* according to the five activities listed. In addition, the current emphasis of the *W3C Architecture Domain Activities* does not include the specification of the *Semantic Web language architecture* specifically, even though all these initiatives may have an impact on the eventual realisation of the Semantic Web.

3.4.2.2 TAG

The purpose of TAG (*Technical Architecture Group*) is to permeate architectural design principles such as discussed in Section 3.4.1, and thus to influence all activities and actions of the W3C. According to the TAG charter, all domains, activities and groups are to be supported by the TAG [249, 250]. The TAG charter was published on 19 July 2001 [249], and summarises the TAG goals as follows:

> *W3C has created the TAG to document and build consensus around principles of Web architecture and to interpret and clarify these principles when necessary, to resolve issues involving general Web architecture brought to the TAG, and to help coordinate cross-technology architecture developments inside and outside W3C* [249].

A number of architectural principles such as those described in Section 3.4.1 underlie the development of the Web. These principles have to be agreed upon and documented in order to facilitate the acceptance, growth and interoperability of the Web. These Web architectural principles are debated, developed, and documented both inside and outside of W3C. Several informal documents on the general topic of *architecture* were published by the W3C [25, 29, 30, 44]. Within these documents the term *architecture* is used to loosely define an organisation of diverse components. However, with the growth of the W3C there is an increasing demand for formal documentation of accepted architectural principles that cut across multiple technologies. Such documented architectural principles will solve a dual purpose in that it would assist developers of new standards, as well as resolve disagreements about architecture that evolved from different working groups. In addition, it is foreseen that W3C architectural recommendations could improve effectiveness, reduce misunderstandings and overlapping work, and improve the consistency of Web technologies and standards [249].

For the purposes of the TAG charter, the term *Web architecture* is defined as the underlying principles that should be adhered to by all Web

components [249]. These principles address issues such as understandability, interoperability, scalability, accessibility, and internationalisation. *Understandability* is promoted by means of specifications that are built using a common framework. Such a framework could potentially clarify interactions and relationships of specifications. *Interoperability* encapsulates principles that cross Working Group boundaries in order to assist technical specifications to interoperate. *Scalability* is a principle that ensures that the different W3C initiatives include aspects that guarantee wide applicability and future extensibility of their recommendations [249].

The mission of the TAG is stewardship of the Web architecture. There are three aspects to this mission:

> - *to document and build consensus around principles of Web architecture and to interpret and clarify these principles when necessary,*
> - *to resolve issues involving general Web architecture brought to the TAG; and*
> - *to help coordinate cross-technology architecture developments inside and outside W3C.*
>
> *The primary activity of the TAG is to develop Architectural Recommendations. An Architectural Recommendation is one whose primary purpose is to set forth fundamental principles that should be adhered to by all Web components* [249].

The primary activity of the TAG is therefore the development of architectural recommendations. *TAG findings* document fundamental principles identified for adherence by all Web components. TAG include these findings in the TAG architectural recommendations, published according to the requirements of the W3C Recommendation Track process [252].

In addition to the development of architectural recommendations, the TAG has as mandate the resolution of *issues* with architectural impact, quickly, consistently, and with as much consensus as possible W3C [253]. Issues are brought before the TAG and are resolved by means of a majority vote. Resolved issues are released as a *statement of architectural principle* [249].

The TAG issues are listed in the *TAG Issues List* W3C [253] and include issues such as:

- **w3cMediaType-1**: Should W3C Working Groups define their own media types?
- **rdfmsQnameUriMapping-6**: Algorithm for creating a URI from a QName?
- **namespaceDocument-8**: What should a *namespace document* look like?

At this stage it suffices to note that *architecture* as defined by TAG has as focus the encapsulation of design principles into W3C Recommendations and specifications. This definition is adopted in the *Architecture of the World Wide Web, Volume One*, released as a W3C Recommendation on 15 December 2004. This Recommendation is the first architectural specification to be issued by the W3C [36], and specifically addresses issues pertaining to *Identification* using URIs, *Interaction* that discusses representation of resources using URIs, as well as *Data Formats* and *General Architecture Principles*. It is beyond the scope of this section to discuss this document in detail. However, it should be noted that the current version of the architecture recommendation of the W3C does not address the Semantic Web, neither layering issues with regard to the languages of the Semantic Web *specifically*.

In this section W3C architectural initiatives are discussed. These architectural initiatives were identified to be the *W3C Architecture Domain* activities (Section 3.4.2.1) and the *TAG* (Section 3.4.2.2). Within these W3C architectural initiatives, the definition of the term *architecture* is generally accepted as a loose framework for infrastructure components (*W3C Architecture Domain*) or a listing of underlying architectural design principles (*TAG*). Presently, none of these initiatives of the W3C address the issues pertaining to the conceptual *layered architecture for the languages of the Semantic Web* specifically.

3.5 RESEARCH PROBLEM

It is not unreasonable to propose that the architecture of any information system is one of the primary aspects to consider during design and implementation thereof. It is thus plausible to state that the proposed architecture of the Semantic Web is crucial to its eventual realisation and that it is therefore necessary to attach undisputable meaning to the specification of the architecture for the languages of the Semantic Web.

The current versions of the layered architecture that exist within literature have been proposed by Berners-Lee, and the literature offers no description or specification of meaning for any of these. Furthermore, the different versions of the Semantic Web architecture proposed by Berners-Lee [28, 31, 34, 35] all depict inconsistencies and discrepancies as discussed in Section 3.3. This leads to confusion, as well as conflicting proposals and adoptions by the Semantic Web community [130, 131, 190]. Furthermore, none of the current formal initiatives by the W3C address the Semantic Web architecture specifically. Within the W3C initiatives, the specification of a comprehensive and functional layered architecture for the Semantic Web could therefore address a current limitation.

In addition, because there is no generally accepted specification of functionality or interfaces, there is a possibility that applications implementing Semantic Web technologies will not be able to interoperate. The interoperation of Semantic Web applications is crucial for the eventual realisation of the founder vision. A comprehensive and functional layered architecture for the Semantic Web that adheres to the fundamental aspects of layered architectures will assist in the development of Semantic Web specifications and applications. Furthermore, several of the current research and implementation issues associated with the implementation of the Semantic Web could potentially be resolved.

This thesis therefore argue for the development of a comprehensive and functional layered architecture for the Semantic Web that adheres to fundamental aspects for layered architectures.

Such an architecture could be regarded as crucial for the development, growth and adoption of the Semantic Web.

3.6 CONCLUSION

The chapter states the research problem for this thesis as the development of a comprehensive, functional architecture for the Semantic Web that adheres to fundamental principles of layered architecture design as embodied in the Software Engineering discipline.

Furthermore, in this chapter, the existing Semantic Web architectural versions and initiatives are discussed. Different versions of the layered architecture for the Semantic Web were proposed by Berners-Lee. There is, however, no description or consensus about the meaning of these architectures. In addition, several problematic issues pertaining to the meaning or structure of the current versions were identified. Furthermore, the architectural initiatives of the W3C do not focus on the Semantic Web or its architecture specifically.

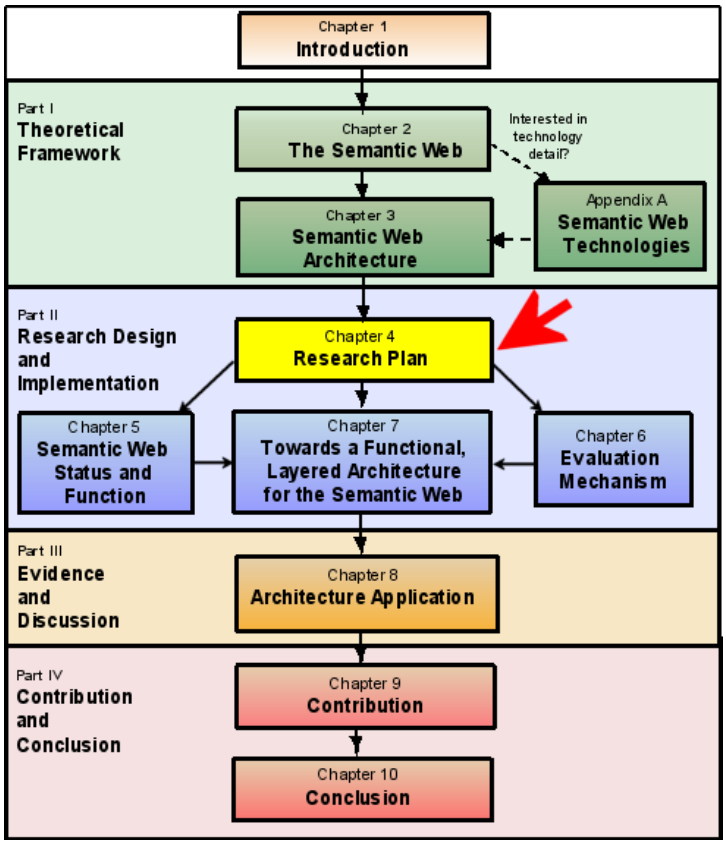
In addition, it is recognised that the architecture of a system is crucial to the development thereof, and research towards formalising an architecture for the Semantic Web is therefore indicated.

Part II

RESEARCH DESIGN AND EXECUTION

CHAPTER 4

RESEARCH DESIGN



Thesis Chapter Layout. Chapter 4 is the first chapter in Part II: Research Design and Implementation.

Chapter Contents

Figures

Tables

4.1 INTRODUCTION

In this chapter, the design of the research executed and reported upon in this study, is discussed.

The questions that are of concern within this chapter, are:

- Which activities were executed in order to complete the research and to answer the research questions?
- Is it a qualitative or a quantitative study?
- What is the epistemological stance of the research?
- Which research approaches were followed?
- Which methods were used for data analysis?

4.2 RESEARCH ACTIVITIES

In Chapter 3, it was established that there are several shortcomings with regard to the proposed Semantic Web architecture and the meaning associated therewith. The chapter concluded with the research problem for this study, namely the development of a comprehensive, functional architecture for the Semantic Web that adheres to the fundamental principles of layered architectures. The research questions are:

	Research Questions
(1)	What is the function of each technology included in the present versions of the Semantic Web layered architecture? ▷ What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?

(2)	To which criteria should a layered architecture conform in order to adhere to system design principles? ▷ Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?
(3)	How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?

Table 4.1: Research questions

The development of a *functional* architecture prescribes an investigation into the functionality provided by the technologies currently part of the layered architecture of the Semantic Web. Chapter 5 addresses the function and status of the respective technologies.

A method to evaluate layered architectures is required on order to develop a *comprehensive* architecture that adheres to fundamental principles of layered architectures. This evaluation mechanism will be used to determine whether the proposed comprehensive and functional layered architecture adheres to fundamental principles for layered architectures. The evaluation mechanism ideally consists of criteria for which it is possible to establish conformance or non-conformance.

Furthermore, it is necessary to calibrate the evaluation mechanism during its development. The evaluation mechanism for layered architectures is therefore used to assess an established layered architecture, and the results of this evluaiton are used to determine the validity of the evaluation mechanism. The evaluation mechanism for layered architectures consisting of criteria derived from fundamental principles is developed in Chapter 6.

Once such an evaluation mechanism is developed and calibrated, it is used to determine possible shortcomings regarding the current versions

of the Semantic Web architecture as proposed by Berners-Lee [28, 31, 34, 35]. The evaluation mechanism is furthermore used to adapt the V2 version of the architecture so that it adheres to the criteria and principles for layered architectures. The development of a comprehensive and functional layered architecture for the Semantic Web (referred to as the CFL architecture for the purposes of this study) is discussed in Chapter 7.

In order to establish whether the proposed CFL architecture is useful, the architecture is applied in case studies in Chapter 8. This chapter is contained within the thesis part on *evidence and discussion*, and the results obtained should provide an indication of the value the proposed CFL architecture adds to the scientific body of knowledge.

This remainder of this chapter is devoted to a discussion of the research design and research approach followed in order to answer the research questions, as well as applicable mechanisms used to verify and test the results.

4.3 QUALITATIVE RESEARCH

According to Myers [174] there is a shift in the research conducted in the Information Systems domain to include Software Engineering aspects such as the human, social, organisational and methodological issues, and hence an increased application of qualitative research methods. Miles and Huberman [168] define qualitative data as data in the form of *words* rather than *numbers*. Qualitative data is rich in meaning and provides opportunities for explanation and integration activities. Furthermore, this type of data provides opportunities to develop artefacts beyond initial conceptions to revised conceptual frameworks and models [168].

As stated by Miles and Huberman [168]:

> *Words, especially organized [sic] into incidents or stories, have a concrete, vivid, meaningful flavor that often proves far more convincing to a reader - another researcher, a policymaker, a practitioner - than pages of summarized numbers* [168].

This study constructs models and tools, and, in addition, uses documents, publications and texts, as well as the researcher's impressions and experience as data sources. The study is therefore qualitative and not quantitative.

4.4 PHILOSOPHICAL AND EPISTEMOLOGICAL STANCE.

In addition to being quantitative or qualitative, all research is executed from a philosophical base or the researcher's stance on aspects such as truth and validity, and that determines acceptable research methods to be adopted [87, 174, 227]. Within this philosophical base, the researcher has assumptions about knowledge, its construction, how it can be obtained, and its validity. This view on what knowledge is and how it can be obtained is referred to as the *epistemological base* of the research study. Putting it differently, epistemology is defined as the *theory of knowledge* and the central question of epistemology is: *Under what conditions does a subject know something to be the case?* [87].

Several epistemological stances are documented in literature, especially within the social sciences. For research in information systems Myers [174] identifies three paradigms namely positivist, interpretive or critical. Positivists generally assume that reality is objectively given and can be described by measurable properties which are independent of the observer (researcher) and his or her instruments. Interpretive researchers start out with the assumption that access to reality (given or socially constructed) is only possible by means of social constructions such as language, consciousness and shared meanings. Critical researchers assume that social reality is historically constituted and furthermore that it is produced and reproduced by people [174, 227].

Miles and Huberman [168], p.8 identify three approaches for qualitative data analysis namely *interpretevism*, *social anthropology* and *collaborative social research*. Interpretevists view everything as a collection of symbols expressing layers of meaning (or *texts*), and in their research they interpret

these texts in order to extract meaning. *Social anthropologists* conduct research using extended contact with an identified community. In *collaborative social research* collective action is undertaken in a social setting [168].

According to the broad categories of Miles and Huberman [168] and Myers [174], this study is of an interpretevist nature.

To expand the notion of the philosophical paradigm and epistemological base of Miles and Huberman [168] and Myers [174], Burrell and Morgan [57] identify two dimensions of research namely the *ontological nature of reality* that is on a scale from internal to external, and the *epistemological nature of science* that is on a scale from subjective to objective. This is graphically depicted in Figure 4.1. Within this context, epistemology is defined as the *theory of knowledge* [87] and *ontology* within the philosophical perspective is defined as *the theory of objects and their ties*. In addition, ontology provides criteria for distinguishing various types of objects and their relations [73].

The *radical humanist* views the social world from a perspective that tends to be nominalist and anti-positivist. Within this research paradigm it is important to transcend limitations or alter existing social arrangements radically. Such a researcher focuses on his or her own consciousness and is mainly concerned with releasing social constraints that limit human potential. The current dominant ideologies are often seen as separating people from their *true selves* and they therefore use this paradigm to justify desire for revolutionary change. Radical humanism is anti-organisation in scope [57, 75].

The *radical structuralist* views the world as consisting of concrete artefacts and the focus is on structural relationships within a realistic social world. Radical structuralists aim of these researchers is to explain the basic interrelationships and change that are built into the nature and structure of society. Based on this paradigm, the radical structuralist see inherent structural conflicts within society that generate constant change through political and economic crises. This has been the fundamental paradigm of Marx, Engles, and Lenin [57, 75].

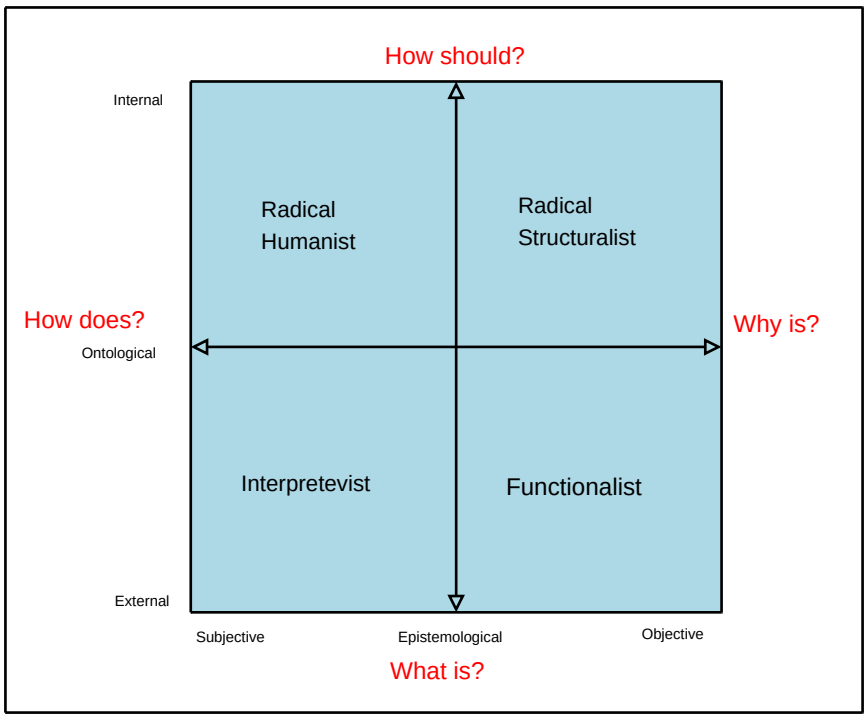


Figure 4.1: The two dimensions of research (adapted from Burrell and Morgan [57], Cronje [75])

The world of the *functionalist* is composed of relatively concrete empirical artefacts with structural relationships. These artefacts can be identified, studied and measured by means of approaches derived from the natural sciences. Modelling is done by making use of mechanical and biological analogies. This has been the primary paradigm for organisational study. The functionalist assumes rational human action and believes understanding of organisational behaviour can be attained by means of hypothesis testing [57, 75].

The *interpretevist* is characterised by a concern to understand the world *as it is*, or to understand the fundamental nature of the social world at the level of *subjective* experience. Such a interpretevist researcher seeks truth and explanations in his or her subjective consciousness, often through constructs or models. In this type-of research the researcher is preferably a participant rather than an observer of a phenomenon. Social researchers in this paradigm aim to describe *on-going processes* for a better understand-

ing of individual behaviour and the *spiritual nature of the world* [57, 75].

The extension of Burrell and Morgan [57] of the broad categories of Miles and Huberman [168] and Myers [174] discussed above, supports the notion that the epistemological stance of this study is *interpretevist*, since it focuses on model-building as well as methodology construction in order to explain the structure of systems and objects.

4.5 RESEARCH APPROACH

The previous section identified this study as a *qualitative* study and the epistemological stance of this research to be of an *interpretevist* nature. In this section, the different research approaches chosen to perform the research are discussed. A research approach defines the processes that are executed during the research.

Within the domain of Computer Science and Information Systems, researchers predominantly perform applied research where artefacts are often built and tested. According to Mouton [172], this research is generally only empirical or non-empirical. In addition, the empirical and non-empirical approaches are extended to include research studies executed within Information Systems research. Mouton [172], p.137 identified several research studies or approaches categorised as either empirical or non-empirical. The two approaches relevant to this thesis are *Theory-building or Model-building studies*, which are *non-empirical*, and *Methodological studies*, which are *empirical*. These studies are discussed in the remainder of this section.

4.5.1 Theory-building or model-building studies

Generally, science makes progress by means of both *models* and *theories*. When models are constructed the researcher attempts to explain phenomena in the world as perceived [172]. A *model* can be defined as *a set of statements that aims to represent a phenomenon or set of phenomena as*

accurately as possible [10, 172]. Furthermore, a *theory* comprises of related concepts and a theory is defined as *a set of statements that makes explanatory or casual claims about reality* [172].

Theory-building or model-building studies are *non-empirical* studies that aim to develop new theories or models in order to explain objects, artefacts or phenomena. In addition, these studies are used to extend or refine existing models and theories. Good, well-constructed theories and models allow for a basis from which one could make predictive claims under certain conditions, as well as bring conceptual coherence to a specific domain of science and from where our understanding of the world could be simplified [172].

More pertinently, theory-building or model-building studies build the intended theories and models mainly though either inductive or deductive reasoning strategies. Typically, inductive reasoning is used in statistical model building where the model is constructed to explain or categorise certain empirical data. In analogical construction of models, a model is constructed on the basis of its similarities to other models of specific phenomena [172]. In contrast deductive reasoning is more formal in that a set of postulates or axioms is formulated. These postulates are used to deductively derive additional theoretical propositions. This process is repeated until the researcher has developed a comprehensive set of theoretical propositions that will ultimately be tested against empirical data [172]. Typically, data analysis methods used in theory-building or model-building studies include mathematical model building as well as grounded theory.

Limitations of theory-building or model-building studies include that these theories are ineffective if they make erroneous claims such as implausible claims of reality, claims that are not testable and vague, or claims that are conceptually incoherent, inconsistent and confusing [172]. In addition, over-abstract formulations of the world that cannot be validated empirically are regarded as a source of error in these studies. Additional sources of error relate to the assumptions that are made in specifying the model, the quality of the data against which the theory or model will be tested and the correct use of statistical and mathematical procedures [172].

4.5.2 Methodological studies

Methodological studies are empirical by nature and are defined as *studies aimed at developing new methods of data collection (such as questionnaires, scales and tests) and are sometimes also used to validate a newly developed instrument through a pilot study* [172]. Methodological studies are typically used to develop measuring instruments or to validate existing tests and scales. These studies are usually done in conjunction with other studies such as surveys, experiments and comparative studies [172].

Methodological studies can use both inductive and deductive reasoning. When using inductive reasoning, empirical data are analysed in order to identify the methodological quality of the data. Deductive reasoning are used to test, for instance, a hypothesis of theory against empirical data [172].

Methodological studies in the fields of experiments-, or surveys and cross-cultural studies have provided insight into sources of error in empirical research. However, limitations include that the results of a methodological study usually cannot be generalised [172]. An example would be a study executed in the USA that cannot be generalised world-wide, or in terms of another specific culture. Sources of error in methodological studies generally entail sampling errors and measuring errors.

4.5.3 Organisation of studies in the thesis

The research questions on page 83 dictates the organisation of the *studies* of Sections 4.5.1 and 4.5.2 in this study.

A non-empirical, model-building study is executed in order to answer Questions 1 (with its sub-question) and Question 3 , namely *What is the function of each technology included in the present versions of the Semantic Web layered architecture?*, *What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?* and *How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?*,

An empirical methodological study is executed in order to develop a measurement tool (referred to as an evaluation mechanism in this thesis) for layered architectures in order to answer Question 2 with its sub-question, namely *To which criteria should a layered architecture conform in order to adhere to system design principles?* and *Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?*. This tool is empirically used to assist in the construction of a new (non-empirical) model. The execution of empirical and non-empirical studies in this thesis is therefore cyclic as depicted in Figure 4.2.

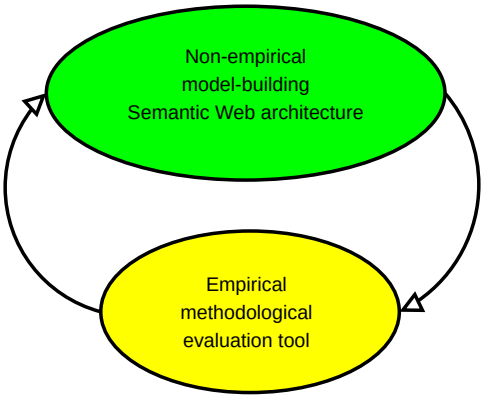


Figure 4.2: The organisation of the studies in this research.

By using both empirical and non-empirical studies, the risk of errors occurring within both methods is limited. Within model-building there is a risk that the model might be too abstract or that it cannot be validated (refer to Section 4.5.1). By constructing an evaluation mechanism based on theoretical grounds, as attempt is made to ensure that the model is validated, that it is not too abstract and that the assumptions are valid.

Moreover, during the construction of an evaluation mechanism for layered architectures, the mechanism is tested against an accepted model to ensure that the mechanism is calibrated. One of the identified risks of measurement tools or evaluation mechanisms is that the tool or mechanism might not be calibrated correctly refer to Section 4.5.2). Once the tool has been calibrated against an accepted layered architectural model, it is applied to the model under construction in this thesis, namely the Semantic

Web layered architecture.

4.6 DATA COLLECTION AND ANALYSIS

In this section the mechanisms used to collect, analyse and verify the data used in this research are discussed. The approach followed was adopted from Miles and Huberman [168], who are cited extensively on the topic of qualitative data analysis.

4.6.1 Qualitative data analysis

Miles and Huberman [168], p.10-11 define qualitative data analysis as consisting of three concurrent flows of activity namely data reduction, data display, and conclusion drawing or verification. These three activities are interwoven before, during and after data collection. Qualitative data analysis is continuous and interactive (refer to Figure 4.3).

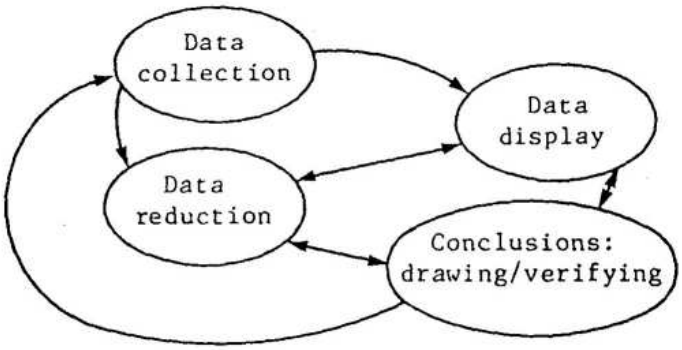


Figure 4.3: Interactive model of the components of qualitative data analysis [168, p.12].

4.6.2 Data reduction

Data reduction is defined as the process of selecting, focusing, simplifying, abstracting, and transforming the written data. Data reduction can also be described as *data condensation*.

Data reduction may include quantification, however, it can take many forms such as data reduction through selection, summary or paraphrase. Data reduction can be done through the subsumption into a larger pattern or conceptualisation [168, p.11].

In this study data reduction was performed by means of selection and subsumption. Referenced publications were selected based on the departure point, which is the architecture of the Semantic Web. The most significant of these documents were analysed and lead to the additional selection of documents. In Section 4.6.2.1 the data collection techniques are discussed in more detail.

In addition, in specific cases analysis of the documents lead to the identification of additional patterns, which is referred to as *data reduction through subsumption*. An example of this is the realisation that the concept *layered architecture* is not defined rigorously in literature but is regarded as an architectural pattern, which lead to the investigation of architectures in general (refer to Chapter 6).

4.6.2.1 Data collection and completeness

The primary source of data used in this thesis is documents that were obtained from formal academic sources. This was supplemented with publications from verified, informal sources such as specific Web sites, as well as from insights gained from experience.

Documents were collected from the primary publication libraries within Computer Science and Information Systems, notably the ACM, the Computer Society Library (IEEE), Elsevier, ScienceDirect and Springerlink. In addition, the Web was used to execute searches and cross-correlate the results with the document collections obtained from the formal libraries.

Several official Web sites also served as an important source for documents, notably the Web site of the W3C (World Wide Web Consortium) [250] and the IETF (Internet Engineering Task Force) [137].

The technique used to ensure completeness of the data or document set, is *saturation* [194]. *Data set saturation* was achieved by utilising the reference lists of significant publications. Documents referenced by significant publications were collected where possible, and if applicable, their reference lists were once again used to increase the document collection. The process continued until between 80-100 percent of the references of the significant publications were included in the data set. This process was continuous and interactive, whenever any new significant document was added to the data set. The process was repeated until the saturation point was reached again.

4.6.3 Data display

The second activity within qualitative data analysis is *data display*. Generally, a display is an organised, compressed assembly of information that permits conclusion drawing. Typical outputs of this activity include extended texts, many types of matrices, graphs and charts. The aim of this activity is to assemble organised information into accessible, compact form in order to draw justified conclusions [168, p.11].

This thesis is initiated with a display of data, in particular the layered Semantic Web architecture of Berners-Lee [28, 31, 34, 35]. However, this display is not accessible and cannot be used to draw justified conclusions as determined in the research problem. A study was therefore initiated towards the adaptation of this display in order to make it accessible and understandable. This resulted in the development of the adapted Semantic Web layered architecture. In addition, a data display activity was used to depict the result of the development of the evaluation mechanism (Chapter 6) as a list of criteria with associated descriptions and questions.

4.6.4 Conclusion drawing and verification

According to Miles and Huberman [168], p.11, qualitative data analysis comprises a third activity, namely *conclusion drawing and verification*, described as follows:

> *From the start of data collection, the qualitative analyst is beginning to decide what things mean - is noting regularities, patterns, explanations, possible configurations, causal flows, and propositions. The competent researcher holds these conclusions lightly, maintaining openness and scepticism, but the conclusions are still there, inchoate and vague at first, then increasingly explicit and grounded...* [168, p.11].

In addition to the drawing of conclusions described above, this activity also includes the verification of such drawn conclusions. Meanings emerging from data have to be tested for plausibility, sturdiness or confirmability, in other words, whether they are valid.

Miles and Huberman [168], p.245 identify several tactics for generating meaning from qualitative data, namely noting patterns and themes, seeing plausibility, clustering, making metaphors, counting, making comparisons, subsuming particulars into the general, factoring, noting relations between variables and finding intervening variables. In addition, the tactics used to assemble a coherent understanding of the data include building a logical chain of evidence and making conceptual and theoretical coherence.

Several of the tactics identified by Miles and Huberman [168] were used in this thesis. *Noting patterns* is used during evaluation of the use of layered architectures in Chapter 6, and *subsuming particulars into the general* with the definition of the Semantic Web in Section 2.3.1. In particular, the definition of an architecture in Section 6.3.4 make substantial use of the *subsuming particulars into the general* tactic. *Clustering* is used to extract the criteria list in the evaluation mechanism.

In the compilation of a thesis, *building a logical chain of evidence* and *making conceptual and theoretical coherence* are used. This is evident in the thesis chapter map displayed at the beginning of each chapter. The

development of the functional architecture in Chapter 7 is performed by means of the *clustering* and *making conceptual and theoretical coherence* tactics.

Miles and Huberman [168] propose the use of more than one tactic when deriving meaning from qualitative data. In addition, the researcher should always remain *sceptic* of conclusions reached and should seek to verify it as often as possible. The authors identify several tactics that could be used for the testing and confirmation of findings [168, p.262]. Several of these tactics verify processes executed when collecting qualitative data through fieldwork, which is not applicable to this thesis. The verification tactics identified that are used in this thesis are *checking for representativeness*, *looking for negative evidence* and *replicating a finding* [168].

The *checking for representativeness* tactic was used during compilation of the document set using *data set saturation*. In addition, an attempt was considered to ensure representativeness when compiling definitions and criteria through the *inclusion tactic* rather than *exclusion*. Every additional definition or criterion identified in literature is included in the definitions or criteria lists through some means. *Looking for negative evidence* is also used when drawing any conclusion from literature. Similarly, *replicating a finding* is used to verify major conclusions. The evaluation mechanism for layered architectures is tested against the ISO/OSI model to calibrate it before it is used to evaluate the Semantic Web layered architecture. In addition, the proposed Semantic Web layered architecture is used in case studies to substantiate the conclusion.

4.7 RESEARCH DESIGN

The qualitative data analysis activities were integrated into the research approaches to form the research design of this thesis. This integrated research design is depicted in Table 4.2. The columns depict the applicable approaches namely a methodological study as well as a model-building study. In both these studies the qualitative data analysis activities of Miles and Huberman [168] are performed. These are indicated in the rows of Ta-

ble 4.2. The data analysis tactics that are applicable to the specific study, are listed in the respective column.

	Methodological study	**Model-building study**
Data reduction	Collection of architectural related publications until data-set saturation was achieved. Tactics: selection and subsumption.	Collection of Semantic Web related publications until data-set saturation was achieved. Tactics: selection and subsumption.
Data display	Determination of the criteria list for layered architectures. Tactics: matrix forming.	Determination of function and status of specific Semantic Web technologies as well as identification of the respective layers of the layered architecture. Tactics: model construction.
Conclusion drawing and verification	Establishing an evaluation mechanism for layered architectures and the calibration thereof using an accepted and established existing layered architecture. Tactics: noting patterns and themes, clustering, making comparisons, subsuming particulars into the general, building a logical chain of evidence and reaching conceptual and theoretical coherence. Tactics for verification: checking for representativeness and replicating a finding.	The compilation of a Semantic Web status model. The adaption of the proposed Semantic Web layered architecture and its evaluation and application in case studies to determine its usefulness. Tactics: noting patterns and themes, observing plausibility, clustering, making comparisons, subsuming particulars into the general, building a logical chain of evidence and reaching conceptual and theoretical coherence. Tactics for verification: checking for representativeness, looking for negative evidence, and replicating a finding.

Table 4.2: Research design

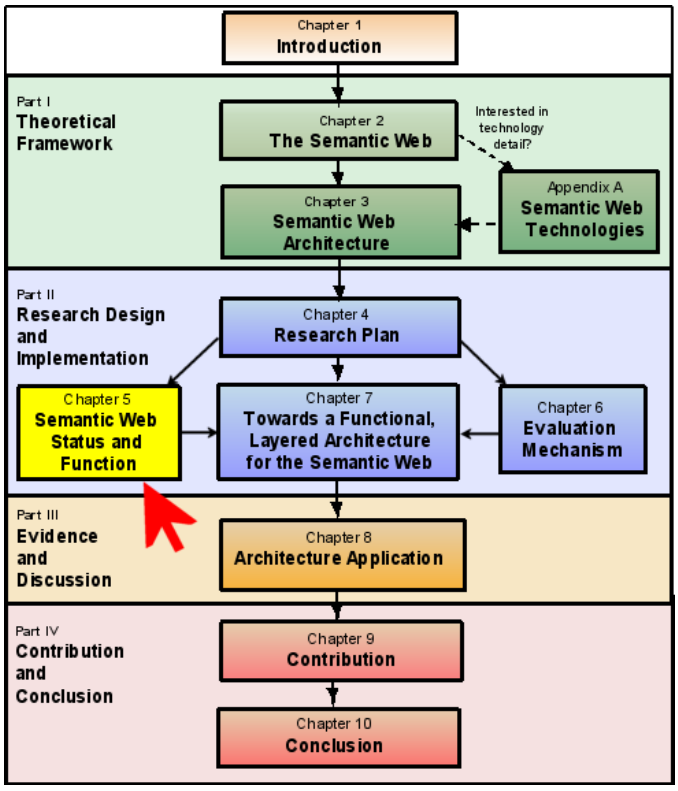
4.8 CONCLUSION

This chapter identified the research in this study as qualitative and in addition, the research was executed from within the epistemological stance of *interpretevism*. In addition, the two studies used in the research approach were a non-empirical, model-building study and an empirical methodological study. Furthermore, the qualitative data analysis activities and tactics of Miles and Huberman [168] were used to ensure valid data collection, analysis, conclusion drawing and verification. The research design is summarised in Table 4.2 that depicts the appropriate data analysis techniques applicable to the identified approaches or studies.

All these approaches, techniques and tactics were used to execute the research activities as identified in the introduction in a responsible manner. This ensures that the resulting contribution of the research may be regarded as trustworthy and valid.

CHAPTER 5

SEMANTIC WEB STATUS AND FUNCTION



Thesis Chapter Layout

Chapter Contents

Figures

Tables

5.1 INTRODUCTION

In order to develop a *functional* and comprehensive layered architecture for the Semantic Web, the extraction of the functionality of the technologies currently depicted in the versions of the layered architecture proposed by Berners-Lee [28, 31, 34, 35] is required. A *functional* architecture depict functionality and not technology implementations, and when the present versions of the layered architecture for the Semantic Web are used as basis, the function of the technologies depicted in these versions need to be extracted. In this chapter, the status and function of the Semantic Web technologies are reviewed. In addition, status models that depict refinements imposed by the function and status quo of current technology, are developed in this chapter. These adapted models may provide valuable insight into the limitations of the technologies currently supporting the Semantic Web layered architecture.

The questions that are of concern within this chapter are:

- What is the function of the technologies depicted in the Semantic Web architecture?
- What is the status of each of the technologies depicted in the Semantic Web architecture?

The status and function of the Semantic Web technologies are discussed in Section 5.2. The status model is developed in Section 5.3. The latest V4 version of the Semantic Web layered architecture in relation with the status model is discussed in Section 5.4. The enhanced status model reflecting technology functions is presented in Section 5.5. The chapter is concluded in Section 5.6.

5.2 SEMANTIC WEB TECHNOLOGY STATUS AND FUNCTION

The discussion in this chapter builds on the technology descriptions contained in Appendix A, and uses versions V1 and V2 of the layered architec-

ture proposed by Berners-Lee [28, 31] in order to organise the discourse. The V1 and V2 versions of the Semantic Web layered architecture were adopted by researchers in literature (refer to Sections 3.2.2, p.56 and 3.2.3, p.60) and are therefore used as the basis for the technology discussion in this chapter. The V1 and V2 versions of the Semantic Web layered architecture are repeated in Figures 5.1 and 5.2 with added 'layer' captions for reference purposes.

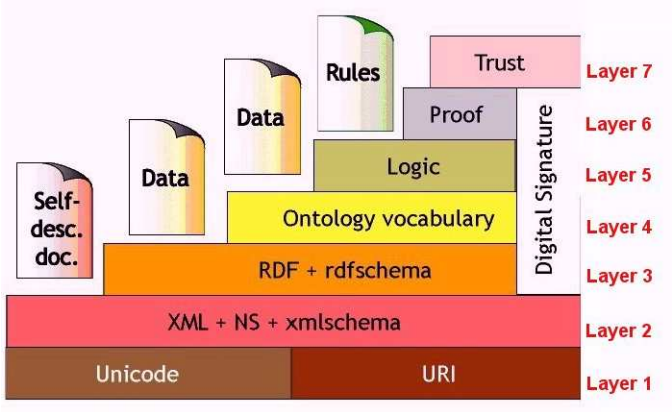


Figure 5.1: The original Semantic Web architecture (V1) [28]

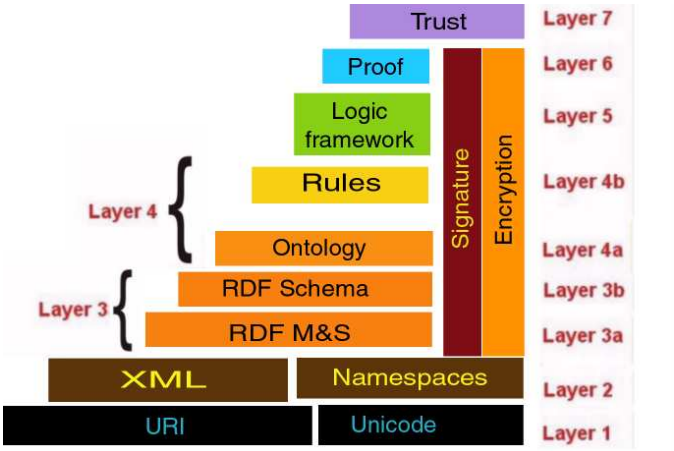


Figure 5.2: The subsequent Semantic Web architecture (V2) [31]

The function and status of the technologies depicted in versions V1 and V2 of the Semantic Web layered architecture are discussed according to

their positioning in the structure, starting with the bottom layer, i.e. layer 1.

5.2.1 Layer 1: Unicode and URI

Layer 1 in the V1 and V2 versions of the proposed Semantic Web layered architecture of Berners-Lee comprises Unicode[1] and URI (Uniform Resource Identifier)[2].

Unicode specifies a universal character encoding standard for the representation of text for computer processing [72] where it replaces the use of legacy character sets and allows data and text to be *exchanged internationally* between different systems [71, 72]. Bettels and Bishop [40] regard the emergence of the Unicode standard and the availability of supporting tools as some of the most significant global recent software technology trends.

URI endeavours to *uniquely identify* resources or objects with a character string and is therefore one of the cornerstone technologies of the Semantic Web [37]. The Semantic Web would be impossible without global identification of resources and hence the use of URIs. To enhance URIs, IRIs (Internationalised Resource Identifiers) are being developed. IRIs use Unicode encoding and will enable global identification of resources across language and character encoding boundaries [245].

A *resource* is defined as anything that has identity, and any discussion about meaning as attempted by the Semantic Web has to uniquely identify and encode the objects or resources of the discussion [37]. It is not possible to collaborate around *data with meaning* without uniformly representing and uniquely identifying the resources under discussion.

5.2.1.1 Layer 1 status

The Unicode Standard is specified and maintained by the Unicode Consortium [71]. The standard supports three encoding mechanisms, UTF-8,

[1] Refer to Section A.3.1 on page 288
[2] Refer to Section A.3.2 on page 290

UTF-16 and UTF-32, allowing the same data to be encoded in a byte, word or double word format (i.e. in 8, 16 or 32-bits per code unit) [72]. At present Unicode is specified and included as part of any emerging W3C standard implying that any application, parser or browser that adhere to W3C standards need to adhere to the Unicode standard [72]. In addition, the Unicode Standard was adopted by various industry leaders and is required by standards such as XML, Java, JavaScript, LDAP (Lightweight Directory Access Protocol) and CORBA 3.0 (Common Object Request Broker Architecture). In general it is supported in modern operating systems and browsers [72]

Both URL (RFC 1738) and URI (RFC 3986) are accepted Internet standards [37, 39, 136].

5.2.1.2 Layer 1 function

From the description of the technologies in the subsections above it is possible to derive that the overarching function of the Layer 1 technologies is to provide a *unique identification mechanism* for upper language technologies of the Semantic Web layered architecture.

5.2.2 Layer 2: Namespaces, XML and XML Schema

Layer 2 in V1 comprises of Namespaces[3], XML (Extensible Markup Language)[4] as well as XML Schema technologies[5] (Figure 5.1). In V2 in Figure 5.2, XML Schema was omitted, but it is possible to derive from a discussion by Berners-Lee [32] entitled *Standards, Semantics and Survival* that it is included under the 'XML' caption as a technology of layer two.

XML is a standard for the exchange of data over a network, notably the Web [61, 84, 159, 257]. XML Schema was developed as a content modelling language and an application of *XML*. An XML Schema describes a model for a whole class of XML documents [233, 234]. Namespaces (NS)

[3] Refer to Section A.4.1 on page 292
[4] Refer to Section A.4.2 on page 294
[5] Refer to Section A.4.3 on page 301

provides a simple method for qualifying element and attribute names used in XML documents [37, 47, 48].

In general XML together with XML Schema and the associated Namespaces can encode anything for which a grammar can be defined [61, 84]. If an XML grammar is accepted as a standard for data exchange by the different stakeholders, any XML parser can parse the XML data and access the content if it is a valid XML document. However, it is difficult to re-engineer the data model from any given XML document if the document type specification or schema is not available [84]. It is not possible to distinguish meaning from an XML document or XML Schema, the meaning or interpretation is provided by the application and end-user of the document [84].

It is also possible within the application space of the Web that two different grammars may define the same terms. This may result in element definition conflicts if the grammars are combined for any reason. XML Namespaces were defined to solve this problem as an element can be qualified against a namespace. Once a namespace is defined, any element used in conjunction with the namespace abbreviation becomes a unique element within the Web context [37, 47, 48].

5.2.2.1 Layer 2 status

The XML 1.0 Recommendation third edition was endorsed by the W3C in 2004 [49]. The W3C endorsed the *XML Schema Part 1: Structures* [234] and the *XML Schema Part 2: Datatypes* [235] in 2001. The *Namespaces in XML 1.1 W3C Recommendation* was accepted as a W3C recommendation in February 2004. The second edition of *Namespaces in XML 1.0* [48] was released as a W3C Recommendation in August 2006 and supersedes the first Recommendation of 1999 [46].

5.2.2.2 Layer 2 function

In V1 and V2 (Figures 5.1 and 5.2, p.105), XML along with XML Schema and Namespaces are depicted on the second bottom layer, thus providing

self-describing document functionality for upper layers. For the Semantic Web these technologies are used as a serialisation mechanism or a *syntax description mechanism* for data interoperability. This implies that an XML document and specification pair describes content entities, but it does not specify any meaning.

5.2.3 Layer 3/Layers 3a and 3b: RDF and RDF Schema

RDF (Resource Descriptive Framework)[6] and RDF Schema[7] technologies reside on Layer 3 in V1 and V2 (Figures 5.1 and 5.2). With the positioning of RDF Schema, Layer 3b, above RDF M&S (Model and Syntax), Layer 3a, in V2, Berners-Lee [32] emphasises the importance of a vocabulary description mechanism on top of the RDF data model as part of the Semantic Web language stack.

The W3C recommends the use of RDF as a mechanism to specify statements about resources with a basic data model. Furthermore, it describes the Resource Description Framework (RDF) as a language for representing meta-data or information about resources on the Web [231, 238]. RDF is intended for the exchange of meta-data about resources between applications, but *without loss of meaning* [86, 231].

In terms of the Semantic Web, the basic *object-attribute-value* data model is the only semantics prescribed in the RDF specifications [238, 239, 243]. RDF has no other data-modelling commitments and specifies no reserved terms for further data modelling or no other mechanisms for declaring property names [231, 239, 240]. For semantic interoperability RDF has significant advantages over XML primarily because of the data model used. RDF allows the definition of statements about resources that an application can process but the application may not actually *understand* the statements. The application provides the meaning [86].

RDF Schema extends RDF and is a W3C Recommendation that provides a technology that will assist in the definition of *meaning* residing on

[6] Refer to Section A.5.1 on page 316
[7] Refer to Section A.5.2 on page 323

top of the basic data structures of the Web by allowing the specification of domain vocabularies. RDF was not specified to describe properties, neither does it provide any mechanism for describing the relationships between properties and resources [231], which is the role of the RDF Schema (the RDF vocabulary description language). RDF Schema defines classes and properties that may be used to describe other classes, properties and resources [231, 241].

5.2.3.1 Layer 3 status

The W3C RDF Core Group has produced six documents forming the W3C RDF Recommendation. Each is aimed at a different audience. The *RDF Primer* [237] is an introduction to, and tutorial on how to use, RDF and RDF Schema. *RDF Concepts and Abstract Syntax* document [243] specifies the fundamental concepts and information model of RDF. The *RDF/XML Syntax Specification (Revised)* document [242] defines how to write RDF in XML syntax. *RDF Vocabulary Description Language 1.0: RDF Schema* [241] describes how to use RDF to describe application and domain specific vocabularies. *RDF Semantics* [239] defines the mathematically precise formal semantics of RDF and RDF Schema. *RDF Test Cases* [240] defines a set of test cases that illustrates aspects of the other specifications and may be used for the automatic testing of implementations.

With the release of the RDF and OWL as W3C Recommendations in February 2004 [244], the W3C intended the migration of the Semantic Web technologies from what was largely a research project to a more practical technology platform that enables more flexible access to structured data on the Web.

5.2.3.2 Layer 3 function

From the position of Layer 3 in V1 and V2 (Figures 5.1 and 5.2), as well as from the description of the technologies it is possible to derive that the function of the technologies on Layer 3 is to provide a meta-data description mechanism (including a data model) for the upper language technolo-

gies. The function of RDF is to provide a *meta-data data model*, whilst RDF Schema provides a primitive *knowledge representation* or ontology technology.

5.2.4 Layer 4/Layers 4a and 4b: Ontology Vocabulary/Ontology

In V1 (Figure 5.1) *Ontology vocabulary* is depicted on Layer 4, whilst this layer is separated as *Ontology*, Layer 4a, and *Rules*, Layer 4b, in V2 (Figure 5.2). OWL is the W3C technology representing an *Ontology vocabulary* or *Ontology* associated with this layer, whilst W3C research efforts aim to establish the technologies required for the implementation of the *Rules* also to be contained in this layer [130].

Ontologies assist in creating a common understanding for communication between people and computer applications [85]. Furthermore, it is envisioned that ontologies will play a crucial role in the processing, sharing, and reuse of knowledge between Web applications [53, 85]. On the Semantic Web, ontologies may be used in applications required to search across, or merge information from diverse communities [120–122]. RDF Schema is often described as a simple ontology language because it allows for the specification of simple semantics associated with identifiers. Using RDF Schema it is possible to define classes with multiple subclasses and super classes, as well as properties with sub-properties, domains, and ranges [19]. RDF Schema is however limited, because RDF Schema cannot, for example, specify that `Person` and `Car` classes are disjoint, or that a string quartet has exactly four musicians as members. In order to achieve interoperation between numerous, autonomously developed and managed ontologies, richer semantics are required and ontologies are thus layered above the RDF Schema layer [19, 133, 228].

5.2.4.1 Layer 4 status

The OWL specification as contained in the W3C OWL documentation set [20, 63, 120, 158, 192, 215] was endorsed as a W3C Recommendation in February 2004. With this Recommendation the W3C intended to draw at-

tention to the OWL specification and to promote its widespread deployment [120].

Furthermore, W3C research efforts aim to establish the technologies required for the implementation of rules to be contained in this layer [130]. As yet no formal submissions in this regard were made by the W3C RIF (Rule Interchange Format) Working Group [132, 254].

5.2.4.2 Layer 4 function

On Layer 4 in V1 and V2 it is noted that the terminology of the labels differs from the three preceding layers, because the *functionality* of the layer rather than the *technology* is mentioned.

The function of Layer 4 is the provision of technologies that will assist in the establishment of common ontologies (or knowledge representation formalisms) and a shared understanding of domain concepts on the Semantic Web. In addition, *Rules* will add the necessary functionality to process inference results from the ontology layer, and would typically be used to capture business process rules.

5.2.5 Layer 5: Logic/Logic framework

The versions of the Semantic Web architecture in V1 and V2 depict *Logic* or *Logic framework* residing above the ontology layer.

Logic is regarded as the foundation of knowledge representation languages, and it is necessary for the provision of the highly expressive language constructs in which knowledge can be captured in a transparent manner [157]. A logic framework provides a well-established formal semantics which assigns unambiguous meaning to logical statements. Without a logic framework, inferencing on the Semantic Web will not be possible [84].

A logic language that enables the necessary expressiveness required to implement the Semantic Web is one of the active focus areas of Semantic Web research at present. RDF as well as RDF Schema with its

extensions to RDF, support the notion of machine-processable information on the Web with the data model of resources and relationship between resources [85], but neither have the necessary expressive power to enable the reasoning required for the Semantic Web [184]. OWL provides basic vocabularies for describing ontologies and thus supports the specification of class hierarchy relationships among data elements in order to support certain kinds of logical inferences. In fact, OIL and DAML+OIL and subsequently OWL all support logic, more specifically DL or Description Logic as part of their specification [100], thus inheriting from Description Logics its formal semantics and reasoning support [54].

The V1 and V2 versions of the Semantic Web architectures depict *Logic* or *Logic framework* residing above the ontology layer even through *Logic* is a central aspect of an ontology language such as OWL [191]. Therefore, the positioning of this layer represents the notion that a richer logical language should be provided on top of an ontology language, which provides additional mechanisms for the capturing of reasoning formalisms. Proposals for web logic languages may therefore employ a special semantics, such as minimal model semantics, to make inferencing more amenable to computer implementation.

5.2.5.1 Layer 5 status

No technologies are available at present to instantiate the envisioned functionality of Layer 5 within the Semantic Web. *Logic* or *Logic framework* is thus at present largely a research effort by institutions such as the W3C [191].

5.2.5.2 Layer 5 function

The function of *Logic* or *Logic framework* is the provision of a logic language or framework on top of an ontology language that allows for additional mechanisms to capture reasoning formalisms and logic language integration.

5.2.6 Layer 6: Proof

In the V1 and V2 versions of the Semantic Web architectures proposed by Berners-Lee (Figures 5.1 and 5.2), *Proof* resides on Layer 6.

Semantic Web proof scenarios are supported by the notion of *proof languages*, which determines the validity of specific statements. An instance of such a proof language scenario generally consists of a list of inference items used to derive the validity of the information in question, as well as the associated trust information of each item [7, 184].

5.2.6.1 Layer 6 status

The concept of *Proof* within the Semantic Web context is at present unattainable and is considered to be a futuristic concept beyond current research efforts [191].

5.2.6.2 Layer 6 function

The function of *Proof* is to provide a mechanism (generally a list of inference items) that are used to determine the validity of a specific statement.

5.2.7 Layer 7: Trust

Trust resides on Layer 7 in V1 and V2 in Figures 5.1 and 5.2.

On the Semantic Web, *Trust* provides a mechanism to establish the trust levels of acquired information or participants in any interaction. The trust levels of information depend on (1) the source of the information, (2) whether the source can be trusted, as well as (3) whether the source is who or what it claims to be, in other words, the authenticity and trustworthiness of the source [223]. It is foreseen that the *context* of the information would assist applications and users of information with aspects regarding the trustworthiness and usefulness of the information [41, 222]. Therefore the *information context* would create a conceptual environment where the

Semantic Web operates and interacts intuitively, without having to rely on complex authentication and checking [184].

5.2.7.1 Layer 7 status

The concept *Trust* within the context of the Semantic Web is at present considered to be unattainable and beyond current research [191].

5.2.7.2 Layer 7 function

The function of *trust* is the provision of a mechanism to establish trust levels of all role-players and information items on the Semantic Web.

5.2.8 Vertical Layers: Signature/Digital Signature and Encryption

Digital Signature and *Signature* are vertical layers associated with Layers 3 to 6 in the V1 and V2 versions of the Semantic Web layered architecture (Figures 5.1 and 5.2). No description of the meaning or intention of either *Digital Signature* or *Signature* could be found in literature and for the purpose of this study, it is assumed that these layers are the same, especially because of their positioning as vertical layers in both V1 and V2.

The V1 version of the Semantic Web layered architecture (Figure A.2) does not depict *Encryption*. It was however added in the V2 version depicted in Figure 5.2 where it is associated with Layers 3 to 6, along with *Signature*.

A digital signature is an electronic signature that can be used to authenticate identity, while encryption is an effective way to achieve data security. Within the context of the Semantic Web a digital signature would be used as a mechanism to verify an identity (such as the author of a document) unambiguously [184].

Encryption is an effective way to achieve data security by obscuring information through and encoding mechanism to make it unreadable without

access to a special encryption key [196, 224].

5.2.8.1 Status of vertical layers

XMLDSig also called XML Digital Signatures or XML Signatures, is a joint IETF/W3C standard that specifies how to digitally sign and verify a signature of a XML data object [17]. XMLDSig enables digital signatures on arbitrary digital content (XML or non-XML) [148].

XMLEnc (XML Encryption) is a W3C standard that specifies how to encrypt/decrypt an XML-formatted data object [139]. XMLEnc supports end-to-end (as opposed to point-to-point) encryption of an XML object, which may be the whole XML document or a part it. The document can be transmitted in XML or non-XML syntax [139].

It is foreseen that XMLDSig will be used in many phases in Semantic knowledge management systems, such as authenticity verification for retrieved/updated knowledge and involved intermediaries, among others [17]. Furthermore, it is foreseen that encryption could be used in knowledge storage, internal/external knowledge transfer as well as authentication [32, 148].

Both XMLDSig and XMLEnc are accepted W3C standards. However, both are only specified with regard to XML security, even though these vertical layers are associated with layers three to six in the V1 and V2 versions of the layered architecture. The association or integration with the other layers of the Semantic Web architectures are unclear at present and it therefore remains a research endeavour [191].

5.2.8.2 Function of vertical layers

The function of both *Digital Signature/ Signature* and *Encryption* is the provision of security mechanisms that support the languages architecture. Digital signatures are used to verify any identity, and encryption is a mechanism used to encrypt/decrypt data transmitted between Semantic Web roleplayers.

5.3 STATUS OF SEMANTIC WEB TECHNOLOGIES

In this section, the status of the technologies covered in Section 5.2 is summarised. In particular, Figure 5.2 as an accepted and adopted version of layered architecture proposed by Berners-Lee [31] is adapted to reflect the status quo of these technologies in Figure 5.3. The proposed adaptations are:

- *OWL* replaces *Ontology* on Layers 4 and 4a,
- *Digital Signature* moves to Layer 1,
- Classification of the bottom four layers as *established technologies*,
- Classification of the top layers as *emerging functionalities*; and
- Classification of the bottom four layers as the *data layer* of the Semantic Web.

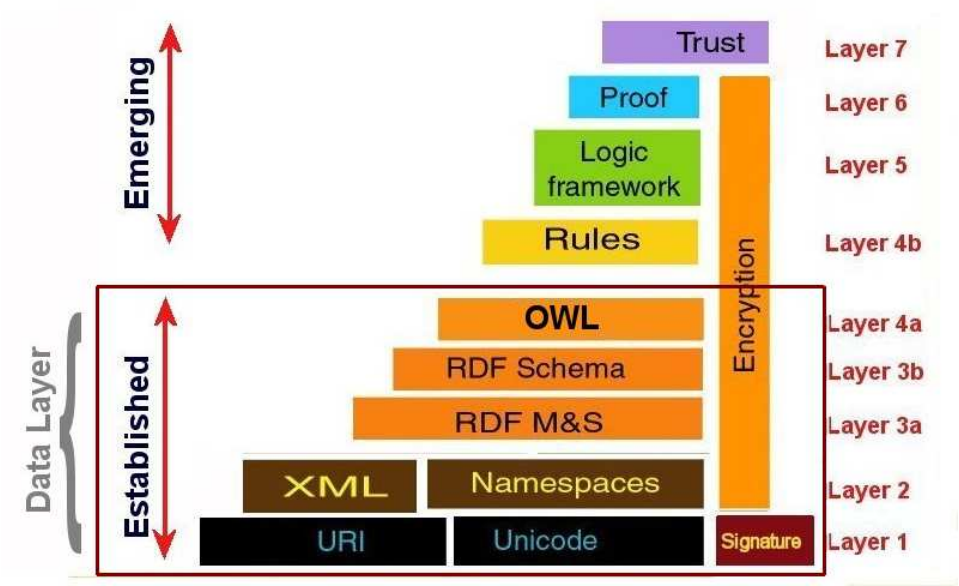


Figure 5.3: The modified Semantic Web status architecture

These adaptations with motivating reasons are discussed in Sections 5.3.1 to 5.3.5.

5.3.1 *OWL* replaces *Ontology* on Layers 4 and 4a

Considering V2 in Figure 5.2, it is noteworthy that the bottom four layers depict *technologies*, more specifically, *W3C Recommendations*. As mentioned in Section 5.2.4, OWL is the W3C technology representing *Ontology* on the V2 Layer 4a in Figure 5.2 in particular. This provides the motivation for one of the changes reflected in Figure 5.3, where *Ontology* is replaced by OWL in order to be consistent with the other W3C technologies represented in the four bottom layers.

5.3.2 *Digital Signature* moves to Layer 1

As discussed in Section 5.2.8, a digital signature is a mechanism used to verify an identity, such as the author of a document, unambiguously [184]. The concept *digital signature* and the use thereof is prevalent in e-commerce and trust negotiations application domains [125, 160].

However, within the context of the Semantic Web the notions of digital signatures, encryption, as well as it's associated roles are still vague [148, 191]. Since the function of Layer 1 technologies was identified to be the provision of a unique identification mechanism for the upper layer technologies, it provides motivation for *Digital Signature* to be incorporated as a technology into Layer 1 because *Digital Signature* as technology assists in the verification of identity. In addition, XML as an immediate upper layer technology on Layer 2, provides a mechanism by means of *XML signatures* to use digital signatures in XML transactions [213]. In a layered architecture, upper layers use functionality provided by lower layers. This provides further motivation for *Digital Signature* to move to Layer 1.

In addition, if *Digital Signature* moves to Layer 1, *Encryption* is layered on top of *Digital Signature*, which implies that *Encryption* uses functionality provided by *Digital Signature*. This is acceptable since encryption mechanisms often obtain their keys from a participants only after the identity of the participant was verified using digital signatures [224]. Furthermore, after the proposed adaptation *Encryption* also resides alongside XML, which is preferable to the previous models since XMLEnc is a W3C Recommen-

dation specified with regard to XML security [139]. The accommodation of *Encryption* into the model after the suggested adaptation provides an additional reason to move *Digital Signature* to Layer 1.

5.3.3 Classification of the bottom four layers as *established technologies*

The technologies of Layers 1 through 4a were adopted as specifications or W3C Recommendations, implying that these technologies are *established* technologies. This provides the motivation for classifying the bottom four layers as established technologies. The following list presents a summary of the status of each of these technologies at present:

- The Unicode Standard was adopted by industry leaders and is required by standards such as XML, Java, JavaScript, LDAP and CORBA 3.0. Unicode is specified as part of any emerging W3C standard. In particular, any application, parser and/or browser that adhere to the W3C standards should therefore adhere to the Unicode Standard. Refer to Section A.3.1 on page 288 and Section 5.2.1 on page 106.
- Regarding URI, RFC3986 currently allows for a subset of ASCII only, comprising about 60 characters. In contrast IRIs are being developed by the Internationalisation Activity of the W3C. IRIs are specified as a sequence of characters from the Universal Character Set (Unicode/ISO10646), which implies that an IRI can include characters from any of the written languages in the world. Refer to Section A.3.2 on page 290 and Section 5.2.1 on page 106.
- The current W3C Recommendation specifies the use of URLs to identify a *namespace* because URLs indicate domain names that should be unique throughout the Internet. However, it is envisioned that IRIs will eventually replace URLs as namespace identifiers. Namespaces are required because of naming conflicts that arise when different authors create grammars for meta-data using, for instance, elements with the same name. XML namespaces avoid naming conflicts when using and re-using multiple vocabularies because an element is qual-

ified against a namespace, thus making it a *unique* element. Refer to Section A.4.1 on page 292 and Section 5.2.2 on page 107.

- ▷ The first *XML* specification was accepted by the W3C in February 1998. The third edition was accepted as a W3C Recommendation in February 2004 [49]. Numerous XML-based languages and applications were developed by organisations that share high volumes of information. Refer to Section A.4.2 on page 294 and Section 5.2.2 on page 107.
- ▷ Regarding XML Schema, the XML 1.0 Recommendation was endorsed by the W3C in 1998. Furthermore, the W3C mandated the XML Schema Working Group to develop an XML Schema Language. In May 2001 the W3C endorsed the language specifications viz. *XML Schema Part 1: Structures* and *XML Schema Part 2: Datatypes*. Refer to Section A.4.3 on page 301 and Section 5.2.2 on page 107.
- ▷ RDF was released as a W3C Recommendation in February 2004 [244]. In doing so, the W3C endeavoured to establish a more practical technology platform as opposed to only a research project environment. This technology platform enables flexible access to structured Web data. The RDF specification set of the W3C provides an extensive overview of RDF. RDF is used to define meta-data vocabularies. Refer to Section A.5.1 on page 316 and Section 5.2.3 on page 109.
- ▷ RDF Schema is a W3C Recommendation [244] that provides a technology that will assist in the description of *meaning* using underlying basic data structures by allowing the specification of domain vocabularies. RDF Schema does not specify a vocabulary of application-specific classes and their associated properties, but it describes the mechanisms necessary to specify such a vocabulary. Refer to Section A.5.2 on page 323 and Section 5.2.3 on page 109.
- ▷ In February 2004 the OWL specification was endorsed as a W3C Recommendation. The current W3C OWL document set consists of six documents. OWL is intended as a Web ontology language, implying that it is specifically designed to be compatible with the architecture of the World Wide Web and Semantic Web [158]. OWL has already permeated the ontology engineering community and a num-

ber of existing OWL ontologies are available on the Web [79, 175]. Refer to Section A.6.1.1 on page 329 and Section 5.2.4 on page 111.

The lower layers of the V1 and V2 versions thus depict existing W3C Recommendations, which provides the motivation for the classification of the bottom four layers as *established technologies*.

5.3.4 Classification of the top layers as *emerging functionalities*

A distinction is made between *established technologies* and *emerging functionalities* in Figure 5.3 below. The *emerging functionalities* reside in the top layers, including layer 4b that contains *Rules*. Emerging functionalities are those functionalities that were identified as necessary to realise the Semantic Web, but which are presently predominantly research efforts. In contrast to this, established technologies are categorised to be those adopted as either W3C Recommendations or Specifications in Layers 1 through 4a.

In the classification of the top layers as *emerging functionalities*, it is necessary to discuss the functionality of these layers within the context of the Semantic Web. With regard to the *Rules* and *Logic framework* layers, it should be pointed out that reasoning and inferencing are two of the driving principles of the Semantic Web. New data is derived from existing data by means of *inferencing*. *Reasoning* deduct meaning and make decisions based on the acquired data that is represented with an ontology language and rules. An expressive, logic language that supports reasoning using the ontology language with the associated rules is required in order to implement the envisioned Semantic Web. Even though RDF together with its data model, as well as OWL, support the notion of machine-processable information on the Web [85], neither RDF nor OWL (residing on the data layer) has the necessary expressive power to enable the inferencing and reasoning required for the Semantic Web. Presently, the *Rules* and required *Logic framework* are active research focus areas of the Semantic Web domain [99, 113, 191].

Trust and *Proof* within the Semantic Web application domain are emergent research concepts [191]. However, both concepts are crucial towards the eventual Semantic Web realisation due to the inevitable and inherent contradictions and duplications in ontology definitions [184]. Applications on the Semantic Web presently depend upon *context* to manage trust and proof. It is an accepted requirement that proof checking mechanisms enhanced by digital signatures will be integrated into the eventual Semantic Web interaction and collaboration activities.

Since no technology instantiations are at present formally accepted for the implementation of Layers 4b to 7, these layers are classified as *emerging functionalities*.

5.3.5 Classification of the bottom four layers as the *data layer* of the Semantic Web

Ontologies that build upon the lower layers depicted in V2 in Figure 5.2 and that capture the meaning and meta-data of information are the realisation of the information space required by the Semantic Web, i.e. the existing technologies enable an information space usable by machines. It is proposed that the established technologies discussed in sections 5.2.1 to 5.2.4 but excluding *Rules* (depicted in Layers 1 to 4a in Figure 5.3), constitute the *data layer* of the Semantic Web. It is thus not unreasonable to state that *some* maturity can already be associated with the data layer of the Semantic Web.

However, an application is required to manipulate or use a data layer. Thus, in the same vein, a Semantic Web application is required to manipulate any ontology. In the vision of Berners-Lee [38], Semantic Web data (ontologies) are used by *agents*, but presently several unresolved issues cloud the realisation of this vision. This is supported by Patel-Schneider and Fensel [191], p.17 who state that:

> *... there are layering decisions to be made whenever a new representation formalism has to be compatible with an existing representation formalism. This will occur at other points of the*

semantic web tower, has occurred in the past, and will undoubtedly occur in the future.

This serves as a further motivation for the term *emerging* associated with the top layers (4b to 7) of the model in Figure 5.3.

5.4 THE STATUS MODEL AND THE V4 VERSION OF THE LAYERED ARCHITECTURE

The purpose of this section is to discuss the status model of Semantic Web technologies (Figure 5.3) with regard to the latest V4 version of the layered architecture proposed by Berners-Lee [35].

During 2005 and 2006 two additional versions (V3 and V4) of the layered architecture for the Semantic Web were presented by Berners-Lee [34, 35]. Because these V3 and V4 versions of the architecture are not yet adopted by Semantic Web authors in literature, the status model developed in the previous section used the accepted and adopted V2 version as basis. However, even though the latest versions are not adopted yet, the status of the technologies in these versions are investigated in order for this study to be complete. In particular, the latest V4 version of the layered architecture [35] is discussed because the term *status* implies that the most recent version of the layered architecture (V4) is applicable. The V4 version is reproduced as Figure 5.4 with added 'Layer' captions.

In order to discuss the status model with regard to the V4 version of the layered architecture, Figure 5.5 that depicts both the adapted status model of Figure 5.2 and the most recent version of the architecture (Figure 5.4), was compiled.

The adaptations made to V2 to compile the proposed status model are:

- *OWL* replaces *Ontology* on Layer 4a,
- *Digital Signature* moves to Layer 1,
- Classification of the bottom four layers as *established technologies*,
- Classification of the top layers as *emerging functionalities*; and

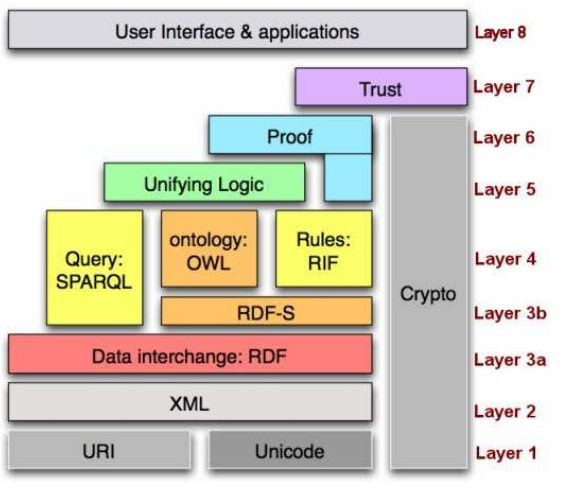


Figure 5.4: The V4 version of the Semantic Web architecture [35].

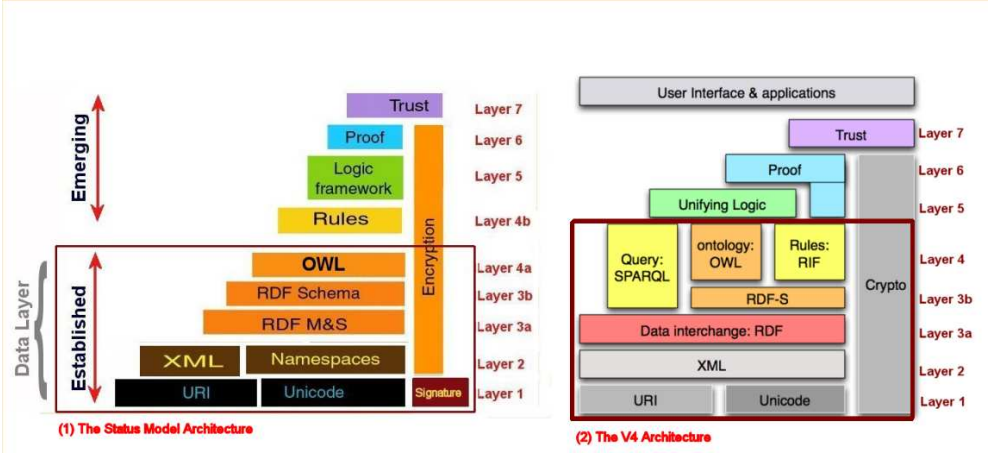


Figure 5.5: The status model and the V4 Semantic Web architectures.

▷ Classification of the bottom four layers as the *data* layer of the Semantic Web.

The suggestion that *OWL* replaces *Ontology* on Layers 4 and 4a in the status model (Figure 5.5:(1)) is supported by the V4 architecture since *ontology: OWL* is the caption for Layer 4 in Figure 5.5:(2). The movement of *Digital Signature* to Layer 1 is upheld since the V4 architecture in Figure 5.5:(2) depicts only one vertical layer, labelled *Crypto*.

The suggestion that the bottom four layers of the status model (Figure 5.5:(1)) are classified as *established technologies* while they also constitute the *data* layer of the Semantic Web is also further supported by V4 in Figure 5.5:(2) with the exception of SPARQL and Rules:RIF that are not yet W3C recommendations. These technologies were however submitted as W3C candidate recommendations and are therefore in the process of being considered full W3C recommendations. In addition, the top three layers in Figure 5.5:(2) can also be classified as *emerging functionalities* in support of the remaining suggestion with respect to the status model (Figure 5.5:(1)).

The *User Interface and applications* layer (Layer 8) in Figure 5.5:(2) is problematic since it neither depicts technologies nor functionalities required for the realisation of the Semantic Web, but rather indicates where Semantic Web applications would reside. It is as such inconsistent with the previous versions of the architecture as well as the status model. This inconsistency again emphasises issues with regard to some irregularities depicted in the versions of the Semantic Web layered architecture as discussed in the problem statement of the thesis (section 3.5).

5.5 LAYERED TECHNOLOGY FUNCTIONALITY

In order to answer the two questions formulated in Research Question 1 namely *What is the function of each technology included in the present versions of the Semantic Web layered architecture?* and *What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?*, the functionality of each layer of the Semantic Web architecture was extracted in Section 5.2. It is necessary to determine the functionality of the layers in the Semantic Web layered architecture that depict technologies in order to develop a *functional architecture*. The bottom layers of the different versions of the architecture depict technologies whilst the upper layers already depict functionality. The functionality of the different layers of the status model (Figure 5.3) thus determined is summarised in Table 5.1 below:

Layer	Functionality
Layer 1	Unique Identification
Layer 2	Syntax Description Language
Layer 3a	Meta-data Data Model
Layer 3b and 4a	Ontology
Layer 4b	Rules
Layer 5	Logic Framework
Layer 6	Proof
Layer 7	Trust

Table 5.1: Layer functionality

The function of Layer 1 was identified to be the provision of a unique identification mechanism for its upper layers that define meta-data for specific resources. The function of Layer 2 is to provide a syntax language for the meta-data, whilst Layer 3a specifies a data model for meta-data declarations. The function of Layers 3b and 4a is the definition of an ontology or meta-data formalism. The rules function depicted on Layer 4b allows for the management of derived information from the ontology layer, whilst the aim of a logic framework on Layer 5 is the integration of the different logic frameworks that might be used by lower layer up to this layer. The function of Layer 6 is the provision of mechanisms to provide proof for derived meta-data conclusions, and the function of trust on Layer 7 is the provision of trust mechanisms for a trust value that can be associated with specific meta-data data.

To conclude the research described in this chapter, Figure 5.6 depicts the status model with the extracted functionality as captions on the left-hand side. This model serves as a building block towards the establishment of a comprehensive and *functional* layered architecture for the Semantic Web.

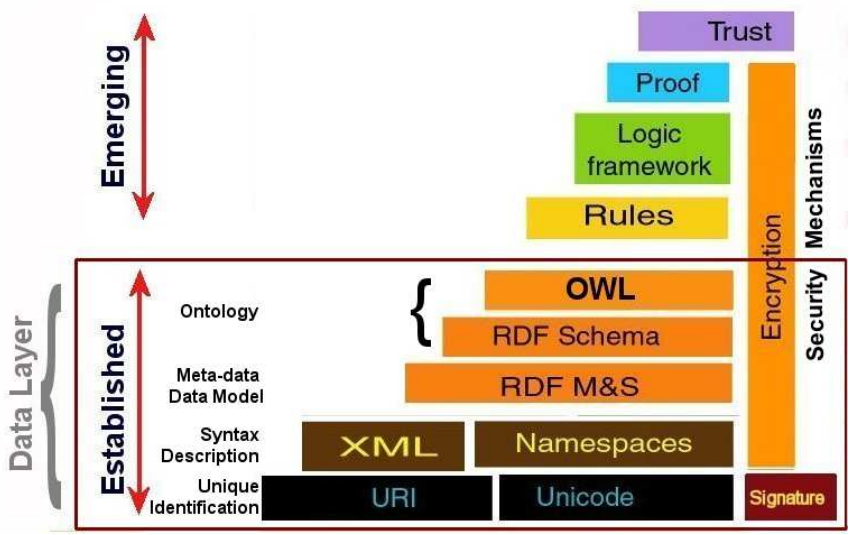


Figure 5.6: The Semantic Web status architecture depicting lower-layer functionality as well.

5.6 CONCLUSION

In this chapter, a discussion of the function and status of present Semantic Web technologies and functionalities, an adapted architecture status model reflecting the status quo of the technologies, and a status model depicting layered functionality, were provided. The status model reflects various refinements imposed by the status quo of current technology, and imparts some insight into the limitations of the technologies currently supporting the proposed Semantic Web layered architecture versions. In addition, the latest version of the architecture is discussed in relation to the status model.

However, the purpose of this chapter is to answer the sub-questions formulated for Research Question 1, namely *What is the function of each technology included in the present versions of the Semantic Web layered architecture?* and *What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?*. In answering this question it is required to determine the *status* of the listed Semantic Web technologies, as well as derive the *function* of the technology layers in the Semantic Web layered architecture. Figure 5.6 presents a functional

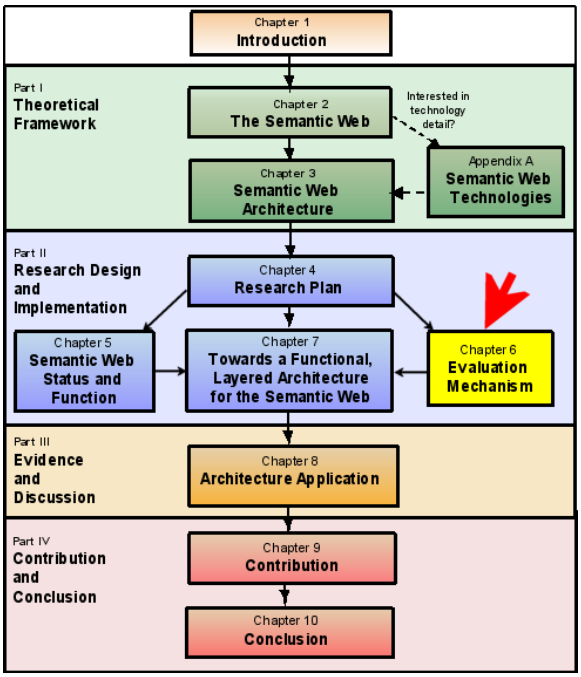
status model.

A summary of the work contained in this chapter and Appendix A was presented at the Ninth World Conference on Integrated Design & Process Technology in San Diego, California, June 25-30, 2006 and published in the proceedings [108].

In the next chapter, Chapter 6, the focus moves to the establishment of the characteristics to be included in the construction of a *comprehensive* architecture. Therefore, the purpose of Chapter 6 is to answer the sub-questions formulated for Research Question 2, namely *To which criteria should a layered architecture conform in order to adhere to system design principles?* and *Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?*

CHAPTER 6

TOWARDS AN EVALUATION MECHANISM FOR LAYERED ARCHITECTURES



Thesis Chapter Layout

Chapter Contents

Figures

Tables

6.1 INTRODUCTION

The goal of the research in this thesis is the development of a comprehensive and functional layered architecture for the Semantic Web. A *functional* layered architecture depicts the required functionality of the respective layers within the context of the system as discussed in the previous chapter, Chapter 5. In order to achieve the goal of a *comprehensive* (adhering to design principles) and *layered* architecture, it is necessary to find a mechanism based on design principles that can be used to evaluate layered architectures.

The research questions that are of concern in this chapter are *To which criteria should a layered architecture conform in order to adhere to system design principles?* and *Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?*.

Architectures and *layered architectures* within Software Engineering span a vast domain. In order to answer these research questions, subquestions were defined which are supported by facts, conclusions, discussions and investigations. This logical structure is graphically depicted in Figures 6.1, 6.2 and 6.3. In these figures, an orange oval depicts a fact or conclusion that serves as a starting point for further questions, discussions or investigations. A green rectangle depicts a task or activity that is executed to answer the question or to enter a discussion. A red hexagon depicts an *abstraction* or the crystallisation of concepts and definitions from a task or activity. These concepts and definitions contribute towards the research results of this chapter.

Figure 6.1 starts with the *fact* that the Semantic Web is a layered architecture as presented in Chapter 3. However, the precise meaning of such an architecture is unclear. Section 6.2.1 therefore comprises an investigation into the use of layered architectures within Computer Science and Information Systems. From this investigation it is concluded that there are several examples of layered architectures, however no concise definition could be found. This leads to the questions namely: *What is meant by layering?* as well as *What is a layered architecture?*. Both these ques-

tions result in further investigations of the concepts *layering* (Section 6.2.2) and *layered architectures* (Section 6.2.3), which concludes into both an extraction of the key concepts of layering, as well as a definition of layered architectures (section 6.2.4). A layered architecture is an architectural pattern or a *type-of* software architecture.

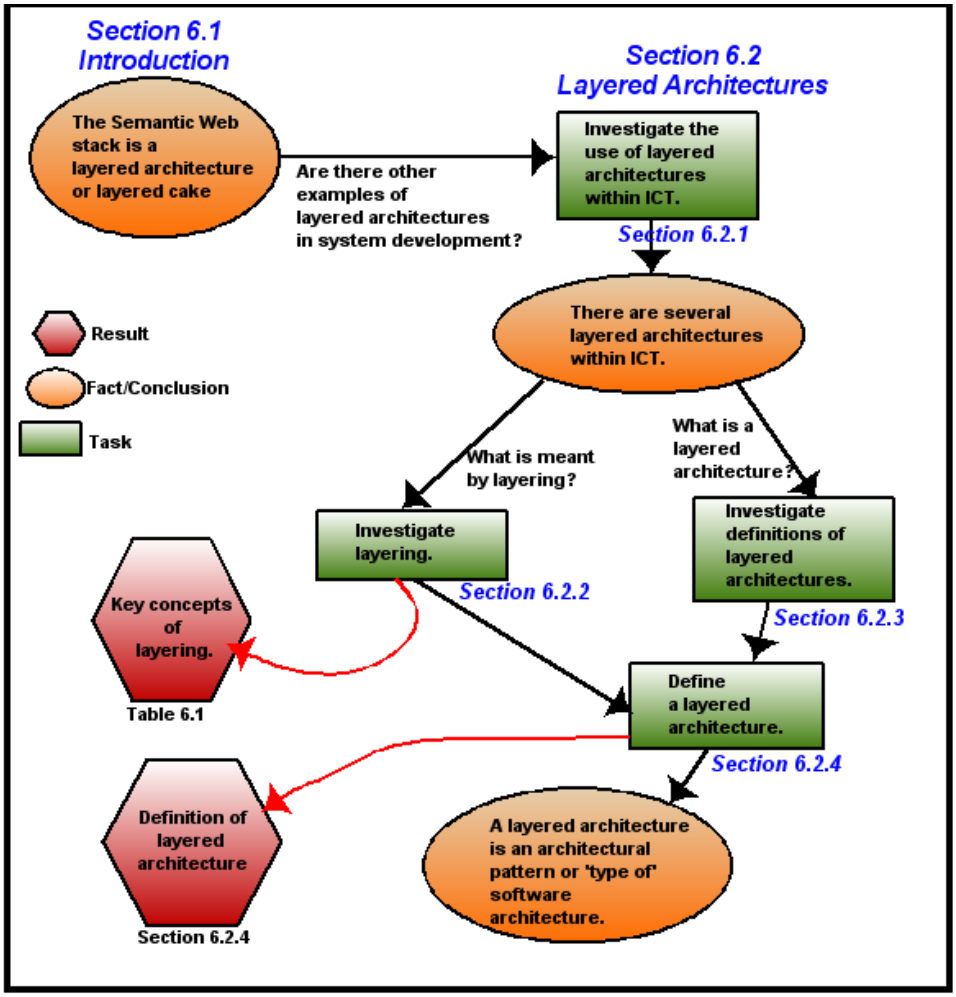


Figure 6.1: Layered architecture analysis

Figure 6.2 thus starts with the layered architecture definition (a result of the previous discussion), which is a *conclusion*. Three questions result from the definition namely *Where did architectures originate?*, *What is an architecture?* and *What is an architectural pattern?*, which in turn result in three investigations presented in Sections 6.3.1, 6.3.2 and 6.3.3. In section

6.3.4 an architecture is defined and in Section 6.3.3 the key concepts of architectures are extracted.

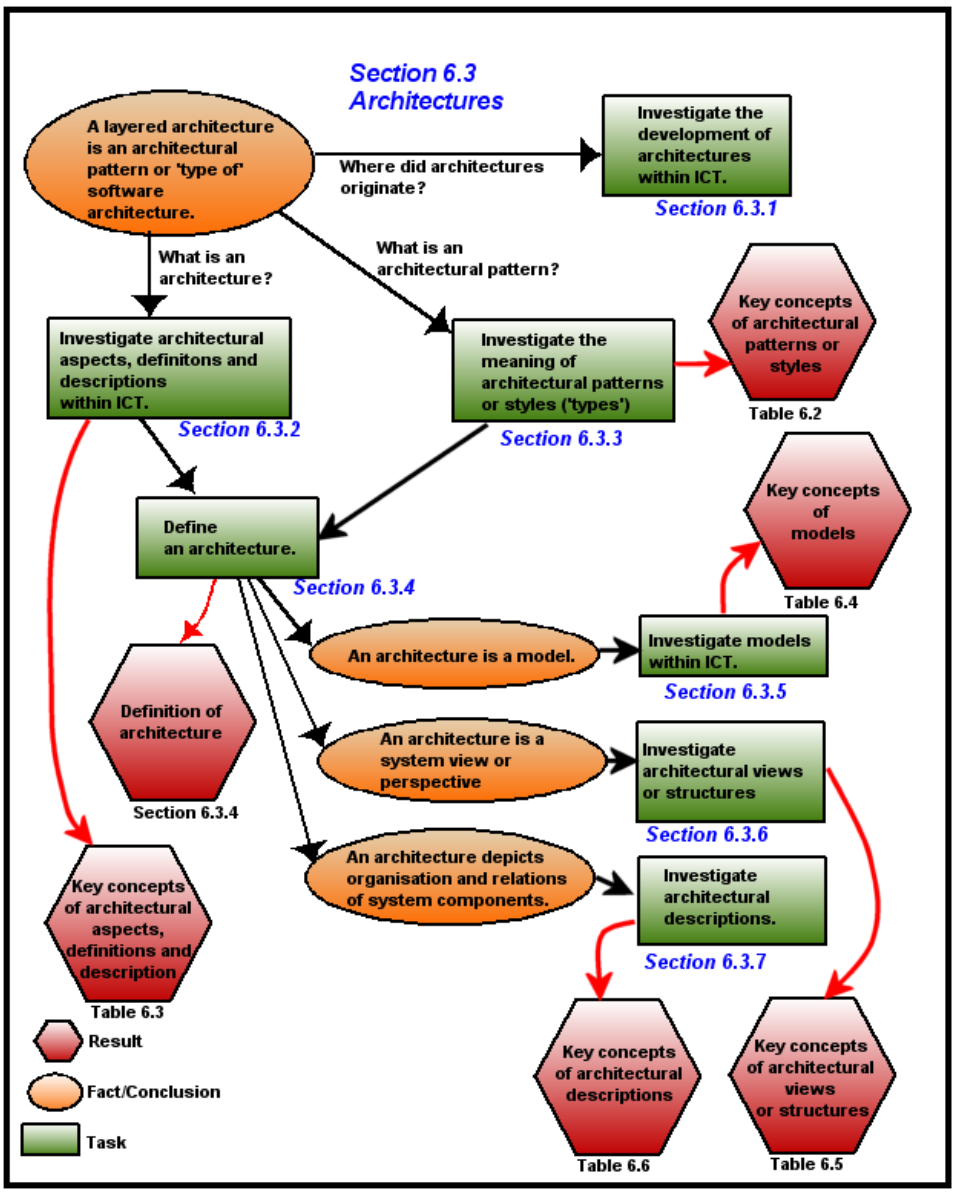


Figure 6.2: Architecture analysis

The architecture definition results in three conclusions: *an architecture is a model*, *an architecture is a system view or perspective* and *an architecture depicts the organisation and relations of system components*. These

three *facts* resulted in further investigations presented in Sections 6.3.5, 6.3.6 and 6.3.7, and in all these investigations, key concepts are extracted.

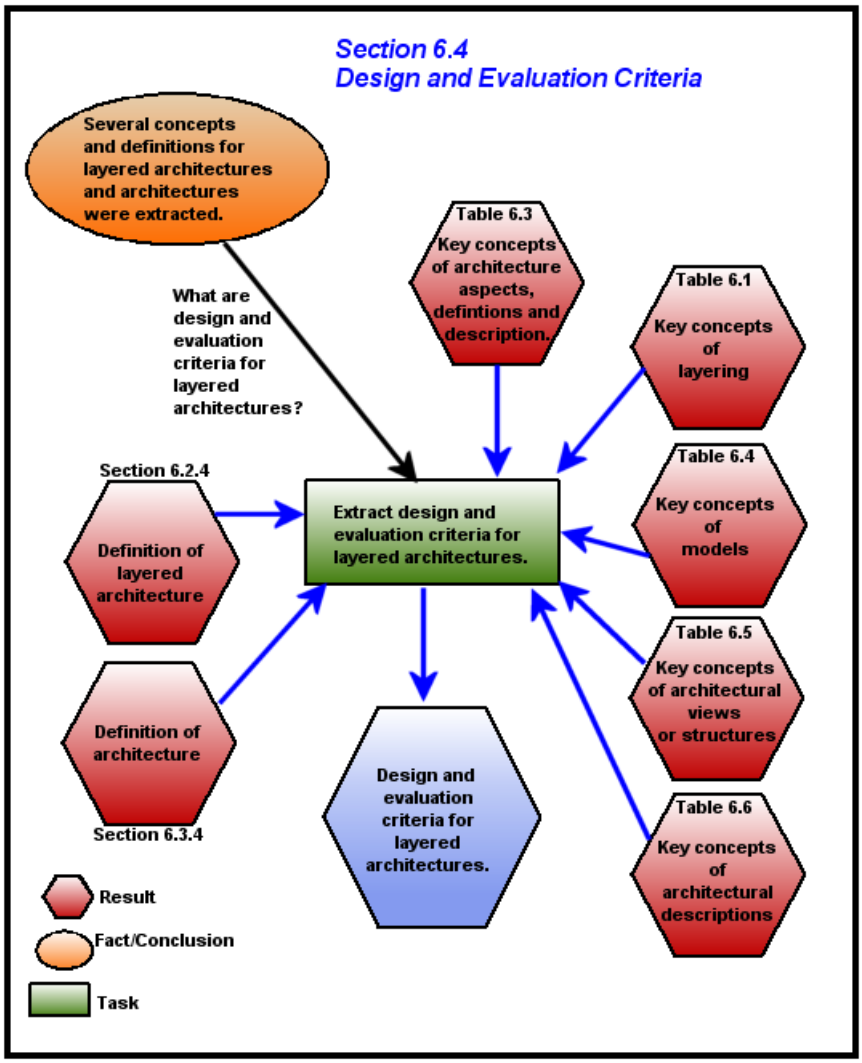


Figure 6.3: Criteria development

In Figure 6.3, all the definitions and key concepts from the previous investigations (repeated in the figure) are combined and the criteria for the evaluation mechanism are extracted (Section 6.4). As stated, the red hexagons in Figures 6.1 and 6.2 depict an *abstraction* or the crystallisation of concepts and definitions from a task or activity.

As discussed in the logical flow descriptions above, Sections 6.2 and

6.3 are devoted to investigating the concepts *layered architecture* and *architecture*. In these sections features and characteristics are identified and extracted from literature and best-practices. From these characteristics a framework of evaluation criteria (or an evaluation mechanism) for specifically layered architectures is developed (Section 6.4). This concludes the first part of this chapter.

However, it is required that the evaluation mechanism for layered architectures be calibrated in order to ensure that it is valid and useful. Therefore, the evaluation mechanism is applied to a prominent and accepted example of a layered architecture, namely the ISO/OSI network model [265] in Section 6.5. In the conclusion (Section 6.6) it is the contention that the use of these evaluation criteria provides insight into the architectural requirements of systems based on layered architectures.

6.2 LAYERED ARCHITECTURES

A *layered architecture* is often used in the modelling of systems by researchers and system architects in the design of information systems [55, 56, 117, 178, 195, 196]. In Section 6.2.1 a discussion of various system description examples that use layered architectures is presented.

6.2.1 Examples of the use of layered architecture within Computer Science and Information Systems

One of the first examples of a layered architecture presented in literature is that of Dijkstra's experimental layered operating system, developed at the Technische Hogeschool Eindhoven (THE). The goal of THE was to design and implement a provably correct operating system. Dijkstra held the notion that the isolation of various aspects of the operating system into distinct layers would result in such a system [88]. This pioneer operating system architecture lead to various layered architecture operating system models. In these operating systems (refer to Figure 6.4), developers isolated layers so that a specific layer only accesses its immediate neighbours

[55, 178, 196].

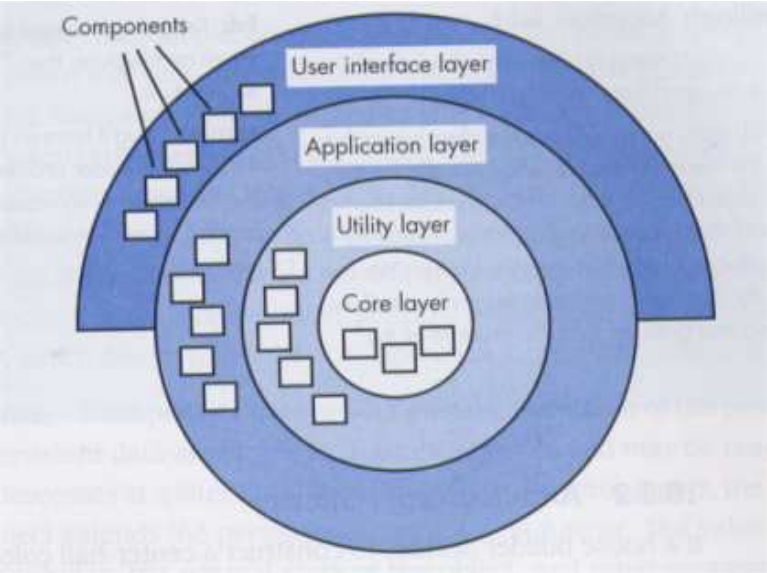


Figure 6.4: An operating system layered architecture [196].

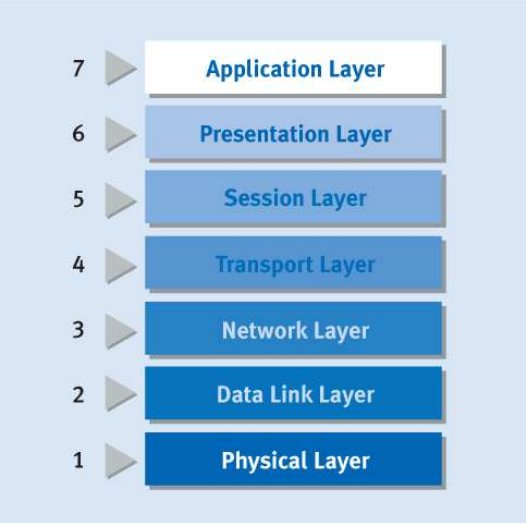


Figure 6.5: The ISO/OSI network architecture [265].

One of the best-known examples of layered architectures is the definition of network protocols found in the ISO/OSI (*International Standards Organisation/Open Systems Interconnect*) architecture (refer to Figure 6.5).

This architecture model defines all the methods and protocols required to connect computers by means of a network [265]. It separates the methods and protocols needed for network connectivity into seven different layers and each higher layer relies on services provided by a lower-level layer. The OSI model is an example of a closed layered architecture with low coupling because a layer may only access the layer immediately below it. A disadvantage of such a closed architecture is the introduction of a speed and storage overhead by each layer [56, 117, 178, 195].

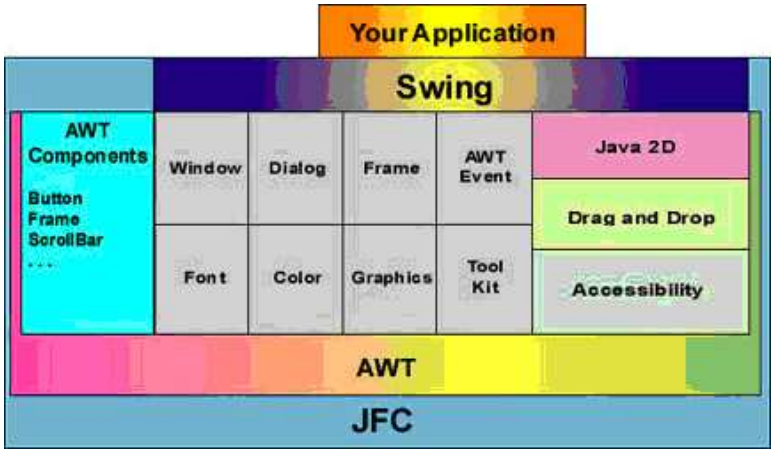


Figure 6.6: A JFC architecture [217].

To minimise the speed and storage overheads, *open* layered architectures were developed. An example of such an *open* layered architecture (where a layer can access the layer immediately below it, but also deeper layers) is the Java Foundation Classes or even the Swing user interface for Java [217]. The JFC (Java Foundation Classes) includes the Swing components that are regarded as an enhancement of the Java AWT (Abstract Windowing Toolkit) (refer to Figure 6.6).

Abmann [1] argues extensively for the use of ***layered*** *constraint frameworks* for Semantic Web application development. He maintains that the structure of layering ensures that, although upper layers require lower layers, every layer can be exchanged independently of each other. In addition, layered frameworks can be partially instantiated on every layer.

There are also several other examples of layered architecture usage.

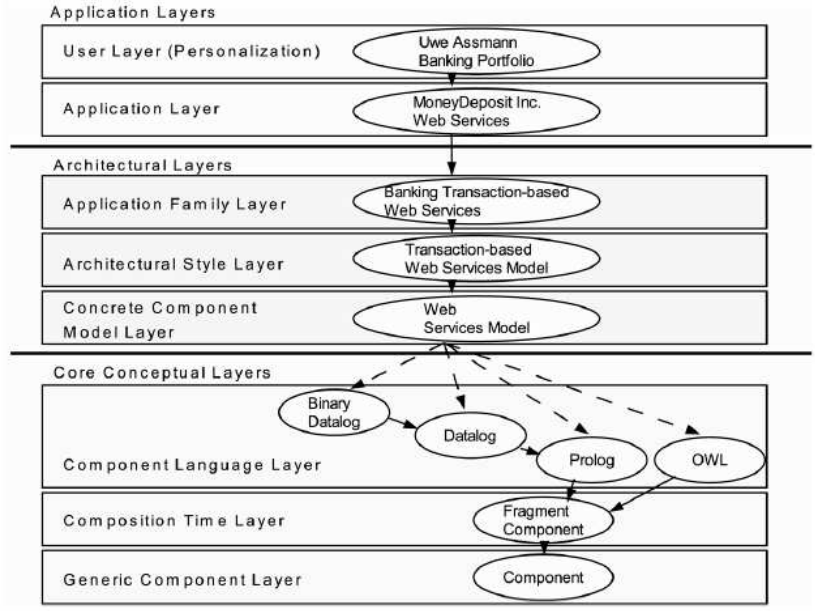


Figure 6.7: Possible component models in the layered framework [1].

The ISO/OSI network model comprises a formal specification. In contrast, the meaning of layered architectures in most other cases are implied. Simpson [214] uses a layered architecture to describe a real-time distributed computing system from the functional, design, distribution and execution viewpoints. Recently, Jeckle and Wilde use the ISO/OSI layered architecture to describe a web services protocol stack [143]. Zachman [264] introduced a layered framework for classifying and organising the descriptive models of an enterprise's architecture. Balakrishnan, Lakshminarayanan, Ratnasamy, Shenker, Stoica and Walfish [15] defines a layered naming architecture for the Internet. More related to the Semantic Web is the architecture for Semantic Web based knowledge management proposed by Davies, Fensel and van Harmelen [81].

It is not the intention of this section to provide an exhaustive list of the use of layered architectures within the Computer Science and Information Systems domain. However, from the preceding discussion it is clear that layered architectures are used extensively for modelling and demonstration purposes but without a precise specification of the intended meaning. The

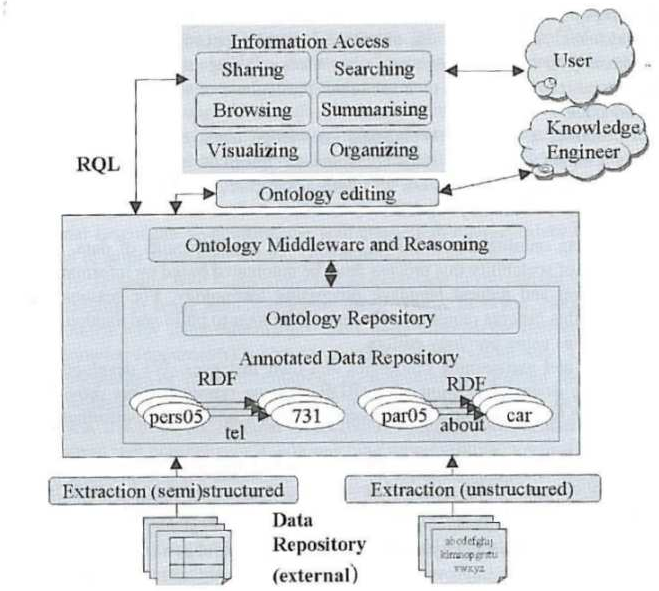


Figure 6.8: Architecture for Semantic Web based knowledge management [81].

intended meaning of layering in all these cases is implied and has to be derived by the user or reader. In the next section, the use of *layering* within software systems is investigated.

6.2.2 Layering

From the discussion in the previous section, it is apparent that layering is a common best practice pattern used by software architects to break apart complex systems [104]. In a layered architecture, the principal elements or components are arranged in the form of a layered cake where each layer rests on a lower layer. Generally, a layer represents a grouping of elements that provides related services. A higher layer may either use various services defined by the immediate lower layer or services by all the lower layers. However, the lower layers are unaware of higher layers [12, 56, 104, 178].

According to Nutt [178], layered architectures were inspired by the desire to divide and conquer system functionality by using horizontal parti-

tioning. The layers above build on and use the functionality of the lower layers. In addition, layered architectures are modular but with a particular set of constraints on how the modules are interconnected. Nutt maintains that the layered architecture uses abstraction to manage detail among the modules. The intellectual challenge to layered architectures is determining the order of the layers, as well as the content thereof [178].

Jacobson, Booch and Rumbaugh [142] define a layer as a set of subsystems that share the same degree of generality and interface volatility. Lower layers are common to several applications and should have more stable interfaces, whilst higher layers are more application-specific and may have less stable interfaces.

Bruegge and Dutoit [56] consider layering to be the result of a hierarchical decomposition of a system that may yield an ordered set of layers. In this latter method of decomposition, a layer is a grouping of subsystems that provides related services. In the resulting layered architecture, layers are ordered in that each layer can depend only on lower level layers and has no knowledge of the layers above it.

Bachman [12] uses guiding principles for system development. One of these principles, called the *Levels of Abstraction guiding principle* creates multilevel object-to-object type hierarchies. In addition, the *Layered architecture guiding principle* organises the multilevel object-to-object type hierarchies into layers with a clear separation of concerns and defined interfaces. The *Layered architecture guiding principle* also protects users in upper layers from the changes brought about by the re-implementation of lower layers. The only changes to have any impact upon users at the top is a new service that is introduced through new implementation or technology and for which an interface is made available at the upper layer.

Fowler [104] describes some of the benefits of decomposing a system into layers:

- A single layer is viewed as a coherent whole without knowledge of the other layers,
- The implementation of a specific layer can be substituted with alternative implementations of the same basic services,

- ▷ Dependencies between layers are minimised,
- ▷ Layers support standardisation because they define how layers, as well as their interfaces, should operate; and
- ▷ Several higher-level services can use a lower-level layer.

In contrast to these benefits, Fowler [104] lists some disadvantages including that layers cannot always encapsulate every functionality, which results in cascaded changes. In addition, the introduction of unnecessary layers influence performance.

6.2.2.1 Key concepts of layering

The key concepts of layering are summarised in Table 6.1 .

Layering is a common best practice pattern used by software architects to break apart complex systems.
Layering depicts system functionality using modular horizontal partitioning or decomposition, minimal dependencies between layers and a particular set of constraints on how the modules are interconnected.
A layer represents a grouping of elements/functionality/sub-systems/components that provides related services (thus implementing the *separation of concerns* principle).
A single layer is viewed as a coherent whole without implementation knowledge of the other layers (thus implementing the *tight coupling* and *cohesion* principles).
In a layered architecture higher layers use functionality provided by lower layers, either the immediate lower layer only or services by all the lower layers. Several higher-level services may use a lower-level layer but lower layers are unaware of higher layers.

Table 6.1: Key concepts of layering

6.2.3 What is a layered architecture?

A layered architecture is the resulting artefact of *layering*. Nutt [178] describes the design of layered architectures as a fundamental technique for dividing complex systems into manageable parts. Layered architectures are used to describe software or hardware hierarchies in computer systems, network protocols (such as the ISO/OSI model) and operating sys-

tem architectures.

A layered architecture is regarded by Pressman [196] as one of the architectural styles . He describes a layered architecture by referring to layered operating systems. The basic structure of this layered architecture is a number of different layers, defined so that each accomplishes operations that progressively approximate the machine instruction set of the CPU. At the outer layer, components service user interface operations. At the inner layer components perform operating system interfacing. Intermediate layers provide utility services and application software functions (refer to Figure 6.4 on p. 138).

Jacobson et al. [142] define a layered architecture as one of several architectural patterns such as the broker pattern for managing object distribution, as well as the client/server, three-tier and peer-to-peer patterns for the design of systems on top of hardware. The layered architecture or layers pattern is applicable to many types of systems. It is a pattern that defines how to organise a design model in layers, implying that components in one layer can reference components only in layers directly below. Layers reduce dependencies and support reuse because lower layers are not aware of details in upper layers.

As mentioned in the previous section on layering, a layered architecture can be *open* or *closed*. In a closed architecture, each layer can access only the layer immediately below it such as for example the ISO/OSI network model [265]. In an open architecture a layer can access the layer below it but also deeper layers. An example of an open layered architecture is the Swing user interface for Java [217]. Closed architectures have the advantage that they lead to low coupling between subsystems and subsystems can be integrated and tested incrementally. Each level however introduces a speed and storage overhead and it may be difficult to add functionality in later revisions [56, 117, 195].

6.2.4 Definition of a layered architecture

From the preceding discussion about layered architectures, the following definition for a layered architecture is compiled:

> *A layered architecture is the resulting artefact of the* layering *technique that is an accepted best practice used to break apart complex systems. Layering performs horizontal decomposition of system functionality. A layered architecture is therefore regarded as an architectural pattern or* type-of *architecture.*

The conclusion that a layered architecture is a *type-of* architecture implies that it is required to investigate the concept *architecture*.

6.3 SOFTWARE ARCHITECTURES

The logical flow of this section is graphically depicted and discussed in Section 6.1 as Figure 6.2 (p.135). As concluded in the previous section, the layered architecture definition indicates that a layered architecture is an *architectural pattern* or *type-of* software architecture. This conclusion results in three investigations, namely *Where did architectures originate?* (Section 6.3.1), *What is an architecture?* (Section 6.3.3) and *What is an architectural pattern?* (Section 6.3.2).

6.3.1 Where did architectures originate?

Although the concept *architecture* in software systems was not formally defined with the introduction of *structured programming*, it was implied in the work of pioneers such as Parnas and Dijkstra [89, 187, 258]. These pioneers derived techniques to model a system as consisting of different and related components. Dijkstra was mainly concerned with program clarity and correctness, and hence a program's structure [88]. Parnas introduced the concept of *modularisation as a mechanism for improving flexibility and comprehensibility of a system while allowing shortening of its development*

time [186]. Parnas also introduced the idea of *a family of programs* rather than a single program [187]. As computer programs became increasingly complex, the separation of related functionality into sub-programs or modules was forced upon developers. The resulting program structure enabled the management of system complexity and the verification of program correctness [188].

The implementation of software as modules invoked related issues such as *low coupling*, *high-cohesiveness* and *the separation of concerns* that are adopted at present as best practices within system design [188, 202]. The abstraction of software system functionality into modules, together with its interfaces is in essence *the determination of the architecture of the system*.

Garlan and Shaw [105], p.5 summarise architecture design in the next quotation.

> *As the size and complexity of software systems increases, the design problem goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organisation and global control structure; protocols for communication, synchronisation, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives. This is the software architecture level of design*.

Modern information systems are complex, intricate and large. There is consensus among developers that the architecture of a system is indispensable towards the understanding, conceptualisation, design and development of the information system. Bass, Clements and Kazman [18] provide three reasons why architecture is important:

- Representations of software architecture enable communication between all parties (stakeholders) interested in the development of a computer-based system.

- ▷ The architecture highlights early design decisions that will have a profound impact on all Software Engineering work that follows and, as important, on the ultimate success of the system as an operational entity.
- ▷ An architecture constitutes a relatively small, intellectually graspable model of how the system is structured and how its components work together.

Similarly, Jacobson et al. [142] assert that an architecture is required in order to understand the system, to organise development, to foster reuse and to evolve the system.

In conclusion it is reasonable to state that software systems made use of architectures since the first program was divided into modules by pioneers such as Dijkstra and Parnas. As system size and complexity increased, programmers were required to decompose systems into modules or components and as a result, they had to define the interfaces and interactions among these modules, as well as the global properties of the assemblage. Historically, architectures have been implicit or accidents of implementation by skilled developers [196]. However, as systems became increasingly complex and the task of building the software becomes exponentially harder, there is consensus that the specification of system decomposition is critical: hence the emphasis on the definition of the software architecture of the system [56, 104].

6.3.2 What is an architectural pattern? (Architectural patterns and styles)

Due to the progression in the design and development of architectural models, some architectural recurrences evolved. These are described as *architectural patterns* [18], also referred to as *architectural styles* [196].

There is no generally accepted definition of a *pattern*. Christopher Alexander coined the phrase *pattern* as follows [3]:

> *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the*

> *solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*.

A pattern is thus the description of a problem that occurs repetitively within a specific environment, as well as the core of the solution to that problem in such a way that the proposed solution may be reused. Patterns are rooted in practice and are referred to as *best practice descriptions*. The focus of a pattern is a specific solution to a recurring problem [104].

Bass et al. [18] define an architectural pattern as a description of element and relation types together with a set of constraints on how these may be used.

The term *architectural style* is often used in literature when reference is made to an architectural pattern. Garlan and Shaw [105] defines an architectural style as a family of system descriptions in terms of a structural organisation. An architectural style determines the vocabulary of components and connectors that may be used in instances of that style, together with a set of constraints on how these may be combined. These constraints may be so specific as to include topological constraints on architectural descriptions.

The remainder of this section summarises several of the most prevalent architectural styles and patterns. It is beyond the scope of this discussion to give a comprehensive overview of the different architectural patterns described in literature. It suffice to note that the *layered architecture* is regarded as an architectural pattern or style. It is therefore a *kind-of* architecture and has to conform to the definition of an architecture as given in Section 6.3.3. In addition, layered architectures as a pattern provide best-practice solutions to recurring problems as discussed in Section 6.2, p.137.

6.3.2.1 Client/Server architectural pattern

In the Client/Server architectural pattern, a server provides services to clients [56].

The separation of functionality into components that could either be executed centrally or with specific focus such as interaction with the user, resulted in the client/server architecture.

The term *client/server* was first used in the 1980s in reference to computers on a network. The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability, and scalability as compared to centralised, mainframe, time-sharing computing [200].

A client is defined as a requester of services and a server is defined as the provider of such services. For example, a client (such as a user's email-package or a web-browser) requests a service from a server, i.e. software running centrally on another device (such as an email-host or web-server). Server software components generally execute on a large central device, whilst a client generally operates on a small remote PC. The nature of coordination and control thus changed: the masters had become servers, and the slaves had become clients [69, 142]. Examples of the client/server architecture (obtained from Clarke [69]) are presented in Figures 6.9 and 6.10.

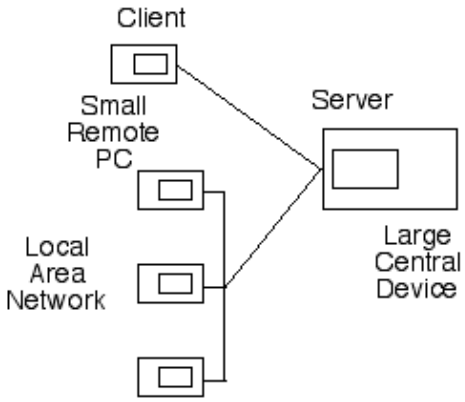


Figure 6.9: Client/Server architecture: several (small) client PC's interact with a (large) server through a local area network [69].

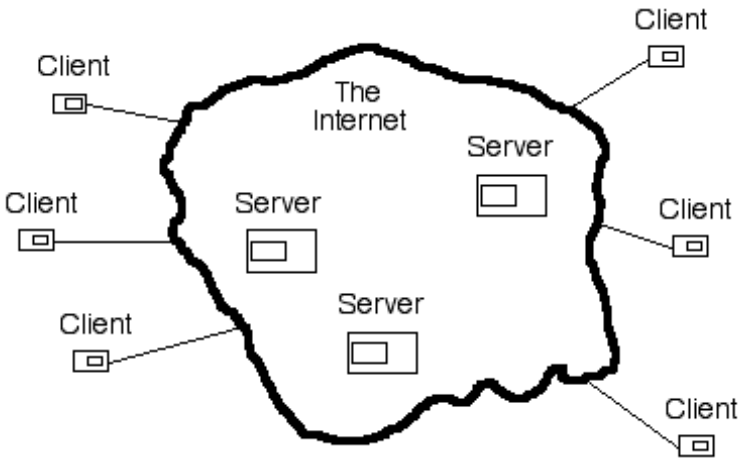


Figure 6.10: Internet Client/Server architecture: clients use the Internet as network to access server functionality [69].

6.3.2.2 P2P (Peer-to-Peer) architectural pattern

P2P (peer-to-peer) is regarded as a generalisation of the client/server architectural pattern in that a subsystem may act either as a client or a server [56, 142]. More specifically, P2P is a distributed computer architecture that is designed for the sharing of computer resources (e.g. content, storage, CPU cycles) by direct exchange, rather than requiring the intermediation or support of a centralised server or authority. P2P architectures are characterised by their ability to adapt to failures and to accommodate transient populations of nodes while maintaining acceptable connectivity and performance [5].

An example of the P2P architecture (obtained from Clarke [69]) is presented in Figure 6.11.

6.3.2.3 Three-tier architectural pattern

The three-tier software architecture (also called three layer architectures) emerged in the 1990s. In this architecture a third tier (middle-tier server) is located between the user interface (client) and the data management (server) components. This middle tier provides process management for the execution of business logic and rules [201]. Thus, the three-tier archi-

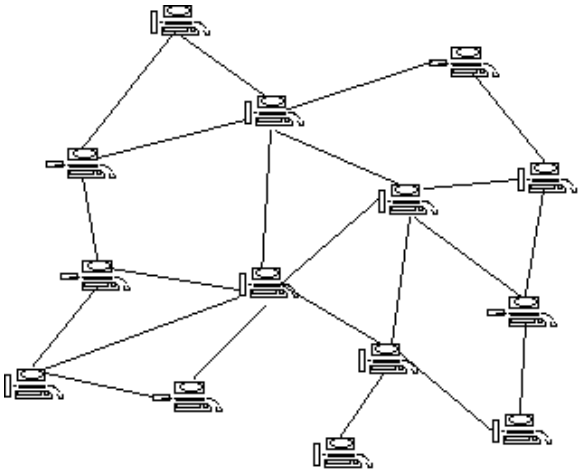


Figure 6.11: P2P architecture [69].

tectural pattern organises subsystems into three layers with different functions. Typically one of the layers would interact with the user, one layer would manage domain knowledge and another layer would contain additional functionality such as storage [56, 104, 142].

The three-tier architecture is used when an effective distributed client/server design is required that would provide increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user [201]. These characteristics have made three-tier architectures a popular choice for Internet applications and net-centric information systems.

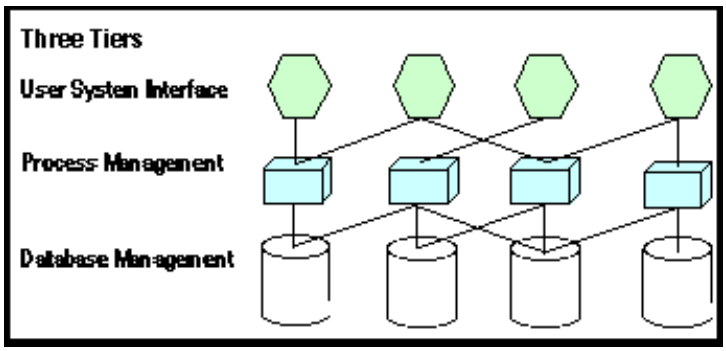


Figure 6.12: Three-tier architecture [201].

Several variations of the three-tier architecture have emerged, such as the MVC (Model/View/Controller) architecture. The *model* sub-system maintains domain knowledge, *view* sub-systems display it to the user and *controller* sub-systems manage sequences of interactions with the user [56]. An example of the three-tier architecture (obtained from Sadoski and Comella-Dorda [201]) is presented in Figure 6.12.

6.3.2.4 Layered architecture or layers pattern

The layered architecture or layers pattern is a pattern that organises functionality in layers, implying that components in one layer can reference components only in layers directly below. Layers reduce dependencies and support reuse because lower layers are not aware of details in upper layers. [142]. Pressman [196] regards the layered architecture as an architectural style. The concept *layered architectures* was discussed in more detail in Section 6.2.3 on p. 143.

6.3.2.5 Key concepts of architectural patterns and styles

Key concepts of architectural patterns and styles are summarised in Table 6.2.

Architectural patterns or architectural styles are architectural model recurrences.
A pattern is a description of a problem that occurs repetitively within a specific environment, as well as the core of the solution to that problem in such a way that the proposed solution can be reused.
The *layered architecture* is an architectural pattern or style. It is therefore a *kind-of* architecture that has to conform to the definition of an architecture.

Table 6.2: Key concepts of architectural patterns and styles

6.3.3 What is an architecture? (Architecture concepts, definitions and descriptions)

During information system development, the conceptualisation, design and development of the system is assisted by means of the *architectural descriptions* of the system [18]. Because of this, discussions recording architecture concepts abound in literature, however, the term *architecture* seems to defy the creation of a common, agreed definition within the Information System application domain both in literature and practice descriptions. Even though comprehensive lists of architecture definitions are available[1], no single accepted definition of software architecture is agreed upon [51, 205].

There is a considerable body of work on the topic of software systems architectures[2] as well as the form of descriptive terms used informally to describe systems. However, there is not yet a well-defined terminology or notation to characterise architectural structures. At present proficient software engineers and developers make use of architectural principles when designing complex software and they have often adopted one or several architecture patterns as strategies for system organisation. Such patterns are generally applied in an informal manner and there are no means to make it explicit in the resulting systems. Many of the principles represent rules of thumb or idiomatic patterns that have emerged informally over time. Others however, are more carefully documented as industry and scientific standards [105, 196].

In this section, architectural descriptions and definitions are summarised. In addition the key concepts of these architecture definitions are listed and in conclusion, a definition is adopted for the purposes of this thesis.

Avison and Fitzgerald [10] define an architecture as a *model* of a sys-

[1] See `http://www.bredemeyer.com/definiti.htm` and `http://www.sei.cmu.edu/architecture/definitions.html`

[2] The distinction between software architecture, systems architecture or software systems architecture is a matter of perspective. This concept is discussed in Section 6.3.6.1 on p. 162.

tem in the given context, where a model is an abstraction of a real-world representation. An architecture thus indicates a simplified representation of the essential aspects at *a specific level*. A model provides a means to view *only* the significant aspects of the entire system from the perspective of the viewer. Software or system architects present the system from different perspectives to understand the design better. These perspectives are elucidated views of the models of the system.

In support of the notion of levels, perspectives or views, Pressman [196], p.289 holds that architectural design is accomplished using four distinct steps. First, the system should be represented in context where the external entities that the software interacts with and the nature of the interactions are defined. Once context has been defined, top-level abstractions or archetypes that represent the pivotal elements of the system's function are specified. Components are defined within these abstractions. Finally specific instantiations of the architecture are developed to prove the design in a real world.

Bass et al. [18], Fowler [104] and Jacobson et al. [142] use the notion of multiple architectures or structures defined by views which implies that an architecture is defined within a *specified context*. The context determines the important aspects of the system at a specific level, as well as the components necessary to realise the system, the properties of components and the relationships between components and external entities.

However, it is still not clear what an architecture depicts, and the remainder of this section is devoted to a discussion of architecture definitions.

Bass et al. [18] defines the software architecture of a program or computing system as a description of the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Furthermore:

- An architecture describes elements and how elements relate to each other. It omits non-relevant information and is therefore an *abstraction* of the system.
- *External visible properties* are the assumptions other elements can make of an element, such as services it provides or its performance

characteristics.

- ▷ Systems comprise more than one structure and no structure alone is *the* architecture.
- ▷ All systems have a architecture because any system can be viewed as a composition of elements with relationships among them.

Pressman [196] states that the architecture of a system comprises of a comprehensive framework that describes the form and structure of the system. This description includes the system components and how they fit together. Architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the system, and the interrelationships that occur among all architectural components of the system. Software architectures provide a holistic view of the system to be built. It depicts the structure and organisation of software components, their properties, and the connections between them.

Bruegge and Dutoit [56] include system decomposition, global control flow, handling of boundary conditions, and sub-system communication protocols into a software architecture.

According to Fowler [104] an architecture has two common elements: one is the highest-level breakdown of a system into its parts; the other, decisions with regard to system functionality implementation that are hard to change. In addition, the identified components should be modular which means that the implementation of the component could change without having an effect on the functionality of the system.

To support the notion of *decisions*, Jacobson et al. [142] define software architecture in the Unified Process as encompassing significant decisions about:

- ▷ the organisation of a software system,
- ▷ the structural elements and their interfaces that will comprise the system, together with their behaviour as specified in the collaborations among those elements,
- ▷ the composition of the structural and behavioural elements into pro-

gressively larger sub-systems; and

- ▷ the architectural style that guides this organisation: the elements and their interfaces, their collaborations, and their composition.

The relevant aspects of an architecture are illustrated in the documentation template developed by Bass et al. [18]. This template was developed to document a software architecture. In the first instance the architect has to choose the *view*. This step is arguably considered to be the most important concept associated with the documentation of an architecture. The next action is to document the view using the seven headings of the template namely the primary presentation, the element catalog, the context diagram, the visibility guide, the architecture background, the glossary of terms and other information. The view description documents the structural aspects of the architecture. In addition to the view documentation, the behaviour in an architecture as well as the interfaces have to be captured in the description. An interface is a boundary between two independent components. The interface to a component remains constant but the implementation of the component may be altered. This supports the notion of modularity. Detail of the documentation template for views and interfaces are contained in Section 6.3.7 on p. 163.

6.3.3.1 Key concepts within architecture description and definition

The key concepts of architecture description and definition are summarised in Table 6.3.

An architecture is a model. Models are used to manage complexity with a simplified representation of a complex idea or application.	6.3.5, p.158
An architecture is a model or representation of a system within a specific context, or from a specific view or perspective.	6.3.6, p.161
A system has multiple architectures, and the view of what is architecturally significant is one that could change over a system's lifetime. An architecture is subjective and depicts whatever is perceived to be important at a specific time or in terms of a certain view.	6.3.6, p.161; 6.3.7, p.163

An architecture depicts the structure and organisation of the software components of which the system comprise at a certain level, as well as their properties and how they relate.	6.3.3, p.153
An architecture is a model of a system within a specified context, depicting the components necessary to realise the system from a particular perspective. The architecture of a system includes the organisation or structure of the identified components, their defining features or properties, as well as the relationships and interfaces between components and outside entities. Components should be modular.	6.3.7, p.163

Table 6.3: Key concepts within architecture description and definition

6.3.4 Definition of an architecture

From the discussions in the previous sections, a definition for the term *architecture* is adopted, namely:

> *An architecture is a model of a system within a specified context, depicting the components necessary to realise the system from a particular perspective or view. The architecture of a system includes the organisation or structure of the identified modular components, their defining features or properties, as well as the relationships and interfaces between components and outside entities.*

Since a layered architecture is a *type-of* or an instance of an *architecture*, it is necessary to map the elements of a layered architecture as discussed in Section 6.2.4 on p. 145 to the concepts in the adopted architecture definition above. The *layers* of a layered architecture map to the *components*, or rather a *grouping of components*. In a layered architecture, the relationships and organisation of the architectural components are represented by the stacking and sequencing of layers as depicted graphically in a layered architecture.

As indicated in Figure 6.2, the architectural definition results in three conclusions namely: *an architecture is a model*, *an architecture is a system view or perspective* and *an architecture depicts the organisation and*

relations of system components. These three *facts* result in further investigations concerning *models and abstractions* (Section 6.3.5), *architectural views or structures* (Section 6.3.6) and *architecture descriptions* (Section 6.3.7). From all these investigations, key concepts are extracted.

6.3.5 Models and abstractions

As stated in the architecture definition of Section 6.3.4, an architecture is a model or abstraction of a system.

Dijkstra [89] introduced the concept of *models* in the early '70s. Models were recommended to simplify unmastered complexity. He argued that the *programmer and his mind are an important part of the computing process* and that *modularised, goto-less programs lead to more efficiency in the use of the computer* [258]. The modularised programs have to be modelled in order to document the functionality decomposition of the program.

Avison and Fitzgerald [10] define a model as an abstraction and representation of part of the real world. *Abstraction* is the process of stripping an idea or a system of its concrete or physical features for a simplified representation of a complex application. Models are used at various levels of system abstraction. A model provides a way of viewing the important aspects of a system at a specific level in such a way that higher levels depict the *essence* of the system and the lower levels show detail that does not compromise the essence. For example, in the three-level system view the system is separated into the conceptual level, the logical level and the physical level. The conceptual level is a high-level overview description of the universe of discourse (UoD) or the domain of interest such as the overall information system, the business system, or even the society. The logical level is a description of the system without any reference to the technology that would be used to implement it. It's scope is the system itself, without the UoD. The physical level describes the implementation of the system, including the required technologies [10].

Lippitt [150] defines a model as a symbolic representation of all the aspects, as well as their interrelationships, of a complex event or situation.

The true value of any model lies in the fact that is an abstraction or representation of reality that is useful for analytical purposes. According to Lippitt [150], p.9:

> *Modelling will help expedite problem-solving and change because it enables those involved to conceptualize the multiple factors through visualized thinking. The interrelationship between the cognitive process of thinking cannot be separated from perception: problem-solving involves cognition, and cognition includes perception. Visualization improves the capability to perceive and, therefore, assists the cognitive process*.

6.3.5.1 Types of models

A model is an abstraction of the real world. There are, however, a number of ways to illustrate such an abstraction. Lippitt [150] proposes a differentiation by category:

- A **Graphic Model** is a two- or three-dimensional diagram that depicts a relationship between aspects such as the relationship between temperature and humidity.
- A **Pictorial Model** aims to transmit an idea by means of a picture or illustration such as a cartoon.
- A **Schematic Model** depicts authority or other relationships within a domain such as a organisation chart of the structure of an organisation, or a flow chart that shows the flow of information.
- **Mathematical Models** usually depict a large degree of abstraction. Mathematical symbols or theorems are used to depict aspects of the real world. These models are useful in studying situations where quantification is possible.
- **Simulation Models** is an approximation of a real-life situation that is used for, for instance, training.

From this categorisation, an architecture is usually depicted by means of a schematic model, and in some cases, where quantification is possible, a mathematical model may be used.

6.3.5.2 Conceptual models

Within Software Engineering conceptual modelling is regarded as a holistic technique used mainly to show various activities of an information system that are logically related, arranged and connected [10]. Conceptual modelling is an abstract process that is used to create an alternative and simplified high-level view or model of a system or problem situation. Often high-level entity models such as E-R diagrams that depict entities with their relationships are referred to as conceptual models [2, 10].

At an upper or conceptual level or system view it is plausible to speculate that an architecture is an instance of a conceptual model as used within the Software Engineering domain.

6.3.5.3 Key concepts of models

Key concepts of models are summarised in Table 6.4.

Models are used to communicate.
Models are used to manage complexity with a simplified representation of a complex idea or application.
A model is an abstraction and a representation of part of the real world.
An abstraction is the process of stripping a system of its concrete or physical features and shows only the *essence* at a certain level.
Models are categorised into different types depending on their purpose.
Models are used at various levels of system abstraction and models provide a way of viewing only what the model indicates are the important aspects at a specific level in such a manner that the *essence* of the system is depicted without the detail that compromises the essence.
Modelling is often concerned with different system views.
Conceptual modelling is an abstract process that is used to create a simplified high-level view or model of a system or problem situation.

Table 6.4: Key concepts of models

6.3.6 Architectural views or structures

The architecture definition of Section 6.3.4 states that an architecture is specified within a specific system context, and that an architecture is a description of the system structure from a particular perspective or view.

In addition, Fowler [104] states that there is not just one way to state a system's architecture; rather, there are multiple architectures in a system, and the view of what is architecturally significant is one that can change over a system's lifetime. In addition, Fowler [104] cites a mailing list post by Ralph Johnson that illustrates the subjective nature of architecture:

> *Architecture is a subjective thing, a shared understanding of a system's design by the expert developers on a project. Commonly this shared understanding is in the form of the major components of the system and how they interact. It is also about decisions, in that it's the decisions that the developers wish they could get right early on because they are perceived to be hard to change. The subjectivity comes in here as well... In the end architecture boils down to whatever is perceived to be important.*

System *views* include possible different levels of abstraction with regard to system composition.

Bachman [12] identifies *levels of abstraction* as one of the guiding principles in the creation of information system architectures. Levels of abstraction is used to study and understand systems and heterogeneous sets of objects of which systems comprise. For instance, the first level of abstraction of Bachman [12] comprises the user's model consisting of meta-objects or the components of a specific organisation's business model and information system implementation. The second level of abstraction is the business model comprising the objects of a comprehensive business information model and the objects of the implementation-oriented models necessary to support it.

Pressman [196] uses four distinct steps representing different levels of abstraction to design an architecture, namely the context, top-level abstrac-

tions or archetypes, components within these abstractions and finally specific instantiations.

Although several authors argue that the similarity between the architecture of a physical building and a software system is superficial [18, 129, 161], it is worthwhile to note that, like a building, a software system is a single entity. However, the software architect and the developers find it helpful to model the system from different perspectives and these perspectives or views clarify different aspects of the system. Some authors describe each of these models as an architectural view of the system [45, 104], whilst others consider that the views together form the architecture [142] of the system.

Related to the notion of architectural views is the distinction made between *system* and *software* architectures discussed in the next section.

6.3.6.1 System and software architectures

The concepts *system architectures* and *software architecture* in essence depict the same structure, however literature usually mentions *software architecture* mainly because it emphasises the crucial nature of the *decisions* made by an architect with regard to software composition [196]. When developing a software architecture, system considerations are always important. A software architect will have to take the whole system with its components, thus the *system* architecture, into account. The system architecture is an abstraction on a higher level of the system than a software architecture that depicts the *software* components within the system with their relationships. In discussions about architecture, literature usually mentions *software* architecture because most of the architect's freedom is presented by software choices, and not by system or hardware choices. Hence authors feel justified to focus on the *software* architectural issues.

In this discussion about architectures, a distinction is made between *software* and *system* in order to emphasise the *perspective* from which the system is viewed. A *system architecture* is a model of the complete system, whilst a *software architecture* only considers the *software* part of the

system. When using the word *architecture*, either the software or system architectures or both, are implied.

6.3.6.2 Key concepts of architectural views and structures

Key concepts of architectural views and structures are summarised in Table 6.5.

Models are used to manage complexity with a simplified representation of a complex idea or application.
Architects and developers often model a system from different perspectives and these perspectives or views clarify different aspects of the system. There is not just one system architecture; rather, there are multiple architectures in a system, and the view of what is architecturally significant is one that can change over a system's lifetime.
What is significant is often subjective. Architecture boils down to whatever is perceived to be important.
System architectures and software architectures are essentially the same, the difference is a matter of perspective. The distinction between *software* and *system* is made in order to emphasise the *perspective* from which the system is viewed.
Literature mentions *software architecture* to emphasise the importance of the architect's software choices, and therefore authors believe that they are justified by focusing on the *software* architectural issues.

Table 6.5: Key concepts of architectural views and structures

6.3.7 Architecture descriptions

In this section the documentation mechanism for architectures as presented by Bass et al. [18] is discussed. The documentation headings identified by Bass et al. are referred to as the major components of an architecture. An architecture is regarded as the blueprint for both the system definition, as well as the project team developing the system. Bass et al. [18], p.201 states that *documenting an architecture is the crowning step to crafting it*.

The first step in the documentation process is the definition and selection of the architectural view. A view is the representation of a coherent

set of architectural elements depicted as a structure. Different views support different goals and uses. The particular view selected for documentation depend on the uses and users thereof. Bass et al. [18] divide views into three groups namely module, component-and-connector and allocation. Module views indicate how the system is to be structured as a set of modules. Component-and-connector views indicate how the system is to be structured as a set of components with runtime behaviour and interactions. The relation of the system to non-software structures such as hardware or networks is depicted in allocation views.

6.3.7.1 Documenting a view

In this section a template for the documentation of a view, obtained from Bass et al. [18] is described.

(1) Primary presentation

The primary presentation of a view depicts the elements or components that populate the view. In addition, it presents the relationships among the components. The primary presentation is usually graphical and should contain a key that explains the notation used.

(2) Element catalogue

The element catalogue details the components and relations depicted in the primary presentation.

(3) Context diagram

The context diagram contextualises the view under discussion and depicts how the view relates to its environment.

(4) Variability guide

The variability guide depicts possible variation points of the architecture in the view under discussion. This includes different variations in decisions contained in the architecture as well as possible options among which a choice can be made. In a module view the possible options are the various versions or parameterisations of modules. In the component-and-connector view the options include constraints on replication, schedul-

ing and choice of protocol. In an allocation view the options may include the conditions under which a software component will be allocated to a particular platform.

(5) Architecture background

The architecture background explains how the design reflected was constructed and why it represent the system under observation. The description includes aspects such as the rationale, the results of analysis that justify the design or explain the impact of modifications, as well as assumptions made.

(6) Glossary of terms

The glossary of terms contains all the terms relevant to the view together with a description of each.

(7) Other information

In this section, any additional necessary information such as management information, authorship and version control or change histories, is included.

6.3.7.2 Documenting behaviour

A view represents structural information about a specific perspective of the system. However, during system development it is necessary to reason about some dynamic system properties such as deadlock, for which a structural description is not sufficient. Therefore, in addition to view documentation, certain behavioural properties of the system need to be documented where applicable [18].

6.3.7.3 Documenting interfaces

An interface is a boundary facilitating the meeting and interaction of two independent entities. The *interface* of an element depicts those properties of the element that are externally visible to other elements [18].

The documentation of an interface comprises the naming and identifi-

cation (or signature) of the interface as well as capturing its syntactic and semantic information. If an interface's resources are invocable programs, the signature names the programs with their parameters. Signatures can be used to check for consistency (for example whether a program compiles and links), however, a signature does not contain a guarantee about system correctness or whether the system would operate successfully. This information is contained in the semantics to the interface which is the information about what happens when the resources are brought into play. The interface to a component remains constant even when the implementation of the component changes. This minimises implementation dependencies and enforce functional dependencies.

An interface is documented with an interface specification that contains aspects of the elements that the architects wish to make public. This information should be kept to a minimum to adhere to the principle of information hiding, but has to be sufficient for the successful implementation of the architecture. Possible *headings* in an interface specification include *interface identity*, *resources provided*, *data type definitions*, *exception definitions*, *variability provided by the interface*, *quality attribute characteristics of the interface*, *element requirements*, *rationale and design issues* as well as a *usage guide* [18].

6.3.7.4 Documentation across views

This type of documentation captures information that applies to more than one view of the system, or the system as a whole. This may include issues such as the structure of the documentation pack, system information such as the system purpose, a view catalog and the mapping between views, the way views are related, a list of components and where they appear as well as a glossary that applies to the whole architecture. Lastly this documentation will capture the rationale behind the design of the architecture and why the architecture is structured in a specific manner.

6.3.7.5 Key concepts of architectural descriptions

Key concepts of architectural descriptions are summarised in Table 6.6.

Documenting and describing an architecture is the crowning step to crafting it.
The first step when documenting an architecture is the definition and selection of the architectural view. A view is the representation of a coherent set of architectural elements depicted as a structure. Different views support different goals and uses.
The second step is a description of the behaviour of the architectural elements.
Next it is necessary to document the interfaces of the architectural elements.
Lastly it is necessary to document the architectural view in relation to other identified views.

Table 6.6: Key concepts of architectural descriptions

6.4 DESIGN AND EVALUATION CRITERIA FOR LAYERED ARCHITECTURES

Figure 6.3 (p. 136) graphically depicts the logical process for the extraction of the evaluation criteria for layered architectures. All the definitions and key concepts from the previous investigations repeated in the diagram are combined so that the criteria for the evaluation mechanism can be extracted. The red hexagons in Figures 6.1, 6.2 and 6.3 depict an *abstraction* or the crystallisation of concepts and definitions from a task or activity.

6.4.1 Extraction of the criteria

In Section 6.2.4 on p. 145 a layered architecture is defined as the resulting artefact of the *layering* technique which is an accepted best practice used to decompose complex systems. Layering facilitates horizontal decomposition of system functionality. In addition, a layered architecture is an architectural pattern or *type-of* software architecture. This lead to an investigation of the concept *software architectures*.

In Section 6.3.4 on p. 157 an architecture is defined as a model of

a system within a specified context, depicting the components necessary to realise the system from a particular perspective. The architecture of a system includes the organisation or structure of the identified modular components, their defining features or properties, as well as the relationships and interfaces between components and outside entities.

The *layers* of a layered architecture are similar to the *components* or *groups of components* in an architecture. In a layered architecture, the relationships and organisation of the architectural components are represented by the stacking and sequencing of layers as depicted in a layered architecture.

In addition, the key concepts extracted from all the supporting discussions of the previous sections are used to identify the criteria. The key concepts can be found in Tables 6.1 (p.143), 6.2 (p.152), 6.3 (p.157), 6.4 (p.160), 6.5 (p.163) and 6.6 (p.167).

The criteria identified are:

- Clearly defined context,
- Appropriate level of abstraction and hiding of implementation details,
- Clearly defined functional layers,
- Appropriate layering, including well-defined interfaces and dependencies; and
- Modularity.

Figure 6.13 graphically depicts the relationship between the criteria and the definitions for layered architectures and architectures. In Table 6.7 the criteria with a short description of each are listed. Furthermore, Table 6.8 questions are listed together with each criterion in order to assist with the creation of the evaluation mechanism.

Evaluation Criteria	Description
Clearly defined context	The context is the result of the identified *view* that is used to analyse the system. The context or view determines the important aspects of the system at that level. It also aids the identification of the main components required to realise the system, the component properties, the structure or organisation of the components, as well as the relationships between the identified components.
Appropriate level of abstraction and hiding of implementation details	The architecture model should be at a sufficiently high level of abstraction so that the system or sub-system under review can be viewed as a whole. Only the aspects of the system that are relevant at a certain level of abstraction should be visible at that level. The hiding of implementation details supports the notion of an appropriate level of abstraction since implementation details should be hidden in an architectural model.
Clearly defined functional layers	This criterion relates to the determination of the architectural components and their grouping into the appropriate layers. This grouping should be the result of functional decomposition and should support system development principles such as tight cohesion of related functional components.
Appropriate layering, including well defined interfaces and dependencies	This criterion relates to the organisation of the identified layers. The layers must clearly build on one another and their relationships and dependencies should be distinguishable. A layer should only access layers below it. This criterion also includes the specification of dependencies or access rules between layers, which is used to determine whether the architecture is open or closed.
Modularity	Components and hence layers should be modular. It should be possible to change the implementation of a layer as long as interfaces and functionality remain the same.

Table 6.7: Evaluation criteria for layered architectures

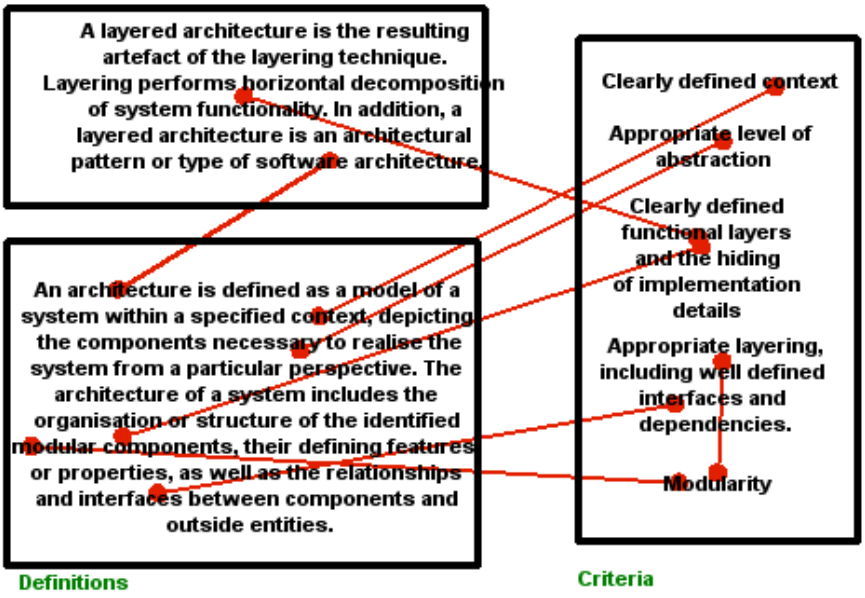


Figure 6.13: Extraction of criteria from architecture definitions.

6.4.2 Criteria framework or evaluation mechanism

In Table 6.8 the criteria are expanded to contain questions that can be used to evaluate layered architectures. In the table possible questions to be asked when evaluating each criterion are indicated by 'Q'.

Evaluation Criteria	Description
Clearly defined context	Q Is it possible to identify the context from the description of the architecture?
Appropriate level of abstraction and hiding of implementation details	Q Can the system within the context be viewed as a whole? Q Are there any components/properties/relationships in the architecture model that could be removed without losing important information at this level of abstraction? Q Are any implementation details visible in the description of the components/properties/relationships/structures of the architecture?
Clearly defined functional layers	Q Does the layer description specify a *function* of the layer within the system? Q Is the function of the layer clear from its description and position in the architecture? Q Could the layer be removed without compromising the integrity of the system?
Appropriate layering, including well-defined interfaces and dependencies	Q Do the layers clearly build on one another? Q Does a specific layer only require functionality defined by lower layers and not those of upper layers? Q Is it possible to determine whether the layered architecture is open or close?
Modularity	Q Is it possible to replace the implementation of a layer with another implementation of the same functionality and interfaces without compromising the integrity of the layered architecture?

Table 6.8: Evaluation criteria for layered architectures, including possible questions for evaluation.

6.5 THE EVALUATION OF LAYERED ARCHITECTURES

The extracted evaluation criteria may be used to assist with the design processes of new architectures or to evaluate existing ones. In order to establish and demonstrate the usefulness and validity of these criteria, the ISO/OSI architecture is evaluated as an accepted example of a layered architecture that is used for the visualisation and design of network functionality.

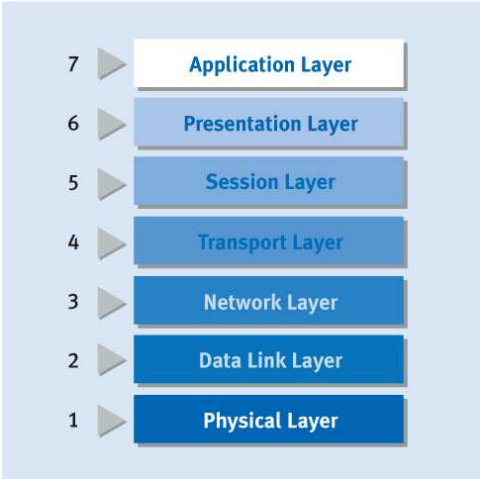


Figure 6.14: The ISO/OSI network architecture [265].

Tanenbaum [220] defines a computer network as a collection of computers, called *hosts*, that communicate with each other directly, or through *nodes* or *IMPs (interface message processors)*. Networks are generally organised as a hierarchy of layers where each layer performs a set of related functions.

The *ISO/OSI model* separates the functionality required to implement a network into seven different layers. These stacks are present on hosts. A layer on one host conceptually facilitates communication with its corresponding layer on another host, even though in practice the data is passed vertically down the layers of the sending machine and up the layers of the receiving machine as depicted in Figure 6.15 [117, 220].

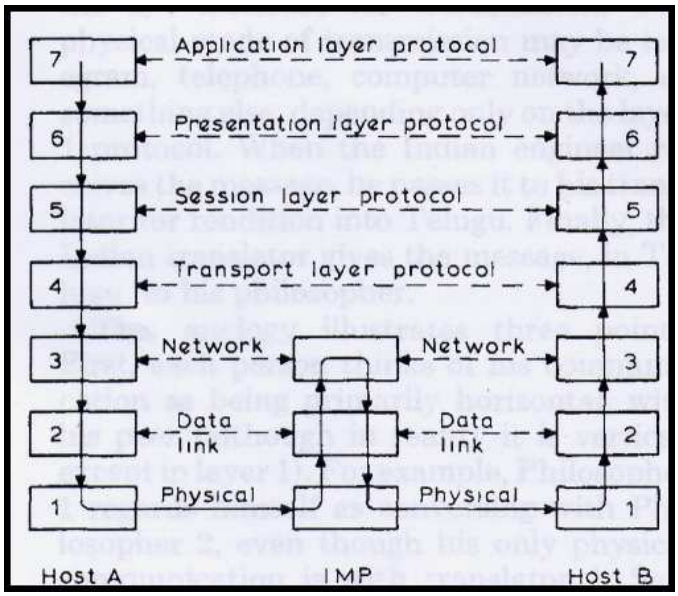


Figure 6.15: Communication between layers on the ISO/OSI network architecture [220].

The ISO model has seven layers and therefore specifies seven protocols. A protocol is defined as the *rules* governing layer conversations. The rules governing layer *k* conversation are called the *layer k protocol*. Layers are specified to be completely independent and no layer is aware of the protocol detail used by other layers. The boundary between adjacent layers is called an interface. The layers, interfaces and protocols in a network together form the network architecture [117, 220].

As stated, the *ISO/OSI model* separates the methods and protocols required for network connectivity into seven different layers. Higher-layers rely on services provided by a lower-level layer [117, 178, 265]. The layers are ordered from bottom to top and include:

- The **Physical Layer** defines the manner in which signals are transmitted among host machines on a network. The network device is responsible for generating and receiving these signals.
- The **Data link Layer** partitions a stream of bytes from the physical layer into a frame.It implements error and flow control on frames. This is generally implemented in the network controller hardware.
- The **Network Layer** encapsulates frames into packets that can be

transmitted between hosts using a high-level addressing and routing scheme. Some of this functionality is implemented in hardware, the remainder will be implemented by means of appropriate software.

- ▷ The **Transport Layer** implements reliable packet delivery for its users. It fragments and regenerates a stream of bytes from a collection of packets that is transmitted over a network layer. The transport layer supports the network layer in maintaining routing tables.
- ▷ The **Session Layer** adds services to the byte stream such as high-level naming and bi-directional transmission services. The session layer is generally less defined than lower layers except in the case of specific applications.
- ▷ The **Presentation Layer** handles tasks such as character mapping and number conversion.
- ▷ The **Application Layer**: the functionality implemented in this layer is application specific.

The ISO/OSI architecture model does not specify how layers are to be implemented. Tanenbaum [220] notes that the model is a framework for describing layered networks. The specification by Zimmermann [265] describes the concept of layering in considerable detail and introduces a uniform terminology. Finally, this specification describes the seven layers including the purpose and functionality of each layer, as well as the services the layer provides to higher layers. According to Tanenbaum [220], the value of the ISO/OSI layered architecture lies in the specification of a uniform nomenclature and a generally agreed-upon standard to split network activities into layers [178].

The ISO/OSI model is not a protocol standard, but by breaking the network functionality up into layers, the model generates the framework for protocol specifications of the different layers. The standards fall outside the domain of the model and are often performed by third-party organisations. At present, the ISO/OSI model is widely integrated into network protocols, and an example thereof is the well-known Internet protocol TCP/IP (Transmit Control Protocol/Internet Protocol) that is an implementation of the network and transport layers of the ISO/OSI model [117].

Criteria	ISO/OSI Architecture Model
Clearly defined context	**Conform to:** The context of the ISO/OSI model is clearly defined as *an architecture for Open Systems Interconnection* [265] or a model defined to assist with the development of network protocols [220].
Appropriate level of abstraction and hiding of implementation details	**Conform to:** All the layers required for network interaction are identified and the network is represented as a whole. The model does not specify implementation details. No unnecessary information is displayed on any layer. No unnecessary or implementation detail is visible on the architecture description [195, 220, 265].
Clearly defined functional layers	**Conform to:** All the layers have well-defined functionality descriptions, and their position within the architecture supports this functionality [265].
Appropriate layering, including well defined interfaces and dependencies	**Conform to:** Each layer build on the layer immediately below and only on this layer, which means that the architecture is closed [265]. In the model, a logical interface is specified for each layer [265], and the protocols which are implementations of layers in the model have detailed interface specifications [117, 220].
Modularity	**Conform to:** There are different implementations of the layers and these can be interchanged without negatively influencing the integrity of the architecture [117, 195].

Table 6.9: Evaluation of the ISO/OSI layered architecture.

It is beyond the scope of this section to enter into a detailed discussion of the layers and the different protocols that were specified to implement the layers. It suffice to provide references in Table 6.9 that depicts the evaluation against the criteria of Section 6.4.2 on p. 170.

Using the proposed criteria, it is established that the ISO/OSI model clearly conforms to all specified criteria. Thus it is possible to conclude that the existing ISO/OSI layered architecture is well designed. The ISO/OSI model is a well accepted industry framework, and is regarded as a proven architecture through several generally accepted adoptions. Examples of such adoptions include the X.21 and RS232-C protocols on the physical layer, HDLC (High-level Data Link Control) or SDLC (Synchronous Data

Link Control) as a data-link protocol, X.25 as a network protocol, and TCP/IP protocol, which is an implementation of the network and transport layers [146, 195, 220].

In addition, it can be stated that the evaluation mechanism is useful and valid for the evaluation of layered architectures, and that the mechanism can therefore be used to evaluate and design a comprehensive and functional layered architecture for the Semantic Web.

6.6 CONCLUSION

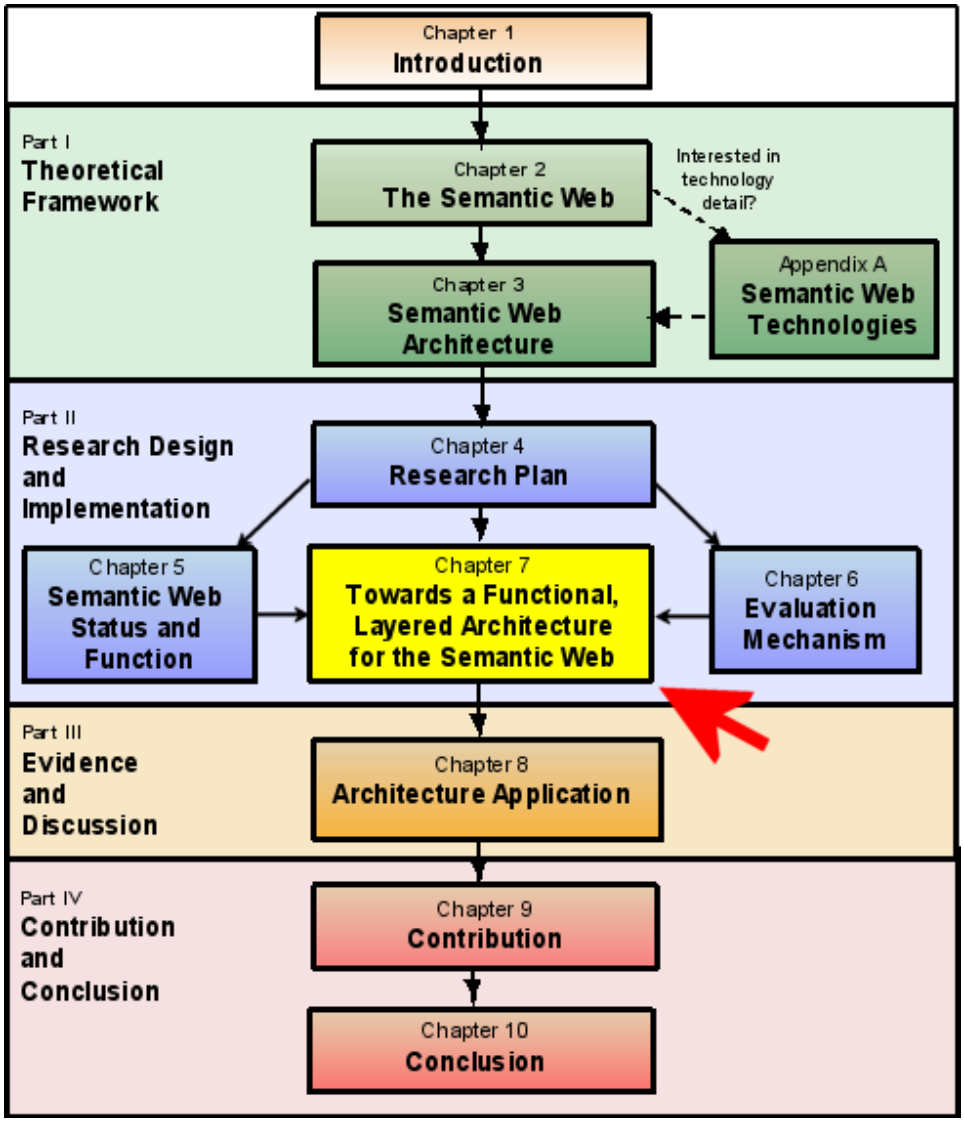
The results of the research presented in this chapter comprise an evaluation mechanism for layered architectures. This evaluation mechanism consists of criteria that were extracted from an investigation into the use of layered architectures, the descriptions and definitions that exist within literature of layered architectures, as well as in-depth investigation into the concept *architecture* with its associated concepts. In order to demonstrate the efficacy of this evaluation mechanism, the ISO/OSI layered architecture obtained from literature was assessed. The result confirmed the notion that this criteria list could assist researchers and system architects to evaluate and design architectures in general, and layered architectures in particular.

The research results contained in this chapter were presented at the MSVVEIS Workshop hosted at the 8th International Conference on Enterprise Information Systems (ICEIS'06), Paphos, Cyprus, on 23-27 May 2006 and were published in the conference proceedings [107].

In this chapter, the principles for the construction of a *comprehensive* architecture were established, whilst in the previous chapter (Chapter 5) the components for the construction of a *functional* architecture were established. Within the next chapter, Chapter 7, the *comprehensive* and *functional* layered architecture for the Semantic Web is constructed and the findings of this chapter and Chapter 5 are used as building blocks.

CHAPTER 7

TOWARDS A COMPREHENSIVE AND FUNCTIONAL LAYERED ARCHITECTURE FOR THE SEMANTIC WEB



Thesis Chapter Layout

Chapter Contents

Figures

Tables

7.1 INTRODUCTION

In this chapter the development of a comprehensive and functional layered (CFL) architecture for the Semantic Web is discussed. The research question relevant to this chapter is *How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?*

The supporting questions that are of concern in this chapter are:

- What is the result of the evaluation of the current Semantic Web architecture against the established criteria of Chapter 6?
- How can the current Semantic Web architecture be adapted to conform to the established criteria for layered architectures?
- What are additional design principles that should be considered in the development of a CFL architecture for the Semantic Web?

In order to develop a comprehensive and functional layered (CFL) architecture for the Semantic Web, the present versions of the Semantic Web layered architecture are evaluated using the evaluation mechanism for layered architectures developed in Chapter 6. This evaluation is discussed in Section 7.2. Additional requirements that the development of the CFL architecture should consider, are discussed in Section 7.3. The suggested adaptations required for the development of a CFL architecture are discussed in Section 7.4. In order to comment on the structure of the suggested CFL architecture for the Semantic Web, this architecture is evaluated using the established evaluation mechanism from Chapter 6 in Section 7.5. The chapter is concluded in Section 7.6.

7.2 THE EVALUATION OF THE SEMANTIC WEB LAYERED ARCHITECTURE

The departure point for the development of a CFL architecture for the Semantic Web is the present versions of the Semantic Web architecture as proposed by Berners-Lee [28, 31, 34, 35]. These four versions of the ar-

chitecture, labelled V1, V2, V3 and V4, were discussed in Chapter 3 and are depicted in Figure 7.1. For this discussion, all four versions (V1-V4) are collective referred to as the *Semantic Web layered architecture*. Where required, a distinction is made by referring to a specific version using the applicable version V1 to V4 label.

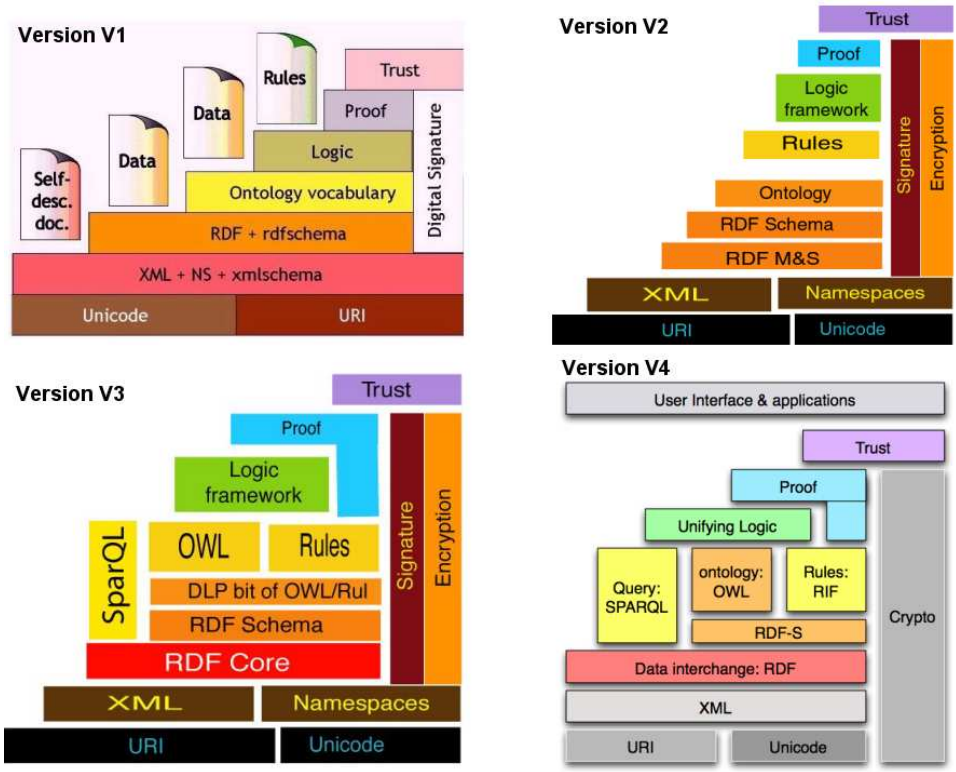


Figure 7.1: The versions of the Semantic Web layered architecture.

To determine shortcomings of the present Semantic Web layered architecture, the evaluation criteria for layered architectures as established in Chapter 6 (section 6.4.2, p.170) are used to evaluate the architecture. These evaluation criteria for layered architectures are:

- A clearly defined context,
- An appropriate level of abstraction and the hiding of implementation details,
- Clearly defined functional layers,
- Appropriate layering, including well-defined interfaces and dependen-

cies; and

- Modularity.

A summary of the evaluation result is presented in Table 7.1, and the remainder of this section (Section 7.2.1 to 7.2.5) expands on the reasons for conformance or non-conformance of the present Semantic Web layered architecture to the evaluation criteria.

Criteria	Semantic Web Architecture
Clearly defined context	**Conform to:** The context for the Semantic Web layered architecture is accepted to be the languages required to implement the Semantic Web.
Appropriate level of abstraction and hiding of implementation details	**Does not conform to:** It is possible to argue that the whole Semantic Web language architecture is visible. However, the layers of the Semantic Web layered architecture (in all four versions) define *functionality* as well as *existing W3C technologies*.
Clearly defined functional layers	**Does not conform to:** In the Semantic Web layered architecture, the higher layers define functionality while the bottom layers specify existing W3C technologies (such as *XML* and *RDF*). This is the case for all the versions of the Semantic Web layered architecture. In addition, the functionality of each layer is not clearly defined.
Appropriate layering, including well-defined interfaces and dependencies	**Does not conform to:** The layers in the Semantic Web layered architecture do not clearly build on one another. It is not clearly specified what the requirements of upper layers with regard to their lower layers are, and it is not possible to establish whether any version is an open or closed architecture. It is not clear what the meaning is of vertical layers, or side-by-side layers depicted in the all the different versions of the architecture.
Modularity	**Undefined:** It is not possible to determine the modularity of the Semantic Web layered architecture since the functionality of the layers is not defined.

Table 7.1: Evaluation of the Semantic Web layered architecture.

7.2.1 Clearly defined context

The evaluation question associated with this criterion (section 6.4.2) is:

- Q *Is it possible to identify the context from the description of the architecture?*

The context of the Semantic Web layered architecture is stated to be the languages required to implement the Semantic Web, or in other words, the languages necessary to describe the machine-processable information containing the meta-data descriptions of Web resources. Fensel [99] refers to the V1 architecture as *the layer language model for the Web*, while Hendler [123] refers to the V1 architecture as the *Semantic Web "layer cake"* and further describes this architecture as a model that shows the proposed layers of Semantic Web languages. Antoniou and von Harmelen [6], p.17-18 refer to the V1 architecture as *... the "layer cake" of the Semantic Web (due to Berners-Lee), which describes the main layers of the Semantic Web design and vision* and they similarly refer to the content of each layer as a *language*.

Berners-Lee [33] refers to the V2 version of the Semantic Web layered architecture as a *stack of expressive power* and he then subsequently discusses the language technologies of the different layers as providing increasing expressiveness. When Berners-Lee [34] presented the V3 architecture in 2005, he explicitly depicted the Semantic Web layered architecture within the context of *Web languages* and *Semantic Web languages*.

The current Semantic Web layered architecture thus *conforms* to this criterion since the context is defined.

7.2.2 Appropriate level of abstraction and hiding of implementation details

The evaluation questions associated with this criterion (according to Section 6.4.2) are:

- Q *Can the system within the context be viewed as a whole?*

- *Are there any components/properties/relationships in the architecture model that could be removed without losing important information at this level of abstraction?*
- *Are any implementation details visible in the description of the components/properties/relationships/structures of the architecture?*

It is possible to argue that the Semantic Web language architecture could be viewed as a *whole* within the context of the *languages required*. However, it is appropriate and commendable to remove information from the model. The top three layers define functionality, but the rest of the layers specify *existing technologies* rather than functionalities. The appropriate level of abstraction implies that an architecture depicts system functionality, and the technology detail depicted in all versions of the Semantic Web layered architecture implies that the architecture does not conform to this criterion.

In addition, to conform to this criterion, an architecture should hide implementation details. Referring to Figure 7.1, it is evident that the bottom layers in all versions depict technologies or *implementation specifications* rather than the necessary functionalities required to implement the language architecture. In addition, it is not clear what the function and interface pertaining to the vertical layers such as *Digital Signatures* in V1, as well as *Signature* and *Encryption* are, neither why *Unicode* and *URI* appear as two side-by-side layers on the bottom layer of all versions. All these layers depict implementation details rather than functionality

It is thus possible to motivate non-conformance of the current Semantic Web layered architecture to this criterion.

7.2.3 Clearly defined functional layers

In order to evaluate conformance or non-conformance to this criterion, the following questions could be asked (section 6.4.2):

- *Does the layer description specify a* function *of the layer within the system?*
- *Is the layer's function clear from its description and position in the*

architecture?

- Q *Could the layer be removed without compromising the integrity of the system?*

The upper layers of the Semantic Web layered architecture (in all versions) define functionality. It is often not clear whether these are in relation to *languages* as the architecture context specifies, or applied functionality necessary to demonstrate the W3C technology initiatives. An example of this is *Rules*. Version V1 does not depict *Rules* and in version V2 *Rules* is depicted above *Ontology*. However, both versions V3 and V4 depict *Rules* residing next to *OWL*. The motivation thereof seems to be the *Rules:RIF* initiative of the W3C. Therefore, even though the caption *Rules* signifies functionality, it is used in versions V3 and V4 in a manner that is inconsistent with the *system functionality* required to implement the languages of the Semantic Web.

As stated, the bottom layers specify existing technologies (such as *XML* and *RDF*) rather than the functions embodied by these layers. The function of *Digital Signatures* or *Signatures* (versions V1, V2 and V3), *Unicode* and *URI* (all versions) is also not clear from its positioning on the architecture.

It is possible to remove, for instance, any *security related* vertical layer in any of the versions of the Semantic Web layered architecture without compromising the integrity of the system as regards to the *languages* of the Semantic Web.

In summary, several layers in the Semantic Web layered architecture (versions V1 to V4) do not depict *a system functionality*. In addition, the function of several layers is not always clear from their position in the architecture, and lastly, it is possible to remove certain layers without compromising the system. Hence, it is possible to motivate the non-conformance if the Semantic Web layered architecture to this criterion.

7.2.4 Appropriate layering

Conformance or non-conformance to this criterion could be evaluated using the following questions (section 6.4.2):

- Q *Do the layers clearly build on one another?*
- Q *Does a specific layer only require functionality defined by lower layers and not that of upper layers?*
- Q *Is it possible to determine whether the layered architecture is open or closed?*

This criterion includes the specification of dependencies. The layering of functionalities in the present Semantic Web layered architecture is not apparent. The presence of vertical and side-by-side layers in all versions of the architecture immediately implies that layers do not clearly build on one another.

In particular, it is not clearly specified in any version what the requirements of upper layers with regard to their lower layers are, nor is it possible to establish whether this is an open or closed architecture.

It is therefore possible to motivate the non-conformance of the present Semantic Web layered architecture to this criterion.

7.2.5 Modularity

The conformance or non-conformance to this criterion could be determined using the following question (section 6.4.2):

- Q *Is it possible to replace the implementation of a layer with another implementation of the same functionality and interfaces without compromising the integrity of the layered architecture?*

It is possible to argue that the conformance of the Semantic Web layered architecture with this criterion is *undefined* since the functionality of all the layers, especially lower layers that depict technologies, has not been determined. For example, Layer 1 in all the versions of the architecture depicts *Unicode* and *URI* and the function of these technologies within the Semantic Web is not defined by the architecture. Alternatively, if an indication of conformance or non-conformance is required, it is possible to argue non-conformance of the Semantic Web layered architecture to this criterion since *modularity* specifies that the implementation of a layer can be replaced with another implementation without compromising the integrity

of the layered architecture. The *technologies* depicted on the lower layers of all versions of the layered architecture cannot be replaced with alternative implementations, which implies non-conformance to the modularity criterion.

7.2.6 Concluding remarks

In this section it is argued that all versions of the proposed Semantic Web layered architecture do not comply with the majority of the established evaluation criteria for layered architectures. The Semantic Web layered architecture is specified within the context of the languages necessary to implement the Semantic Web, but the architecture fails to conform to criteria with regard to abstraction, layering, functionality description and modularity.

7.3 ADDITIONAL REQUIREMENTS

The evaluation and development of the Semantic Web layered architecture using the established criteria for layered architectures represent a *top-down* process. Alternatively, in this section specific principles are scrutinised and possible refinements to the architecture from a *bottom-up* perspective are investigated. The design principles were obtained from the W3C architecture initiatives and the ISO/OSI architecture definition.

7.3.1 W3C design principles

As part of the ongoing W3C design initiatives, Berners-Lee argues for the inclusion of certain Software Engineering design principles into W3C design efforts, and he identifies four principles namely simplicity, modularity, decentralisation and tolerance [29]. These design principles are discussed in more detail in Section 3.4 (p.70).

In summary, *simplicity* strives to use only a few basic elements to achieve the required results. *Modularity* suggests a system design adhering to *loose coupling* and *tight cohesion* of system elements. *Decen-*

tralisation avoids any common point that may be a single point of complete failure. *Tolerance* specifies liberal requirements and conservative implementations. In this section the integration of these *design principles* into a proposed layered architecture of the Semantic Web, is discussed.

7.3.1.1 Integration with the Semantic Web CFL architecture requirements

The *simplicity* and *modularity* design principles of the W3C are supported by the already established criteria for layered architectures, specifically *functional layers, appropriate level of abstraction* and *modularity*. Therefore, when a proposed Semantic Web layered architecture (the CFL architecture) adheres to the criteria of the evaluation mechanism, the *simplicity* and *modularity* W3C design principles will be supported.

The *tolerance* principle is applicable when specifying the functionality contained within a layer, and the functionality descriptions of the layers will have to be evaluated against this principle continuously.

Decentralisation is a principle that is applicable on the deployment level of the Web or Semantic Web. This principle would be relevant for the definition of ontologies or knowledge bases on the Semantic Web. To adhere to this principle, it is preferable that not one, but several ontologies within a domain and across domains are specified, even if it will result in terminology conflicts.

7.3.2 ISO/OSI design principles

Section 6.5 concluded that the layered ISO/OSI (International Organisation for Standardisation / Open Systems Interconnection) architecture conforms to the evaluation criteria for layered architectures. In order to refine a proposed Semantic Web layered architecture (the CFL architecture), in this section the ISO/OSI architecture model as defined by Zimmermann [265] in *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*, is scrutinised.

The ISO/OSI layered architecture model was developed due to the identification of an urgent need for standardisation by the ISO for protocols that would allow for the composition of heterogeneous computer networks. The model was developed as a layered architecture for open systems interconnection, which could serve as a framework for the definition of standard protocols [265]. The architecture is not a specification in itself, to be more precise, it is considered a framework for specifying layered networks. Zimmermann [265] discusses the adopted concepts of the generally agreed-upon layering in the architecture in considerable detail, and introduces a uniform terminology or nomenclature [220].

The initial experimental development of computer networks such as ARPANET was rapidly followed by computer manufacturers' network implementations. The universal need for interconnecting systems from different manufacturers rapidly became apparent, which lead to the establishment of the ISO activities for the development of standards required for *Open Systems Interconnection (OSI)* [265]. *Open* here means that an implementation conforms to the international standards and that it should be open to any global system obeying the same standards [195, 265].

The basic objective of the ISO/OSI model is to standardise the rules of interaction between interconnected systems. Thus, only the external behaviour of Open Systems should conform to the architecture [265]. The internal organisation and functioning of each individual system fall outside the scope of the ISO/OSI model since these are not visible from other systems with which such a system interconnects. [195, 265].

7.3.2.1 Layering principles

In the ISO/OSI model description, Zimmermann [265] discusses the meaning of layers in substantial detail. Layering is seen as a *structuring technique which permits the network of Open Systems to be viewed as logically composed of a succession of layers, each wrapping the lower layers and isolating them from the higher layers* [265]. Each individual system is viewed as a logically composed succession of layered subsystems. Using another view, a layer can be viewed as a logically composed subsystem of

the same rank in all the interconnected systems [265]. The aspects of layered architectures described below were identified by Zimmermann [265]:

- **Value addition:**
  In a layered architecture, each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered the set of services it requires for its applications. Layering thus *divides the total problem into smaller pieces*.
- **Independence and modularity:**
  Independence of each layer is ensured by defining services provided by a layer to the next higher layer, regardless of how these services are implemented. This technique correlates with structured programming design where only the functions performed by a module (and not its internal functioning) are known to its users. Independence and modularity enable changes of functions or protocols within a layer without affecting the other layers. In addition, this aspect permits changes to be made in the way a layer or a set of layers operate, provided such layers still offer the same services to the next higher layer. This principle also endorses the creation of layers with localised functions in such a manner that any layer could be completely redesigned with major protocol changes to take advantage of advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers.
- **Simplicity:**
  This principle guides the number of layers. There should not be so many layers that the system-engineering task describing and integrating the layers become cumbersome.
- **Functional layering:**
  Functional layering implies that separate layers are created to handle different functions of the process performed or the technology involved. Similar functions should be collected into the same layer. A separate layer should only be created when there is a need for a different level of abstraction in the handling of any data, e.g., morphology, syntax, semantics.

- **Interface:**
  This principle implies that, for each layer, interfaces with only its upper and lower layer are established. A boundary should be created at a point where the description of services is small where and the number of interactions across the boundary is minimised. A *boundary* exists between layers, and the specific boundary between adjacent layers is called an interface. A boundary should be created where it may be useful to have the corresponding interface standardised.
- **Sub-layers:**
  The principle of sub-layers implies that further sub-grouping and organisation of functions to form sub-layers within a layer should be created in cases where distinct communication services require it. Two or more sub-layers with a common, and therefore minimum, functionality should be created where required in order to allow interface operation with adjacent layers. It should be possible to bypass sub-layers.

7.3.2.2 Integration with the Semantic Web CFL architecture requirements

The *value addition*, *independence and modularity*, *simplicity*, *functional layers* and *interface* principles of the ISO/OSI architecture are captured by the evaluation mechanism criteria descriptions and conform to the general principles of layering as discussed in Section 6.2. However, *sub-layers* add an additional principle that could enhance the model, and it is worthwhile to consider this during the development of the proposed Semantic Web layered architecture (the CFL architecture).

As regards the approach followed for the development of the ISO/OSI architecture, it is worthwhile to analyse the situation at the time of its development. As discussed in section 7.3.2, several network protocols were being developed at the time by different vendors [220], and it is plausible to speculate that a network architecture depicting these standards could be presented as architecture (1) in Figure 7.2.

There is a recognisable similarity between the two architectures de-

picted in Figure 7.2, which represents a network layered architecture with protocols, and (for demonstration purposes) the V1 Semantic Web layered architecture. Both architectures depict technologies on the lower layers and functionalities on higher layers.

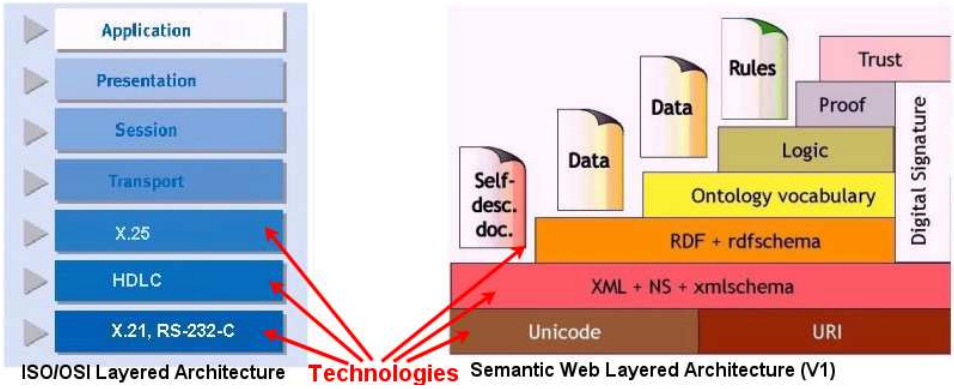


Figure 7.2: Layered architectures depicting technologies.

It is also possible to speculate that, had the only network architecture at the time of the development of the ISO/OSI architecture been similar to (1) in Figure 7.2, there would have been many discussions about layering and the implementations of different specifications in literature, as is presently the case with the Semantic Web (refer to Section 3.2, p.55).

Thus, the role played by the ISO to abstract functionalities required for a network into the seven defined layers of the ISO/OSI layered architecture was fundamental in order to realise network interconnectivity. This abstraction of network functionality into the ISO/OSI layered architecture had a considerable effect on the development, standardisation and acceptance of network protocols, and very few debates on the layering of technologies were ever published. The ISO/OSI layered architecture enabled the development of different and diverse protocol standards for the different layers that could still interconnect and interoperate in spite of their differences.

A similar approach to that of the ISO with the creation of the ISO/OSI layered architecture is recommended in this thesis. The abstraction of the language functionality required for the implementation of the Semantic Web into a functional layered architecture could prove to be valuable for the re-

alisation of the Semantic Web and could ensure data interoperability. Such a layered architecture could serve as a framework for the development of *language* standards in a way similar to the ISO/OSI architecture, which serves as a framework for protocols to ensure network interoperability.

7.4 TOWARDS A COMPREHENSIVE AND FUNCTIONAL LAYERED (CFL) ARCHITECTURE FOR THE SEMANTIC WEB

The criteria developed in Section 6.4.2, the additional design principles of Section 7.3, as well as the reasons for the non-adherence of the present Semantic Web layered architecture of Berners-Lee (encompassing all versions) as presented in Section 7.2, are combined in order to propose a comprehensive and functional layered architecture for the Semantic Web, for reference purposes dubbed the CFL architecture in this thesis. The CFL architecture is presented in Figure 7.3.

When referring to the present *Semantic Web layered architecture* of Berners-Lee, all versions of the architecture are, by implication, included (Figure 7.1). When a specific version of the Semantic Web layered architecture has to be identified, it is referred to by using the V1 to V4 indicator for versions 1 to 4 respectively.

The proposed CFL architecture for the Semantic Web varies from the previously suggested versions (V1-V4) of the architectures mainly because it adheres to the evaluation criteria for layered architectures as established in Section 6.4.2. In addition, it is noticeable that the CFL architecture abstracts and depicts related functionalities rather than the W3C technologies used to instantiate these functionalities. In the remainder of this section the adaptations according to the clarification of the architecture context, as well as the abstraction of functionalities and the development of a security stack, are discussed. In Section 7.4.5 detail on the functionality of the layers is provided.

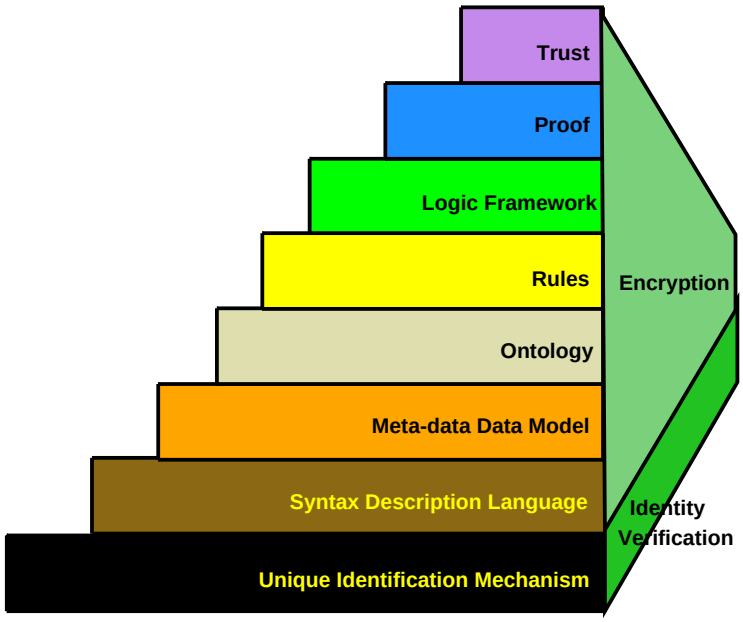


Figure 7.3: The proposed CFL architecture for the Semantic Web comprises two orthogonal architecture stacks, the *language* stack and the *security* stack.

7.4.1 The clarification of the architecture context

The architecture of a system defines the main components and their relationships as per the definition in Section 6.3.4. In order to describe a system with regard to its components and the relationships between the components, it is necessary to state the context of the architecture clearly.

The context of the Semantic Web layered architecture is accepted to be the *language functionalities* required to implement the Semantic Web (refer also to Section 7.2.1). The present Semantic Web layered architecture does conform to this criterion, although the context is not stated as clearly as preferred.

For *context specification*, the emphasis should be on the language *functionality* in favour of the language *technologies* that could be used to implement a functionality. In the architecture proposed in Figure 7.3 all the layers in both the *language* and *security* stacks represent functionalities.

7.4.2 The abstraction of functionalities

All layers in the proposed CFL architecture for the Semantic Web should portray the *functionalities* necessary to implement the languages of the Semantic Web. At present only the upper layers in the present Semantic Web layered architecture (all four versions) as depicted in Figure 7.1 define functionality, the remainder of the layers specify *existing W3C technologies* rather than functionalities.

This discussion builds upon the functional status model as developed in Chapter 5 (section 5.5 p.125). From the abstraction of the functionalities from the related technologies, it is proposed (as a first iteration) that:

- Unicode and URI be replaced with *Unique Identification Mechanism*. Unicode aims to uniquely identify all the characters in all the written languages of the world [71] and URI endeavours to uniquely identify any object with a character string [37]. Together, the functionality of these technologies could be described as the provision of a *unique identification mechanism* within the language stack for the Semantic Web.
- XML, XML Schema and Namespaces are depicted as *Syntax Description Language*. XML (Extensible Markup Language) provides a W3C standard for the exchange of data over various networks, especially the Web or WWW [257]. An XML schema is an XML document that defines the content and structure of one or more derived XML documents [233–235]. XML Namespaces provide a simple method for qualifying element and attribute names used in XML documents by associating them with Namespaces identified by URIs [46]. These technologies provide a syntax language for among others, the envisioned Semantic Web.
- RDF is replaced by the *Meta-data Data Model* function. RDF is a W3C Recommendation designed to standardise the definition and usage of meta-data with a simple data model. This layer therefore provides a mechanism to model the meta-data required to implement the Semantic Web.
- RDF Schema should be integrated into the *Ontology* function. RDF

Schema provides a predefined, basic type system for RDF models, and provides, although limited, a mechanism to add semantics to meta-data or to represent knowledge. RDF Schema extends RDF by providing an externally specified semantics to specific resources [231, 243].

If these changes are implemented, the criterion stating the *appropriate level of abstraction* will be adhered to because each layer depicts only the functionality required for that layer within the context of the languages for the Semantic Web. As a result the system under review (the languages for the Semantic Web) can be regarded as a whole and only the aspects that are relevant for a specific layer are visible.

In a layered architecture, upper layers access the functionality of lower layers by means of interfaces to the functionality provided by the lower layers. An interface is the mechanism through which the functionality of a specific component is invoked and through which data input and output are achieved [196].

Concerning the technologies depicted in the present versions of the Semantic Web layered architecture, the following observations regarding interfaces can be made:

- The specifications and/or Recommendations of the IETF and W3C could be regarded as interface specifications to these technologies. For example, the XML Recommendation [50] describes in detail how to serialise or encode any data by means of XML.
- The depicted technologies including the W3C Recommendations do not encapsulate functionality. In order to use any of these technologies, applications that implement the technology specifications have to be developed. For example, a parser or serialiser is necessary to use, interpret or generate XML data. This is one of the stumbling blocks of the realisation of the Semantic Web at present because few of these technology applications exist.
- The technology application that implements a standard would define interfaces to functions it provides. For instance, the XML Recommendation [50] defines clearly what a well-formed XML textual

object is. The XML application could therefore define an interface `CheckWellFormed(AnObject:textobject)` that would check whether the provided text object is well-formed.

With regard to the interfaces of layered architectures, Garlan and Shaw [105], p.11 maintain that:

> *Layered systems have several desirable properties.....they support reuse. Like abstract data types, different implementations of the same layer can be used interchangeably, provided they support the same interfaces to their adjacent layers. This leads to the possibility of defining standard layer interfaces to which different implementors can build. (A good example is the OSI/ISO model and some of the X Window System protocols.)*

In order to specify such interfaces within the CFL architecture, the functionality and components within each layer should be defined unambiguously. Once the components that implement the functionality are specified, the definition of its interfaces will be a pre-requisite for the implementation of the Semantic Web language architecture. The specification of *clearly defined interfaces* is therefore one of the aspects of the architecture that needs to be addressed in future research.

7.4.3 The development of a security stack

Security aspects do not play a *functional* role within, specifically, the *language* stack of the proposed Semantic Web architecture (Figure 7.3). Therefore the development of a separate, parallel architecture to implement the security functionality required for the Semantic Web is proposed. However, this parallel security architecture is required to interface with the language architecture. In the proposed CFL architecture, the security based layered architecture is depicted as adjacent to the language architecture.

Figure 7.3 depicts an orthogonal, multi-dimensional architecture with different *views* (as discussed by Bass, Clements and Kazman [18], Fowler [104], Jacobson, Booch and Rumbaugh [142]). One of these views is the

security stack. In addition, the structure of the layered architecture implies that the upper layers use only part of the functionality defined by the interfaces of the immediate lower layer.

In order to be consistent with the definition of functionality components, it is proposed that *Digital Signatures* be replaced with *Identity Verification Mechanism*.

The four versions (V1-V4) of the present Semantic Web layered architecture of Berners-Lee depict vertical layers focusing on security aspects on the right hand side, such as *Digital Signatures* in V1, *Signatures* and *Encryption* in V2 and V3, and *Crypto* in V4. It is possible to argue that these specific *vertical* layers support the notion of an alternative *security* view other than that of the *language* layers. Thus, in the proposed CFL architecture for the Semantic Web, an orthogonal architecture depicting security functionality, is presented. Even though these security technologies are not part of the *language* architecture of the Semantic Web, they are indispensable for the eventual realisation of the Semantic Web.

General security principles have to be incorporated in the development of the security stack. Security encompasses at least the following aspects [230]:

- **Integrity**: ensures that the data is not altered in transit.
- **Confidentiality**: ensures that only the intended recipient accesses the data being exchanged.
- **Authenticity**: ensures that the data was sent by the person who claims to be the originator.
- **Non-repudiation**: ensures that the sender of data cannot deny sending it.

These aspects need to be integrated into the security functionality of all layers of the Semantic Web. At present it is possible to derive what, for instance, *Integrity* would entail in reference to the *Unique Identification*, *Syntax Description* and *Meta-data Data Model* language layers. Integrity ensures that the data should not be altered in transit between communicators that interact on these language layers. However, *Ontology* layer functionality includes the possibility to *derive* data, and it not clear how *de-*

rived data influences the *Integrity* security aspect on the Semantic Web. Future research should address the incorporation of the mentioned security aspects on each layer of the Semantic Web. Refinement of the security architecture for the Semantic Web therefore remains a topic for future research.

At present, no publication specifically addressing a security architecture for the Semantic Web could be found in literature or the activities of the W3C, even though there is active interest amongst researchers in related security aspects of the Semantic Web. For example, Ashri, Payne, Marvin, Surridge and Taylor [8] support the development of a Semantic Web security infrastructure that combines conventional security solutions with the ability to reason about security at the semantic level. Yague, Mana, Lopez and Troya [263] argue for the application of Semantic Web concepts and technologies to access control. Nejdl, Olmedilla and Winslett [176] extend research on policy languages for trust and security requirements to access control policies controlled at run time for trust establishment.

Within the activities of the W3C, XMLSig or XML Signature [251] is a W3C Recommendation that specifies how to achieve integrity, message authentication and/or signer authentication services for XML data. On the other hand, XMLEnc or XML Encryption [139] provides end-to-end security for applications that require secure exchange of structured data. XML Encryption specifies that each participant in the data exchange can maintain secure or insecure states with any other communication participants. In addition, both secure and non-secure data can be exchanged in the same document [139]. Furthermore, the W3C initiated a Security Activity [256] that focuses on the challenges that arise when Web users encounter currently deployed security technology. The formulation of initiatives that address specific security aspects pertaining to the Semantic Web is a requirement for future W3C activities.

7.4.4 The layered structure

The proposed CFL architecture retains the triangular structure depicted in the present versions of the Semantic Web layered architecture (Figure 7.1).

However, in contrast with the present versions of the Semantic Web layered architecture where the meaning of the structure is not defined, the structure of the CFL architecture implies that applications or systems could exchange data on any layer as discussed in section 2.4.2. In addition, a layer may be represented by more technology functionality than what is required by upper layers. For instance, in the case of Layer 2, XML with its associated technologies, such as DTD and XML Schema is regarded as an instantiation of the *Syntax Description Language* layer. Any application exchanging XML data will probably use XML with one or more of the associated technologies such as XML Schema. However, its upper layer (the *Meta-data data model*) only requires XML, which is the reason for a wider *Syntax Description Language* Layer than a *Meta-data Data Model* Layer.

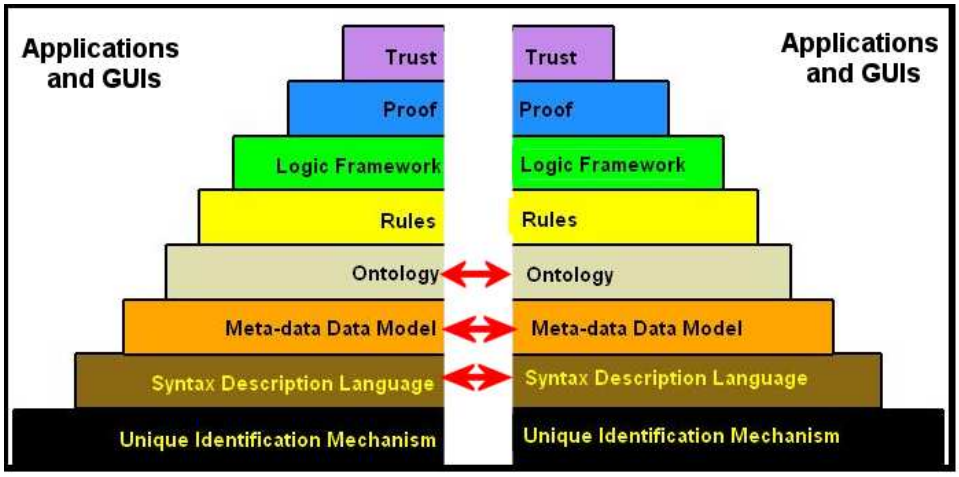


Figure 7.4: Applications and GUIs can be implemented to interface at any layer with the architecture. However, both sides must use a similar implementation of the chosen interaction layer.

Figure 7.4 depicts *Applications and GUIs* that are able to interface with any layer on account of the triangular structure (in contrast with V4 of the layered architecture by Berners-Lee depicted in Figure 7.1 (p.183) where *User Interface and Applications* are only depicted as the uppermost layer, Layer 8). However, the exchange of data, information or knowledge between applications (refer to Section 2.4.2) require a similar implementation of the layers within each application as depicted by the horizontal arrows in

Figure 7.4.

7.4.5 The definition of layer functionality

In this section, a more detailed discussion of the functionality to be contained in each layer of the proposed Semantic Web architecture, is presented. This functionality is derived from the definitions of the Semantic Web as discussed in Chapter 2 (Section 2.3.1 the present Semantic Web layered architecture as discussed in Chapter 3 (Section 3.2), and the technology discussions contained in Appendix A (p.281).

Conforming to the principle of *value addition*, the layers increasingly specify functionality required to describe meaning, or add semantics to data. It is a prerequisite that the layer implementations of the specific layer with which applications choose to communicate, are implemented similarly in both applications as depicted by the arrows in 7.4. For example, if systems communicate at a specific layer within the ISO/OSI architecture, the implementation of the protocol on that layer should be the same (refer to Figure 7.5). Using the proposed CFL architecture for the Semantic Web, system developers may therefore decide to allow applications to communicate at different layers, however the instantiation of the technology used at the layer of communication should be similar.

7.4.5.1 Unique identification mechanism

The function of this layer within the Semantic Web architecture is to uniquely identify resources used. Any discussion about data or meaning on the Web will have to uniquely identify the resources under discussion. Unique identification entails two aspects, namely unique representation and unique encoding.

Unique representation of any resource is required before the meta-data of the specific resource can be captured. On the Semantic Web, an additional requirement for any unique representation method is that it should be universal. The Semantic Web is not restricted by geographical bound-

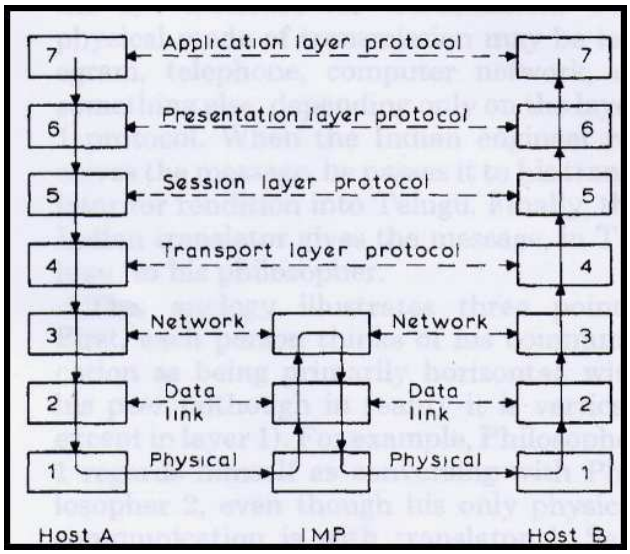


Figure 7.5: Communication at different layers of the ISO/OSI architecture. At the layer at which the applications decide to communicate, the protocol implementation should be the same.

aries, and any mechanism that uniquely represents a resource, should do so globally.

Unique encoding supports unique representation in that the encoding of the resource representation is done in such a way that it is globally understood by computer systems on global networks.

By combining unique encoding with unique representation, it is ensured that the meta-data that is captured for the purposes of the Semantic Web of any resource, is uniquely associated with the specific resource and is captured in such a way that it is uniquely interpreted by global computer systems.

At present this functionality is instantiated by the URI and Unicode technologies. URI instantiates unique representation, whilst Unicode instantiates unique encoding. The depiction of these two technologies as side-by-side layers in V1-V4 of the present Semantic Web layered architecture (Figure 7.1 on page 183) is confusing since URI makes use of Unicode to encode the specified resource identifications. By means of Unicode the URI is interpreted in the same way globally.

7.4.5.2 Syntax description language

The function of this layer is to provide a syntax language or a serialisation mechanism for data transfer. At this level, *meaning* is contained within the application or system. Syntax descriptions do not describe meaning.

A *Syntax Description Language* has as its main objective the packaging of meta-data data into a structure recognisable by the applications or users. The acceptable and common mechanism to achieve this is through markup tags, hence the development of technologies such as HTML, SGML and XML. When using HTML, the *applications* exchanging data is primarily concerned with its display (on a Web page for instance) and the packaging of the data therefore concentrates on capturing its display information.

When exchanging data for use within applications that are not primarily concerned with the display of the data, alternative syntax description mechanisms are used that package the data in such a way that it can be interpreted by the different applications. However, the participant applications must receive an agreed-upon data schema that captures the structure of the data for the application. That way the different participant applications all use and interpret the data in a similar manner. DTD's and XML Schema are additional mechanisms that assist with the verification and validity checks of such encoded data documents.

Systems that exchange XML data using for instance DTD or XML Schema, operate on Layer 2 of the architecture.

For the purpose of the Semantic Web, meta-data has to be captured with increasingly semantic richness. After resources were unique identified with the chosen *unique identification mechanism* technologies, the data about the resources have to be packaged. If the *meaning* of the packaging is prescribed by applications using a predefined schema, the scenario is as depicted in Figure 7.6. However, the meaning of the data may be prescribed by a semantically richer meta-data language when moving to a higher layer in the proposed CFL architecture for the Semantic Web.

The instantiation of this layer at present is XML with its associated technologies such as XML Schema and DOM (Document Object Model). Only

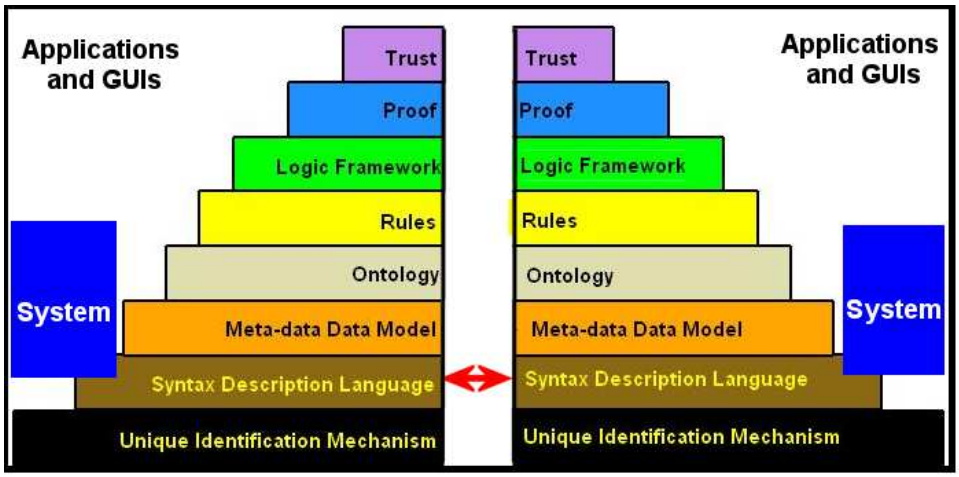


Figure 7.6: Communication on Layer 2 of the proposed CFL architecture for the Semantic Web.

the XML functionality is required by the higher Semantic Web languages.

7.4.5.3 Meta-data data model

The Meta-data Data Model describes the model that will be used to add descriptions of meaning to data. The lower layers firstly provide a means to uniquely identify a resource through the *Unique Identification Mechanism* layer, and a method to package the meta-data through the *Syntax Description Language*. In order to provide a next level of semantically enriched data, it is necessary to define a structure or model for the description of the meta-data by means of basic statements. This function is encapsulated by the *Meta-data Data Model* layer in the CFL architecture for the Semantic Web.

The technology that is defined by the W3C on this level is RDF [231]. RDF provides a mechanism for the representation of meta-data about resources on the Web by means of a basic data model for *meta-data statements*. A *statement* describes an entity (resource) in terms of *properties*, which have *values* [85, 92, 238].

It is proposed that RDF Schema is integrated into the next layer repre-

senting the *Ontology* function. RDF Schema is based on RDF and is an extension of RDF. RDF Schema provides a predefined, basic type system for RDF models, and specifies a (limited) mechanism to add semantics to meta-data or to represent knowledge [243]. Because it defines structure and meaning above the data model, RDF Schema is already elevated to a higher semantic layer.

7.4.5.4 Ontology

The function of the Ontology layer is to provide a mechanism to represent knowledge formally in such a way that basic inferencing is supported. After a meta-data data model has been specified, the formalisation of the resources and their relationships is facilitated by means of an *ontology*. Inferencing is possible once the description of resources with their relationships have been formalised.

RDF Schema and OWL are the W3C technologies that instantiate this layer at present. OWL requires RDF Schema as a building block and this provides another reason for RDF Schema to be integrated into this layer. If we exchange OWL with its associated DL (Description Logics) for another formalism, it will build on either RDF, or on another data model.

At this layer it is possible to argue for the inclusion of a sub-layer as identified in the layering principles established by the ISO/OSI architecture (refer to Section 7.3.2.1). In the ISO/OSI model description, Zimmermann [265] discusses the meaning of the defined layers in substantial detail and introduces the principle of sub-layers that implies that further sub-grouping and organisation of functions in a layer to form sub-layers is possible. In addition, he states that it should be possible to bypass sub-layers. In presenting the instantiations of OWL and RDF Schema, OWL could be included as a sub-layer as depicted in Figure 7.7. The *Rules* instantiation above RDF Schema could be RIF, no technology for the *Rules* functionality above OWL has been specified yet.

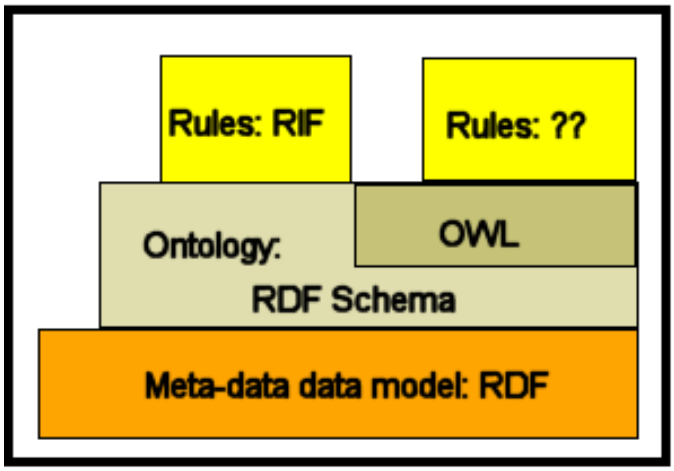


Figure 7.7: OWL as a sub-layer in the instantiation of *Ontology*.

7.4.5.5 Rules

The function of *Rules* is to provide a mechanism to introduce rules (such as business rules) above the knowledge base and its inferencing. A simplified explanation at this stage would suffice - if an inference action on an ontology produces the result A, a *Rules* instantiation might add an *if A then action B else action C* formalism. The `make` build environment popular for programming is an example of a primitive rule system. The function of *Rules* is therefore to add a mechanism for extending the inferencing capabilities of the *Ontology* layer. According to Grosof [114] there are at present four noteworthy families of rule systems namely (1) SQL or relational database, (2) Prolog, (3) production rules systems such as CLIPS and Jess, and (4) ECA (Event-Condition-Action) Rules. All of these systems could possibly be integrated into the Semantic Web *Rules* layer.

The W3C is working on RIF and SWRL as possible technology instantiations for the *Rules* layer [132, 247]. The SWRL proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. RIF allows for a *Rules* extension above RDF-Schema.

However, at present no technologies are formally defined to instantiate this layer and the *Rules* layer remains a research endeavour.

7.4.5.6 Logic Framework

The purpose of the Logic Framework Layer is to provide an overarching mechanism for the integration and extension of different Logic formalisms of lower layers. It is proposed that this function remains the same for the first version of the CFL architecture.

As example of the function of the *Logic Framework* layer, the work of Thomans and Sheth [221] is presented. These authors argue that logic-based systems require the maintenance of a consistent knowledge base for optimal functioning. They further argue that a decidable logic language is required to maintain this consistent knowledge base computationally and that it is possible to extend OWL to be able to express uncertainties. In addition, Thomans and Sheth [221] propose increased expressiveness in the layers of the Semantic Web, notable the *Logic framework* layer, to enable this extension or augmentation of OWL. For the extension, they propose an incomplete graph-based algorithm with more rigid logic formalisms. This approach may in future represent multiple ontologies and may furthermore allow for inferencing across the ontologies.

However, there is at present no formal technology specification that can instantiate this layer, and possible instantiations remains a future research endeavour.

7.4.5.7 Proof

The Proof Layer has to provide the *proof* functions within the context of the Semantic Web. *Proof* is defined as whether a result due to inferencing and rules can be regarded as valid. In essence, this layer has to answer the question *'How can I believe what the computer tells me?'*. This is done by providing the *proof* that the result as achieved is valid, as well as by giving an explanation as to how this proof was obtained.

Recent relevant research includes the proposal of PML, a *Proof Markup Language for Semantic Web Services* [77]. PML allows Semantic Web services to explain their results by generating justifications in an exchangeable and combinable format. PML aims to represent different kinds of proofs

ranging from formal natural deduction derivations to proof traces required by optimised theorem provers. In addition to the representation of proofs, different kinds of explanations have to be represented, ranging from summaries of assertions to formal derivation paths [77].

It is proposed that this function remains the same for the first version of the Semantic Web CFL architecture. There are no formally specified technologies available at present that can instantiate this layer.

7.4.5.8 Trust

The Trust Layer has to provide *trust* functions within the context of the languages of the Semantic Web. In addition, *Trust* has to provide a mechanism to ensure that any conclusion or subsumption can be trusted to be valid.

The issue of *trust* on the Semantic Web is a wide-ranging topic, from digital signature implementations to social networks [198]. The essential issue with regard to trust is to decide to what degree credence should be allocated to a source on the Semantic Web. One possible solution to this problem is the specification of a language that allows for the building of a *trust network* where a user specifies whom he or she trusts and how much [198]. This notion is extended by Nejdl, Olmedilla and Winslett [176] who propose the development of policy languages for trust negotiations between peers on the Semantic Web.

It is proposed that this function description remains the same for the first version of the Semantic Web CFL architecture. There are no formal technologies available at present that instantiate this layer, and as such, it remains a future research endeavour.

7.4.5.9 Security stack

The proposed CFL architecture for the Semantic Web includes the security functionality into an orthogonal stack with two layers. This security stack is based on the security aspects depicted as vertical layers in versions V1-V3

of the presented Semantic Web layered architecture (Figure 7.1). V4 of the present Semantic Web layered architecture seems to depict the realisation that the security issues of the Semantic Web are unresolved because of the inclusion of only one vertical layer with caption *Crypto*.

In the proposed CFL architecture, the function of *Digital Signatures* is incorporated into the *Identity Verification* layer. The function of this layer in the security stack is to unambiguously verify an identity on the Semantic Web. It has a functionality grain similar to the *Unique Identification Mechanism* layer in the language stack, and was therefore incorporated as the bottom layer. *Encryption* is an overarching term used to ensure data security, and encryption functionality is required on all layers of the Semantic Web language stack. The *Encryption* layer is therefore depicted as interfacing with all the language layers excluding the bottom Unique Identification Mechanism layer.

Security encompasses at least the following aspects [230]:

- **Integrity**: ensures that the data is not altered in transit.
- **Confidentiality**: ensures that only the intended recipient accesses the data being exchanged.
- **Authenticity**: ensures that the data was sent by the person who claims to be the originator.
- **Non-repudiation**: ensures that the sender of data cannot deny sending it.

It is necessary to address the incorporation of these security aspects when accessing the functionality of each language layer of the Semantic Web in a responsible manner. The development of the *Security stack* in the CFL architecture for the Semantic Web remains largely a topic for future research and it has been excluded from the scope of this thesis. At this stage, the functional layers of such an architecture as well as the integration thereof into the *language* architecture are not yet resolved. It is possible to speculate that there are convincing arguments for the inclusion of certain security functionality into language layers, as well as arguments for the extraction of certain security functionality into a separate layered architecture with definite boundary interfaces with the language layers.

7.5 THE EVALUATION OF THE PROPOSED CFL ARCHITECTURE FOR THE SEMANTIC WEB

In this section the proposed criteria for the evaluation of layered architectures as derived in Chapter 6 are used to evaluate and discuss the proposed CFL architecture for the Semantic Web.

Criteria	Adapted Semantic Web Architecture
Clearly defined context	**Conform to:** The context is the languages necessary to implement the Semantic Web.
Appropriate level of abstraction and hiding of implementation details	**Conform to:** All layers depict functionality. In addition, all the envisioned functions as required for data interoperability on the Semantic Web, are depicted.
Clearly defined functional layers	**Conform to:** All layers depict functionality, in addition, the functionality of each layer is defined.
Appropriate layering, including well-defined interfaces and dependencies	**Conform to:** The functional layers build on one another. Furthermore, an initial attempt is made to define the functionality requirements of upper layers with regard to their lower layers. In addition, the CFL architecture is defined as a closed architecture, implying that any layer may only access its immediate lower layer and may not bypass any lower layers.
Modularity	**Conform to:** It is possible to instantiate the functionality of a specific layer with different technologies.

Table 7.2: Evaluation of the proposed CFL architecture for the Semantic Web.

The evaluation of the original Semantic Web architecture is presented in Table 7.1 (p.184) where it is argued that the existing Semantic Web layered architecture (including its versions) does not comply with the majority of the evaluation criteria for layered architectures. In contrast, it is possible to argue that the adapted CFL architecture conforms to the criteria for layered architectures. The remainder of this section (Section 7.5.1 to 7.5.5) expands on the reasons for conformance of the proposed Semantic Web CFL architecture to the evaluation criteria.

7.5.1 Clearly defined context

As discussed in Section 7.4.1, the CFL architecture for the Semantic Web is accepted to be the *meta-data language functionalities* required to implement the Semantic Web.

7.5.2 Appropriate level of abstraction and hiding of implementation details

As discussed in Section 7.4.2, all layers in the proposed CFL architecture for the Semantic Web portray the *functionalities* necessary to implement the languages of the Semantic Web. As a result the system under review (the languages for the Semantic Web) can be viewed as a whole and only the aspects that are relevant for a specific layer are visible.

7.5.3 Clearly defined functional layers

As discussed in Section 7.4.5, the functionality of each layer in the proposed Semantic Web CFL architecture is defined. In addition, the layers increasingly specify functionality required to describe meaning, or to add semantics to data.

7.5.4 Appropriate layering, including well-defined interfaces and dependencies

As discussed in Section 7.4.4, the proposed CFL architecture retains the triangular structure depicted in the present versions of the Semantic Web layered architecture (Figure 7.1). However, in contrast to the current Semantic Web layered architecture where the meaning of the structure is not defined, the structure of the CFL architecture implies that applications or systems could exchange data on any layer given that the implementations on both sides are the same. In addition, any layer functionality may be instantiated by more than one technology.

7.5.5 Modularity

As discussed in sections 7.4.5 and 7.4.4, the functionality of any layer may be instantiated by more than one technology without compromising the integrity of the layered architecture.

7.5.6 Concluding remarks

It is possible to argue that the proposed CFL architecture for the Semantic Web conforms to the established criteria for layered architectures.

7.6 CONCLUSION

In this chapter a first iteration of the development of a proposed CFL architecture for the Semantic Web, is undertaken. This architecture adheres to the fundamental aspects of layered architectures within the Information Systems domain. Using a previously compiled evaluation mechanism for layered architectures, the present Semantic Web layered architecture of Berners-Lee is evaluated, and it is established that this architecture (including all its versions) does not conform to the majority of the evaluation criteria for layered architectures. The results of the evaluation of the present architecture, the criteria of the evaluation mechanism for layered architectures, as well as additional design criteria, constitute the building blocks for the development of the initial version of the CFL architecture for the Semantic Web. This CFL architecture conforms to the criteria for layered architectures as presented in the evaluation mechanism for layered architectures.

The research results in this chapter were accepted for presentation and publication in the proceedings of the IASTED International Conference - SE2007, part of the 25th IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria, To be held 13-15 February 2007 [110].

At this stage a caveat is presented and this caveat provides the rea-

son for the inclusion of *towards* in the title of the thesis and this chapter. The ISO/OSI architecture was developed over 18 months, which was considered remarkable at the time [265]. More than hundred participants took part in the discussions [265]. The process for the establishment of a *generally agreed upon* layered architecture for the Semantic Web prescribes many more participants and have to be the result of considerably more debate. However, this thesis argues for the approach to establish an agreed-upon CFL architecture, and proposes a preliminary version of such a architecture for the Semantic Web.

From Figure 7.3 a reader may conclude that the proposed CFL architecture depicts only a simplification of the original architecture. The *abstraction* of the functionality from the original versions of the Semantic Web layered architecture does indeed entail a simplification of the model, but this simplification introduces several advantages such as a mode understandable and universal model that facilitates debate. One of the most significant advantages is that the abstraction enables the use of diverse and non-related technologies to implement the functionality of a specific layer. In other words, different technologies could be used to instantiate the functionality of a specific layer in the proposed Semantic Web layered architecture.

A generalised layered architecture for the Semantic Web with well-defined functionalities will assist in the acceptance of diverse technologies for implementation of the required functionalities. This will aid the eventual realisation of the Semantic Web. It is acknowledged that all issues with regard to the layering of the Semantic Web languages have not been resolved by this approach. However, the establishment of a first iteration CFL architecture for the Semantic Web is an important and necessary step towards realising the eventual notion of the Semantic Web. In addition, the CFL architecture for the Semantic Web should assist in the development of Semantic Web specifications and applications, or even W3C Recommendations, and several of the current research and implementation issues associated with the implementation of the Semantic Web could potentially be resolved.

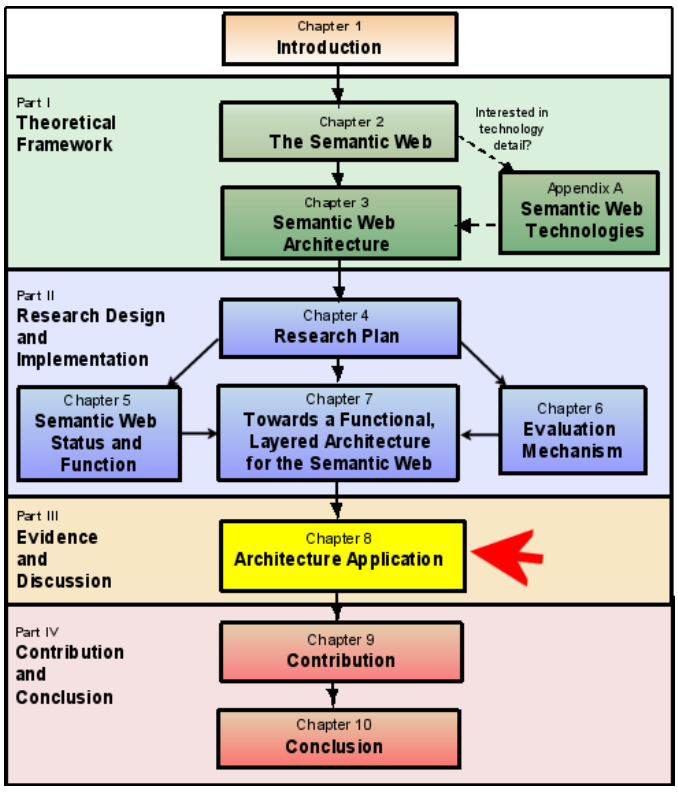
Within the next chapter, Chapter 8, the proposed CFL architecture for the Semantic Web is applied to four usage scenarios in order to determine the usefulness of this architecture.

Part III

EVIDENCE AND DISCUSSION

CHAPTER 8

ARCHITECTURE APPLICATION



Thesis Chapter Layout

Chapter Contents

Figures

8.1 INTRODUCTION

In this chapter, the proposed CFL architecture for the Semantic Web as developed in Chapter 7, is applied to usage scenarios. The results of these applications are used to determine the value and therefore possible contribution of the proposed CFL architecture for the Semantic Web.

The first usage scenarios were obtained from literature and the proposed CFL architecture is applied to solve some of the issues with regard to the layering of the Semantic Web (Sections 8.2 and 8.3). In addition, the current V4 Semantic Web layered architecture of Berners-Lee [35] is investigated as being an instantiation of the proposed CFL architecture (section 8.4). Lastly, a practical application example was developed using the proposed CFL architecture in Section 8.5. Section 8.6 concludes the chapter.

8.2 SCENARIO: THE TWO-TOWER ARCHITECTURE

Originally, versions V2 and V3 of the present Semantic Web architecture as proposed by Berners-Lee were discussed by Kifer, Bruijn, Boley and Fensel [147], who argued strongly in support of *multiple independent, but interoperable, stacks of languages*. They argued that a single stack architecture as depicted in V2 (Figure 8.1) is *unrealistic* and *unsustainable* and they regarded the side-by-side layering of *OWL* and *Rules* in V3 (Figure 8.2) as an implementation of the *interoperable stacks of languages* concept. They therefore supported the *multi-stack architecture* as depicted by V3. In particular, they discuss the incorporation of the Datalog rule-based languages as *Rules*, which could not be layered on top of OWL, as a separate stack alongside the stack of OWL-based languages with their prescribed Description Logics.

However, this discussion does not seem to concur with the proposal of Berners-Lee and the W3C in their *Rules* discussions (refer to Section A.11.2.3 on page 348 for a description of the RIF Working Group of the W3C).

These diverse interpretations of the meaning of layers in the present

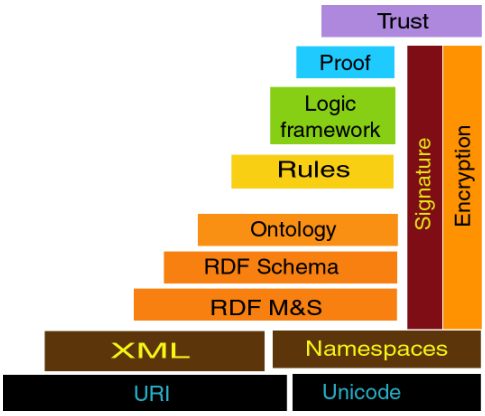


Figure 8.1: The V2 version of the Semantic Web architecture [31–33].

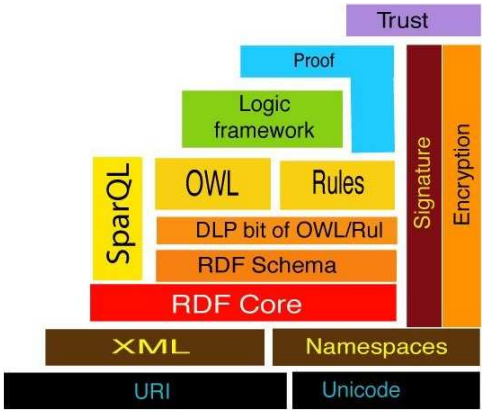


Figure 8.2: The V3 version of the Semantic Web architecture [34].

versions of the Semantic Web layered architecture emphasise the need for a standardised architecture (such as the CFL architecture) with specific meaning associated with the layers as proposed in this thesis.

Horrocks, Parsia, Patel-Schneider and Hendler [130] entered the debate about the positioning of *Rules* with their *Semantic Web Architecture: Stack or Two Towers?* article. They further argued that a realistic architecture for the Semantic Web had to be based on multiple independent, but interoperable, stacks of languages. In particular, they pointed out that DLP/Datalog and RDFS/OWL are not semantically compatible and cannot be layered as proposed in the V3 version of the present Semantic Web layered architecture (Figure 8.2). They consequently proposed a two-tower architecture

(Figure 8.3) as the solution for the acceptance of both RDFS/OWL and DLP/Datalog as Semantic Web languages or technologies [130].

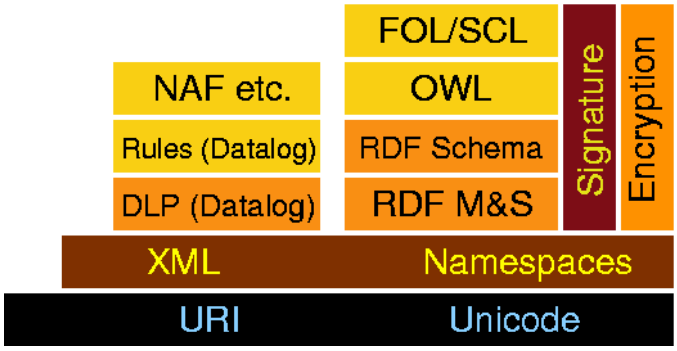


Figure 8.3: The two-tower architecture of Horrocks, Parsia, Patel-Schneider and Hendler [130]

This two-tower architecture of Horrocks et al. [130] depicts two possible instantiations of the Semantic Web layered architecture. In order to demonstrate the value of the proposed CFL architecture for the Semantic Web presented in this thesis, the instantiations in the two towers of Horrocks et al. [130] are related to the CFL architecture, which is based on fundamental Software Engineering principles that allow for the inclusion of different technologies. The relation of the two towers to the CFL architecture for the Semantic Web is depicted in Figures 8.4 and 8.5.

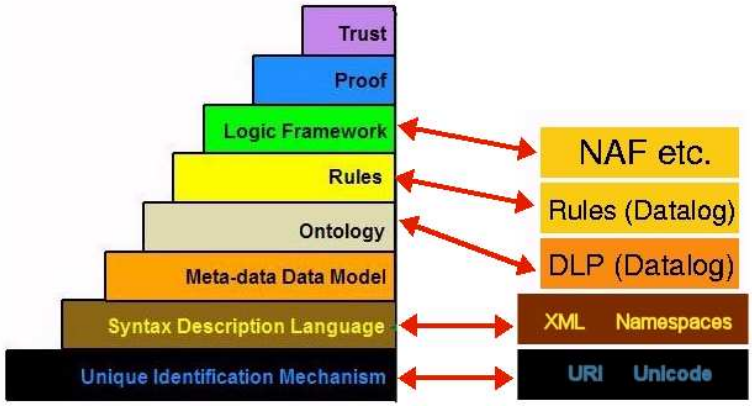


Figure 8.4: The two-tower architecture: Tower 1 as instantiation of the Semantic Web CFL architecture

In *Tower 1* on the left-hand side in Figure 8.3, Horrocks et al. omitted an explicit data representation layer. Datalog was developed as a rule and query language for deductive databases and syntactically Datalog is a subset of Prolog, hence it does not constitute a data representation layer. Datalog supports both relational and object databases, and to improve the composition of Tower 1, a *meta-data data model that is either an entity-relationship model or a UML model* could be added. Thus, the adapted Semantic Web architecture supports more than one technology to be used in this layer.

In addition, in Tower 1 in Figure 8.3, the Ontology functionality is implemented by DLP (Description Logic Programs) where these DLP extend Datalog to include knowledge representation. The Rules functionality is implemented with Datalog Rules and NAF. This relationship is depicted in Figure 8.4.

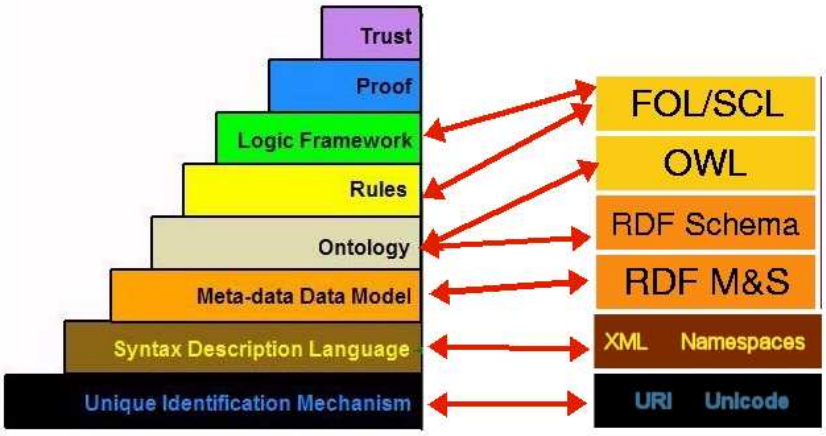


Figure 8.5: The two-tower architecture: Tower 2 as instantiation of the Semantic Web CFL architecture

In contrast to Tower 1, Tower 2 on the right-hand side in Figure 8.3 implements the meta-data data model functionality by means of RDF. The Ontology layer functionality is implemented using RDF Schema and OWL. In this study, the contention is that the FOL/SCL layer provides for Rules and to a limited extent, Logic Framework functionality. This instantiation is depicted in Figure 8.5.

Note that the purpose of this scenario is not to resolve the layering debate of the different technologies. In this study, the development of a functional layered architecture of the Semantic Web that is based on fundamental principles of layered architectures, is argued. Such an architecture defines *functionality* that has to be implemented by the different layers, and which allows for the acceptance of diverse technology *instantiations* that implement the requisite functionality. This is aptly demonstrated in this usage scenario where both the *towers* are accommodated by the proposed Semantic Web CFL architecture, even though they represent different semantic bases.

8.3 SCENARIO: A LAYERED APPROACH TO INFORMATION MODELLING AND INTEROPERABILITY ON THE WEB

Because of the variety of information models available to fulfil the diverse information needs of users today, Melnik and Decker [163] proposed a *layered approach to information modelling and interoperability on the Web*. Their layered approach borrowed from the layered software structuring technique used in internetworking, notably the ISO/OSI layered architecture. They regarded the provision of a clear distinction between services, interfaces and protocols used in internetworking as one of the major contributions of the ISO/OSI architecture model.

Melnik and Decker [163] argued that interoperability between data models could be simulated with interoperability of networks. Their model of network interoperability is depicted in Figure 8.6. They suggested that a layered approach could greatly facilitate interoperation between data models. In their paper they discussed several proposed Semantic Web languages notably RDF and RDF Schema (RDFS), SHOE (Simple HTML Ontology Extensions), UML (Unified Modeling Language) and OIL, in order to identify how data-modelling primitives are used in the languages, and how these languages can be organised into layers to support interoperability as

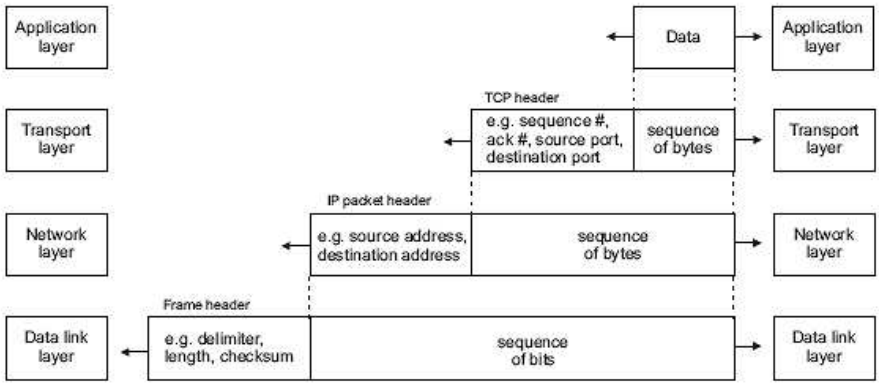


Figure 8.6: An example of networking layers [163].

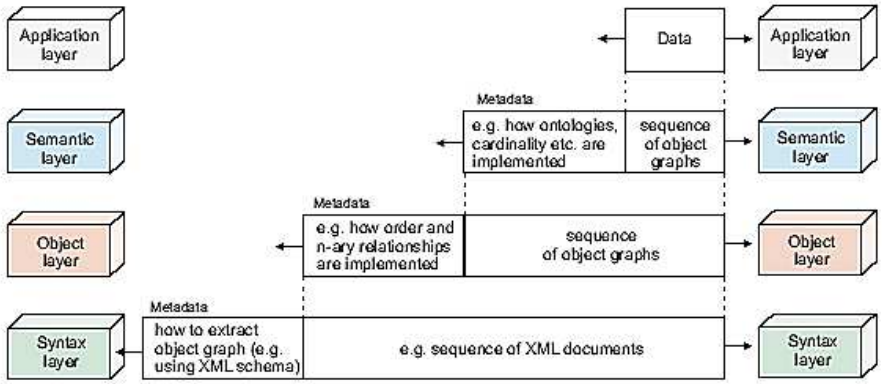


Figure 8.7: An example of data-modelling layers [163].

depicted in Figure 8.7. They identified three layers, the semantic layer, the object layer and the syntax layer, which are depicted in Figure 8.8. Within the layers are sub-layers and a sub-layer corresponds to a specific data-modelling feature.

Melnik and Decker [163] mention several advantages of layering, including simplification, reduced costs of application development due to well-defined interfaces, as well as independence and modularity. However, these authors focus mainly on the implementation of their *object* layer and include RDF as well as SHOE, OIL and OWL into this layer. It is not clear what the implementation of the *semantic* layer would entail.

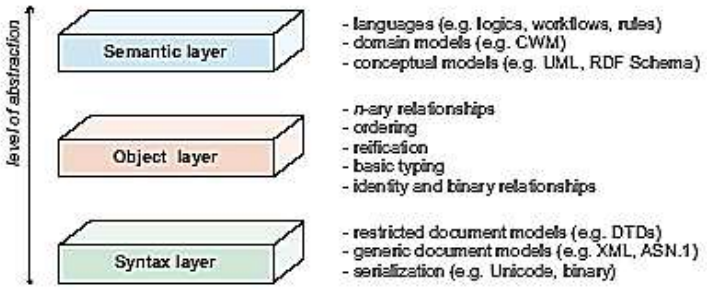


Figure 8.8: The syntax, object and semantic layers [163].

The concept of layering adopted by Melnik and Decker [163] supports the approach followed in this study. A layered approach for the implementation of the language stack for the Semantic Web has many advantages that have been identified by both models. The noteworthy difference is that the CFL architecture proposed in this thesis is based on both the current layered architecture for the Semantic Web as well as fundamental principles of layered architectures within Software Engineering. It is plausible to state that the approach followed in this thesis to develop the Semantic Web CFL architecture is therefore more thorough, but the same observations regarding the advantages of layering is upheld.

8.4 SCENARIO: THE SEMANTIC WEB LAYERED ARCHITECTURE

In this section, the latest version (V4) of the layered architecture as proposed by Berners-Lee [35] is investigated as being an instantiation of the proposed CFL architecture for the Semantic Web. Figure 8.9 depicts the mapping of the depicted W3C technologies to the functionality layers of the CFL architecture.

Upon scrutiny, the mapping depicted in Figure 8.9 indicates irregularities, which is due to the inconsistencies in the V4 architecture as discussed in Section 3.3 (p.65).

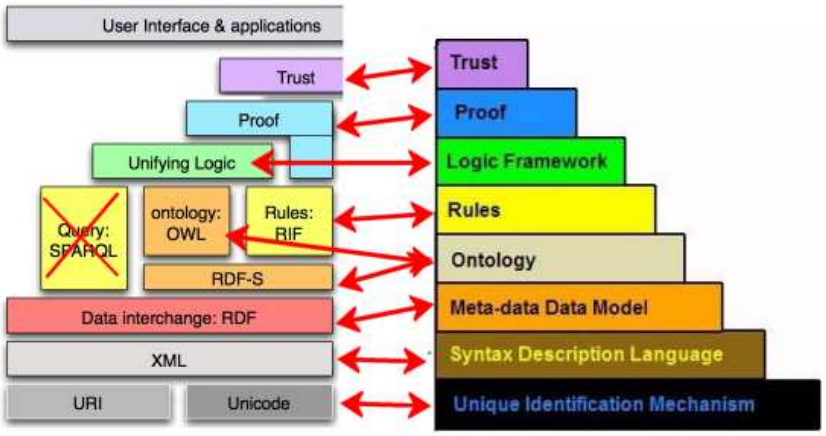


Figure 8.9: The V4 version of the Semantic Web architecture [35] as instantiation of the adapted architecture.

The *Unique Identification Mechanism* layer maps to Unicode and URI as before. URI essentially uses Unicode and therefore the meaning of the side-by-side layering of Unicode and URI in V4 is unclear. However, it is plausible to state that URI making use of Unicode is an instantiation of the layer 1 *unique identification mechanism* functionality of the CFL architecture. XML instantiates the *Syntax Description Language* layer and the *meta-data data model* is instantiated with RDF.

The *Ontology* layer of the CFL architecture is instantiated with either RDF Schema (depicted as RDF-S in V4) or RDF Schema and OWL (RDF-S and OWL). If RDF-S as instantiation is used, RIF is the instantiation of the *Rules* layer. *RIF* is an acronym for *Rule Interchange Format* and it is a draft specification for a rule language using RDF Schema as its ontology basis. RIF is discussed in Section A.11.2.3.

Above RIF, no *technologies* are depicted in V4 and therefore the *instantiations* cannot be discussed. *Unifying Logic* maps to *Logic Framework* and the *Proof* and *Trust* layers are not instantiated with any technology, thus remaining *Proof* and *Trust*. It is unclear what is meant by both *Unifying Logic* and *Proof* residing on top of *Rules:RIF*. It is possible to reason that both reside alongside each other above *Rules:RIF* or it might mean that the layered architecture is *open* in this instance so that *Proof* might

access functionality of *RIF* directly without having to access the *Unifying Logic* layer. However, in general the technologies from URI and Unicode through to RIF can be regarded as an instantiation of the adapted, functional architecture.

In contrast, if the *Ontology* layer is instantiated with RDF Schema together with OWL (RDF-S and OWL), a deviation is depicted in the layers because of the omission of a *Rules* layer above *OWL*. It is however foreseen that a rule language will be required above an OWL ontology, hence the SWRL initiative of the W3C (RIF builds on RDF Schema only). It is plausible to speculate that this omission will have to be rectified in future.

This confirms the argument that a comprehensive and functional layered architecture will allow for diverse technology specifications for the instantiation of a layer. As before, above OWL, no technologies are depicted and therefore no instantiations can be discussed. *Unifying Logic* maps to *Logic Framework* and the *Proof* and *Trust* layers relate to the *Proof* and *Trust* layers in the Semantic Web CFL architecture.

It can be argued that the latest version of the Berners-Lee Semantic Web architecture also presents two towers of possible technology instantiations, and that both can be regarded as instantiations of the adapted comprehensive and functional architecture presented in this study.

8.5 APPLICATION USAGE SCENARIO

In this section the usefulness of the CFL architecture is illustrated by simulating two applications using two different meta-data data models.

In this usage scenario, the proposed system is a distributed management application for book publishers using Semantic Web technologies of the bottom three layers as depicted in Figure 8.10.

As discussed in Section 7.4.5, the implementation or instantiation of the technology at the layer of communication should be similar. In this scenario, the meta-data data model could be either an entity relationship model, or an RDF diagram. Because the system implements up to and

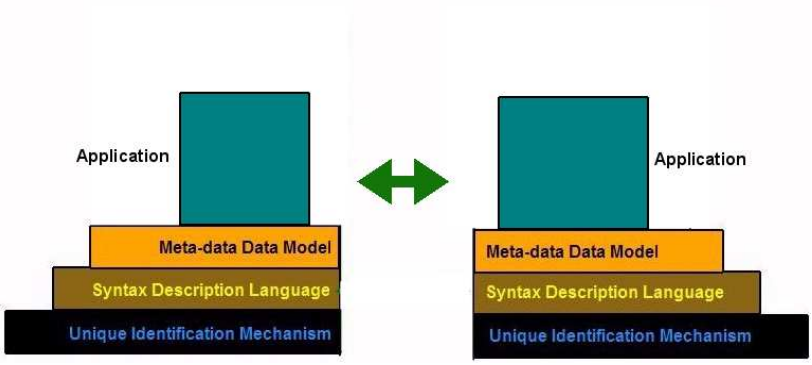


Figure 8.10: Applications using the first three layers of the proposed Semantic Web CFL architecture.

including layer three, the layer three models have to be implemented using the same technology.

The important aspect is that there is a *choice* of meta-data data models when the proposed CFL architecture for the Semantic Web is used. In contrast, the original architecture of Berners-Lee would have limited the choice to RDF.

In the scenario data model, a *publication*, which is of a specific *type* (a book in this case), has a *title ('My life story')*, an *isbn* (0-07-777123-3) and an *author*. The *author* is a *Person* with a *fullname* (John Smith) and a *mailbox* (mail:js@abc.com).

The Layer 3 implementation will be a data model using either an ER (entity relationship) diagram or an RDF diagram. A possible ER diagram is depicted in Figure 8.11. Similarly, this data can be modelled in an RDF diagram. An RDF diagram modelling the same data is depicted in Figure 8.12.

Figures 8.11 and 8.12 therefore depict two possible choices of meta-data data modelling technologies that can be used in this scenario as the proposed CFL architecture does not prescribe the technology. As long as the implementation adheres to the principles of layering and the specified functionality of the layer it implements, any meta-data data modelling technology can be used.

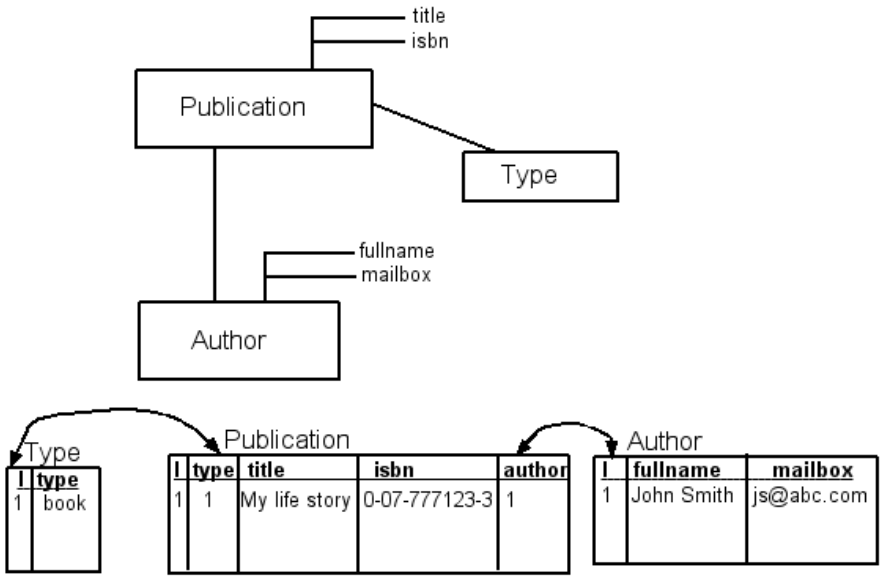


Figure 8.11: An entity relationship diagram with a database implementation modelling the *publication* data of the scenario.

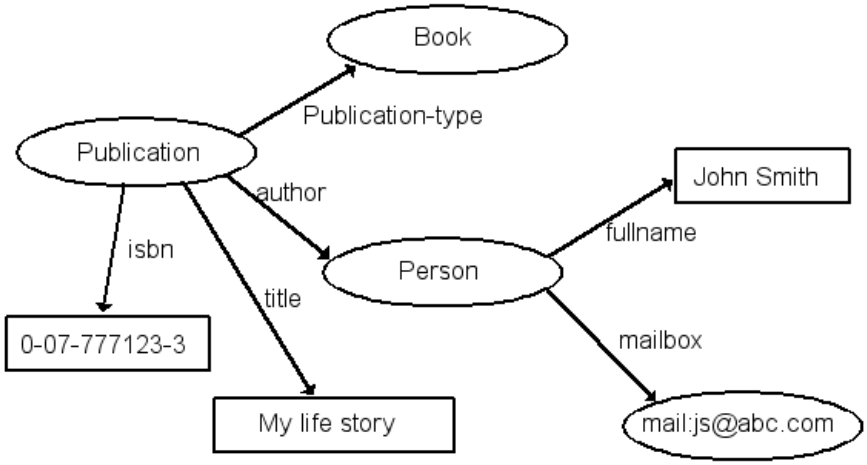


Figure 8.12: An RDF diagram modelling the *publication* data of the scenario.

The implementation of the next layer that is used (Layer 2) dictates the serialisation of the data model to XML. The code in Figure 8.13 depicts

a possible serialisation of the ER diagram to XML. Note that the functionality to serialise a database implementation to XML is implemented by most RDBMS (Relational Database Management Systems) nowadays. The `<?getUI entity ?>` simulates an XML processing instruction.

```
<?xml version="1.0"?>
<db:ERD xmlns:erd=
        "http://www.w3.org/1999/02/22-db-syntax-ns#">
 <Publication>
   <type> <?getUI typeBook ?> </type>
   <title> My life story </title>
   <isbn> 0-0-777123-3 </isbn>
   <author>
     <fullName> John Smith </fullName>
     <mailbox> <?getUImail mailbox ?> </mailbox>
   </author>
 </Publication>
</db:ERD>
```

Figure 8.13: Serialisation of the ER diagram to XML.

The XML code that is a possible serialisation of the RDF diagram is depicted in Figure 8.14. As stated, the `<?getUI entity ?>` simulates an XML processing instruction invoking a function to obtain a *Unique Identifier* from the next lower layer, which is the *Unique Identification Mechanism* Layer. This layer can also be instantiated with any technology, and this layer is accessed by calling a *getUI* function that will return a URI in this case, and the *getUImail* function will return a unique mail address for John Smith.

The XML that will be transferred between the applications when the unique identifiers are processed will possibly be as indicated in the code in Figure 8.16, and the RDF diagram depicting this data is depicted in Figure

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:publication=
         "http://www.w3.org/publication#">
         xmlns:person="http://www.w3.org/contact#">

 <publication:Publication>
   <publication:about> <?getUI this ?> </publication:about>
   <publication:type> <?getUI typeBook?></publication:type>
   <publication:title> My life story </publication:title>
   <publication:isbn> 0-0-777123-3 </publication:isbn>
   <contact:Person>
     <contact:about> <?getUI author ?> </contact:about>
     <contact:fullName> John Smith </contact:fullName>
     <contact:mailbox>
        <?getUImail mailbox John Smith ?>
     </contact:mailbox>
   </contact:Person>
 </publication:Publication>
</rdf:RDF>
```

Figure 8.14: Serialisation of the RDF diagram to XML.

8.15.

In this section a scenario is described in which an application is implemented using the proposed CFL architecture for the Semantic Web. Developers are able to choose between alternative technologies to implement the different layers of the CFL architecture for the Semantic Web effectively. In this usage scenario, it is demonstrated how two possible meta-data data-modelling technologies can be used to implement Layer 3. Both models

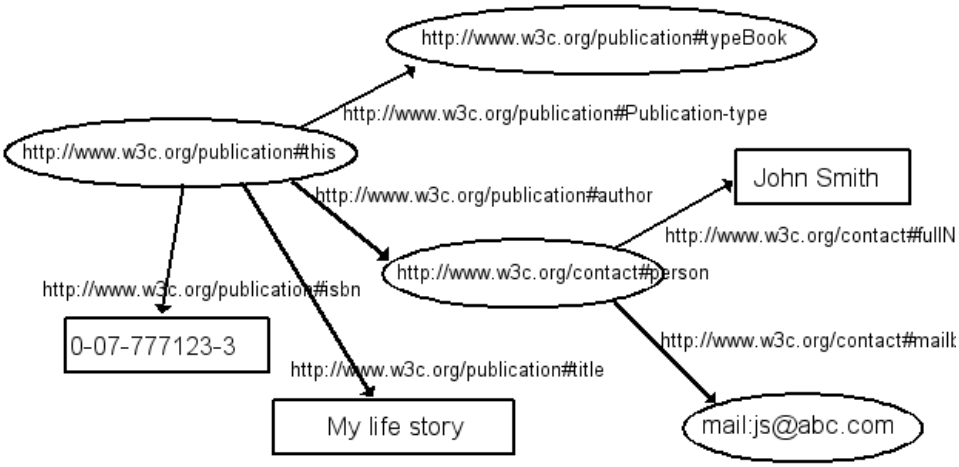


Figure 8.15: An RDF diagram with decoded URIs modelling the *Publication* data of the scenario.

use a Layer 2 that is instantiated with XML, and a Layer 1 that decodes to Unicode URIs.

The Layer 1 implementation for both systems uses URI to uniquely identify the resources under discussion, such as the `http://www.w3.org/People/EM/contact#author` URI in Figure 8.12, to identify the author. In addition, the systems encode the serialised XML using Unicode to ensure that it is uniquely interpreted across the world.

8.6 CONCLUSION

In this chapter, the proposed CFL architecture for the Semantic Web was applied to various scenarios. In all of the scenarios, the CFL architecture proved to be of value.

In the first scenario, the CFL architecture solved issues with regard to the *layering* of Semantic Web technologies. In addition, *diverse* technologies can be used to implement the same functionality as encapsulated by a layer definition.

In the second scenario, the proposed CFL architecture for the Semantic

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:publication=
         "http://www.w3.org/publication#">
         xmlns:person="http://www.w3.org/contact#">

 <publication:Publication rdf:about=
             "http://www.w3.org/publication#this"/>
   <publication:type rdf:type=
             "http://www.w3.org/publication#typeBook"/>
   <publication:title>My life story</publication:title>
   <publication:isbn>0-0-777123-3</publication:isbn>
   <contact:Person rdf:about=
            "http://www.w3.org/People/EM/contact#author">
     <contact:fullName>John Smith</contact:fullName>
     <contact:mailbox rdf:resource="mailto:js@abc.com"/>
   </contact:Person>
 </publication:Publication>
</rdf:RDF>
```

Figure 8.16: Serialisation of the RDF diagram to XML including URIs.

Web improved a layered model developed for data interoperability.

The V4 version of the present Semantic Web layered architecture is investigated as an instantiation of the proposed CFL architecture for the Semantic Web in the third scenario. The fourth scenario developed applications using different meta-data data models when incorporating the bottom three layers of the proposed CFL architecture for the Semantic Web.

It is plausible to speculate that the present versions of the Semantic Web layered architecture of Berners-Lee were developed to depict the pro-

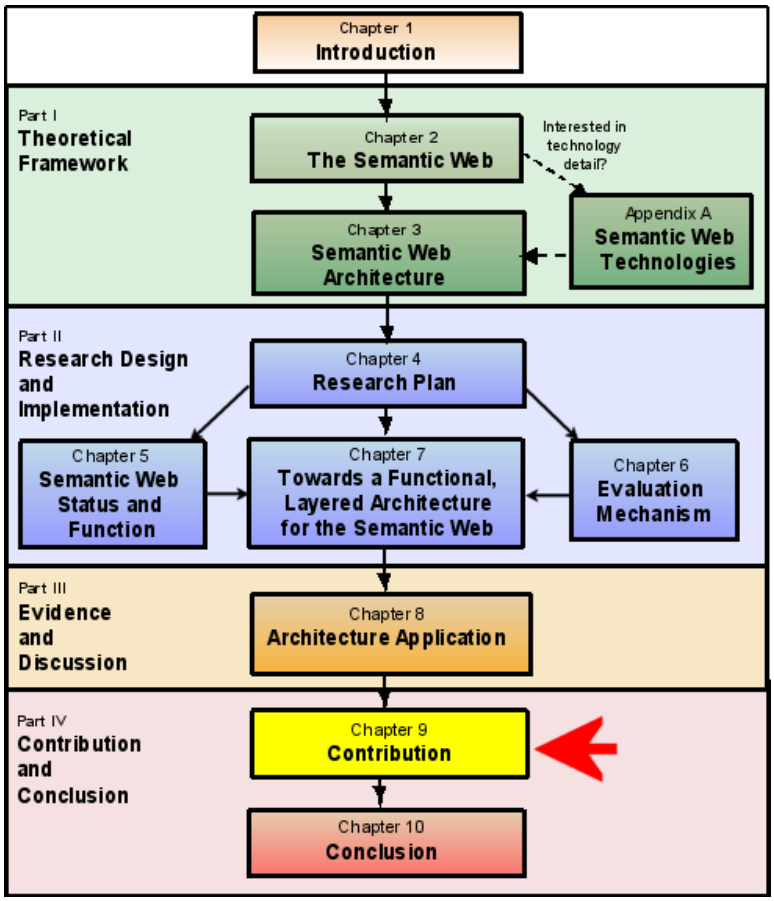
gression pertaining to the development of W3C technologies. The definition of a general, accepted layered architecture with functional components (CFL architecture) could possibly assist the W3C to develop different technology specifications for the implementation of a specific layer functionality. This approach would *include*, rather than *exclude* technologies that would have as benefit a more rapid adoption of the Semantic Web vision. In addition, the definition of W3C specifications requires an appropriate level of abstraction to be able to penetrate sufficiently into the implementation domain, and it is the contention of the author that this can be provided by the CFL architecture for the Semantic Web as proposed in this study.

Part IV

CONTRIBUTION AND CONCLUSION

CHAPTER 9

CONTRIBUTION



Thesis Chapter Layout

Chapter Contents

Figures

Tables

9.1 INTRODUCTION

In this chapter, the research contributions of this study are discussed. The purpose of this study is to develop a comprehensive and functional layered (CFL) architecture for the Semantic Web. In order to develop such an architecture, three research questions are formulated. The research that supports the answering of each of these questions constitutes the respective contributions of the study. The research questions formulated are:

	Research Questions
(1)	What is the function of each technology included in the present versions of the Semantic Web layered architecture? ▷ What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?
(2)	To which criteria should a layered architecture conform in order to adhere to system design principles? ▷ Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?
(3)	How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?

Table 9.1: Research questions

The first contribution, labelled as (1) in Figure 9.1, is extracted from the research that supports the answer to Research Question 1. This contribution is discussed in Section 9.2. The first contribution comprises a Semantic Web status model enhanced with function descriptions of listed Semantic Web technologies.

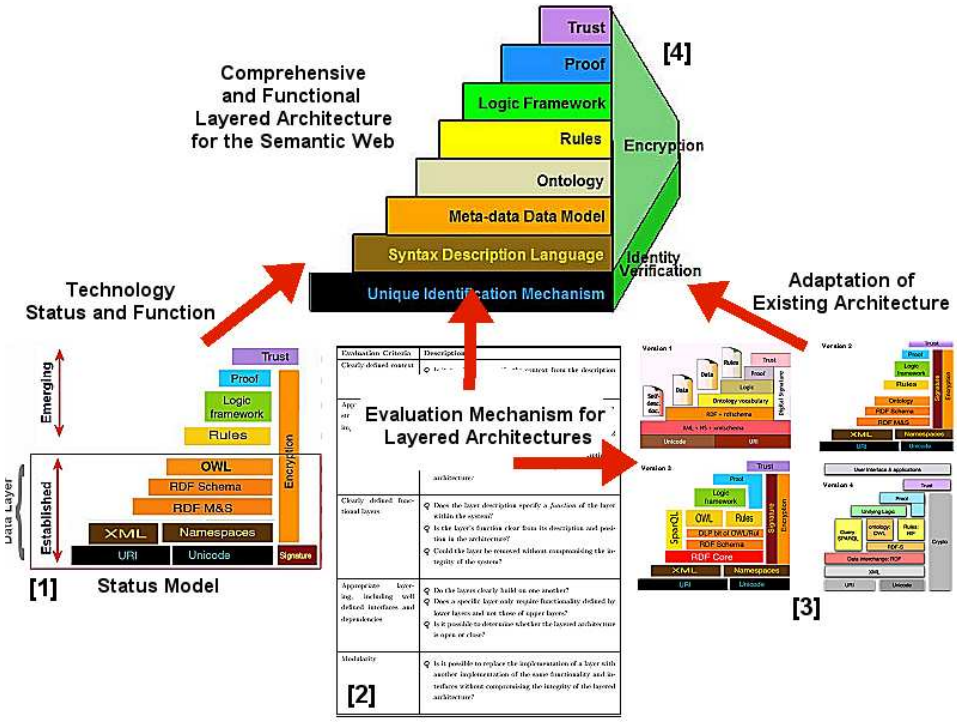


Figure 9.1: The development of the CFL architecture for the Semantic Web is supported by three research contributions.

The research that supports the answer to Research Question 2, labelled as (2) in Figure 9.1, is used to formulate the second contribution. This second research contribution comprises an evaluation mechanism for layered architectures discussed in Section 9.3.

The third contribution, labelled as (4) in Figure 9.1, is formulated from the research that supports the answer to Research Question 3. The third contribution adapts the existing layered architecture for the Semantic Web as proposed by Berners-Lee [28, 31, 34, 35] (labelled as (3) in Figure 9.1) to conform to the principles and extracted criteria for layered architectures. This contribution is discussed in Section 9.4.

In addition, three research contributions identified within the Software Engineering discipline are discussed in Section 9.5.

9.2 RESEARCH CONTRIBUTION: SEMANTIC WEB STATUS MODEL (RESEARCH QUESTION 1)

The first research question is defined as:

What is the function of each technology included in the present versions of the Semantic Web layered architecture?

- ***What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?***

9.2.1 The research activities devised for Research Question 1

In order to answer this question, an investigation into the status and function of the proposed technologies associated with the Semantic Web is performed in Chapter 5. This investigation is supported by the technology discussions contained in Appendix A. These technologies are discussed according to their positions in the layered organisation of the first two versions (V1 and V2) of the layered architecture proposed by Berners-Lee [28, 31, 33].

As result of this investigation into the different technologies, a status model reflecting the current status is compiled. Figure 9.2 presents the status model as developed in Section 5.3 (p.117). The status model is developed by adapting V2 (the second version of the layered architecture proposed by Berners-Lee [31] (Figure 5.2, p. 105)) to reflect the current status of each of the Semantic Web technologies. These adaptations comprise:

- *OWL* replacing *Ontology* on Layers 4 and 4a in order to be consistent with the bottom layers.
- *Digital Signatures* moves to Layer 1 since its function should be incorporated into the *unique identification layer*.

- ▷ Classification of the bottom four layers as *Established Technologies* since they are W3C Recommendations.
- ▷ Classification of the top layers as *Emerging Functionalities* since none of these layers has an associated technology to instantiate them.
- ▷ Classification of the bottom four layers as the *data* layer of the Semantic Web.

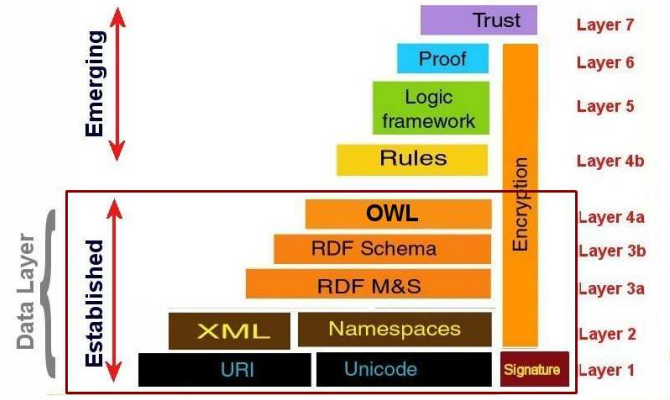


Figure 9.2: A Semantic Web status model.

However, in order to answer Research Question 1 fully, it is necessary to extract the function of the layers that depict technologies instead of functionality. Figure 9.3 depicts the status model with added functionality descriptions (refer to Section 5.5 (p.125)).

A functional status model for the Semantic Web, graphically depicted in Figure 9.3, is suggested as an answer to Research Question 1.

9.2.2 Scientific contribution: Research Question 1

The Semantic Web was introduced in 2001 by Berners-Lee et al. [38] in their vision of a new intelligent Web. In order to depict the relationship with existing W3C technology recommendations, several versions of the Semantic Web architecture were proposed (section 3.2, p.55).

The original Semantic Web vision presents [38] a description of a usage scenario and this description does not contain any implementation sugges-

tions. In addition, no meaningful descriptions associated with the proposed versions of the Semantic Web layered architecture presented by Berners-Lee [28, 31, 34, 35] could be found in literature.

This study contributes in three ways in finding an answer to Research Question 1. Firstly, the proposed *Semantic Web status model* reflects the status of the technologies that comprise the Semantic Web according to Berners-Lee [31]. The status model also reflects various refinements imposed by the status quo of current technology, and imparts some insight into the limitations of the technologies currently supporting the proposed Semantic Web layered architectures. No comprehensive discussion of the status of associated Semantic Web technologies could be found in literature, and the status model is developed from a detailed investigation into the Semantic Web with its associated technologies as currently presented in literature.

Secondly, the Semantic Web status model serves as a starting point for a prospective user or developer, who faces the daunting task of assimilating and understanding the Semantic Web and the role and purpose of its associated technologies.

The final contribution pertaining to Research Question 1 is the identifica-

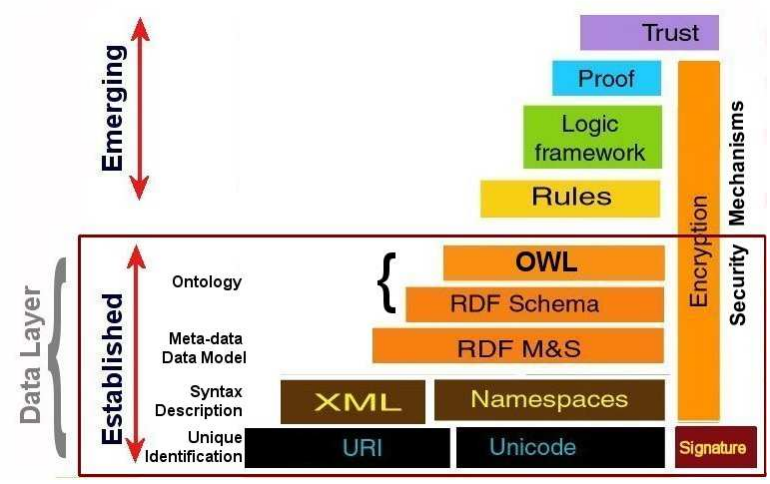


Figure 9.3: A Semantic Web status model depicting status as well as the functionality of layers.

tion of the *functionality* of technologies listed within the Semantic Web architecture. The lower layers of the proposed versions of the Semantic Web layered architecture of Berners-Lee [28, 31, 34, 35] all depict technologies without a description of function. The functionality of these technologies is extracted from the detailed investigation into the use and characteristics of these technologies (Appendix A), and the functionality descriptions are included into a functional status model of the Semantic Web. This contribution directly supports the development of a *functional* architecture for the Semantic Web that requires the identification of the function of each technology layer.

9.3 RESEARCH CONTRIBUTION: EVALUATION MECHANISM FOR LAYERED ARCHITECTURES (RESEARCH QUESTION 2)

The second research question is formulated as:

To which criteria should a layered architecture conform in order to adhere to system design principles?

- ***Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?***

9.3.1 The research activities devised for Research Question 2

In order to answer this question, an investigation into architectures and layered architectures is executed and reported on in Chapter 6.

Layering is a common best practice pattern used by software architects to structure complex systems. Even though layered architectures are often used by developers, a precise description of the architecture seldom

accompanies the architecture. No definition of a layered architecture was found during a comprehensive literature investigation except for a description of a layered architecture as an *architectural pattern*.

For the purpose of this study, a definition for a *layered architecture* is compiled in Section 6.2.4, reading as follows:

> *A layered architecture is the resulting artefact of the layering technique that is an accepted best practice used to break apart complex systems. Layering performs horizontal decomposition of system functionality. A layered architecture is therefore regarded as an architectural pattern or a type-of architecture.*

A layered architecture is an *architectural pattern* and could therefore also be described as a *type-of* architecture. The conclusion that a layered architecture is a *type-of architecture* leads to an investigation into the concept *architecture*.

There is general consensus that an *architecture* is an indispensable mechanism for the successful implementation of complex information systems. In addition, researchers and developers agree that an architecture, amongst other things, comprises the structural organisation of system components. There are several definitions for the term *architecture*, but there no single universally agreed-upon definition for the term *architecture* could be found in the literature.

For the purpose of this study, a definition for the term *architecture* is formulated following a comprehensive literature investigation in Section 6.3.4, namely:

> *An architecture is a model of a system within a specified context, depicting the components necessary to realise the system from a particular perspective. The architecture of a system includes the organisation or structure of the identified modular components, their defining features or properties, as well as the relationships and interfaces between components and outside entities.*

Within the architectural domain and the structuring of system components, *layering* is a system functionality structure using modular horizontal partitioning. A *layered architecture* is a type-of architecture and can thus also be described as an instance of an *architecture*. The elements of a layered architecture are mapped to the concepts of the adopted architecture definition as follows:

- The *layers* of a layered architecture map to specific *components*, or a *grouping of components* in an architecture.
- The stacking and sequencing of layers as depicted in a layered architecture represents the relationships and organisation of the architectural components.

In this study, the focus was not only on what architectures and layered architectures are, but also on the characteristics of such architectures. Therefore, in addition to the definitions for layered architecture and architectures compiled and listed above, key concepts of architectures and layered architectures were extracted from a comprehensive literature study, refer to Chapter 6 Tables 6.1 (p.143), 6.2 (p.152), 6.3 (p.157), 6.4 (p.160), 6.5 (p.163) and 6.6 (p.167).

The definitions and key concepts of the architectural investigations as well as additional criteria to which a layered criteria should conform, are used to compile a criteria list for layered architectures, namely: a clearly defined context; an appropriate level of abstraction and the hiding of implementation details; clearly defined functional layers; appropriate layering, including well defined interfaces and dependencies; and modularity. These criteria are combined into an evaluation mechanism for layered architectures (see Section 6.4 (p.167)) and this evaluation mechanism is depicted in Table 9.2.

Evaluation Criteria	Description
Clearly defined context	Q Is it possible to identify the context from the description of the architecture?
Appropriate level of abstraction and hiding of implementation details	Q Can the system within the context be viewed as a whole? Q Are there any components/properties/relationships in the architecture model that could be removed without losing important information at this level of abstraction? Q Are any implementation details visible in the description of the components/properties/relationships/structures of the architecture?
Clearly defined functional layers	Q Does the layer description specify a *function* of the layer within the system? Q Is the layer's function clear from its description and position in the architecture? Q Could the layer be removed without compromising the integrity of the system?
Appropriate layering, including well defined interfaces and dependencies	Q Do the layers clearly build on one another? Q Does a specific layer only require functionality defined by lower layers and not those of upper layers as well? Q Is it possible to determine whether the layered architecture is open or close?
Modularity	Q Is it possible to replace the implementation of a layer with another implementation of the same functionality and interfaces without compromising the integrity of the layered architecture?

Table 9.2: Evaluation criteria for layered architectures, including possible questions for evaluation.

In order to calibrate the evaluation mechanism that was developed for

layered architectures, this evaluation mechanism was used to evaluate an accepted and widely adopted layered architecture, notably the ISO/OSI layered architecture for network interoperability. The result of such evaluation indicated that this architecture conforms to the established criteria for layered architectures. It is therefore plausible to state that the evaluation mechanism can be used to assist with the design and assessment of layered architectures.

9.3.2 Scientific contribution: Research Question 2

Conducting a comprehensive literature study lead to the conclusion that no universally accepted design and evaluation mechanism for architectures in general, nor layered architectures in particular, has been formulated as yet.

The results of the research conducted in order to answer Research Question 2 are condensed into an evaluation mechanism for layered architectures. The evaluation mechanism consists of criteria that were extracted from, amongst other, an investigation into the use of layered architectures, the descriptions and definitions available in literature pertaining to layered architectures, as well as in-depth investigation of the concept *architecture* and related concepts. In order to demonstrate the efficacy of this criteria list, the ISO/OSI layered architecture obtained from literature is assessed, and it is concluded that the established evaluation mechanism can possibly assist researchers and system architects to evaluate and design architectures in general, and layered architectures in particular.

9.4 RESEARCH CONTRIBUTION: PROPOSED CFL ARCHITECTURE FOR THE SEMANTIC WEB (RESEARCH QUESTION 3)

The third research question is formulated as:

> ***How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and to conform to the criteria identified for layered architectures?***

9.4.1 The research activities devised for Research Question 3

In order to answer this question, the functional status model depicting the functionality of the technologies (addressed by Research Question 1), as well as the criteria list of the evaluation mechanism (derived for Research Question 2) are combined and an adapted CFL Semantic Web architecture, as depicted in Figure 9.4 below, is constructed (also refer to Chapter 7).

The adaptations required for construction of the CFL architecture are dictated by the clarification of the architecture context, the abstraction of functionalities and the development of a security stack. In addition, the functionality of the the language layers, starting with the lowest layer, is defined to be:

- **Unique Identification Mechanism:** The function of this layer within the Semantic Web architecture is to *uniquely identify* resources used. Any discussion about data or meaning on the Web will have to uniquely identify the resources under discussion.
- **Syntax Description Language:** The function of this layer is to provide a *syntax language* or a serialisation mechanism for *data transfer*. At this level, *meaning* is contained within the application or system.

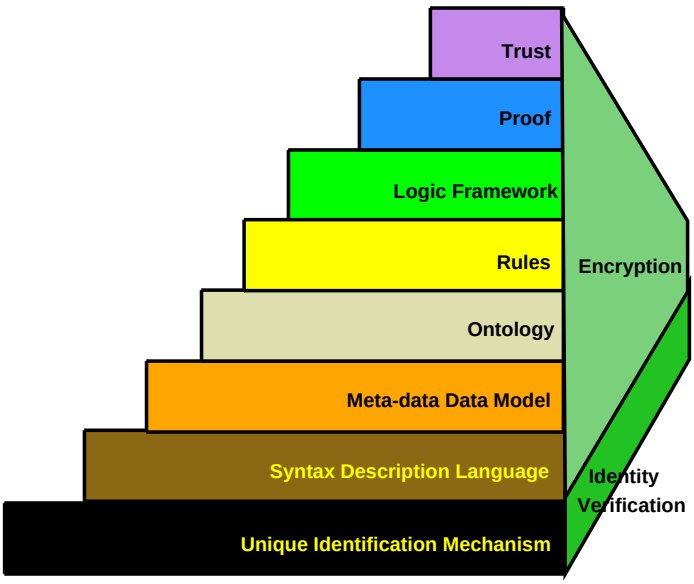


Figure 9.4: The CFL Semantic Web architecture comprises two orthogonal stacks, the *language* stack and the *security* stack.

Syntax descriptions do not describe meaning.

- **Meta-data Data Model:** The *Meta-data Data Model layer* describes the model that is used to model meta-data within the architecture. A data model is required to add descriptions of meaning to data.
- **Ontology:** The function of the Ontology layer is to provide a mechanism to *formally represent knowledge* in such a manner that basic inferencing is supported.
- **Rules:** The function of the *Rules layer* is to provide a mechanism to introduce *rules* (such as business rules) above the knowledge base and its inferencing.
- **Logic Framework:** The purpose of the *Logic Framework layer* is to provide an overarching mechanism for the *integration of different logic formalisms*.
- **Proof:** The *Proof layer* provides *proof* functionality within the context of the Semantic Web. *Proof* means that any result due to inferencing and applied rules can be regarded as valid.
- **Trust:** The *Trust layer* provides *trust* functions within the context of the Semantic Web. *Trust* has to provide a mechanism to ensure that

we can *trust* conclusions.

The functionality of the the security layers is defined to be:

- **Identity Verification Mechanism:** The function of this layer in the security stack is to unambiguously verify an identity on the Semantic Web. It has a functionality grain similar to the *Unique Identification Mechanism* layer in the language stack, and was therefore incorporated as the bottom layer.
- **Encryption:** *Encryption* is an overarching term used to ensure data security, and encryption functionality is required on all layers of the Semantic Web language stack. The *Encryption* layer is therefore depicted as interfacing with the all the language layers excluding the bottom Unique Identification Mechanism layer.

In order to determine the usefulness of the adapted architecture, four usage scenarios for the CFL architecture were investigated in Chapter 8. Two of these usage scenarios represents problem cases obtained from literature, and in these cases, the CFL architecture was applied as a suggestion to solve the problems. The third scenario describes a practical application where a simplified data model was instantiated with two technologies, namely an ER model and an RDF model on the *Meta-Data Data Model layer* (layer 3). Lastly, the most recent (2006) V4 version of the proposed architecture of Berners-Lee was discussed as an instantiation of the adapted architecture. In all of the case studies conducted, the CFL architecture for the Semantic Web contributed towards a solution, and it is therefore plausible to state that the adapted model constitutes a contribution.

9.4.2 Scientific contribution: Research Question 3

To answer Research Question 3, the previously compiled evaluation mechanism and criteria list are used to evaluate the original Semantic Web layered architecture versions proposed by Berners-Lee and these versions were found lacking. To address the identified shortcomings, a first iteration adaption of a functional and comprehensive layered Semantic Web archi-

tecture is proposed as the CFL architecture for the Semantic Web. The architecture is *comprehensive* since it adheres to the fundamental aspects of layered architectures within the information systems domain. The architecture is *functional* because it depicts the required functionality of the language layers that constitute the envisioned Semantic Web.

One of the advantages of the proposed CFL architecture depicted in Figure 9.4 is that this architecture is a simplification of the original architecture versions proposed by Bernes-Lee as a result of the *abstraction* of required functionality. This simplification enables the use of diverse and non-related technologies to implement the same functionality. In other words, different technologies could be used to instantiate the functionality of a specific layer in the proposed Semantic Web layered architecture. An agreed-upon and generalised functional and comprehensive layered architecture for the Semantic Web will aid the acceptance of diverse technologies for implementation of the required functionalities. Such acceptance is important for the eventual realisation of the Semantic Web.

Similarly to the ISO/OSI model description, the proposed CFL architecture aims to present a structure that permits the meta-data languages for the Semantic Web to be viewed as logically composed of a succession of layers, each wrapping the lower layers and isolating them from the higher layers [265]. It is also necessary that, for each layer, interfaces with its upper and lower layers are established in future research. Interfaces are used to standardise access to the functionality provided by a layer.

The present versions of the Semantic Web layered architecture of Berners-Lee were developed to depict technologies adopted by the W3C as standards or W3C Recommendations. This correlates with the mandate of the W3C that includes the development of *specifications* for the Web. However, given the result of this research, it is possible to argue that the definition of a general, accepted layered architecture with functional components might assist the W3C in developing diverse technology specifications for the implementation of the relevant functionalities. This could assist the W3C to include all technology developments rather than to make a choice for the adoption of only one standard to the exclusion of other

technologies. This position is similar to the position adopted by the network community with the acceptance of the ISO/OSI layered architecture as architectural framework for network interoperability standards.

It is acknowledged that not all issues pertaining to the layering of the Semantic Web languages have been resolved by this approach. However, this is an important step towards realising the notion of the Semantic Web. This approach would *include*, rather than *exclude* technologies that would have as benefit a more rapid adoption of the Semantic Web. The definition of W3C specifications needs an appropriate level of abstraction to be able to penetrate sufficiently into the implementation domain, and this is provided by the proposed CFL architecture for the Semantic Web.

Finally, the process followed, the results of the individual research questions, as well as the successful application of the adapted model constitute a contribution in the formulation of a convincing *argument* for the development of a comprehensive and functional layered architecture for the Semantic Web. This argument is based on the following premises:

- ▷ The Semantic Web comprises several diverse technologies spanning a vast application domain.
- ▷ It is possible to argue that the present versions of the layered architecture, proposed by Berners-Lee [28, 31, 34, 35], upon scrutiny, depict inconsistencies and irregularities, and no description of their meaning has been formally published.
- ▷ At present, there is no generally accepted comprehensive and functional architecture for the Semantic Web in literature.
- ▷ Furthermore, no formal W3C initiative has so far been established to address the issue of a conceptual architecture for the languages required to describe the meta-data of the Semantic Web specifically .

There is a requirement for a comprehensive and functional layered architecture for the Semantic Web due to the following reasons:

- ▷ In order to adopt and apply the Semantic Web, an understanding of the technologies, their function and status is required. This requirement is addressed with the first part of Research Question 1.
- ▷ In order to discuss and develop the Semantic Web with its associ-

ated technologies, a *functional* framework or architecture is required (as was done with the ISO/OSI layered architecture for network interoperability). This requirement is addressed with the second part of Research Question 1.

- ▷ In order to adapt the Semantic Web or the current proposed technologies, a framework and evaluation mechanism are required to determine whether the adaptation improves the status quo. This requirement is addressed with Research Question 2.
- ▷ An architecture reflects design decisions, and design decisions need to be made based on design principles or corollaries in order to be *comprehensive*. This requirement is addressed by considering the criteria established for Research Question 2.

From the premise and the established requirements, it is possible to argue convincingly that the development of a comprehensive and functional layered architecture for the Semantic Web indeed constitutes a contribution within the Semantic Web application domain, thus presenting research Question 3.

9.5 ADDITIONAL RESEARCH CONTRIBUTION: SOFTWARE ENGINEERING DISCIPLINE

Apart from the research contributions discussed in Section 9.2 to Section 9.4, it is also possible to reflect on the contribution of the study to the Software Engineering discipline in general. In addition, the research process of this study is regarded as a contribution towards qualitative research processes within the Software Engineering discipline.

With regard to the contribution within the Software Engineering discipline, the first contribution is a discussion on the comprehensive and functional layered architecture as a *model* (Section 9.5.1). Secondly, a *method* is proposed for architectural developers to follow during architectural development (Section 9.5.2). Lastly, comments are made in Section 9.5.3 on the research approach followed within this study as an innovative approach for qualitative research within the Software Engineering discipline, especially

for other studies similar to this one.

9.5.1 The comprehensive and functional architecture as model

Models are abstractions of reality and are indispensable for the effective use of any kind of knowledge within the Software Engineering discipline [181]. Objects within Software Engineering have to be considered from different viewpoints by different audiences, and for this powerful and consistent modelling techniques are necessary. In this respect architectures are models used to model the invisible aspects of software systems [10, 181, 259].

In order to comment on the proposed CFL architecture as model, the characteristics of successful models according to Wentzel [259] are considered. He proposes that a successful model is:

- useful - it must solve a problem,
- intuitive - the target user must be able to understand and interpret the model; and
- predictive - a user should be able to use the model to plan his/her future activities.

In the case of the CFL architecture for the Semantic Web, it is possible to argue that the architecture is:

- useful - it contributed towards solving the problems of the usage scenarios of Chapter 8, namely problems related to the layering of Semantic Web technologies, as well as the accommodation of diverse but functionally similar technologies,
- intuitive - the target users should be able to understand and interpret the model because the model depicts functionality rather than technology. In addition, the functionality of each layer is clearly defined. The meaning of the layering is also described, and the architecture conforms to the criteria of the evaluation mechanism; and
- predictive - a user should be able to use the model to plan future activities. The model allows for the instantiation of the layers making use of different technologies. In addition, any developer that wants to

adopt the Semantic Web architecture will be able to incorporate the CFL architecture into system design activities, and in such a case the architecture would assist with the inclusion of fundamental layered architectural principles into the design.

Since the proposed CFL architecture for the Semantic Web adheres to all three of Wentzel's prerequisites for a successful model, it is the contention of the author that the proposed CFL architecture may be regarded as a successful model.

9.5.2 A method for layered architecture development

Architectural developers should consider a number of aspects during the development of layered architectural structures. Omitting to do so may result in a layered architectural structure that represents poor design decisions and a system that is not sustainable. However, architectural developers are often so preoccupied with the analysis of the application domain where the detail of the different components of the layered architecture is established, that the development of the layered architecture adhering to established design decisions and acting as a comprehensive communication tool, is neglected. Furthermore, there is no set of guidelines on the development of layered architectures.

In addition to the contribution made for layered architectures in the Semantic Web application domain, the method followed to establish the comprehensive and functional layered architecture is suggested as a method for layered architectural development. The steps suggested for establishment of layered architectures, graphically depicted in Figure 9.5, include:

- Step 1: Consider technological implications,
- Step 2: Establish an evaluation mechanism,
- Step 3: Construct a layered architecture; and
- Step 4: Assess the layered architecture.

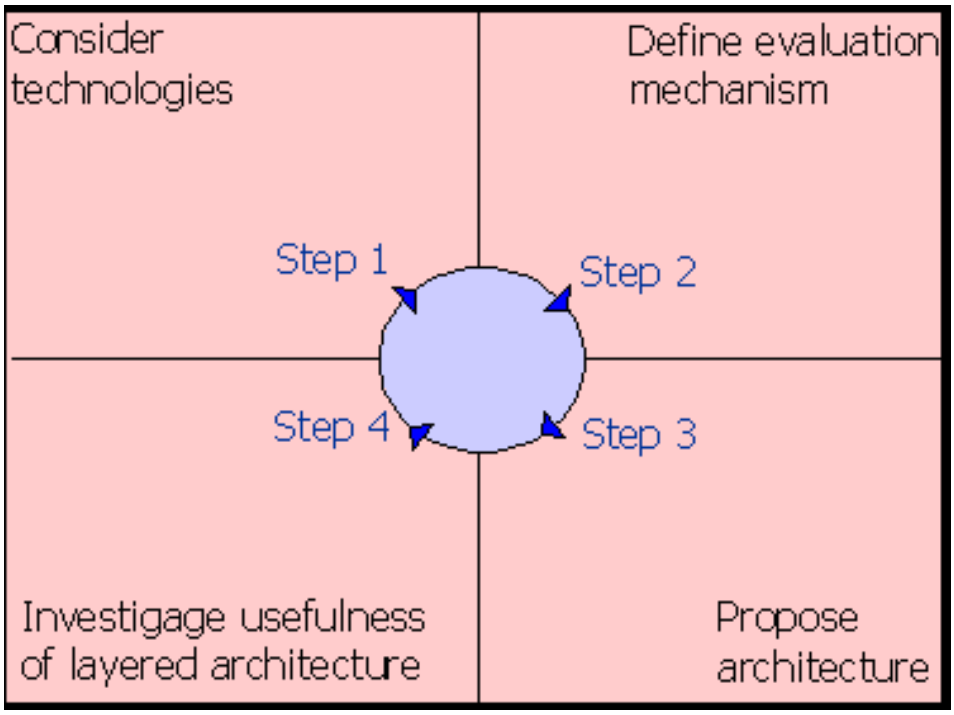


Figure 9.5: Architectural development method

9.5.2.1 Step 1: Consider technological implications

In this study, Step 1 comprised an investigation of the technologies depicted by the current versions of the layered architecture as proposed by Berners-Lee [28, 31, 34, 35]. From this investigation the status and function of the reflected technologies are extracted (Chapter 5 and Appendix A).

In this first step, the architectural developer therefore investigates existing technologies and develops a status model reflecting the current status quo. Considerations include the limitations of current technology specifications, as well as technology interactions and the impact of layering. Risks involved in Step 1 execution include that any status quo may change rapidly and a status model could therefore rapidly become outdated.

9.5.2.2 Step 2: Establish an evaluation mechanism

In the second step, the architectural developer should consider the evaluation mechanism to be used as guideline during development of the architectural structure. In this study Step 2 comprised the development of an evaluation mechanism for layered architectures based on an extensive literature study because it was determined that there is no generally accepted evaluation mechanism for layered architectures in current literature. This study included the establishment of applicable design corollaries and their integration into the evaluation mechanism criteria. In addition, this evaluation mechanism was calibrated against the ISO/OSI architecture.

Considerations during this step are:

- A proof that the evaluation mechanism are useful (e.g. proof to show that the mechanism can be used for different architecture evaluations).
- Using the evaluation mechanism as starting point to indicate that the existing structures are not adequate.
- If an evaluation mechanism is constructed, it should be calibrated before it is used for evaluation.
- The criteria of the evaluation mechanism should integrate applicable Software Engineering design principles.

Developers should guard against an evaluation mechanism that is not applicable or is an invalid measurement instrument for the object being evaluated. The calibration of any evaluation mechanism against an object that may be regarded as a standardised object will ensure that the evaluation mechanism is valid.

9.5.2.3 Step 3: Construct a layered architecture

The next step is to construct an architecture, which should be based on existing architectures (if any) and furthermore to consider the characteristics of architectures. The evaluation mechanism established (or identified) during Step 2 could be used as guideline during the definition of the architecture.

Within this study, both the functionality extracted in Step 1 and the criteria of the evaluation mechanism of Step 2 were used as building blocks to compile a first iteration comprehensive and functional layered (CFL) architecture for the Semantic Web. By adhering to the previously established evaluation criteria, it can be argued that the adapted architecture complies with being *functional* and *comprehensive*.

9.5.2.4 Step 4: Assess the layered architecture

In this study, Step 4 comprised an assessment of the proposed CFL architecture by evaluating it using the evaluation mechanism as well as applying it to usage scenarios. However, it is recognised that the proposed CFL architecture of this study is only a first version, and in order to be agreed-upon and fully functional for general adoption, more role players need to participate in the construction of such an architecture.

To generalise this assessment action for the purpose of an architectural development method, Step 4 should include the assessment of the constructed architecture from at least two perspectives. The layered architecture should firstly be evaluated from an architectural perspective using the evaluation mechanism, and secondly from an application perspective by applying this architecture to usage scenarios.

The evaluation from an architectural perspective using the evaluation criteria, establishes how the developed architecture adheres to the defined characteristics, and shortcomings are subsequently identified. Where possible, the architecture is adapted to address these shortcomings and to reflect stricter adherence to the established criteria.

For the assessment from an application perspective, the architecture should however be applied to a number of usage scenarios to reflect on the feasibility of the model within different application domains. Should the architecture be found not to be useful in practice, the architectural developer need to revisit the model. In addition, the chosen usage scenarios should be as diverse as possible in order to allow for a comprehensive assessment of the usefulness of the architecture.

9.5.3 The qualitative research process

A further contribution identified for this research comprises the qualitative research approach followed during this study. Traditionally, research within the Software Engineering discipline inherited the more rigorous approach followed by Computer Scientists where the focus was primarily on the proof of techniques in practice. Within Software Engineering, where the focus is on model construction, tools, methods, methodologies etc, the researcher is faced with complex problems, which often require a variety of research methods. For example, in the last decade within Software Development the focus shifted to more user-centred application development, which caused the introduction of new concepts within new disciplines, such as Human Computer Interaction (HCI). Similarly, with new technologies developed daily, the Software Engineering discipline need to investigate research methods (or a combination of research methods) that will enable the researcher within this discipline to establish applicable standards, tools and techniques.

In the research conducted in this study, the focus is on the development of a layered architecture for the Semantic Web application domain. No single research method was found to be adequate and it was therefore necessary to investigate a combination of two different qualitative approaches, namely a model-building study combined with a methodological study (Figure 9.6).

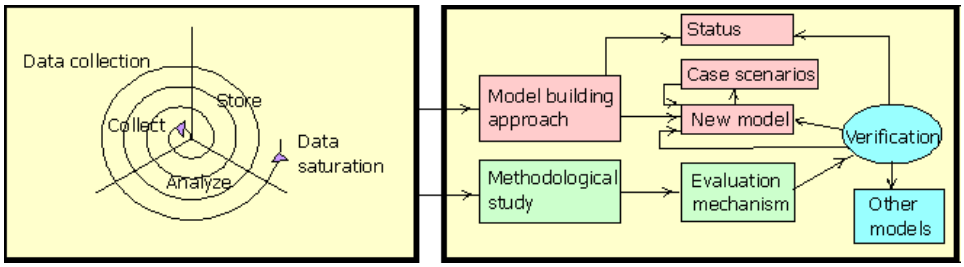


Figure 9.6: Research approach used

In this study, the initial research activity was the collection of data. The *data set saturation* technique [194] was utilised to ensure completeness

of the qualitative data set. In addition, two specific qualitative research techniques were identified and used to execute three qualitative studies. A *model-building study* was applied to build the functional status model of the Semantic Web, as well as the construction of the proposed CFL Semantic Web architecture. Furthermore, a *methodological study* was applied to develop an evaluation mechanism for layered architectures.

For studies similar to this study where the focus is on the development of models that may act as guidelines or standards, such as a layered architecture, the following steps are suggested (apart from the first step, the data collection step, the following steps are not necessarily in a specific order of sequence):

- ▷ Initiate the study with a data collection activity until data saturation is achieved (this step consists of a data collection, analysis, and storage of relevant literature phases).
- ▷ Follow a model-building study to build a status model reflecting the current status of technologies within the problem domain and propose the new/adapted model or layered architecture.
- ▷ Conduct a methodological study where an evaluation mechanism is identified for verification purposes.
- ▷ Verify the evaluation mechanism against existing accepted models to calibrate the mechanism and ensure that it can be trusted.
- ▷ Verify the status model in order to ensure that it adheres to the evaluation criteria and if not, identify the shortcomings.
- ▷ Evaluate the suggested model against the evaluation criteria. Repeat the design of the model to eliminate any criteria to which the structure does not adhere (if possible).
- ▷ Test the new model against usage scenarios to investigate the usefulness thereof in the relevant application domain. Identify any shortcomings/weaknesses and redesign the proposed model if necessary.

Prescribed verification mechanisms are generally lacking within qualitative research approaches and this is problematic. In this study, verification was incorporated throughout the process. An evaluation mechanism for layered architectures was constructed as a building block in the construc-

tion of the CFL architecture. However, even in the construction of the evaluation mechanism, verification was ensured by calibrating the mechanism against an existing, accepted layered architecture (the ISO/OSI architecture). The established evaluation mechanism was used to construct, as well as validate, the CFL architecture. As additional verification mechanism and to counteract the problem of models being too abstract, the CFL architecture was applied to usage scenarios to ensure that it is useful.

The described qualitative approach used that includes data set completion, the execution of model-building and methodological studies, as well as the establishment of qualitative verification mechanisms, could be regarded as a further research contribution within qualitative research techniques in the Software Engineering discipline.

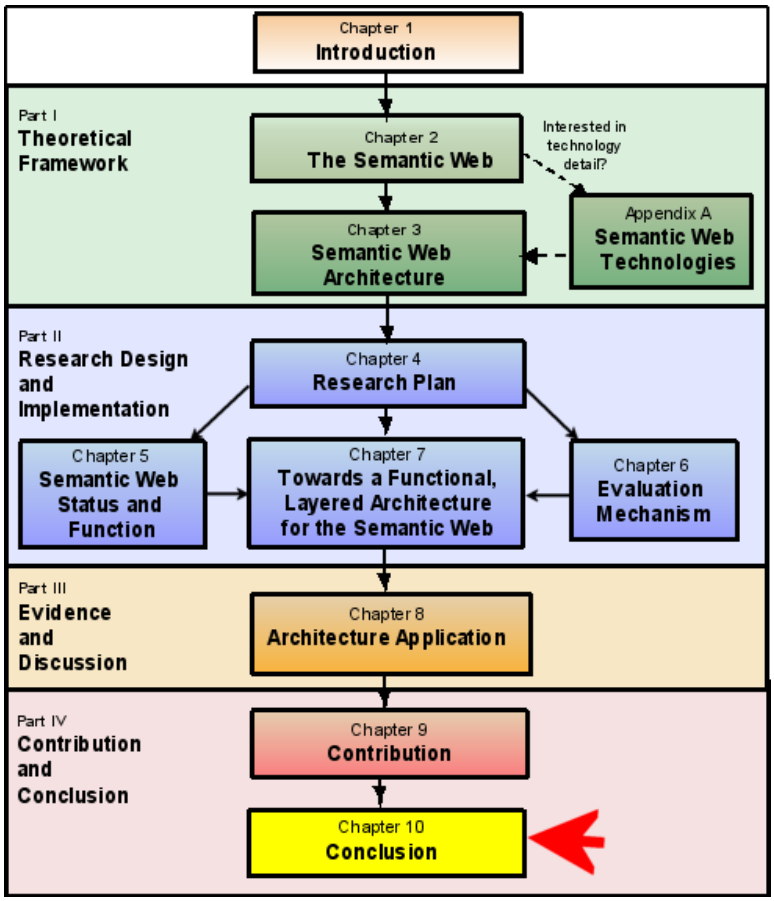
9.6 CONCLUSION

In this chapter, the identified contributions of this study are discussed by referring to the research questions that were formulated. In answer to Research Question 1, the study contributed a functional Semantic Web status model. The research that supports Research Question 2 constitutes a contribution in the form of an evaluation mechanism for layered architectures. The third contribution in answer to Research Question 3 is a proposed comprehensive and functional layered (CFL) architecture for the Semantic Web.

Three additional contributions of the research contained in this study, were identified as contributions within the Software Engineering discipline as well as in qualitative research processes. Within Software Engineering, the first contribution assesses the comprehensive and functional layered architecture against characteristics of a successful model. The secondly contribution is a method proposed for architectural developers to follow during the construction of layered architectures. Lastly, the research approach followed in this study is discussed as an innovative approach for qualitative research within the Software Engineering discipline for studies similar in nature to this study.

CHAPTER 10

CONCLUSION



Thesis Chapter Layout

Chapter Contents

Tables

10.1 INTRODUCTION

In this chapter, this study is concluded through a summary, a reflection and several recommendations. The research findings pertaining to this study are summarised in Section 10.2. In Section 10.3, reflections on the research from a methodological and substantive perspective are presented. In Section 10.4, recommendations based on this study are presented, including recommendations for policy and practice, as well as for further research and developmental work. This is followed by a short closure in Section 10.5.

10.2 SUMMARY

The remarkable adoption of the Web into society resulted in several usage problems such as information overload, discrepancy and trustworthiness. The Semantic Web is regarded as the inevitable successor of the current Web and according to its founder [38], it will, amongst other things, solve these problems according to its founder . In this vision, the Semantic Web is portrayed as an information space usable by sophisticated software agents that act on behalf of their users to solve their problems.

In this study a definition for the Semantic Web was compiled following a presentation of Semantic Web discussions as found in literature (Section 2.3.1, p.33). This definition describes the Semantic Web as:

1. a Web enriched with semantic meta-data that will enable agents to execute complex information management tasks on behalf of their users,
2. a mechanism that contributes towards data, information and knowledge exchange and integration across communities and applications; and
3. a comprehensive architecture of meta-data language functionality that can be instantiated with different technology standards and specifications.

In this definition, the Semantic Web is described, amongst other things,

as a comprehensive *architecture* of meta-data language functionality. An architecture depicts the structure and components that comprise a system within a specific context [18, 105]. Often architectures depict the organisation of components according to several identified patterns, one of which is the *layered architecture* [18]. An architecture also depicts design decisions and is indispensable for the successful implementation of a system. Design decisions should be guided by Software Engineering design principles and corollaries as presented in Section 1.1 [14, 91].

Since the Semantic Web vision was envisaged, Berners-Lee presented several versions of a Semantic Web architecture where existing W3C technologies and functionalities are layered into an increasingly expressive stack [28, 31, 34, 35]. In the context of the *languages of the Semantic Web*, the existing versions of the architecture conforms to architecture definitions because its intention is a depiction of the language components necessary to describe meta-data for the Semantic Web. However, it is unclear what exactly is intended by these representations since a description of its precise meaning is lacking in literature. Upon scrutiny, the proposed architecture versions contain several inconsistencies and discrepancies. This lack of intended meaning results in obstacles for adopters and developers of the Semantic Web and its associated technologies. In addition, the Semantic Web is diverse and its associated technologies span various application domains. The lack of a functional architecture causes problems with regard to the assimilation of the context, application domain, and functionality of the Semantic Web and its associated technologies for prospective users. In addition, the lack of an agreed-upon and accepted architecture is also problematic for the specification, acceptance and adoption of W3C Recommendations since a useful framework is lacking.

A comprehensive and functional layered (CFL) architecture for the Semantic Web is thus required, and this study describes the research conducted towards the development of such an architecture. The purpose of this study is thus stated to be the development of a comprehensive and functional layered architecture for the Semantic Web, where:

- An *architecture* is a model that depicts the structure of components

of which a system within a specific context comprises.

- ▷ A *layered architecture* is an architecture that organises the system components or groups of components into successive layers logically similar and of equal rank. Any layer uses functionality presented by its lower layers and isolates these lower layers from layers above.
- ▷ A *functional* architecture is an architecture that depicts components identified by their function within the system.
- ▷ A *comprehensive* architecture is an architecture reflecting design decisions based on established Software Engineering principles.
- ▷ Finally, the *Semantic Web* is regarded as the inevitable successor of the present Web, providing an intelligent information space comprising of a layered architecture of increasing semantically expressive languages for software agents to roam and to perform sophisticated information management tasks on behalf of their users.

The research questions are defined to support the purpose of the study, namely the development of a comprehensive and functional layered (CFL) architecture for the Semantic Web. These research questions are:

	Research Questions
(1)	What is the function of each technology included in the present versions of the Semantic Web layered architecture? ▷ What is the status of the specified technologies within the present versions of the Semantic Web layered architecture?
(2)	To which criteria should a layered architecture conform in order to adhere to system design principles? ▷ Which aspects should be considered when architectures, and in particular layered architectures, are evaluated?

(3)	How can the proposed Semantic Web layered architecture be adapted to be comprehensive and functional, and conform to the criteria identified for layered architectures?

Table 10.1: Research Questions

The theoretical framework underpinning these questions is contained in Part I of the study, and comprises Chapters 2 and 3, as well as Appendix A that presents a discussion of the Semantic Web, the Semantic Web architecture and Semantic Web technologies respectively. A research approach and design to answer the research questions have been included in Chapter 4. In order to construct a comprehensive and functional layered architecture for the Semantic Web, the research activities with their associated results that answer the respective research questions are discussed in the remainder of this section.

10.2.1 Summary: Research Question 1

Research Question 1 aims to determine the status and function of the technologies associated with the Semantic Web according to the two initial architecture versions of the Semantic Web suggested by Berners-Lee [28, 31, 33]. The research activities and results are described in Chapter 5, where an adapted architecture model reflecting the current status quo as well as the functionality of the Semantic Web technologies is constructed. This status model (Figure 5.6, p.127) reflects various refinements imposed considering the status quo of current technology, and imparts some insight into the limitations of the technologies currently supporting the proposed Semantic Web layered architecture versions. In order to construct a comprehensive and functional layered (CFL) architecture for the Semantic Web, the identified functionality of the reflected technologies was used towards the construction of a *functional* layered architecture for the Semantic Web.

10.2.2 Summary: Research Question 2

In order to comment on, adapt or design any architecture, a need exists for the establishment of evaluation criteria for architectures based on established Software Engineering design principles. Research Question 2 aims to establish the criteria that a layered architecture should conform to in order to adhere to system design principles, as well as define the aspects to consider during an evaluation of software architectures and, in particular, layered software architectures. These research activities and results are described in Chapter 6 where an evaluation mechanism for architectures, in particular layered architectures is constructed (Section 6.4.2, p.170). This evaluation mechanism consists of criteria extracted from investigations into various aspects of architectures and layered architectures. To calibrate this evaluation mechanism, the ISO/OSI layered architecture obtained from literature was assessed, and it conformed to the criteria. It is plausible to state that the evaluation mechanism can assist researchers and system architects to evaluate and design architectures in general, and layered architectures in particular. In order to construct a comprehensive and functional layered architecture for the Semantic Web, the identified criteria were used towards the construction of a *comprehensive* layered architecture for the Semantic Web.

10.2.3 Summary: Research Question 3

Research Question 3 focuses on the construction of a comprehensive and functional layered (CFL) architecture. In order to develop such an architecture, the proposed V2 version of the Semantic Web layered architecture [31] is adapted to conform to the established criteria for layered architectures. These research activities and results are described in Chapter 7 where a first iteration of the CFL Semantic Web architecture is developed from both the extracted functionality of Chapter 5 and the evaluation mechanism of Chapter 6. The architecture is presented in Figure 7.3 on page 196. The architecture adheres to the criteria of the evaluation mechanism and is thus both functional and comprehensive. It is acknowledged that

not all issues with regard to the layering of the Semantic Web languages have been resolved by this approach. However, the first version of the architecture is an important step towards realising the notion of the Semantic Web.

10.2.4 Summary: Contribution to the Software Engineering discipline

In Section 9.5, the additional research contributions are discussed to include the argument that the layered architecture developed is a model since it adheres to the characteristics of a successful model as formulated by Wentzel [259], which states that a successful model is useful, intuitive and predictive.

The *method* used for the development of layered architecture models constitutes the second contribution towards the Software Engineering discipline. This method is summarised to include four steps namely (1) Consider technological implications; (2) Establish an evaluation mechanism; (3) Construct a layered architecture; and (4) Assess the layered architecture.

Lastly, the *research approach* formulated to conduct the research is considered as a research contribution. In this approach the data saturation technique was used for data collection, and two qualitative research approaches, namely a *model-building study* and a *methodological study*, were combined to constitute the research approach. This approach is presented as a feasible research design for studies of a similar nature within the Software Engineering discipline.

10.3 SUBSTANTIVE AND SCIENTIFIC REFLECTION

In this section some lessons learnt during the execution of the research for this study are discussed. The goal with substantive reflection is to compare the results of the research reported on in this study with other related research in the same field, while scientific reflection focuses on what the research contributes to the *scientific body of knowledge*.

The Semantic Web is an active research field that has captured the imagination of researchers, developers and users, resulting in a prolific amount of publications. Therefore, any reader who wants to keep abreast of current trends faces a daunting task. However, the status of Semantic Web technologies is prescribed the W3C, which follows a rigourous process for the inclusion of such technologies. Any research within the Semantic Web application domain should take cognisance of the rapid development of this domain. As far as the status and function model of the Semantic Web is concerned, the scientific contribution of the status model was discussed in Chapter 5. However, due to the growth and development of the Semantic Web, the substantive reflection includes the notion that such a model could soon be obsolete.

Within the Software Engineering discipline, publications on the development of architectures are less common. No publications on criteria or evaluation mechanisms for architectures could be found, although Bass et al. [18] provides measures to determine whether a software architecture reflects the user requirements and contribute successfully towards system design and implementation. There are several publications in literature that present or adopt architectures, but few discuss the processes and methods necessary to construct an architecture. In this regard, the evaluation mechanism for layered architectures constructed in this study seems to be distinctive and this constitutes a unique contribution.

No version of a Semantic Web architecture with associated meaning descriptions has been formally published. The versions put forth by Berners-Lee as discussed in Chapter 3 were part of his presentations and generally depicted an organisation of W3C technology recommendations that

form part of the Semantic Web. In addition, the formally specified W3C initiatives do not specifically address the development of a Semantic Web architecture. At this stage the focus of Semantic Web research seems to be on specific technology issues rather than on the design of the Semantic Web. As regards a Semantic Web architecture, the contention is that this research is unique and this study contributes to the scientific body of knowledge in this respect. With regard to the substantive reflection, and to the best of the author's knowledge, no similar research has been published to date apart from Gerber et al. [110].

Publications on topics concerning the construction of models within Software Engineering, as well as studies describing model-building techniques and tactics, particularly using qualitative data analysis techniques, are limited. In this regard the study also contributes to the body of scientific knowledge.

10.4 RECOMMENDATIONS

10.4.1 Recommendations for policy and practice

The adapted comprehensive and functional layered architecture for the Semantic Web provides a framework for the development of data-interoperability standards or languages. By constructing and applying the layered architecture in a similar fashion to the ISO/OSI architecture, diverse technology standards could be accommodated to realise the Semantic Web. It is therefore recommended that the W3C as standards body consider adoption of such an architectural framework.

10.4.2 Recommendations for further research

The initial version of the comprehensive and functional layered architecture for the Semantic Web is by no means conclusive, and key stakeholders and role players have to be engaged to refine the model. Further research could include the establishment of the process necessary to ensure that the

eventual architecture is comprehensive and functional, as well as the establishment of the various Software Engineering processes and techniques required to facilitate participation by all interest parties for the establishment of the eventual CFL architecture.

The development of clear functionality implementations within the CFL Semantic Web architecture will assist with the determination of boundaries as well as interface specifications. Interface specifications to the components contained within a layer thus remains a topic for future research.

The development of the security stack of the proposed CFL Semantic Web architecture is recommended for future research. The functional layers of such an architecture as well as the integration thereof into the *language* architecture are unresolved issues. It might be required to include security functionality within each language layer, or the security functionality might be extracted into a separate layered architecture with definite boundary interfaces with the language layers.

10.4.3 Recommendations for further development work

The development of diverse technology standards with their interfaces as instantiations of specific functional layers is a requirement for the successful realisation of the Semantic Web.

There are several diverse technologies with application possibilities within the Semantic Web, and the refinement of the proposed CFL Semantic Web architecture as well as the technology specifications are clear requirements to be addressed in future development work.

In addition, the incorporation of the proposed CFL Semantic Web architecture into the design and development activities of Semantic Web applications would shed light on additional requirements for these applications as well as impose possible refinements of the CFL Semantic Web architecture.

Successful Semantic Web applications that solve the problems associated with information overload and discrepancy resolution will be the next *major application* on the Web. This application is required by, but not lim-

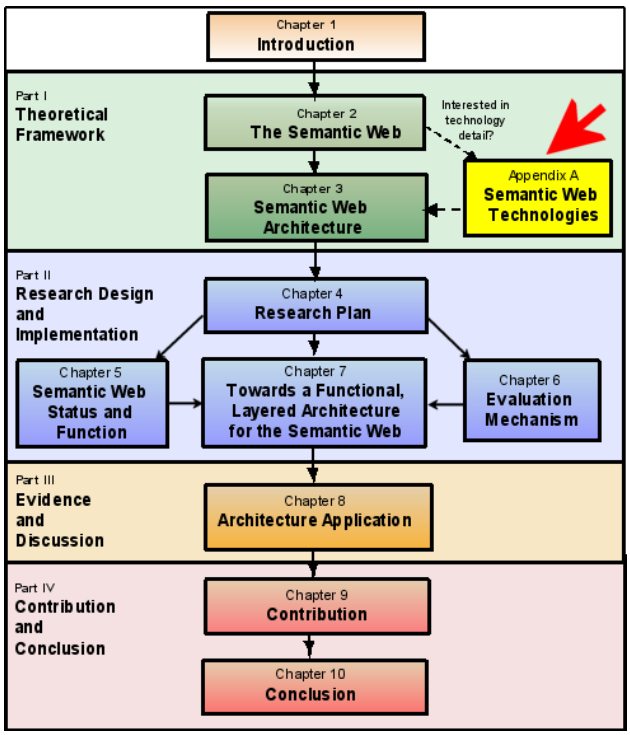
ited to, all users of the Web. Any user who are confronted by vast and diverse information sources, would benefit from such an application. The CFL architecture for the Semantic Web might assist with the resolution of the problems currently experienced by researchers and developers within the Semantic Web application domain.

10.5 CLOSURE

Thus, within this study a comprehensive and functional layered architecture for the Semantic Web is constructed. It is concluded that the construction of such an architecture is indeed required and feasible due to the many advantages thereof, such as communication, inclusiveness and the growth and adoption of the Semantic Web with its associated technologies. To support the construction of a comprehensive and functional layered architecture for the Semantic Web, three contributions are constructed, namely a status and function model for the Semantic Web, an evaluation mechanism for layered architectures, and the first iteration of a comprehensive and functional layered architecture for the Semantic Web. In addition, the study firstly contributes to the Software Engineering discipline by establishing that the constructed architecture is a successful model according to the characteristics determined by Wentzel [259]. Secondly the study contributes by proposing a method for architectural development, and lastly, by formulating a qualitative research approach for use in similar research studies.

APPENDIX A

SEMANTIC WEB TECHNOLOGIES



Thesis Chapter Layout.

In this appendix Semantic Web technologies are discussed.

Semantic Web Technologies.

by
Aurona Gerber
Prof. Andries Barnard
Prof. Alta J. van der Merwe

Technical Report

UNISA-TR-2006-02
http://www.osprey.ac.za/TechnicalReports/

UNIVERSITY OF SOUTH AFRICA

November 2006

Chapter Contents

Figures

Tables

A.1 INTRODUCTION

The purpose of this report is to present an overview of the technologies envisioned as part of the Semantic Web.

The questions of concern in this report are:

- What are the different technologies depicted on the layers of the Semantic Web architecture?
- What are the interrelationships between the technologies depicted in the Semantic Web architecture?

The Semantic Web was introduced in 2001 by Tim Berners-Lee [38] in his vision of a new intelligent Web. Complementing this vision, he proposed four versions of the Semantic Web architecture [28, 31, 34, 35], for the purpose of this report labelled V1, V2, V3 and V4 respectively, as indicated in Figure A.1.

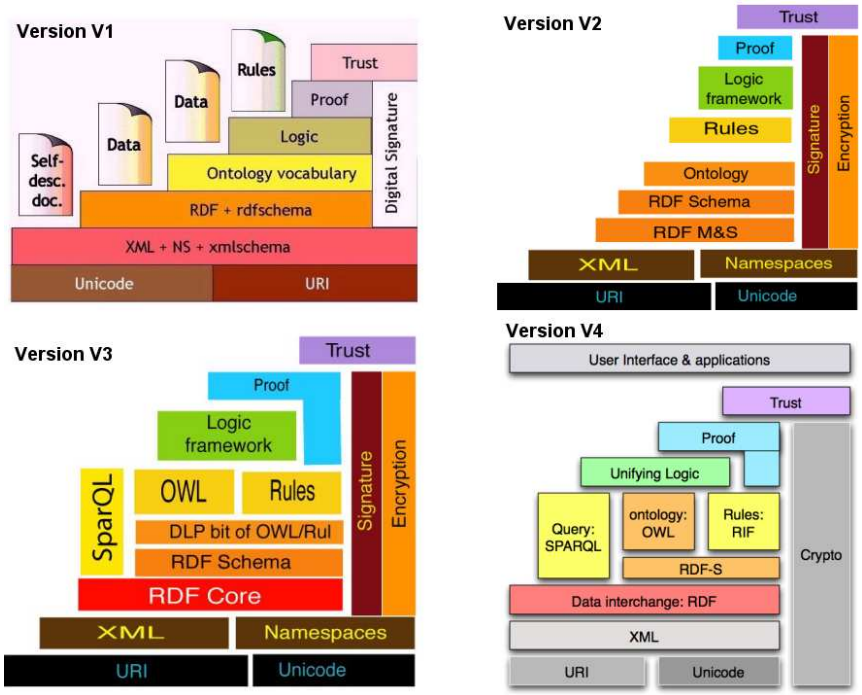


Figure A.1: The four versions of the Semantic Web architecture [28, 31, 34, 35]

Versions V1 and V2 have been adopted more than once in literature by authors who regard this model as the Semantic Web architecture of choice [6, 101, 124, 130, 147, 190, 191, 223]. In contrast, there was no significant adoption of versions V3 and V4 of the architecture. In this report, the technologies of the Semantic Web are discussed with reference to the *adopted* versions (V1 and V2) of the layered architecture [28, 31]. For each technology the appropriate terms together with its history and relation to other important concepts are discussed.

The technology investigation in relation to the underlying architecture is discussed in Section A.2. Sections A.2 to A.10, discuss the technologies according to their position in the different layers in the architecture. In order to be comprehensive, the additional technologies depicted in versions V3 and V4 of the layered architecture, are summarised in Section A.11. The report is concluded in Section A.12.

A.2 SEMANTIC WEB TECHNOLOGIES

The Semantic Web is an information space used by *machines* rather than *humans*. Instead of processing and manipulating Web information, a user would have a personal *agent* on his/her computer that would solve problems related to information overload, acquisition and discrepancy resolution [84]. Once an agent has executed the first level of information management, a user would access or manipulate the results. In order to execute these tasks, the information the agents uses has to be presented in an increasing semantically enriched format by means of several technology layers. These technology layers are depicted in the different versions of the Semantic Web layered architecture.

Versions V1 and V2 of the architecture are presented in Figures A.2 and A.3 with added *Layer* captions for reference purposes. In both these versions of the Semantic Web architecture, a higher level layer language use the syntax and semantics of its immediate lower level layer.

In Sections A.3 through A.9 the different layers of V1 and V2, as presented in Figures A.2 and A.3, are considered. The discussion of each

layer comprises a description of the residing technologies. Sections A.10.1 and A.10.2 discusses the vertical layers, digital signatures and encryption, which serves as identification authentication as well as security mechanisms for layers three to six.

A.3 LAYER 1: UNICODE AND URI

Layer 1 in both models comprises *Unicode* and *URI (Uniform Resource Identifier)* technologies.

A.3.1 Unicode

Unicode aims to uniquely identify the characters in all the written languages by assigning a unique number to each character. In order to uniquely identify each character, Unicode specifies the universal character encoding standard used for representation of text for computer processing. In general, character encoding standards define not only the identity of each character and its numeric value (code point), but also the representation of the value in bits. Unicode extends ASCII by assigning a unique numeric value and name for each character used in all the written languages of the world [72].

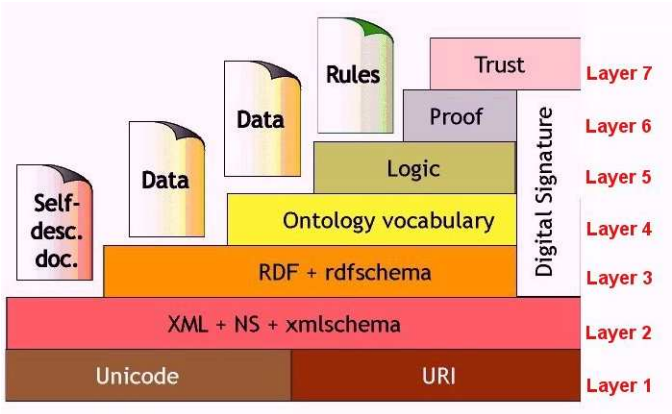


Figure A.2: The V1 Semantic Web architecture ([28])

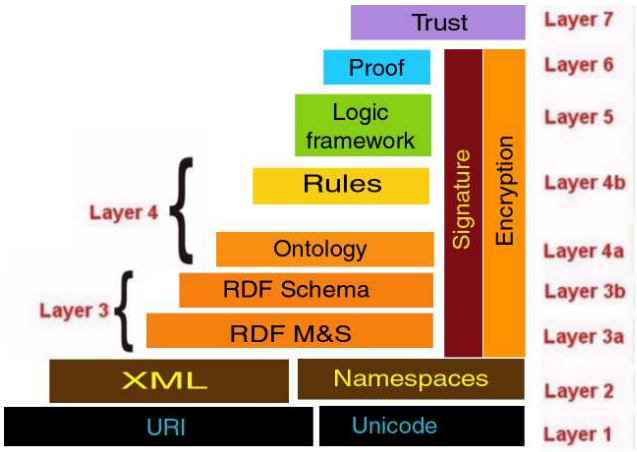


Figure A.3: The adapted Semantic Web architecture (V2) ([31])

The Unicode Standard is specified by the Unicode Consortium [71]. The Unicode Consortium is a non-profit Organisation founded to develop, extend and promote use of the Unicode Standard [72]. Its membership represents a broad spectrum of corporations and organisations in the computer and information processing industry. The standard supports three encoding mechanisms, UTF-8, UTF-16 and UTF-32, allowing the same data to be encoded in a byte, word or double word format (i.e. in 8, 16 or 32-bits per code unit). All three encoding mechanisms encode the same common characters and can be transformed into one another. Any of these encoding methods is endorsed as a way to implement the Unicode Standard [72].

UTF-8 is generally used for HTML or similar protocols. UTF-8 transforms all Unicode characters into a variable length encoding of bytes. It has the advantages that the Unicode characters corresponding to the familiar ASCII set have the same byte values as ASCII, and that Unicode characters transformed into UTF-8 can be used with existing software without rewrites. UTF-16 is utilised in environments that need to balance efficient access to characters with economical use of storage. It is more compact than UTF-8 and the characters that are used in general fit into a single 16-bit code unit, whilst all other characters are accessible via pairs of 16-bit code units. UTF-32 is popular where memory space is no concern and

where fixed width, single code unit access to characters is desired. When using UTF-32 each Unicode character is encoded as a single 32-bit code unit [72].

The three encoding forms of Unicode use a common collection of characters but also allows for the encoding of more characters as required. The standard makes provision for all known character encoding requirements, including full coverage of the historic scripts of the world, as well as punctuation marks, diacritics[1], mathematical symbols, technical symbols, arrows, and even characters such as dingbats [72].

The Unicode Standard has been adopted by various industry leaders and is required by standards such as XML, Java, JavaScript, LDAP and CORBA 3.0. In general it is supported in modern operating systems and browsers.

A.3.2 URI

A URI (Uniform Resource Identifier) is defined as an extendable, compact string of characters that is used for the identification of a resource. Furthermore, a URI is used to identify either an abstract or a physical resource. The standard URI specification of the IETF (Internet Engineering Task Force) is RFC3986 [37]. The IETF is an open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the operation of the Internet. The technical work of the IETF is done in its working groups, which are organised by topic into several areas (e.g., routing, transport, security, etc.) [137, 138].

According to RFC3986, a *resource* is defined as anything that has identity. A resource is the *conceptual mapping* to an entity or set of entities and not necessarily the entity itself, which corresponds to that mapping at any particular instance in time. A resource, therefore, remains constant even when its content changes over time, provided that the conceptual mapping

[1] Diacritics are modifying character marks such as the tilde ˜ , that are used in conjunction with base characters to represent accented letters such as ñ.

is not changed in the process. This is an area of debate called *the Semantic Web Identification Problem* and the related issues have not been resolved yet [218].

An *identifier* is an object that can act as a reference to something that has identity. In the case of URI, the object is a sequence of characters, ideally Unicode, with a restricted syntax.

In addition, the URI specification proposes *uniformity*, which enables the use of different types of resource identifiers in the same context, as well as the reuse of a defined URI in many different contexts. This implies that new applications or protocols can refer to an existing specified set of resource identifiers that are in use. The URI specification aims to assist in the *uniform* semantic interpretation of common syntactic conventions across different types of resource identifiers.

URLs (Uniform Resource Locators) are a sub-set of URI that specifically identify resources by using their network *location* rather than identifying the resource by name or by other attributes. More specifically, a URL is a compact string representation of the location for a resource that is available via the Internet [39]. It generally takes the form: `http://www.w3c.org/Addressing/Activity`

URNs (Uniform Resource Names) refers to the sub-set of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable [76, 171]. A URN takes the form: `urn:NID:NSS` where `NID` is the namespace identifier and `NSS` is the namespace-specific string. An example, obtained from Carey [61], uniquely identify a book with an ISBN as follow: `urn:isbn:0-619-01969-7`

At present non-ASCII characters are not allowed in URIs [37]. To lift this restriction, however, IRIs (Internationalised Resource Identifiers) are being developed in the W3C Internationalisation Activity [140]. In contrast to an URI, an IRI is a sequence of *Unicode characters*[2]. The present endeavours towards IRIs suggest the use of UTF-8 as the preferred character encoding for URIs, and it also supports IRI-to-URI conversion [245].

[2]Refer to Section A.3.1 on page 288 for Unicode

Any Web development is inextricably involved with URIs because of the necessity of global identification. In particular, data on the Semantic Web is ideally described using IRIs rather than URIs. This is supported at present by W3C activities. URIs ensure that concepts are not just text but are tied to a unique definition that can be found by any Web user. With IRIs, concepts are universally accessible across language boundaries.

A.4 LAYER 2: NAMESPACES, XML AND XML SCHEMA

Layer 2 comprises of *Namespaces*, *XML (Extensible Markup Language)* as well as *XML Schema technologies* (V1 in Figure A.2). In V2 as depicted in Figure A.3, *XML Schema* was omitted, however, for this the purpose of this discussion it will be included as a Layer two technology.

A.4.1 Namespaces

Namespaces (NS) provides a simple method for qualifying element and attribute names used in XML documents. Namespaces are identified by URI references. The W3C Namespace Recommendation defines an XML namespace as a collection of names, identified by a URI reference [37, 47], which are used in XML documents as element types and attribute names. The Second Edition of *Namespaces in XML 1.0* [48] was released as a W3C Recommendation on 16 August 2006 and supersedes the first Recommendation of 14 January 1999 [46].

This section discusses the concept of Namespaces as a qualifier for a domain specifying a grammar or vocabulary on the Semantic Web in XML using XML Schema. This discussion must therefore be read in conjunction with the sections on XML (Section A.4.2 and XML Schema (Section A.4.3)). However, Namespaces is presented before these sections as it is an underlying concept to any grammar specification such as XML. A namespace represents a collection of element types and data-type names and is identified by a unique name [49, 216, 261]. Namespaces are used to

manage naming conflicts that invariably arise when different authors create grammars or vocabularies for the Semantic Web.

At present the W3C Recommendation recommends the use of a URL to identify a namespace because URLs indicate domain names that are unique and are used throughout the Internet. For the purpose of a namespace declaration, this specific URL does not mean a *Web address*, but *unique identifier*. XML Namespaces differ from the *namespaces* conventionally used in computing disciplines in that the XML version has internal structure and is not, mathematically speaking, a set [49].

An example illustrates the use of namespaces. Namespaces are usually declared as an attribute of the root element in an XML Schema in the following manner:

```
<aElement xmlns:abc="http://www.abc.com" />.
```

In the declaration of the attribute `xmlns:abc`, xmlns is a reserved word used only to declare a namespace or to bind namespaces, and it is not itself bound to any namespace. In addition the *prefix* `abc` is bound to the namespace `http://www.abc.com`.

It is convention to use the phrase `XSD` or `XS` as a prefix for the *XML Schema Namespace*. When defining dedicated application namespaces, the use of meaningful namespace prefixes is recommended since it assists with the clarity of XML documents. Prefixes are used as placeholders and are expanded by the namespace-aware XML parser to use the actual namespace bound to the prefix. In the next example the elements `Title` and `Author` are associated with the namespace `http://www.literature.com`:

```
<?xml version="1.0"?>
  <Book xmlns:lib="http://www.literature.com">
    <lib:Title>Emma</lib:Title>
    <lib:Author>Jane Austen</lib:Author>
</Book>
```

It is possible to declare a *default* namespace, meaning that any element within the scope of the default namespace declaration will be qualified im-

plicitly if it is not already qualified explicitly using a prefix. As with prefixed namespaces, a default namespace can be overridden.

The *namespaces* concept is defined to manage naming conflicts in the information space of the Web where different vocabularies containing the same name might co-exist. A namespace defines an information space wherein all the declared names are unique. Thus, when vocabularies are combined, names are unique when referenced in association with their namespaces.

A.4.2 XML

XML (Extensible Markup Language) specifies a standard for the exchange of data over networks, notably the Web. XML is considered to be both a *meta-language* and a *markup language* [61, 84, 159, 257]. The function of a markup language is to describe information, usually for storage, transmission or processing by an application. The function of a meta-language is to describe another language formally. XML as meta-language allows for the specification of the content of documents according to a predefined and specific structure. All documents conforming to this specification will have the same structure or represent data items in the specified structure [233, 257]. In addition, XML as markup language allows for the insertion of markup tags into text to define the logical structure of a document, or to add information regarding information contained in a document (meta-data) [233, 234].

An example adapted from McKinnon and McKinnon [159] could assist with clarification of the dual role of XML. XML as *meta-language* may be used to specify the document structure of documents that is used to store the contact information of customers in a specific application. For instance, it might specify the first item as a surname, followed only by initials and relevant telephone number. In addition, all the documents that contain this contact information will use *XML tags* to indicate that the first field in the document is the surname and initials and in this way, XML is used as a *markup language*.

An XML document is a *text* document, which in itself does not have any functionality. It is used only to describe data, information or meta-data [61, 257]. Thus, XML is a means for defining common grammars to enable data exchange. XML does not specify semantics. All parties participating in the data exchange *must agree* on the data model and document structure for XML data exchange to be successful. If an XML grammar is accepted as a standard for data exchange, any XML parser can parse the XML data and access the content if it is a valid XML document. It is however difficult to re-engineer the data model from any given XML document if the document type specification is not available [84, 189].

A.4.2.1 The history of XML

XML was developed from SGML or GML, originally developed by IBM, who foresaw the need for a way to separate data and its display information [61, p.1.04]. IBM released GML or *Generalised Markup Language* in 1973. With GML there was a first attempt to separate the specific formatting instructions of a document from the content of the document. GML's generic encoding approach made a document transportable, meaning that it could be displayed or rendered in different styles without any changes to the original document. In 1978, ANSI (the American National Standards Institute) initiated the development of a standard based on GML [159, 208]. The result was SGML, the *Standardised General Markup Language*, approved in 1986 [159, 208, 210].

SGML is an extensive, versatile standard using generic descriptive markup so that the content of a document is defined completely separate from its processing. SGML also formalises the concept of a *document type* associated with a document in another file called the DTD (Document Type Definition). DTD is discussed in Section A.4.3.1 on page 301. A DTD identifies all the elements and their structural relationships to be contained in a document of that type and a document of a specific type can be verified against the document type definition to ensure that it conforms to its type declaration [208, 210].

In spite of its advantages, the versatility of SGML made it too cumber-

some and too resource-intensive to be adapted or fully incorporated into applications.

A.4.2.2 HTML and XML

HTML (Hypertext Markup Language) was developed by CERN (European Organisation for Nuclear Research) mainly because SGML was considered too cumbersome to develop documents for the Web. HTML is an application profile of SGML making HTML a SGML document type [208]. A fully compliant SGML system would be able to process HTML documents. Thus, HTML documents are SGML documents with predefined markup tags that are appropriate for the representation of information on the Web [159, 209].

By the 1990's the Internet was widely adopted as information exchange medium and the simplified specification of HTML could not meet the functionality demands of Web developers any more. The W3C (World Wide Web Consortium) initiated an activity to simplify SGML for Web application development in 1996, which resulted in XML [208]. Like HTML, XML is an application profile of SGML and any SGML system that fully conforms will be able to process XML documents. However, XML does not require a system that is capable of understanding full SGML. XML is not expected to replace HTML, rather it is designed to deliver structured content over the Web [61, 199].

XML was readily adopted by Internet users, and the first specification, the Extensible Markup Language (XML) 1.0 (First Edition) specification, was accepted by the W3C in 1998 whilst the third edition was accepted as a W3C recommendation in February 2004 [49].

XML was never intended as replacement of HTML, XML was designed to describe data that has to be exchanged or transported over networks, whilst HTML was designed to *display* Web content and make it acceptable for general users of the Web.

A.4.2.3 XML document structure

An XML document is a *text* document in that it does not have any functionality in itself. It is used only to describe data, information or meta-data according to the XML specification. In order to use XML, an *XML processor* or *XML parser*, or *application* is required. An *XML processor* or *XML parser* reads an XML document, performs validity checks and provides access to the content and structure of the XML document on behalf of users or software applications. *Applications* are software programs that use XML documents. Note that there is a distinction between *application* and *XML application* in that an XML application is an application of XML, or an XML language that has been developed according to the XML specification [61, 84, 159].

The W3C XML Recommendation [49] specifies that XML documents have a *physical structure* made up of storage units or *entities*. Entities are fragments of XML documents, which range in type and scope from single characters to complete external documents. Entities can be *parsed* or *unparsed*. Parsed entities contain markup or content text and should be parsed by an XML processor. An unparsed entity may be text or any other format, is not parsed by the XML processor and is generally passed without changes to the application using the XML document.

Furthermore, XML documents have a *logical structure* according to the W3C Recommendation, which means that an XML document have a prolog and one or more *elements* or containers of information that can be nested [49, 61].

An XML document prolog is the first major component in any XML document and it can consist of up to five possible components:

- An XML declaration,
- Processing instruction(s),
- A Document Type Declaration,
- Comments(s),
- White Space.

All these components except for the first one (XML declaration), are

optional. The first component, the XML declaration, is the first line in any XML document and nothing should precede it:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes">
```

There are three attributes defined in the XML declaration, namely the the XML version number, `1.0`, the document language encoding designation, `UTF-8`, and the standalone specification `yes`. The version (`1.0` in this case) corresponds to W3C XML Recommendation 1.0. The encoding attribute is optional, if nothing is specified, the default is UTF-8. Other character set options such as Unicode are available. The `standalone` attribute is also optional with the default being `"yes"`. *Yes* means that the document exists alone and there is no need to refer to any external document [49, 61, 159].

Elements are the basic building blocks of an XML document, and each document must at least have one root or parent element . The elements in an XML document are indicated by start and end tags as indicated below:

```
<elementname>
  content
</elementname>
```

All other elements contained in an XML document must be nested inside the root element's start and end tags. This forms the *element hierarchy* of an XML document. Elements may have *attributes* or *attribute specifications*. Element attributes are data that can be specified for elements and they appear in the form of name-value pairs inside the start tag of an element as indicated below [49, 61, 159].:

```
<elementname attribute="value">
  content
</elementname>
```

XML documents conforming to all the XML syntax rules such as that all start tags must have end tags etc. are called *well-formed* XML documents.

A well-formed XML document consists of a balanced tree of nested pairs of open and close tags. Each pair can include several attribute-value pairs [49, 61, 159].

A.4.2.4 Valid XML documents

Valid XML documents are well-formed XML documents that also conform to their document-type specification as contained in the document's respective *document type declaration (DTD)* or *schema*[3]. The role of DTDs and that of XML Schemas are essentially the same. They create a grammar for specific XML documents when they specify, for instance, allowable combinations and nestings of tag names, as well as attribute names. Both DTDs and XML Schemas specify only syntactic conventions. Any intended semantics are outside the realm of an XML specification [49, 84].

Section A.4.3 on page 301 contains a discussion on XML Schema.

A.4.2.5 XML language specifications

There is no fixed vocabulary for XML documents. XML vocabularies are created for each specific application as required [49, 61, 159, 233].

Since the XML 1.0 Recommendation was endorsed by the W3C in 1998, numerous XML-based vocabularies or languages have been developed in academia and industry by organisations that have to share high volumes of information. Some of these are depicted in Table A.1 (adapted from McKinnon and McKinnon [159], p.17-18).

[3] Section A.4.3 on page 301

Language Acronym	Description	URL / Reference
CDF	The CDF (Channel Definition Format) is an open specification that permits a web publisher to offer frequently updated collections of information, or channels, from any web server for automatic delivery to compatible receiver programs on PCs or other information appliances	http://www.w3.org/TR/NOTE-CDFsubmit.html (accessed July 2006)
CML	Chemical Markup Language (CML) is an extensible base for chemically aware markup languages. The World Wide Molecular Matrix is a molecular repository and contains and manages chemical information and molecules entirely in XML and CML (chemical markup language) / CMLComp (computational chemical markup language).	http://www.xml-cml.org/ (accessed July 2006)
ETD-ML	The Electronic Thesis and Dissertation Markup Language (ETD-ML) allows semantic encoding of ETDs independent of visual appearance and allows simplified hypertext and multimedia. ETD-ML converts theses from documents generated in word processors, for example, to SGML/XML.	http://etd.vt.edu/etd-ml/userguid.htm (accessed July 2006)
SMIL	The Synchronised Multimedia Integration Language (SMIL) enables simple authoring of interactive audiovisual presentations. SMIL is typically used for ”rich media”/multimedia presentations which integrate streaming audio and video with images, text or any other media type. SMIL is an easy-to-learn HTML-like language, and many SMIL presentations are written using a simple text-editor.	http://www.w3.org/AudioVideo/ (accessed July 2006)

MathML	MathML 2.0, a W3C Recommendation was released on 21 Feb 2001. A product of the W3C Math working group, MathML is a low-level specification for describing mathematics as a basis for machine to machine communication. It provides a much needed foundation for the inclusion of mathematical expressions in Web pages.	`http://www.w3.org/Math/` (accessed July 2006)

Table A.1: XML based languages

A.4.3 XML Schema

An *XML schema* is an XML document defining the content and structure of one or more derived XML documents. Generally, a *schema* is a model for describing the structure and content of data. XML Schema is a content modelling language as well as an application of *XML* that applies only to XML-related languages and documents. In particular, an XML Schema describes a model for a whole class of XML documents. The model describes the possible arrangement of elements, their attributes and text that would be present in a schema-valid document.

As discussed in Section A.4.2.1, XML was developed from SGML or GML. SGML specifies the DTD (document-type declaration) as part of a grammar specification [96]. However, because SGML with its associated DTDs are regarded as too cumbersome for the specification of data exchange vocabularies on the Web, XML Schema was defined by the W3C to replace DTDs [233]. For completeness the next section will briefly discuss DTDs.

A.4.3.1 Document Type Declaration (DTD)

SGML introduced the concept of the *document-type declaration* or DTD as a mechanism to describe the structure and content for derived documents. Specifically, a DTD in an XML (or HTML) document provides a specification of the elements, attributes, comments, notes, and entities contained in the

document [208]. It also indicates the relationship between these elements within the document [159].

DTDs, however, have several disadvantages, some of which are listed below [159, p.105]:

- ▷ DTDs have their own syntax (EBNF or Extended Backus Naur Form) that differs from XML. This means that the same tools cannot be used to process documents and their document models.
- ▷ DTDs have limited ability to describe elements and their attributes, an example being that it is not possible to indicate whether character data should be numbers, date format or currency.
- ▷ It is difficult to specify the cardinality of sub-elements in DTDs. It is possible to specify ”one or more” of a sub-element, but support for the specification of any other type of cardinality is limited.
- ▷ Lastly, but probably most significant, DTDs have limited support for namespaces meaning that they can't define or restrict the content of elements based on context sensitivity.

It is possible to revise or extend the DTD specification as inherited from SGML to address the above issues, but this would require the revision of the SGML standard's DTD language. The W3C opted for the development of XML Schema specifically for XML documents to overcome some of the shortcomings of DTD [165].

Using DTD

This section provides a short summary and some examples of DTD notation. The discussion is by no means exhaustive and the purpose is only to give an indication of the DTD syntax and to demonstrate how it is used.

A DTD specifies the grammar of an XML document and there can only be one DTD per XML document. A document-type declaration is specified in the prolog of an XML document. The document type is specified either in the XML document itself (internal DTD) or referenced as an external document with the *standalone* attribute (external DTD) in the prolog of the XML document [96, 159, p.117].

An example of the usage of DTDs is shown below. The XML file `hallo.xml` has a root element `Greeting` and this XML file can be validated against `hallo.dtd`:

```
<?xml version="1.0"?>
<!DOCTYPE Greeting SYSTEM "hallo.dtd">
  <Greeting>
    Hallo World!
</Greeting>
```

The DTD file hallo.dtd simply specifies the element as:

```
<!ELEMENT Greeting (#PCDATA)>
```

For the hallo DTD, any XML content as shown below will be valid:

```
<Greeting> any text but no markup </Greeting>
```

The following example would be invalid:

```
<Greeting>
  <aTag>various text</aTag>
  <someEmptyTag/>
</Greeting>
```

DTD Elements

DTDs define five different types of element content [61, p.3.08]:

- ▷ Any elements where there is no restriction on the element's content,
- ▷ Empty elements where the element cannot store any content,
- ▷ Character Data where the element can only contain a text string,
- ▷ Elements where the element may only contain child elements; and
- ▷ Mixed element content where the element contains both a text string and child elements.

An example indicating that the `Book` element has two children namely `Title` and `Author` is shown below:

```
<!ELEMENT Book (Title, Author)
<!ELEMENT Title (#PCDATA)
<!ELEMENT Author (#PCDATA)
```

`#PCDATA` stands for *parsed character data* which is any well-formed text string not containing special characters.

DTD Element Attributes

To enforce any attribute properties on an XML element, attribute-list declarations must be added to the document's DTD [61, p.3.16-17]. Attribute-lists:

- ▷ lists the names of all attributes associated with a specific element,
- ▷ specifies the data type of an attribute,
- ▷ indicate whether an attribute is required or optional; and
- ▷ provides a default value for the attribute where necessary.

The syntax for declaring a list of attributes is

```
<! ATTLIST element attribute1 type1 default1
                  attribute2 type2 default2
                  attribute3 type3 default3 ... >
```

An example of a `Customer` element that must have a `CustomerID` attribute with a value `ID` is shown below:

```
<! ATTLIST Customer CustomerID ID #REQUIRED>
```

The purpose of DTD and XML Schema is the same, but their approach differs. Table A.2 represents an adaption from Carey [61], p.4.14 and highlights some of the essential differences between XML Schema and DTD.

DTDs are still used with regard to the specification of XML documents. However, for the validation of XML documents on the Web, the W3C hopes

XML Schema	DTD
XML based, written in XML	Written in different format (EBNF)
Schema can be validated by XML parser	Format not supported by XML parser
Supports more than 40 data types	Supports ten data types
Supports the creation of customised data types	No customised data types
Easily handles mixed element content	Difficult to specify mixed element content
Schemas can be attached to namespaces	DTDs cannot be associated with a namespace
No support for entities	Entity support

Table A.2: Comparing XML Schema and DTD

to eventually completely replace DTDs with XML Schema declarations [233]. It is however noteworthy that the adherence of XML as a sub-set of the SGML standard was the reason for DTD validation. If any new grammar as a sub-set of SGML is defined or developed, it will again use DTD as a validation mechanism. *XML Schema* is a sub-set of *XML* and is only applicable to XML [96, 165].

A.4.3.2 XML Schema

When the XML 1.0 Recommendation was endorsed by the W3C in 1998 [257], the shortcomings of DTD were known. The XML Schema Working Group was specifically formed by the W3C to develop an XML Schema Language. As a result the W3C endorsed the XML Schema Part 1: Structures [234] and the XML Schema Part 2: Data-types [235] in 2001.

A *schema* is a model for describing the structure and content of data, and XML Schema was developed as a content modelling language and an application of *XML*, not *SGML*. XML Schema therefore applies only to XML-related languages and documents [61, 159] .

An XML Schema describes a model for a whole class of XML docu-

ments [233]. The model describes the possible arrangement of elements, their attributes and text that would be present in a schema-valid document. The schema-models are described in terms of *constraints* where a constraint defines what can appear in a given document. *Content model constraints* define the *elements* that can appear, as well as the number and type of components, the order they appear in and whether they are required or optional. *Data-type constraints* describe the *units of data* that the schema considers valid [61, 159, 233].

A schema defines a *class* of XML documents, and therefore the term *instance document* is often used to describe an XML document that conforms to a particular schema. The *instance document* of the schema represents a specific instance of the structure that is specified in the schema document, and it contains relevant content in the structure. [61, 233].

A.4.3.3 XML Schema documents

This section discusses the format and structure of XML Schema documents, enabling the reader to form a concept of what is required when developing an XML Schema grammar.

An XML Schema document is in the first place a well-formed XML document. Therefore the first line is mandatory and is the *XML declaration* required in all XML documents [233].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema example -->
  <!-- Next we declare the root element = schema -->
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      document specifications
  <\xsd:schema>
```

The line beginning with <xsd:schema ..> is the start tag for the *schema element* which is the root element of the XML document. The <schema> element is therefore the parent element of all other elements

or *subelements* in the schema. The start tag may also include attributes defining for instance the namespaces and unqualified local elements. The *document specifications* are schema components that declare the schema, in other words the properties and contents of the data that the instance documents should contain [61, 159, 233].

An XML Schema consists of components. A Schema component is the generic term for the building blocks that comprise the abstract data model of the schema [234]. An XML Schema is therefore a set of *schema components*, such as:

- ▷ Simple type definitions,
- ▷ Complex type definitions,
- ▷ Attribute declarations,
- ▷ Element declarations.

Each of the elements in a schema usually has a prefix xsd: which is associated with the XML Schema namespace through the declaration,

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

that appears in the schema element. The prefix `xsd:` is used by convention to denote the XML Schema namespace, although any prefix can be used [61, 159, 233].

XML Schema differentiate between simple types (which cannot have element content and cannot carry attributes) and complex types (which allow elements in their content and may carry attributes). There is also a distinction between definitions which create new types (both simple and complex), and declarations for document instances which allow elements and attributes with specific names and types (both simple and complex) to appear [233].

XML Schema simple types

Any XML schema declares several elements and attributes that have simple types. Some of these simple types, such as `string` and `decimal`, are built-in to XML Schema, while others are derived from the built-in's. Both built-in simple types and their derivations can be used in all element and attribute declarations [159, 233–235].

There are 44 *built-in* data types that are part of the XML Schema specification. The built-in data types consist of either *primitive* also called *base types*, or *derived types*. The primitive types consist of 19 fundamental data types that are not defined in terms of other data types. They are also called *atomic types*, and the value of an atomic type is indivisible from XML Schema's perspective [61, 235]..

The derived data types are a collection of 25 data types that were created based on the 19 primitive types [61, 233]. An example of a built-in type declaration is presented below:

```
<element name="ID-number" type="positiveInteger" />
```

This example declares that the ID-number element is limited to only positive integers.

New simple types are defined by deriving them from existing simple types (built-in's and derived). A new simple type can be defined by restricting an existing simple type (signifying that the legal range of values for the new type are a sub-set of the existing type's range of values). The `simpleType` element is used to define and name a new simple type. The restriction element is used to indicate the existing (base) type, and to identify the constraints or the range of values.

For example, defining a new type of integer called newInteger with a range of values is between 10000 and 99999 (inclusive) is declared by using the built-in simple type integer [234, 235]:

```
<xsd:simpleType name="newInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

User-derived data types are therefore specifically created by the schema author, and consist of the built-in types and/or other user-derived

types.

XML Schema also defines other simple types such as *union types* and *list types* (for instance a list of Provinces of South Africa as shown below). This is in addition to the atomic types described above [234, 235]..

```
<xsd:simpleType name="SAProvList">
 <xsd:list itemType="SAProv"/>
</xsd:simpleType>

<xsd:simpleType name="NineSAProvinces">
 <xsd:restriction base="SAProvList">
  <xsd:length value="9"/>
 </xsd:restriction>
</xsd:simpleType>
```

Elements whose type is `NineSAProvinces` should have nine items, and each of the nine items must be one of the (atomic) values of the enumerated type `SAProv`.

Atomic types and list types enable an element or an attribute value to be one or more instances of one atomic type. In contrast, a union type enables an element or attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types [234, 235].

XML Schema complex types

As discussed, XML Schema differentiates between complex types (which may have attributes and/or elements in their content), and simple types which cannot have element content and cannot carry attributes [233]:.

New complex types are defined using the `complexType` element and such definitions typically contain a set of element declarations, element references, and attribute declarations. The next section discusses element declarations. The declarations are not themselves types, but rather an association between a name and the constraints which govern the appearance of that name in documents governed by the associated schema

[234, 235].

Any addition of attributes or child elements to simple types will result in the specification of a complex type. If we want to specify a currency, for example, we have to declare a complex type. Decimal is a simple type, and the next example, obtained from W3C [233] defines a new complex type:

```
<xsd:element name="internationalPrice">
 <xsd:complexType>
  <xsd:simpleContent>
   <xsd:extension base="xsd:decimal">
    <xsd:attribute name="currency" type="xsd:string"/>
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>
</xsd:element>
```

In the example, the `complexType` element is used to start the definition of a new (anonymous) type. To indicate that the content model of the new type contains only character data and no elements, we use a `simpleContent` element. Finally, the new type is derived by extending the simple decimal type. The extension consists of adding a currency attribute using a standard attribute declaration. The `internationalPrice` element declared in this way can appear in an instance document as follows [234, 235]:

```
<internationalPrice currency="EUR">423.46</internationalPrice>
```

Elements that contain sub-elements or which carry attributes are said to have *complex types*, whereas elements that contain numbers (and strings, and dates, etc.) but which do not contain any sub-elements are said to have *simple types*. Some elements have attributes; attributes always have simple types [234, 235].

The next section focuses on element declarations.

XML Schema element declarations

XML Schema allows for the specification of two types of elements: complex and simple (note: simple and complex *elements* vs. simple and complex *types*). A *simple type* element contains only character data. A *complex type* element is an element that has attributes or is a parent for child elements [234, 235].

A simple element is declared as follows:

```
<element name="name" type="type" />
```

Or if we use a namespace prefix (in this case `xsd`):

```
<xsd:element name="name" type="xsd:type" />
```

A complex element contains attributes and/or other elements, for example:

```
<element name="name">
  <complexType>
    compositor
      element declarations
    compositor
      attribute declarations
  </complexType>
</element>
```

In this example, `name` is the name of the element in the instance document, `element declarations` are simple type declarations for each child element, `compositors` define how the list of elements are organised (of value either sequence, choice or all), and `attribute declarations` are declarations that define the attributes of the element [199, 234, 235].

One advantage XML Schema has over DTDs is the ability to specify mixed content elements. If an element contains both text and child el-

ements, the *mixed* attribute is added to the `complexType` tag [61]. For example, the following XML content:

```
<Description>
  Author <Name>Charles Darwin</Name> wrote
  <Book>On the Origin of Species</Book>
  in the 19th century.
</Description>
```

can be declared in XML Schema as

```
<element name="Description">
  <complexType mixed="true">
    <sequence>
      <element name="Name" type="string" />
      <element name="Book" type="string" />
    </sequence>
  </complexType>
</element>
```

XML Schema is very versatile and allows the content text to appear before, between and after child elements as in the example above [61, 159].

XML Schema element occurrences

To specify the number of times an element occur, one uses the `minOccurs` and `maxOccurs` attributes, for example [61]:

```
<element name="BookTitle" type="string" minOccurs="1"
         maxOccurs="unbounded" />
```

In this example, there should be at least one `BookTitle` element, and there is not an upper limit to the number of occurrences. If `minOccurs` is

set to 0, the declared item is optional, if `minOccurs` and `maxOccurs` are not specified, their value is assumed to be 1. The `minOccurs` and `maxOccurs` attributes can also be used in compositors to specify the repeat occurrence of entire sequences of items [235].

XML Schema attributes

In XML Schema, any element that contains an attribute, is also a complex type. The syntax for an attribute is:

```
<attribute name="name" type="type" use="use"
  default="default" fixed="fixed" />
```

where `name` is the name of the attribute, `type` is the data type, `use` states whether the attribute is required or not (possible values are *required*, *optional* and *prohibited*), `default` is the default value of the attribute and `fixed` is a fixed value for the attribute [61, 235].

An attribute should be declared with the element to which it pertains. If an element is empty, such as:

```
<Student No="123456" Gender="male" />
```

an attribute is declared using the same syntax as child elements:

```
<element name="Student">
  <complexType>
      <attribute name="No" type="string" />
      <attribute name="Gender" type="string" />
  </complexType>
</element>
```

If an element contains child elements in addition to attributes, the attributes are placed after the element declarations. For example:

```
<Student No="123456" Gender="male">
  <Name>John Smith</Name>
  <Age>18</Age>
</Student>
```

is specified

```
<element name="Student">
  <complexType>
    <sequence>
      <element name="Name" type="string" />
      <element name="Age" type="positiveInteger" />
    </sequence>
    <attribute name="No" type="string" />
    <sttribute name="Gender" type="string" />
  </complexType>
</element>
```

A.4.3.4 Namespaces and XML Schemas

XML Schemas provide namespace support for the qualification of elements[4].

The target namespace declared in the prolog of an XML document provides the information for the processor to check any instance document to see whether it validates against a schema. A namespace therefore indicates to the XML processor that the definitions of elements and other data-types in the schema are adopted from the declared namespace [47].

The namespace declaration can be done as an attribute of the *schema* element as shown [47, 159].

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema".../>
```

This code snippet declares the attribute `xmlns:xsd` with value

[4]Namespace concepts are discussed in Section A.4.1 on page 292

"http://www.w3.org/2001/XMLSchema". Breaking the statement up into parts:

- The xmlns name indicates that it is a namespace declaration.
- The xsd portion is the abbreviation used to relate the respective elements and data-types to the namespace.
- The "http://www.w3.org/2001/XMLSchema" portion is the unique URI or URL identifying the namespace.

In this case the W3C Namespace Recommendation is the namespace used by the XML processor for the definitions of elements and other data-types. When the processor encounters any data-types with the prefix xsd: (the prefix represents the URL) the meaning for those data-types are identical to the definition found in the W3C Recommendation [47, 234].

After the namespace has been defined, the element used in conjunction with the abbreviation, becomes a unique element. An example is <xsd:sequence>, where the local part of the unique name is sequence [61].

Namespaces do not have to be declared explicitly with prefixes, as in:

```
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema" ..../>
```

Without a prefix, the shown URL is presumed to be the default namespace to which an element or attribute without a prefix in the document is associated [61].

A.5 LAYER 3/LAYERS 3A AND 3B

RDF (Resource Descriptive Framework) and RDF Schema technologies reside on Layer 3 (refer to Figures A.2 and A.3). With the positioning of RDF Schema, Layer 3b, above RDF M&S (Model and Syntax), Layer 3a, in Figure A.3, Berners-Lee [31] emphasises the importance of a vocabulary description mechanism on top of the RDF data model as part of the Semantic Web layered architecture.

A.5.1 RDF

RDF (Resource Descriptive Framework) is a W3C Recommendation designed to standardise the definition and usage of meta-data, or in the context of the Semantic Web, a mechanism to capture data about web resources. The W3C describes the Resource Description Framework (RDF) as a language for representing meta-data or information about resources on the Web [238, 239]. RDF is intended for the exchange of meta-data about resources between applications, but *without loss of meaning* [86].

The purpose of RDF is to declare meta-data that is machine-processable. RDF provides a mechanism to declare statements that describe resources by means of a basic data model [240]. A *statement* describes an entity (resource) in terms of *properties*, which have *values*. Furthermore, an RDF statement is a subject, predicate, object triple [85, 92, 238]. The *subject* is the resource of the statement. The *predicate* is the property or characteristic of the subject specified by the statement (examples include creator, creation-date, or language), and the value of the property is the *object* [238, 239].

The W3C RDF specification was developed to provide a common framework so that application developers can use common RDF parsers and processing tools as is the case with XML using common XML parsers and tools. According to the W3C in *RDF Concepts* [238, 239, 243], the design of RDF is intended to meet the following goals:

- having a simple data model,
- having formal semantics and provable inference,
- using an extensible URI-based vocabulary,
- using an XML-based syntax,
- supporting use of XML schema data-types; and
- allowing anyone to make statements about any resource.

The next section discusses the RDF model in more detail. The reader not interested in the detail of RDF could continue to RDF Schema, Section A.5.2 on page 323.

A.5.1.1 RDF model

As stated previously, RDF is specifically intended for the representation of meta-data about Web resources. RDF describes the resources in terms of their properties and property values. More specifically, RDF makes statements about resources by using an subject, predicate, object triple [85, 238].

The statement `http://www.thesis.org/SWTechnologies.html has a creator whose value is Student AJG`, which is described in RDF terms as follows:

- ▷ the subject is the URL `http://www.thesis.org/SWTechnologies.html`,
- ▷ the predicate is the word `creator`; and
- ▷ the object is the phrase `Student AJG`.

The RDF statement above can be depicted in a graph as shown in Figure A.4.

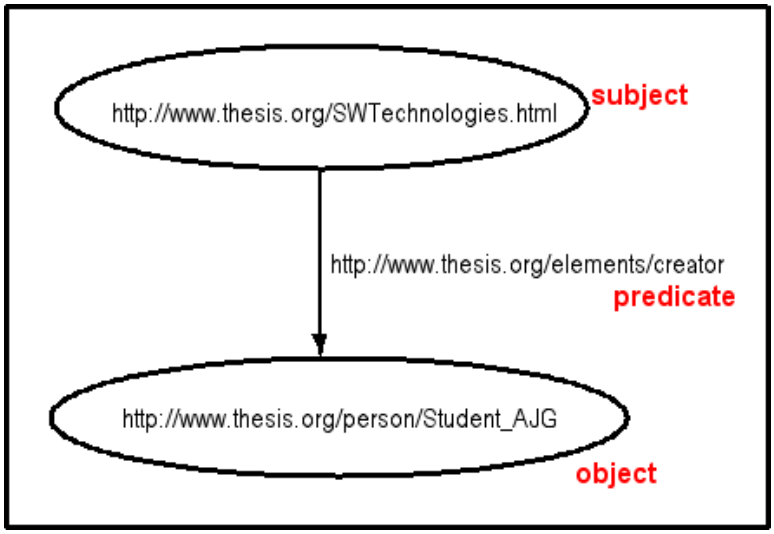


Figure A.4: An RDF graph for a basic statement: `http://www.thesis.org/SWTechnologies.html has a creator whose value is` *`Student AJG`*

An RDF statement or subject-predicate-object-triple, (sometimes also called an object-attribute-value-triple) can also be indicated as $A(O, V)$, which means that an object O has an attribute A with value V. This re-

lationship can also be modelled in a graph as a labelled edge A between two nodes, O and V: $[O]-A \rightarrow [V]$ as shown by the arrow in the graph in Figure A.4 [62, 238]. The underlying datamodel of RDF can be labelled a hypergraph, with each statement being a predicate-labelled link between object and subject. The graph is a hypergraph since each node can itself again contain an entire graph [54].

RDF uses URI[5], or more specifically URIrefs (URI References) as its mechanism for identifying the subjects, predicates, and objects in statements. A URIref is a URI with an optional fragment identifier at the end. To clarify, the URI reference `http://www.thesis.org/SWTechnologies.html#section2` consists of the URI `http://www.thesis.org/SWTechnologies.html` and the fragment identifier `section2`. RDF URIrefs are specified using Unicode[6] characters. In general, RDF defines a resource as anything that is identifiable by a URI reference. RDF can therefore be used to represent information about anything that can be identified on the Web, even when it cannot be directly retrieved on the Web.

The RDF graph in Figure A.5 was obtained from the RDF Primer of the W3C [238] and it is used to further illustrate how RDF describes meta-data. This figure represents the group of statements *there is a Person identified by* `http://www.w3.org/People/EM/contact\#me`*, whose name is Eric Miller, whose email address is* `em@w3.org`*, and whose title is Dr*.

In figure A.5, URIrefs are used to identify:

- individuals, e.g., Eric Miller, identified by `http://www.w3.org/People/EM/contact#me`,
- kinds of things, e.g., Person, identified by `http://www.w3.org/2000/10/swap/pim/contact#Person`,
- properties of those things, e.g., mailbox, identified by `http://www.w3.org/2000/10/swap/pim/contact#mailbox`; and
- values of those properties, e.g. `mailto:em@w3.org` as the value of the mailbox property (RDF also uses character strings such as "Eric Miller", and values from other data-types such as integers and dates,

[5] See Section A.3.2 on page 290 for a discussion of URIs.

[6] See Section A.3.1 on page 288 for a discussion of Unicode

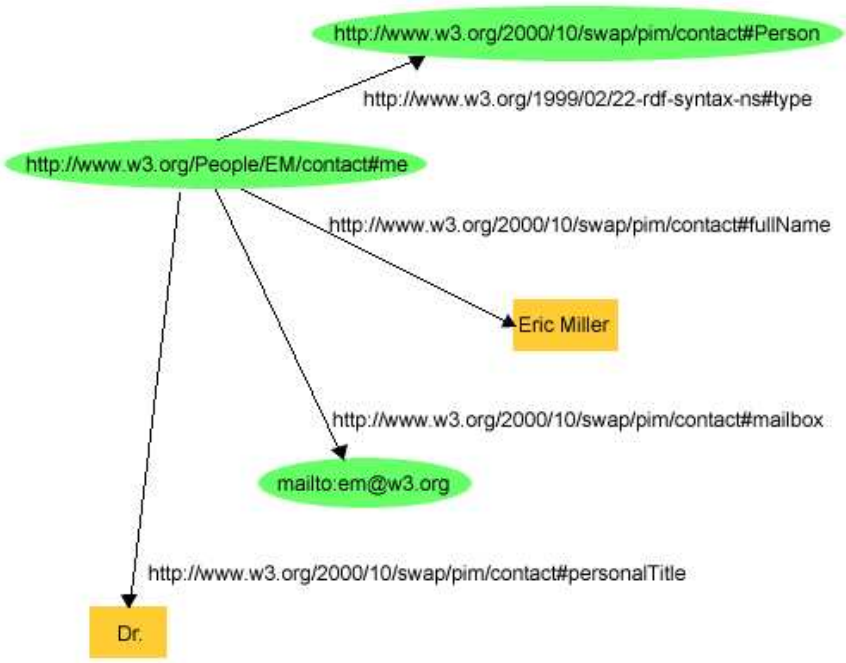


Figure A.5: An RDF graph describing Eric Miller [237]

as the values of properties).

The purpose of RDF is to declare meta-data that is machine-processable and RDF therefore uses XML as syntax. RDF defines a specific XML markup language, RDF/XML, to represent RDF information. The graph in Figure A.5 is represented in RDF/XML [22] as follows:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

RDF allows objects and values to be interchanged. This means that

labelled edges in a graph can be chained. It is also possible to make an RDF statement or triple the object of another statement, allowing for the nesting of RDF statements [54, 85].

A.5.1.2 Blank RDF nodes

An RDF graph can also be used for expressions that are not necessarily triples. The representation of such a structure requires the use of a blank node. In the following example (obtained from the W3C RDF Primer [237]), RDF is used for the representation of John Smith's address. John's address needed to be recorded as a structure consisting of separate street, city, state, and postal code values.

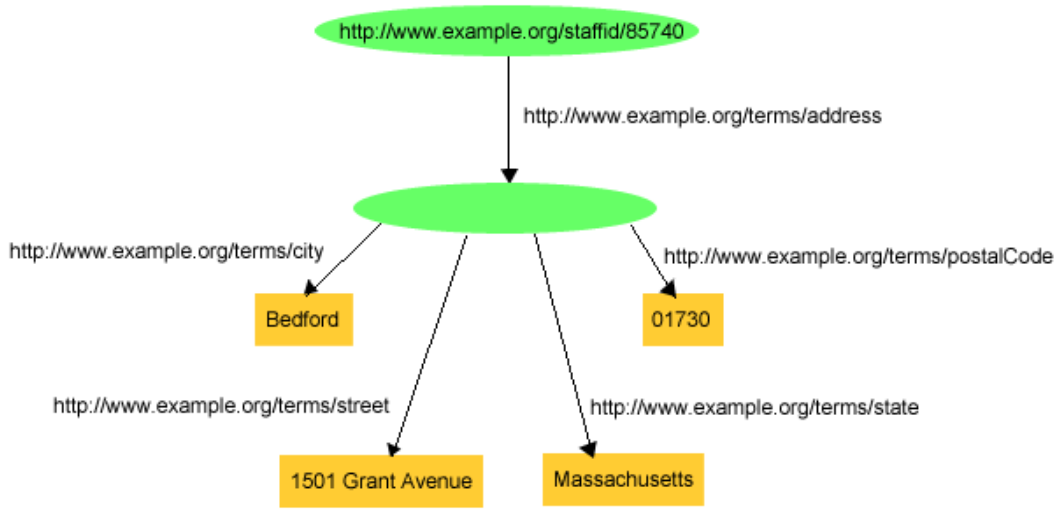


Figure A.6: An RDF graph with a blank node [237]

Figure A.6 is a valid RDF graph and uses a node without a URIref to stand for the concept of "John Smith's address". The node provides the necessary connectivity between the various other parts of the graph. However, some form of explicit identifier for that node is needed in order to represent this graph as triples.

One possibility of representation of the triples corresponding to Figure A.6 is:

```
exstaff:85740       exterms:address         _:johnaddress .
_:johnaddress       exterms:street          "1501 Grant Avenue" .
_:johnaddress       exterms:city            "Bedford" .
_:johnaddress       exterms:state           "Massachusetts" .
_:johnaddress       exterms:postalCode      "01730" .
```

where _:johnaddress (a blank node identifier) refers to the presence of the blank node.

There is often a need for statements about *collections* of resources. For this purpose RDF defines the concept of a *container* [238, 243]. RDF defines three types of containers that can represent collections of resources or literals:

- ▷ *Bags* are unordered lists. Bags do not enforce set semantics, so a value may appear several times in a Bag,
- ▷ *Sequences* are ordered lists. Like Bags, Sequences permit duplicate values; and
- ▷ *Alternatives* are lists from which the property can use only one value.

The previous sections gave a rudimentary description of the RDF model and some of the basic ways in which the model represents meta-data. More detailed information can be obtained from the W3C RDF documentation set [237, 239–243].

A.5.1.3 RDF vocabularies

RDF is used to define vocabularies. The term *vocabulary* is used to refer to a RDF graph depicting a set of meta-data or data definitions. A RDF vocabulary is therefore a set of URIrefs and its relationships defined for some specific purpose, such as a shared domain definition for participants of a domain, or even the set of URIrefs defined by RDF for its own use, for instance in RDF Schema [237, 239, 241]

RDF provides a way to make statements that RDF applications can process although the specific application does necessarily ”understand” the statements contained in the definition. An example might assist in explain-

ing the concepts. A user could search the Web for a review of all the scuba diving sites in South Africa. This user could then create an average rating for each diving site and publish the information on the Web. Another Web site could take the list of diving sites and create a *"Top Ten Tourist Adventure Sites"* page. Because a shared vocabulary exists, different users might built a commonly understood and increasingly powerful (as additional contributions are made) *information base* about topics on the Web.

One of the most quoted examples of a RDF vocabulary is at the DCMI (Dublin Core Meta-data Initiative) [82]. The DCMI is an organisation dedicated to promoting the widespread adoption of interoperable meta-data standards and developing specialised meta-data vocabularies for describing resources that enable more intelligent information discovery systems. The mission of DCMI is to make it easier to find resources using the Internet through the following activities:

- ▷ developing meta-data standards for discovery across domains,
- ▷ defining frameworks for the interoperation of meta-data sets; and
- ▷ facilitating the development of community- or disciplinary-specific meta-data sets that are consistent with items 1 and 2.

In addition to the DMCI, the RDF Resources page compiled by Beckett [21] contains a list of several RDF applications and projects which is updated regularly.

Section A.5.2 discusses RDF Schema which is an example of a RDF vocabulary.

A.5.1.4 The application of RDF, XML and XML Schema on the Semantic Web

RDF is preferred above XML for the description of meta-data on the Semantic Web because RDF specifies a data model and the information specified by RDF can be mapped directly and unambiguously to the model [84]. This model is part of an external specification and several generic, third-party parsers are already available to interpret such an RDF document [238]. An RDF model separates semantic and syntactic information. Be-

cause the RDF model is a W3C standard, all users that adopt RDF will be able to make this distinction [184].

XML Schema is not necessarily used in conjunction with RDF since XML Schema is a language for restricting the syntax of XML applications [86]. However, XML Schema might be useful to specify certain data-types or similar restrictions on the syntax. XML Schema is therefore not used to control the syntax of RDF, rather the syntax as contained in an XML document [184].

A.5.2 RDF Schema

RDF Schema specifies extensions to RDF that are used to define common vocabularies in RDF meta-data statements. RDF itself provides the data model and does not prescribe any application-specific classes and properties, this is accomplished by RDF Schema. RDF Schema specifies extensions to RDF, which are provided by the *RDF Vocabulary Description Language 1.0: RDF Schema* [241]. RDF Schema provides a predefined, basic type system for RDF models, thus extending RDF by assigning an externally specified semantics to specific resources. RDF Schema expressions are valid RDF expressions, and therefore RDF Schema is a semantic extension of RDF [53, 119]. Software that can interpret RDF can also be used to interpret an RDF Schema implementation although it will not attach the intended meaning to the built-in schema definitions [85].

The RDF vocabulary description strategy contained in RDF Schema acknowledges that there are many techniques that enable *description of meaning* of classes and properties. To extend the description of meaning, ontology languages (such as DAML+OIL, OIL and OWL), inference rule languages and other formalisms are used [85].

In the definition of a particular vocabulary for RDF data, RDF Schema assists in a similar way to the function of XML schema that provides a vocabulary-definition facility for XML. RDF schema provides a predefined, basic type system for RDF models [84]. RDF Schema extends RDF by giving an externally specified semantics to specific resources. e.g., to

`rdfs:subclassOf` or to `rdfs:Class`. It is because of this external semantics that RDF Schema is useful [54].

RDF Schema does not specify a vocabulary of application-specific classes, but it describes the mechanisms necessary to specify such a vocabulary [238]. It provides the facilities required to describe application specific classes and their associated properties. For example, the property `ex:breedType` should be used in describing an `ex:Dog`. RDF Schema can therefore also be described as providing a typing mechanism for RDF, similar to the type systems of object-oriented programming languages. An example (obtained from [241]), states that a class `ex:Dog` might be defined as a sub-class of `ex:Mammal` which is a sub-class of `ex:Animal`, meaning that any resource which is in class `ex:Dog` is also implicitly in class `ex:Animal` as well [52, 231].

As mentioned, RDF Schema facilities are themselves specified in the form of an RDF vocabulary; that is, as a specialised set of predefined RDF resources. Usually, resources in the RDF Schema vocabulary have URIrefs with the prefix `http://www.w3.org/2000/01/rdf-schema#` or `rdfs:`. Any vocabulary that is defined using RDF Schema is also a legal RDF graph and any software that can interpret RDF can therefore also be used to interpret an RDF Schema implementation although it will not attach the intended meaning to the built-in Schema definitions. For this an RDF application must be written to process an extended language that includes not only the `rdf:` vocabulary, but also the `rdfs:` vocabulary with the built-in meanings [53, 241].

A.5.2.1 Summary of RDF Schema constructs

In order to illustrate the nature of RDF Schema as well as the way in which RDF is utilised, an RDF Schema summary obtained from the W3C *Resource Description Framework (RDF) Model and Syntax Specification* [241] is presented in Tables A.3 and A.4. These tables presents an overview of the vocabulary of RDF, formed from the original vocabulary defined in the *RDF Model and Syntax Specification* [231] as well as the classes and properties that originated with RDF Schema [241].

Class Name	Comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties.
rdf:List	The class of RDF Lists.

Table A.3: RDF Classes

Property name	Comment	Domain	Range
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List

rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values.	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

Table A.4: RDF Properties

RDF is used to represent information in the Web using a simple data model using an subject, predicate, object triple. RDF Schema is a semantic extension of RDF [239], which signifies that RDF Schema expressions are valid RDF expressions in the form of the subject-predicate-object triple. In RDF Schema, there is an agreement on the semantics of certain terms and thus on the interpretation of certain statements [53].

A benefit of this approach is that the description of existing resources could be extended [239]. The RDF vocabulary description strategy also acknowledges that there are many techniques through which the meaning of classes and properties can be described. This is where ontology languages such as DAML+OIL[7] and OWL[8], inference rule languages and other formalisms (for example temporal logics) will be used to extend the specification of meaning.

[7] See Section A.6.1.2 on page 329 for a discussion of DAML+OIL.

[8] See Section A.6.1.1 on page 329 for a discussion of OWL.

A.6 LAYER 4 / LAYERS 4A AND 4B

In V1 (Figure A.2) *Ontology vocabulary* is depicted on Layer 4, whilst this layer is separated as *Ontology*, Layer 4a, and *Rules*, Layer 4b, in V2 (Figure A.3). In Figure A.3, Berners-Lee [31] acknowledges that an ontology is a *knowledge representation language* capturing the syntax (ontology) as well as semantics (rules) of a specific domain [157, 158]. OWL is the W3C technology representing an *Ontology vocabulary* or *Ontology* associated with this layer, whilst W3C research efforts aim to establish the technologies required for the implementation of the *Rules* to be contained in this layer [130].

It should be noted that the terminology on this layer differs from that pertaining to the three preceding layers, because *functionality* rather than *technology* is mentioned.

A.6.1 Ontology vocabulary / Ontology

An ontology specifies a machine-readable vocabulary in computer systems technology descriptions. Generally it is defined as a shared, formal, explicit specification or conceptualisation of a particular domain [54, 84, 121]. An ontology typically describes a hierarchy of resource concepts within a domain and associates each concept's crucial properties with it. Ontologies are used to define and manage concepts, attributes and relationships between concepts in a precise manner [58].

The concept of an ontology was inherited from philosophy and has only recently become commonplace in computer systems technology descriptions where an ontology specifies a machine readable vocabulary. Ontologies on the Semantic Web and expert systems or AI (artificial intelligence) technologies of the 1980s are based on the same motivations but they emerged from different architectures which implies that the technologies are deployed or applied differently [184].

Ontologies assist in creating a common understanding for communication between people and computer applications. The Web Ontology Work-

ing Group at the W3C identified six main areas for Ontology use within the *OWL Use Cases and Requirements document* [120], namely:

- *Web Portals*
  - Categorisation rules used to enhance search
- *Multimedia Collections*
  - Content-based searches for non-text media
- *Corporate Web Site Management*
  - Automated Taxonomical Organisation of data and documents
  - Mapping Between Corporate Sectors (mergers!)
- *Design Documentation*
  - Explication of ”derived” assemblies (e.g. the wing span of an aircraft)
  - Explicit Management of Constraints
- *Intelligent Agents*
  - Expressing User Preferences and/or Interests
  - Content Mapping between Web sites
- *Web Services and Ubiquitous Computing*
  - Web Service Discovery and Composition
  - Rights Management and Access Control

From the above, it is possible to deduce that ontologies will be used in applications that search across or merge information from diverse communities. XML, DTDs and XML Schemas are sufficient for exchanging data between parties who have agreed to common definitions beforehand, but the lack of semantics prevent this when new XML vocabularies emerge. RDF and RDF Schema specify simple semantics associated with identifiers. With RDF Schema, one could define classes with multiple subclasses and super classes, as well as properties with sub-properties, domains, and ranges. Therefore, RDF Schema is regarded as a simple ontology language. However, RDF Schema is limited, for instance, RDF Schema cannot specify that `Person` and `Car` classes are disjoint, or that a string quartet has exactly four musicians as members. In order to achieve interoperation between numerous, autonomously developed and managed schemas, richer semantics are required [19, 133, 228].

It is foreseen that ontologies will play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications in the future, and therefore the establishment of Semantic Web. The W3C developed OWL as a Web Ontology language which satisfies most of the current foreseen requirements of an ontology specification for the Semantic Web [120, 122].

A.6.1.1 OWL

The acronym *OWL* was derived from *Web Ontology Language*. The W3C Working Group decided against the direct acronym *WOL* and decided on *OWL* instead. OWL as a Web Ontology Language means that the language was designed to be compatible with the architecture of the Web, specifically the Semantic Web [158]. The next section provides some insight into the development of OWL and the relationship between known Ontology language efforts such as OIL and DAML+OIL.

A.6.1.2 OWL, OIL and DAML+OIL

Research to represent ontologies with languages dates back to the work on frame-languages in the early days of AI. With the emergence of the Web, efforts of designing ontology-representation languages that are Web enabled started [100]. Several initiatives were created around this research topic. On-to-Knowledge [182] is an European IST project that focused on the development of Semantic Web technology, specifically the development of ontology-based tool environments for knowledge management. These tools have to deal with large numbers of heterogeneous, distributed, and semi-structured documents typically found in large company intrados and the Web. The project aimed to develop:

- ▷ a toolset for semantic information processing and user access,
- ▷ OIL, an ontology-based inference layer on top of the Web; and
- ▷ an associated methodology and validation by industrial case studies.

OIL (Ontology Inference Layer) as well as several tools and architectures providing a Web-based representation and inference layer for ontolo-

gies were results of this work. A complete RDF Schema implementation for OIL was developed [54, 100, 101, 182].

OIL stands in a similar relationship to RDF Schema than RDF Schema to RDF in that it defines semantics to extend RDF Schema. It is possible to capture meaning in OIL that cannot be captured in RDF Schema, but this extension is done in such a way that the RDF Schema document is still valid [54, 84].

OIL adopted the essential modelling primitives of frame-based systems into its language, basing its formal semantics on DL (Description Logics). DL models knowledge in terms of concepts, role restrictions and derived classification taxonomies, and provide a principled method to reason with the presented knowledge. In addition, OIL was developed with support for the XML and RDF syntaxes. OIL has a well-defined syntax in XML based on a DTD and a XML schema definition, and is also defined as an extension of RDF.

Concurrently with the On-to-Knowledge initiative, DARPA (the Defense Advanced Research Projects Agency), developed DAML (the DARPA Agent Markup Language) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the Web. Because of similar work these organisations decided to co-operate [79]. DAML was a significantly funded project for the development of Semantic Web technology. As a result of this co-operative research, a new language called DAML+OIL was defined. To continue work in this area, the *Joint EU/US ad hoc Agent Markup Language Committee* was formed and some of the OIL standards were adjusted to conform with an international standard backed up by the US defence department [79]. Fensel [100] describes DAML+OIL as an expressive description logic supported with Web syntax. It does not provide layered sub-languages with different complexity nor language primitives that are defined around a modelling paradigm.

Because of its inheritance from DAML and OIL, DAML+OIL builds on existing Web technologies such as XML, URI and RDF. DAML+OIL markup is a specific kind of RDF markup. RDF, in turn, can be serialised to XML with its associated Namespaces and URIs [157, 229].

It was inevitable that the W3C, because of its mandate, set up a *Semantic Web Activity* initiative. The Activity was initiated early 2001 with a Web-Ontology (WebOnt) Working Group tasked to investigate all the efforts and consolidated them into a Web Ontology Language [158, 236]. OWL is the result of the activities of this group and the OWL Recommendation consisting of six documents was issuedby the W3C in February 2004 [244]. Some of the results of this work are discussed in the next section, Section A.6.1.3.

A.6.1.3 OWL specification

OWL includes concepts and design aspects of DAML+OIL. OWL extends RDF Schema in order to express complex relationships between different classes specified in RDF Schema. In addition OWL enhances the specification of constraints applicable to classes and properties [120, 158].

The W3C OWL document set consists of six documents aimed at different audiences or users, namely (1) a presentation of the OWL use cases and requirements [120], (2) an overview document which briefly explains the features of OWL [158], (3) a comprehensive OWL guide that provides a walk-through of the features of OWL [215], (4) a reference document that provides the details of every OWL feature [20], (5) a test case document [63], as well as (6) a document presenting the semantics of OWL and details of the mapping from OWL to RDF [192].

Within this document set, OWL Specifies three sub-languages. These languages were specified to be increasingly expressive and ontology designers should choose the most appropriate version:

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, see the section on OWL Lite in the OWL Reference for

further details.

- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a sub-class of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full. Complete OWL Full implementations do not exist with the writing of this document.

Each of these sub-languages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not [158, 215].

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

A.6.1.4 Structure of an OWL ontology

A typical OWL ontology begins with a namespace declaration. Namespaces[9] provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation more readable indicating precisely

[9] Section A.4.1 on page 292 discusses Namespaces.

what vocabularies are being used. Therefore a standard initial component of an ontology includes a set of XML Namespace declarations enclosed in an opening `rdf:RDF` tag [20, 120, 158].

Once namespaces are established, a collection of assertions about the ontology grouped under an `owl:Ontology` tag are normally included. These tags support for instance a name or reference for the ontology, comments, version control and might also specify the inclusion of other ontologies [20]. At this stage in the ontology document it is appropriate to specify the basic elements of the ontology. Elements of an OWL ontology consists of classes, properties, instances of classes, and relationships between these instances. OWL provides the language components essential to define these elements [20, 120, 158].

OWL support the definition of simple named classes with the `Class` and `rdfs:subClassOf` constructs. An ontology needs to specify the basic concepts in the domain. These concepts should correspond to classes that are the roots of various taxonomic trees. Every class is a member of the class `owl:Thing`. Thus each user-defined class is implicitly a sub-class of `owl:Thing`. Domain specific root classes are defined by declaring a named class. In addition, OWL also defines the empty class, `owl:Nothing`.

A class definition has two parts: a name introduction or reference and a list of restrictions. Each of the immediate contained expressions in the class definition further restricts the instances of the defined class [20, 120, 158, 215].

In addition to classes OWL allows for the description of members of classes. These can be described as individuals in the universe of things described by the ontology. An individual is minimally introduced by declaring it to be a member of a class. It is important to note the difference between a *class* and an *individual* in OWL. A class is simply a name and collection of properties that describe a set of individuals. Individuals are the members of those sets. Thus classes should correspond to naturally occurring sets of things in a domain of discourse, and individuals should correspond to actual entities that can be grouped into these classes [215].

OWL *properties* are used to assert general facts about the members of

classes and specific facts about individuals. A property is a binary relation. Two types of properties are distinguished [215]:

- ▷ data-type properties that are used to describe relations between instances of classes and RDF literals and XML Schema data-types,
- ▷ object properties that are used to describe relations between instances of two classes. Note that the name object property is not intended to reflect a connection with the RDF term `rdf:object`.

OWL includes a number of mechanisms that are used to further specify properties. It is possible to specify property characteristics which provides a powerful mechanism for enhanced reasoning about a property. In addition to designating property characteristics, it is possible to further constrain the range of a property in specific contexts in a variety of ways. This is done with property restrictions. For the detail of OWL, a reader is referred to the W3C OWL documentation set [20, 63, 120, 158, 192, 215].

At present a number of OWL ontologies are available on the Web, including an ontology library at DAML [79], which contains about more than 300 examples written in OWL or DAML+OIL. In addition, several ontologies described as *large* ontologies have been released in OWL. These include a cancer ontology in OWL developed by the US NCI's (National Cancer Institute) Center for Bio-informatics, which contains about 17,000 cancer related terms and their definitions [175]. The NCI Thesaurus is a public domain description logic-based terminology produced by the National Cancer Institute. It is deep and complex compared to most broad clinical vocabularies, implementing rich semantic interrelationships between the nodes of its taxonomies. The semantic relationships in the Thesaurus are intended to facilitate translational research and to support the Bio-Informatics infrastructure of the Institute.

To realise the vision of the Semantic Web, the ontologies of the Semantic Web need to be widely shared and re-used. An example of such re-use as portrayed from Smith et al. [215] states that a user might adopt a date ontology from one source and a physical location ontology from another and then extend the notion of location to include the time period during which it holds. Much of the effort of developing an ontology is devoted to

hooking together classes and properties in ways that maximise implications. Simple assertions about class membership should have broad and useful implications. This is a difficult part of ontology development. If an existing ontology that has already undergone extensive use and refinement can be found, it makes sense to adopt it [158, 215].

A.6.1.5 Description Logics

Any discussion of ontologies or OWL would be incomplete without a section on DL (Description Logics). DL is the logical foundation of all three the OWL sub-languages [215].

To discuss the relationship between OWL and DL, it is necessary to investigate knowledge representation languages. A knowledge representation language is specified when both the syntax and the semantics of the language is described [157]. In the syntax definition the legal statements in the language are defined, and the semantic description specifies each legal statement's intended meaning. The semantics can be formally specified in multiple ways within a logical framework such as with FOL (First Order Logic) or the various Description Logics [11, 84].

The OWL language provides a specific sub-set in the form of OWL DL to provide a language sub-set that has the computational properties necessary for reasoning systems [20, 158, 215]. DL offers a formal foundation for frame-based systems where meaning is provided by interpretations that define the formal semantics of the logic [113] as well as support for some automated reasoning (e.g. class consistency checking) [54, 157].

Where the central modelling primitives of predicate logic are predicates, in frame-based and object-oriented approaches the central modelling primitives are classes (or frames) with certain properties, also referred to as attributes, that do not have a global scope. Description Logics describe knowledge in terms of concepts and role restrictions that are used to automatically derive classification taxonomies. In spite of some of the discouraging theoretical complexity of results with regard to expressing structured knowledge and accessing and reasoning with it in a principled way, there

are now efficient implementations for DL languages [100].

OWL incorporates the essential modelling primitives of frame-based systems, and therefore the formal semantics of Description Logics.

Grau [113] summaries Description Logics as follow:

> *Description logics (DL) are a set of knowledge representation formalisms, whose semantic characterisation is based on standard first-order logics. Meaning is provided by interpretations, which define the formal semantics of the logic. An interpretation in DL is a mathematical structure* $I = \{\Delta^I, .^I\}$ *consisting of:*
>
> - *A nonempty set* Δ^I*, called the domain of the interpretation. The domain is divided into two disjoint sets:*
>   - *The abstract domain is the set of all the individuals*,
>   - *The concrete domain is composed of data values and is used to integrate data-types in description logics*.
> - *An interpretation function* $.^I$ *that maps:*
>   - *Every concept (class) name to a sub-set of* Δ^I,
>   - *Every role (property) name to a sub-set of* $\Delta^I \times \Delta^I$; and
>   - *Every individual to an element of* Δ^I.
>
> *The interpretation function can be extended to complex concepts and roles and can be used to provide meaning to axioms in the knowledge base.*

As the purpose of this section is to provide an overview of the Semantic Web technologies, the reader is referred to one of the most substantial books on the subject, *The Description Logic Handbook, Theory, implementations and application* by Baader et al. [11] for an in depth discussion of DL.

A.6.2 Rules

Rules is not depicted in V1 (Figure A.2). In contrast, Layer 4 is separated as *Ontology* (Layer 4a) and *Rules* (Layer 4b) in V2 (Figure A.3). With the issue of the *OWL Recommendation* by the W3C in 2004, *Rules* was stated as a requirement, and it is plausible to speculate that this is the reason for its inclusion into V2. Rules are defined as executable pieces of declarative knowledge, important in managing complex and dynamic operations [127].

A.6.2.1 SWRL

At present there is a W3C proposal for a Semantic Web Rule Language (SWRL) based on a combination of the OWL DL and OWL Lite sub-languages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sub-languages of the Rule Markup Language [132]. The proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. A high-level abstract syntax is provided that extends the OWL abstract syntax described in the *OWL Semantics and Abstract Syntax* document [192]. An extension of the OWL model-theoretic semantics is also given to provide a formal meaning for OWL ontologies including rules written in this abstract syntax [132, 254].

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold [132]. An OWL ontology in the abstract syntax contains a sequence of axioms and facts. Axioms may be of various kinds, e.g., `subClass` axioms and `equivalentClass` axioms. It is proposed to extend this with rule axioms: `axiom ::= rule`

At present SWRL is a W3C member submission. The W3C Team evaluates this submission in the context of W3C activities and the work is therefore in process [132, 254].

A W3C Workshop on Rule Languages for Interoperability was held during April 2005 to gather data and explore options for establishing a stan-

dard web-based language for expressing rules. More than a dozen use cases were presented for rule language standardisation, and about a half-dozen candidate technologies were presented and discussed. The workshop gave many indications that a W3C Recommendation here would be useful. As a result of the workshop discussions, the Semantic Web Activity group of the W3C established a RIF (Rule Interchange Format) Working Group at the end of 2005 to assist with the establishment of a core rule language plus extensions which together allow rules to be translated between rule languages and thus transferred between rule systems [254].

A.7 LAYER 5

The Semantic Web architectures in Figures A.2 and A.3 depicts *Logic* or *Logic framework* residing above the ontology layer.

A.7.1 Logic/Logic framework

Logic is regarded as the foundation of knowledge representation languages, and it is required to provide the highly expressive language constructs in which knowledge can be captured in a transparent way. A logic framework provides a well-established formal semantics which assigns unambiguous meaning to logical statements. Without a logic framework, inferencing on the Semantic Web will not be possible.

McGuinness et al. [157] defines a knowledge representation language as a language that specifies both the syntax and the semantics of the language. In the syntax definition the legal statements in the language are defined, and the semantic description specifies each legal statement's intended meaning. The semantics can be formally specified in multiple ways with a logical framework such as FOL (First Order Logic) or the various Description Logics [11, 84]. The V1 and V2 Semantic Web architectures in Figures A.2 and A.3 depict *Logic* or *Logic framework* residing above the ontology layer even through *logic* is an aspect included in the OWL Ontology Language through DL formalisms [191]. Therefore, the po-

sitioning of this layer represents the notion that a richer logical language should be provided on top of the *Ontology language*, which provides additional mechanisms for the capturing of reasoning formalisms. Proposals for web logic languages may therefore employ a special semantics, such as minimal model semantics, to make inference more amenable to computer implementation. The technologies that might implement *Logic* or *Logic Framework* are at present largely research efforts by institutions such as the W3C.

A.8 LAYER 6

In the V1 and V2 Semantic Web architectures of Berners-Lee (Figures A.2 and A.3), *Proof* resides on Layer 6.

A.8.1 Proof

Proof as concept exists within the theorem proving domain, for instance as applied in artificial intelligence [197]. To support Semantic Web proof scenarios, *proof languages* were developed. A proof language determines the validity of a specific statement. An instance thereof generally consists of a list of inference items used to derive the information in question, as well as the associate d trust information of each item [7, 184].

A Semantic Web will probably not require proof generation and in general proof validation will be adequate. The search for and generation of a proof for an arbitrary question, is typically an intractable process for many real world problems, and the Semantic Web does not require this to be solved [191]. For perceived Semantic Web applications construction of a proof is performed according to constrained rules, and only the validation thereof is required from other parties. For example, when a user is granted access to a web site, an accompanying document explains to the web server why the user in question should be granted access. Such proof for example, could be a chain of assertions and reasoning rules with pointers to all supporting material [27]

Even though the trust and proof aspects have not really been explored and is considered beyond current research [191], it is a crucial aspect of the Semantic Web. To illustrate the concept with a use case, an example of Palmer [184] is adopted:

> *If one person says that x is blue, and another says that x is not blue, doesn't the whole Semantic Web fall apart? Of course not, because*
>
> *a) applications on the Semantic Web at the moment generally depend upon context; and*
>
> *b) because applications in the future will generally contain proof checking mechanisms, and digital signatures.*

To support scenarios such as the one above, the notion of *proof languages* has to be developed. A proof language is simply a language that is used to prove whether or not a specific statement is true. An instance of a proof language will generally consist of a list of inference *items* used to derive the information in question, and the trust information for each of those items that can then be checked [184].

A.9 LAYER 7

Trust resides on Layer 7 in Figures A.2 and A.3.

A.9.1 Trust

Semantic Web interaction requires different collaborators to communicate, implying that they have to determine how to trust one another, as well as how to establish the trust levels of acquired information [223]. The trust levels of information depend on (1) the source of the information, (2) whether the source can be trusted, as well as (3) whether the source is who it claims to be, in other words, the authenticity and trustworthiness of the source. When dealing with user interactions on the Web, McKnight, Choudhury

and Kacmar [160] define the term *trust* as the belief that another entity is benevolent, competent, honest, or predictable in a given situation. Trust also includes the participants' willingness to depend on one another in a specific interaction.

Within the Semantic Web the concepts trust and proof are dependent on the interaction context. However, an all-encompassing definition of context is problematic [41]. An appropriate meaning of context is therefore explicated by means of the following example:

> A user on the Semantic Web receives data from a friend regarding the best music performances. The data can be *trusted* as it originates from a *known* (implying verified) friend, whose musical interests are familiar. It is thus possible to *use* the data because the user either shares or disagrees with the musical tastes of the friend.

Within the domain of the Semantic Web, *context* therefore assists applications or users regarding the trustworthiness and usefulness of information [41]. *Context* will enable applications or users to decide whether or not received data can be trusted and therefore how to handle information [222].

Trust levels of information might include *user groups* and shared context. If a group or organisation develop a Semantic Web application, then any user's trust of that application, amongst other things, depends upon how much the entire *group* can be trusted. The *information context* would therefore create an conceptual environment where the Semantic Web operate and interact intuitively, without having to rely on complex authentication and checking [184].

A.10 VERTICAL LAYERS

In versions V1 and V2 of the Semantic Web layered architecture of Berners-Lee *Digital Signature* is associated with layers three to six (see Figures A.2 and A.3).

A.10.1 Digital signatures

The DSS (Digital Signature Standard) is a cryptographic standard or a particular application of public key cryptography promulgated by NIST (National Institute of Standards and Technology) [64]. A digital signature is an electronic signature that can be used to authenticate identity. Digital signatures are easily transportable, cannot be imitated, and can be automatically time-stamped. A digital signature can be used with any kind of message, whether it is encrypted or not [27, 42].

XMLDSig (XML Digital Signatures), also called XML Signatures, is a joint IETF/W3C standard that specifies how to digitally sign and verify a signature of a XML data object [17]. XMLDSig enables digital signatures on arbitrary digital content (XML or non-XML) [148]. XML Signatures are digital signatures designed for use in XML transactions [213].

For the Semantic Web a digital signature is a mechanism used to unambiguously verify an identity such as the author of a document [184]. The implementation of digital signatures on the Semantic Web could result in a system which can express and reason about relationships across the whole range of public-key based security and trust systems. It is foreseen that XMLDSig will be used in many phases of semantic knowledge management systems, such as the authenticity verification for retrieved/updated knowledge and involved intermediaries, among others [17].

A.10.2 Encryption

The V1 version of the Semantic Web layered architecture of Berners-Lee (Figure A.2) does not depict *Encryption*. It was however added in version V2 depicted in Figure A.3 where it is associated with layers three to six, along with *Signature*.

Encryption is an effective way to achieve data security. XMLEnc (XML Encryption) is a W3C standard that specifies how to encrypt/decrypt an XML-formatted data object. XMLEnc supports end-to-end (as opposed to point-to-point) encryption of an XML object, which can be the whole or a part of an XML document. The document can be transmitted in XML or

non-XML syntax [139]. indexXMLEnc

On the Semantic Web it is foreseen that encryption would be used in knowledge storage, internal/external knowledge transfer as well as authentication [31, 148]. At present, the implementation of the *Encryption* functionality within the Semantic Web remains largely a research effort [191].

A.11 THE TECHNOLOGIES OF THE LATER VERSIONS OF THE LAYERED ARCHITECTURE

Tim Berners-Lee published additional versions of the architecture in 2005 and 2006, referred to in this thesis as V3 and V4 respectively [34, 35]. A third version (V3) of the Semantic Web architecture was introduced in his keynote presentation at the 2005 World Wide Web Conference [34], and the latest version (V4) of the architecture was presented in a AAAI2006 keynote presentation [35]. These versions are reproduced as Figures A.7 and A.8 with 'Layer' captions corresponding to the first two versions as depicted in Figures A.2 and A.2.

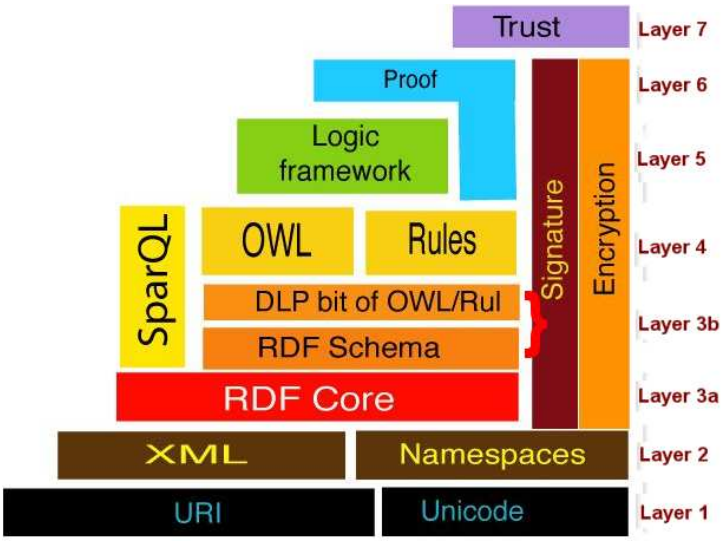


Figure A.7: The V3 version of the Semantic Web architecture [34].

In this section the technologies of the last versions (V3 and V4) of the

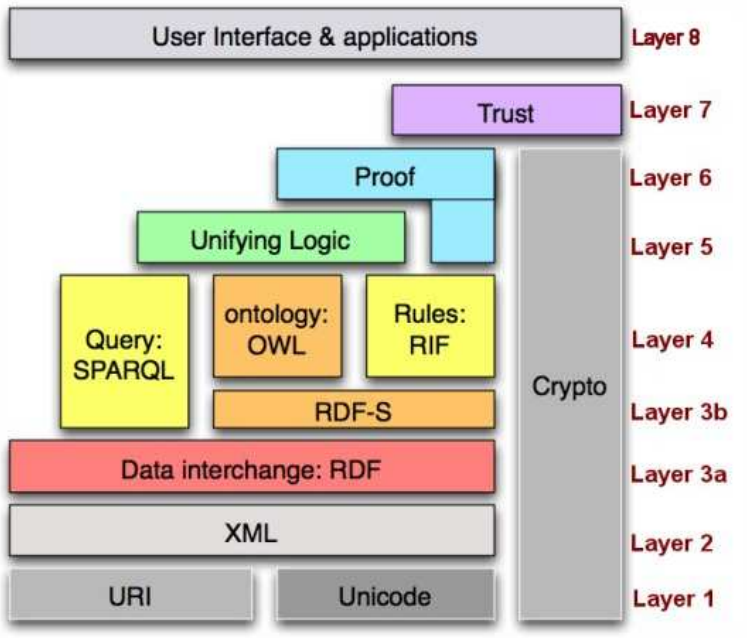


Figure A.8: The V4 version of the Semantic Web architecture [35].

architecture is discussed by relating it to the previous versions V1 and V2 in Figures A.2 and A.3 on pages 288 and 289. In Section A.11.1 the focus is on the general adaptations that appear in versions V3 and V4 when referring to versions V1 and V2 discussed in Section A.2. Section A.11.2 discusses the specific adaptations to Layer 4 in versions V3 and V4 with the addition of *Rules* and *Rules:RIF* and *SPARQL*.

A.11.1 General adaptations

Versions V3 and V4 of the Semantic Web layered architecture proposed by Berners-Lee [34, 35] introduce some noteworthy adaptations, which are discussed in this section. These versions were introduced as part of keynote presentations and thus the specific meaning of the architecture adaptations (as in the case with the previous versions of the architecture) has not been discussed in literature.

Layer 1 in versions V3 and V4 still depicts *URI* and *Unicode*. Layer 2

in V3 in Figure A.7 is similar to previous versions, however, *Namespaces* is omitted from the V4 architecture in Figure A.8. Generally, Namespaces are regarded as included in the XML specifications[10] and it is proposed that the omission is regarded as superficial.

On Layer 3a in both V3 and V4 additional descriptions of RDF are included, namely either *RDF Core* in V3 or *Data interchange: RDF* in V4. However, the specific meaning thereof is unclear. RDF provides a metadata data model for data description. The *Data interchange: RDF* caption in V4 support the functionality description of RDF as mechanism for data interchange on the Semantic Web.

On Layer 3b in both V3 and V4 *RDF Schema* or *RDF-S* still resides above RDF, however *SPARQL* is added. SPARQL provides a mechanism to query RDF data. SPARQL is discussed in more detail in Section A.11.2.1. In V3 Figure A.7 *DLP bit of OWL/Rul* is added as a layer above *RDF Schema* but the intended meaning of this addition is unclear. *DLP bit of OWL/Rul* is discussed in more detail in Section A.11.2.2.

In V3 and V4 on Layer 4, *OWL* and *ontology: OWL* replace *Ontology* and *Rules* or *Rules:RIF* is introduced. *Rules* are therefore moved down from Layer 4b in V2 of Figure A.3 to Layer 4 on the same level as OWL in V3 and V4. OWL is the W3C recommendation that are used for the creation of ontologies and it is discussed in Section A.6.1 on page 327. RIF is an acronym for Rule Interchange Format that is at present a Working Group of the W3C. RIF is discussed further in Section A.11.2.3.

On Layer 5 of both V3 and V4 *Proof* is extended down to above *Rules*, thus residing both above both the *Rules: RIF* layer as well as above the *Unifying Logic* layer. The purpose of this layer seem to be to assist with the integration of different formalisms. In V4 the caption *Unifying Logic* replaces *Logic Framework* but it is plausible to speculate that the intention of the layer remain the same and this adaption is a matter of semantics. *Trust* still resides on Layer 7 in bother V3 and V4. In addition, a layer (Layer 8) is added above Layer 7 in V4 depicting *User Interface and applications*

[10] Refer to Section A.4 on page 292 for a discussion of XML, Namespaces and XML Schema.

that seems to represent the notion that all applications and user interfaces of the Semantic Web will reside above Layer 7. Layer 8 is problematic since it neither depicts technologies nor functionalities required for the realisation of the Semantic Web, but rather where Semantic Web applications would reside, and thus the terminology of this layer deviates from all previous versions.

The vertical layers in Figure A.3 (*Digital Signatures* and *Encryption*) are still depicted in V3 in Figure A.7. However, in V4 (Figure A.8) they have been replaced with a single vertical layer called *Crypto*. What is significant is that the Crypto vertical layer does not start on top of Layer 2 (XML) as in previous versions such as V3, but reside alongside the whole stack. It is however not clear what is the meaning of Crypto and it is possible to speculate that it is a combined layer reflecting the security needs of the Semantic Web architecture.

A.11.2 Layer 4 adaptations

Layer 4 in V3 (Figure A.7) and V4 (Figure A.8) depicts the addition of new technologies to the architecture, namely SPARQL, RIF and *DLP bit of OWL/Rul*. There is however a deviation from the previous versions V1 and V2 of the Semantic Web layered architecture in the introduction of these technologies since none of them are W3C recommendations whilst previously only W3C recommendations were depicted. The remainder of this section discusses SPARQL, RIF and *DLP bit of OWL/Rul*, as well as highlight the present status of these technologies.

A.11.2.1 SPARQL

As mentioned in Section A.5.1, RDF is a data model and is considered to be a flexible and extensible way to represent information about Web resources. It is used to represent diverse information such as personal information or meta-data about any digital artefacts. RDF provide a mechanism to integrate diverse sources of information.

In order to use RDF, a query language is considered to be an urgent requirement [246, 248]. Without such a language it will not be possible to efficiently extract information from an RDF data store. SPARQL has as goal a standardised query language for RDF data. SPARQL Protocol And RDF Query Language is the result of the W3C endeavour to develop a query language and it offers developers and end users a way to write and to consume the results of queries across this wide range of information. Used in conjunction with a common protocol, applications can access and combine information from across the Web [248]. SPARQL is at present a W3C candidate recommendation.

A.11.2.2 *DLP bit of OWL/RuI* layer

DLP bit of OWL/RuI is added as a layer above *RDF Schema* but the intended meaning of this addition is unclear.

DLP (Description Logic Programs) is described as a mechanism to transform any OWL ontology into a (disjunctive) logic program [90]. However, Hitzler, Haase, Krotzsch, Sure and Studer [128] states that an entirely satisfactory definition of DLP is not straightforward. According to them, DLP was originally conceived in [115] as a fragment of OWL DL, but subsequently it has been a source of confusion and controversy.

The addition of the *DLP bit of OWL/RuI* layer into V3 seems to be to incorporate the work of Grosof, Horrocks, Volz and Decker [115] who argue for a mechanism to interoperate, semantically and inferentially, between rules (as RuleML Logic Programs) and ontologies (as OWL/DAML+OIL Description Logic). They define DLP as an intermediate knowledge representation contained within this intersection between rules and ontologies. The intention is for DLP to extend Datalog to include knowledge representation. In this case it is plausible to speculate that, in order for *Rules* to reside above RDF Schema, it was considered to necessary to extract the *DLP bit of OWL/RuI* from OWL and insert this into a layer above RDF Schema.

However, this was since refuted by Horrocks, Parsia, Patel-Schneider and Hendler [130] who argue that (1) Datalog cannot be layered on DLP

due to semantic differences, unless a different semantics for DLP (DLP-Datalog) is adopted, (2) DLP-Datalog cannot be layered on RDF Schema and is not compatible with RDF semantics, and (3) OWL cannot be layered on top of DLP-Datalog.

Datalog is a (function-free) variant of Horn predicate logic which is widely used for deductive databases [177]. It is also described as a query and rule language for deductive databases that, syntactically, is a sub-set of Prolog [65]. Datalog was popular in academic database research but never succeeded in becoming part of a commercial database system. Advantages of Datalog over SQL such as the clean semantics or recursive queries were not sufficient. Datalog is however used in knowledge representation applications, such as KAON-2, a Datalog system that represents an approximation of OWL. In addition, Ontoprise implemented OntoBroker, which is based on a Datalog reasoner [4].

In summary, the *DLP bit of OWL/Rul* layer in the V3 version (Figure A.7) of the architecture introduced Datalog extensions to the architecture underlying OWL [60]. However, Horrocks et al. [130] pointed out that OWL and Datalog are not semantically compatible. This is probably the reason for the removal of this layer in the subsequent V4 version (Figure A.8).

A.11.2.3 RIF

The Semantic Web Activity group of the W3C established a RIF (Rule Interchange Format) Working Group at the end of 2005 in order to specify a core rule language with extensions, which together, allow rules to be translated between rule languages and thus transferred between rule systems [255].

Rule-languages and rule-based systems have been used in Computer Science and Information Technology in several types of applications such as expert systems or deductive databases. Automated inferencing based on symbolic representations has a rich history and continues to be a key technology driver [43, 127]. Due to the innovations prevalent in the Semantic Web domain, there is now a need for research in this technology area

[255]. Some issues pertaining to automated inferencing will require further research, but others can be addressed by enabling existing rule-based technologies to interoperate according to standards-based methodologies and processes. This is the basic goal of RIF (the Rule Interchange Format) [247]. The RIF Working Group attempts to devise the required standards that are useful in the present situation as well as being easily extensible in order to deal with the evolution of rule technology and other enabling technologies [255].

The RIF Working Group published a new Public Working Draft of RIF called *RIF Use Cases and Requirements* [247]. This version includes requirements in addition to use cases, as well as goals and a rough cut of the space of rule systems likely to be addressed in Phase 1. RIF is at present a W3C member submission as working draft and the W3C Team is evaluating this submission within the context of W3C activities. The W3C has not endorsed the submission of RIF yet.

A.12 CONCLUSION

At present, the information overload experienced by information technology users, specifically on the Web, necessitates the introduction of automated information management functionality, one realisation of which constitutes the envisioned Semantic Web. In addition, the Semantic Web and its associated technologies are permeating various fields and domains within the ICT (Information and Communications Technology) domain. In order to understand the nature and impact of these technologies, specifically with regard to the envisioned Semantic Web by Berners-Lee and the relation with the proposed versions of his architecture, it is necessary to discuss these technologies and concepts. In this report, the research provided a starting point to assimilate Semantic Web terminology and associated concepts. In addition, the information about the technologies presented in this report serves as a foundation to any subsequent discussions of the different versions of the Semantic Web layered architecture.

References

[1] ABMANN U. **Composing Frameworks and Components for Families of Semantic Web Applications**. *Lecture Notes in Computer Science*, vol. 29(01) **2003**: pp. 1–15.

[2] ABUGESSAISA I.E.A. AND SIVERTUN A. **Ontological Approach to Modeling Information Systems**. In *Proceedings of the Fourth International Conference on Computer and Information Technology (CIT0*, vol. 0-7695-2216-5/04. IEEE **2004**.

[3] ALEXANDER C., ISHIKAWA S. AND SILVERSTEIN M. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA **1977**. ISBN 0195019199.

[4] ALLEMANG D. **S is for Semantics**. Dean Allemang's Blog **2005**. URL: `http://dallemang.typepad.com/my_weblog/2005/11/dl_vs_dlp.html`. Last accessed 12/8/2006.

[5] ANDROUTSELLIS-THEOTOKIS S. AND SPINELLIS D. **A Survey of Peer-to-Peer Content Distribution Technologies**. *ACM Computing Surveys*, vol. 36(4) **2004**: pp. 335–371. ISSN 0360-0300. doi: 10.1145/1041680.1041681.

[6] ANTONIOU G. AND VON HARMELEN F. *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, London, England **2004**. ISBN 0-262-01210-3.

[7] APPEL A.W. AND FELTEN E.W. **Proof-carrying authentication**. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*. ACM Press, New York, NY, USA **1999**.

ISBN 1-58113-148-8, pp. 52–62. doi:http://0-doi.acm.org.oasis.unisa.ac.za:80/10.1145/319709.319718.

[8] ASHRI R., PAYNE T., MARVIN D., SURRIDGE M. AND TAYLOR S. **Towards a Semantic Web Security Infrastructure**. In *Proceedings of Semantic Web Services, 2004* **2004**.

[9] AVISON D. AND FITZGERALD G. *Information Systems Development: Methodologies, Techniques and Tools*. ISBN: 0-07-709233-3. McGraw-Hill, UK **1997**.

[10] AVISON D. AND FITZGERALD G. *Information Systems Development: Methodologies, Techniques and Tools.* 3rd edition. McGraw-Hill, UK **2003**.

[11] BAADER F., CALVANESE D., MCGUINNESS D.L., NARDI D. AND PATEL-SCHNEIDER P.F., eds. *The Description Logic Handbook, Theory, implementations and application*. Cambridge University Press, London, UK **2003**.

[12] BACHMAN C. **Personal Chronicle: Creating Better Information Systems, with Some Guiding Principles**. *IEEE Transactions on Knowledge and Data Engineering* **1989**: pp. 17–32.

[13] BACHMANN F., BASS L., CARRIERE J., CLEMENTS P., GARLAN D., IVERS J., NORD R. AND LITTLE R. **Software Architecture Documentation in Practice: Documenting Architectural Layers**. *Technical Report CMU/SEI-2000-SR-004*, Carnegie Mellon Software Engineering Institute **2005**.

[14] BAHRAMI A. *Object Oriented Systems Development Using the Unified Modeling Language*. Irwin McGraw-Hill **1999**. ISBN 0-256-25348-X.

[15] BALAKRISHNAN H., LAKSHMINARAYANAN K., RATNASAMY S., SHENKER S., STOICA I. AND WALFISH M. **A layered naming architecture for the internet**. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and*

protocols for computer communications. ACM Press, New York, NY, USA **2004**. ISBN 1-58113-862-8, pp. 343–352. doi:http://0-doi.acm.org.oasis.unisa.ac.za:80/10.1145/1015467.1015505.

[16] BALDONI M., BAROGLIO C. AND HENZE N. **Personalization for the Semantic Web**. *Lecture Notes in Computer Science: Reasoning Web: First International Summer School 2005*, vol. 3564 / 2005 **2005**: p. 173. ISSN 0302-9743. doi:10.1007/11526988_5.

[17] BARTEL M., BOYER J., FOX B., LAMACCHIA B. AND SIMON E. **XML-Signature Syntax and Processing**. W3C Web site **2002**. URL: `http://www.w3.org/TR/xmldsig-core/`. Last accessed 13/7/2006.

[18] BASS L., CLEMENTS P. AND KAZMAN R. *Software Architecture in Practice*. Addison Wesley Professional **2003**. ISBN 0-321-15495-9. URL: `http://www.aw-bc.com/catalog/academic/product/0,4096,0321154959,00.html`. Last accessed 12/8/2006.

[19] BECHHOFER S. **Standardisation Report: Ontology Language Standardisation Efforts**. Web Document **2004**. URL: `http://wonderweb.semanticweb.org/deliverables/documents/D4.pdf`. Last accessed 8/3/2006.

[20] BECHHOFER S., VAN HARMELEN F., HENDLER J., HORROCKS I., MCGUINNESS D.L., PATEL-SCHNEIDER P.F. AND STEIN L.A. **OWL Web Ontology Language Reference**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`. Last accessed 15/3/2005.

[21] BECKETT D. **Dave Beckett's Resource Description Framework (RDF) Resource Guide**. Web Document **2004**. URL: `http://www.ilrt.bris.ac.uk/discovery/rdf/resources/`. Accessed 13/3/2006.

[22] BECKETT D. **RDF/XML Syntax Specification**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`. Last accessed 27/10/2006.

[23] BELLINGER G., CASTRO D. AND MILLS A. **Data, Information, Knowledge, and Wisdom**. Systems Thinking.org Web site **2004**. URL: http://www.systems-thinking.org/dikw/dikw.htm. Last accessed 12/10/2006.

[24] BERGEY J.K., FISHER M.J. AND JONES L.G. **Use of the Architecture Tradeoff Analysis MethodSM (ATAMSM) in Source Selection of Software-Intensive Systems**. *Technical Report CMU/SEI-2002-TN-010*, Software Engineering Institute, Carnegie Mellon University. **2002**. URL: ftp://ftp.sei.cmu.edu/pub/documents/02.reports/pdf/02tn010.pdf. Last accessed 2/2/2006.

[25] BERNERS-LEE T. **Axioms of Web Architecture: Metadata Architecture**. W3C Web site **1997**. URL: http://www.w3.org/DesignIssues/Metadata.html. Last accessed 2/10/2006.

[26] BERNERS-LEE T. **Frequently asked questions**. W3C Web site **1998**. URL: http://www.w3.org/People/Berners-Lee/FAQ.html. Last accessed 2/10/2006.

[27] BERNERS-LEE T. **The Semantic Web Road Map**. W3C Web site **1998**. URL: http://www.w3.org/DesignIssues/Semantic.html. Last accessed 12/8/2006.

[28] BERNERS-LEE T. **Semantic Web - XML2000**. W3C Web site **2000**. URL: http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html. Last accessed 11/8/2006.

[29] BERNERS-LEE T. **Axioms of Web Architecture: Principles of design**. W3C Web site **2002**. URL: http://www.w3.org/DesignIssues/Principles.html. Last accessed 5/10/2006.

[30] BERNERS-LEE T. **Web Architecture from 50,000 feet**. W3C Website **2002**. URL: http://www.w3.org/DesignIssues/Architecture.html. Last accessed 4/10/2006.

[31] BERNERS-LEE T. **The Semantic Web and Challenges**. W3C Web site Slideshow **2003**. URL: http://www.w3.org/2003/Talks/01-sweb-tbl/Overview.html. Last accessed 10/10/2006.

[32] BERNERS-LEE T. **Standards, Semantics and Survival**. *SIIA Upgrade* **2003**: pp. 6–10.

[33] BERNERS-LEE T. **WWW Past and Future**. W3C Web site **2003**. URL: `http://www.w3.org/2003/Talks/0922-rsoc-tbl/slide30-0.html`. Last accessed 10/10/2006.

[34] BERNERS-LEE T. **WWW2005 Keynote**. W3C Web site **2005**. URL: `http://www.w3.org/2005/Talks/0511-keynote-tbl/`. Last accessed 10/10/2006.

[35] BERNERS-LEE T. **Artificial Intelligence and the Semantic Web: AAAI2006 Keynote**. W3C Web site **2006**. URL: `http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html`. Last accessed 12/10/2006.

[36] BERNERS-LEE T., BRAY T., CONNOLLY D., COTTON P., FIELDING R. ET AL. **Architecture of the World Wide Web, Volume One**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-webarch-20041215/`.

[37] BERNERS-LEE T., FIELDING R., IRVINE U.C. AND MASINTER L. **IETF RFC3986 - Uniform Resource Identifiers (URI): Generic Syntax**. Web Document **2005**. URL: `http://www.rfc-archive.org/getrfc.php?rfc=3986`. Last accessed 24/8/2006.

[38] BERNERS-LEE T., HENDLER J. AND LASSILA O. **The Semantic Web**. *The Scientific American*, vol. 5(1) **2001**. URL: `http://www.scientificamerican.com/2001/0501issue/0501berbers-lee.html`. Last accessed 20/9/2006.

[39] BERNERS-LEE T. AND MASINTER L. **IETF RFC1738 - Uniform Resource Locators (URL): Generic Syntax**. Web Document **1994**. URL: `http://www.rfc-archive.org/getrfc.php?rfc=1738`. Last accessed 24/8/2006.

[40] BETTELS J. AND BISHOP F. **Unicode: A Universal Character Code**. *Digital Technical Journal*, vol. 5 no. 3(3) **1993**.

[41] BHARGAVA B., LILIEN L., ROSENTHAL A., WINSLETT M., SLOMAN M. ET AL. **The Pudding of Trust**. *IEEE Intelligent Systems*, vol. 1541-1672/04 **2004**: pp. 74–88.

[42] BIDDLE C.B. **Legislating Market Winners: Digital Signature Laws and the Electronic Commerce Marketplace**. *World Wide Web Journal (W3J)*, vol. 1 **1996**. URL: `http://www.w3journal.com/7/s3.biddle.wrap.html`. Last accessed 12/8/2006.

[43] BOLEY H., TABET S. AND WAGNER G. **Design Rationale of RuleML: A Markup Language for Semantic Web Rules 2001**. URL: `citeseer.ist.psu.edu/boley01design.html`. Last accessed 3/7/2006.

[44] BOS B. **What is a good standard? An essay on W3C's design principles**. W3C Web site **2003**. URL: `http://www.w3.org/People/Bos/DesignGuide/introduction`. Last accessed 2/10/2006.

[45] BOSCH J. **Product-line architectures in industry: a case study**. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*. IEEE Computer Society Press, Los Alamitos, CA, USA **1999**. ISBN 1-58113-074-0, pp. 544–554.

[46] BRAY T., HOLLANDER D. AND LAYMAN A. **Namespaces in XML Recommendation - REC-xml-names-19990114**. W3C Web site **1999**. URL: `http://www.w3.org/TR/REC-xml-names/`. Last accessed 8/8/2006.

[47] BRAY T., HOLLANDER D., LAYMAN A. AND TOBIN R. **Namespaces in XML 1.1: W3C Recommendation**. W3C Web site **2004**. URL: `http://www.w3.org/TR/xml-names11/`. Last accessed 12/8/2006.

[48] BRAY T., HOLLANDER D., LAYMAN A. AND TOBIN R. **Namespaces in XML 1.0 (Second Edition)**. W3C Web site **2006**. URL: `http://www.w3.org/TR/REC-xml-names/`. Accessed 1/10/2006.

[49] BRAY T., PAOLI J., SPERBERG-MCQUEEN C.M., MALER E. AND YERGEAU F. **Extensible Markup Language (XML) 1.0 (Third Edi-**

tion). W3C Web site **2004**. URL: http://www.w3.org/TR/REC-xml/. Last accessed 16/9/2006.

[50] BRAY T., PAOLI J., SPERBERG-MCQUEEN C.M., MALER E. AND YERGEAU F. **Extensible Markup Language (XML) 1.0 (Fourth Edition)**. W3C Web site **2006**. URL: http://www.w3.org/TR/2006/REC-xml-20060816/. Last accessed 20/12/2006.

[51] BREDEMEYER. **Software Architecture, Software Architects and Architecting**. Web site **1999-2005**. URL: http://www.bredemeyer.com/. Last accessed 17/6/2006.

[52] BRICKLEY D. AND GUHA R. **RDF Vocabulary Description Language 1.0: RDF Schema**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Last accessed 27/10/2006.

[53] BROEKSTRA J., KAMPMAN A. AND VAN HARMELEN F. **Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema**. In *Proceedings of The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy*, vol. Volume 2342 / 2002 **2002**, p. 54.

[54] BROEKSTRA J., KLEIN M., DECKER S., FENSEL D., VAN HARMELEN F. AND HORROCKS I. **Enabling Knowledge Representation on the Web by Extending RDF Schema**. In *Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong*, vol. ACM 1-58113-348-0/01/0005 **2001**, p. 467. URL: http://www10.org/cdrom/papers/291/. Last accessed 18/3/2006.

[55] BROOKS F.P. *The Mythical Man-month: Essays on Software Engineering*. 3rd edition. Addison-Wesley Publishing Company, Reading, Massachusetts **1979**. ISBN 0-201-00650-2.

[56] BRUEGGE B. AND DUTOIT A.H. *Object-oriented Software Engineering using UML, Patterns, and Java*. 2nd edition. Prentice-Hall **2004**. ISBN 0-13-0471100.

[57] BURRELL G. AND MORGAN G. *Sociological Paradigms and Organisational Analysis*. Heinemann, London **1979**. ISBN 1-857421140.

[58] BUSSLER C., FENSEL D. AND MAEDCHE A. **A conceptual architecture for semantic web enabled web services**. *ACM SIGMOD, SPECIAL ISSUE: Special section on semantic web and data management*, vol. 31, issue 4 **2002**: pp. 24 – 29.

[59] BUTLER M.H. **Barriers to real world adoption of semantic web technologies**. *HP Labs Technical Report*, vol. HPL-2002-333 **2002**. URL: `http://www.hpl.hp.com`. Last accessed 15/3/2006.

[60] CALVANESE D., GIACOMO G.D., LENZERINI M., NARDI D. AND ROSATI R. **Description Logic Framework for Information Integration**. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR'98),* **1998**, pp. 2–13.

[61] CAREY P. *New Perspectives on XML - Introductory*. ISBN: 0-619-10187-3. Thomson Course Technology **2003**. ISBN 0-619-10187-3.

[62] CARROLL J.J., BIZER C., HAYES P. AND STICKLER P. **Named graphs**. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3(4) **2005**: pp. 243–366.

[63] CARROLL J.J. AND ROO J.D. **OWL Web Ontology Language Test Cases**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-owl-test-20040210/`. Last accessed 15/3/2006.

[64] CENTER E.P.I. **Digital Signatures**. EPIC Web site **2005**. URL: `http://www.epic.org/crypto/dss/`. Last accessed 15/3/2006.

[65] CERI S., GOTTLOB G. AND TANCA L. **What You Always Wanted to Know About Datalog (And Never Dared to Ask)**. *IEEE Transactions on Knowledge and Data Engineering*, vol. 01(1) **1989**: pp. 146–166. ISSN 1041-4347. doi:http://doi.ieeecomputersociety.org/10.1109/69.43410.

[66] CHEN H., PERICH F., CHAKRABORTY D., FININ T. AND JOSHI A. **Intelligent Agents Meet Semantic Web in a Smart Meeting Room**. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), New York, USA*, vol. 2. ACM **2004**, pp. 854–861.

[67] CHEN P.P.S. **The entity-relationship model: toward a unified view of data**. *ACM Transactions on Database Systems (TODS).*, vol. 1(1) **1976**: pp. 9–36. ISSN 0362-5915. doi:http://doi.acm.org/10.1145/320434.320440.

[68] CHERRY S.M. **Weaving A Web of Ideas**. *IEEE Spectrum, Special RD Report*, vol. 39 **2002**: pp. 65–69.

[69] CLARKE R. **Peer-to-Peer (P2P) - An Overview**. Webpage: Xamax Consultancy Pty Ltd **2004**. URL: `http://www.anu.edu.au/people/Roger.Clarke/EC/P2POview.html`. Last accessed 5/6/2006.

[70] CONFOTO. Website URL: `http://www.confoto.org/`. Last accessed 23/8/2006.

[71] CONSORTIUM U. **The Unicode Consortium**. Unicode Web site **2004**. URL: `http://www.unicode.org/`. Last accessed 23/9/2006.

[72] CONSORTIUM U. **The Unicode Standard**. Unicode Web site **2006**. URL: `http://www.unicode.org/versions/latest/`. Last accessed 23/9/2006. The versions of Unicode are available at http://www.unicode.org/standard/versions/.

[73] CORAZZON R. **Ontology. A Resource Guide for Philosophers.** Web site **2006**. URL: `http://www.formalontology.it/index.htm`. Last accessed 14/8/2006.

[74] COST R.S., FININ T., JOSHI A., PENG Y., NICHOLAS C. ET AL. **ITtalks: A Case Study in the Semantic Web and DAML+OIL**. *IEEE Intelligent Systems*, vol. 17(1) **2002**: pp. 40–47. doi:http://dx.doi.org/10.1109/5254.988447.

[75] CRONJE P.J. **Presentation: Research Indaba at Meraka Institute**. Johannes Cronje's Web site **2006**. URL: http://hagar.up.ac.za/catts/abchome.html. Last accessed 13/8/2006.

[76] CURRAN C.J. AND DAIGLE L. **Uniform Resource Names (urn)**. IETF Web site **2001**. URL: http://www.ietf.org/html.charters/OLD/urn-charter.html. Last accessed 15/9/2006.

[77] DA SILVA P.P., MCGUINNESS D.L. AND FIKES R.E. **A Proof Markup Language for Semantic Web Services.** *Technical report*, Technical Report KSL-04-01. Knowledge Systems Laboratory, Stanford University. **2004**. URL: http://xml.coverpages.org/ni2004-01-16-a.html.

[78] DACST. **The White Paper on Science and Technology, Department of Arts, Culture, Science and Technology**. DACST Web site **1996**. URL: http://www.dacst.gov.za/.

[79] DAML. **DARPA Agent Markup Language 2005**. URL: http://www.daml.org/. Last accessed 21/5/2006.

[80] DARPA. **Defence Advanced Research Projects Agency**. DARPA Web site **2004**. URL: http://www.darpa.mil/. Last accessed 28/4/2006.

[81] DAVIES J., FENSEL D. AND VAN HARMELEN F. *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley and Sons, West Sussex, England **2003**. ISBN 0470-84867-7.

[82] DCMI. **Dublin Core Metadata Initiative (DCMI) 2005**. URL: http://dublincore.org/. Last accessed 15/3/2006.

[83] DE BRUIJN J., LARA R., POLLERES A. AND FENSEL D. **OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web**. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*. ACM Press, New York, NY, USA **2005**. ISBN 1-59593-046-9, pp. 623–632. doi:http://doi.acm.org/10.1145/1060745.1060836.

[84] DECKER S., MELNIK S., VAN HARMELEN F., FENSEL D., KLEIN M., BROEKSTRA J., ERDMANN M. AND HORROCKS I. **The Semantic Web: The Roles of XML and RDF**. *IEEE Internet Computing*, vol. 4 **2000**: pp. 63–74.

[85] DECKER S., MITRA P. AND MELNIK S. **Framework for the Semantic Web: An RDF Tutorial**. *IEEE Internet Computing*, vol. 4(6) **2000**: pp. 68–73.

[86] DECKER S., VAN HARMELEN F., BROEKSTRA J., ERDMANN M., FENSEL D., HORROCKS I., KLEIN M. AND MELNIK S. **The Semantic Web - on the respective Roles of XML and RDF 2000**.

[87] DEROSE K. **What Is Epistemology? A Brief Introduction to the Topic**. Web site **2004**. URL: `http://pantheon.yale.edu/%7Ekd47/What-Is-Epistemology.htm`.

[88] DIJKSTRA E.W. **The structure of the THE-multiprogramming system**. *Commun. ACM*, vol. 11(5) **1968**: pp. 341–346. ISSN 0001-0782. doi:http://doi.acm.org/10.1145/363095.363143.

[89] DIJKSTRA E.W. **The end of computing science?** *Communications of the ACM*, vol. 44(3) **2001**: p. 92. ISSN 0001-0782. doi:http://doi.acm.org/10.1145/365181.365217.

[90] DLP. **DLP - Description Logic Programs**. KAON Web site URL: `http://kaon.semanticweb.org/alphaworld/dlp`. Last accessed 1/11/2006.

[91] DO S.H. AND SUH N.P. **Systematic OO Programming with Axiomatic Design**. *IEEE Computer* **1999**: pp. 121–124.

[92] DONG J.S., LEE C.H., LEE H.B., LI Y.F. AND WANG H. **Semantic web foundations: A combined approach to checking web ontologies**. In *Proceedings of the 13th International Conference on World Wide Web*, ISBN:1-58113-844-X. ACM Press, New York, NY, USA **2004**, pp. 714 – 722.

[93] DORI D. **ViSWeb - The Visual Semantic Web: Unifying Human and Machine Knowledge Representations with Object-Process Methodology**. *The VLDB Journal*, vol. 13(2) **2004**: pp. 120–147. ISSN 1066-8888. doi:http://dx.doi.org/10.1007/s00778-004-0120-x.

[94] DST. **The Department of Science and Technology**. Web site . URL: `http://www.dst.gov.za/`. Last accessed 10/11/2006.

[95] DST. **South Africa's National Research and Development Strategy, Department of Science and Technology**. DST Web site **2002**. URL: `http://www.dst.gov.za/publications/reports/sa_nat_rd_strat.pdf`. Last accessed 10/11/2006.

[96] DTD. **DTD - Document Type Definition**. W3C Website URL: `http://www.w3.org/TR/html4/sgml/dtd.html`. Last accessed 31/10/2006.

[97] ESWC06. **The 3rd European Semantic Web Conference**. Web site **2006**. URL: `http://www.wswc2006.org/`.

[98] EUZENAT J. AND NAPOLI A. **The Semantic Web: Year One**. *IEEE Intelligent Systems*, vol. 1094-7167(3) **2003**.

[99] FENSEL D. **The Semantic Web and its Languages**. *IEEE Intelligent Systems*, vol. 15(6) **2000**: pp. 67–73.

[100] FENSEL D. **Language Standardization for the Semantic Web: The Long Way from OIL to OWL**. In *Distributed Communities on the Web: 4th International Workshop, DCW 2002, Sydney, Australia*, vol. 2468 / 2002 **2002**, pp. 215–227.

[101] FENSEL D. **Ontology-Based Knowledge Management**. *IEEE Computer*, vol. 35 no 11 **2002**: pp. 56–59.

[102] FENSEL D., VAN HARMELEN F., HORROCKS I., MCGUINNESS D.L. AND PATEL-SCHNEIDER P.F. **OIL: An Ontology Infrastructure for the Semantic Web**. *IEEE Intelligent Systems*, vol. 1094-7167/01 **2001**: pp. 38–45.

[103] Flink. Website URL: http://prauw.cs.vu.nl:8080/flink/. Last accessed 23/9/2006.

[104] FOWLER M. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA, USA **2003**. ISBN 0-321-12742-0.

[105] GARLAN D. AND SHAW M. **An Introduction to Software Architecture 1994**. URL: http://www.inf.ed.ac.uk/teaching/courses/seoc1/2005_2006/resources/intro_softarch.pdf. Last accessed 15/9/2006.

[106] GEIGER K. *Inside ODBC*. Microsoft Press Redmond, WA, USA **1995**.

[107] GERBER A., BARNARD A. AND VAN DER MERWE A. **Design and Evaluation Criteria for Layered Architectures**. In *Proceedings of the MSVVEIS Workshop hosted at the 8th International Conference on Enterprise Informatiopn Systems, Paphos, Cyprus* **2006**. ISBN 972-8865-49-8, pp. 163–172.

[108] GERBER A., BARNARD A. AND VAN DER MERWE A. **A Semantic Web Status Model;**. In *Proceedings of the Ninth World Conference on Integrated Design & Process Technology, San Diego, California*. IEEE **2006**. ISSN 1090-9389.

[109] GERBER A., BARNARD A. AND VAN DER MERWE A. **Semantic Web Technologies**. *Technical Report UNISA-TR-2006-02*, UNISA (University of South Africa) **2006**. URL: http://www.osprey.ac.za/TechnicalReports/.

[110] GERBER A., VAN DER MERWE A. AND BARNARD A. **Towards a Semantic Web Layered Architecture**. In *Proceedings of IASTED International Conference on Software Engineering, (SE2007), Innsbruck, Austria*. IASTED **2007**. ISBN 978-0-88986-641-6, pp. 353–362.

[111] Google. Website. URL: http://www.google.com. Last accessed 26/9/2006.

[112] Google Scholar. Website. URL: http://scholar.google.com. Last accessed 26/9/2006.

[113] GRAU B.C. **A Possible Simplification of the Semantic Web Architecture**. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM Press, New York, NJ, USA **2004**. ISBN 1-58113-844-X, pp. 704–713. doi:http://doi.acm.org/10.1145/988672.988769.

[114] GROSOF B. **A Roadmap for Rules and RuleML in the Semantic Web**. *Technical report*, MIT Sloan School of Management, Cambridge, MA, USA **2003**. URL: http://ebusiness.mit.edu/bgrosof/paps/ruleml-ieee-intell-sys-2003.pdf.

[115] GROSOF B.N., HORROCKS I., VOLZ R. AND DECKER S. **Description Logic Programs: Combining Logic Programs with Description Logic**. In *Proceedings of Twelfth International World Wide Web, WWW2003*, ACM 1-58113-680-3/03/0005. **2003**.

[116] GUHA R., MCCOOL R. AND FIKES R. **Contexts for the Semantic Web**. *Lecture Notes in Computer Science: The Semantic Web ISWC 2004*, vol. 3298/2004 **2004**: pp. 32–46. doi:10.1007/b102467.

[117] HALLBERG B. *Networking: A Beginner's Guide.* 2nd edition. McGraw-Hill, Berkeley, California, USA **2001**. ISBN 0-07-213231-0.

[118] HAUBEN M. **A History of ARPANET. Behind the Net - The untold history of the ARPANET**. Web document **2004**. URL: http://www.dei.isep.ipp.pt/docs/arpa.html. Last accessed 15/9/2006.

[119] HAYES P. **RDF Semantics**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-mt-20040210/. Last accessed 9/9/2006.

[120] HEFLIN J. **OWL Web Ontology Language, Use Cases and Requirements**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-webont-req-20040210/. Last accessed 9/9/2006.

[121] HEFLIN J. AND HENDLER J. **A Portrait of the Semantic Web in Action**. *IEEE Intelligent Systems*, vol. 16 **2001**: pp. 54–59.

[122] HEFLIN J.D. *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment*. Ph.D. thesis, Faculty of the Graduate School of the University of Maryland, College Park **2001**.

[123] HENDLER J. **Agents and the Semantic Web**. *IEEE Intelligent Systems*, vol. 16 **2001**: pp. 30–37.

[124] HENDLER J. **Agents and the Semantic Web**. *IEEE Intelligent Systems*, vol. 16 **2001**: pp. 30–37.

[125] HESS A., HOLT J., JACOBSON J. AND SEAMONS K.E. **Content-triggered trust negotiation**. *ACM Transactions on Information and System Security (TISSEC)*, vol. 7(3) **2004**: pp. 428–456.

[126] HGARET P.L. **The Architecture Domain**. W3C Web site **2006**. URL: `http://www.w3.org/Architecture/`. Last accessed 4/10/2006.

[127] HITZLER P., ANGELE J., MOTIK B. AND STUDER R. **Bridging the Paradigm Gap with Rules for OWL**. In *In Proceedings of the W3C Workshop on Rule Languages for Interoperability* **2005**. URL: `http://www.w3.org/2004/12/rules-ws/`. Last accessed 9/9/2006.

[128] HITZLER P., HAASE P., KROTZSCH M., SURE Y. AND STUDER R. **DLP isnt so bad after all**. In *OWL Workshop* **2005**. URL: `http://www.mindswap.org/2005/OWLWorkshop/sub2.pdf`. Last accessed 1/11/2006.

[129] HOFMANN C., HORN E., KELLER W., RENZEL K. AND SCHMIDT M. **The Field of Software Architecture**. *Technical Report TUM-I9641*, Faculty of Mathematics and the Institute for Informatics at the Technical University of Munich. **1996**.

[130] HORROCKS I., PARSIA B., PATEL-SCHNEIDER P. AND HENDLER J. **Semantic Web Architecture: Stack or Two Towers?** *Lecture Notes in Computer Science: Principles and Practice of Semantic Web Reasoning: Third International Workshop*, vol. 3703 / 2005 **2005**. ISSN 0302-9743. doi:10.1007/11552222.

[131] HORROCKS I. AND PATEL-SCHNEIDER P.F. **Three theses of representation in the semantic web**. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*. ACM Press, New York, NY, USA **2003**. ISBN 1-58113-680-3, pp. 39–47. doi: http://0-doi.acm.org.oasis.unisa.ac.za:80/10.1145/775152.775159.

[132] HORROCKS I., PATEL-SCHNEIDER P.F., BOLEY H., TABET S., GROSOF B. AND DEAN M. **SWRL: A Semantic Web Rule Language Combining OWL and RuleML**. W3C Web site **2004**. URL: http://www.w3.org/Submission/SWRL/. Last accessed 9/9/2006.

[133] HORROCKS I., PATEL-SCHNEIDER P.F. AND VAN HARMELEN F. **From SHIQ and RDF to OWL: The Making of a Web Ontology Language**. *Journal of Web Semantics* **2003**. URL: http://www.cs.man.ac.uk/~horrocks/Publications/download/2003/HoPH03a.pdf. Last accessed 9/9/2006.

[134] HUANG W. AND WEBSTER D. **Enabling Context-Aware Agents to Understand Semantic Resources on The WWW and The Semantic Web**. In *Web Intelligence, IEEE/WIC/ACM International Conference on (WI'04)* **2004**, pp. 138–144.

[135] ICCC. **International Conference on Computer Communication**. ICCC Web site **1972**. URL: http://www.icccgovernors.org/. Last accessed 9/9/2006.

[136] IETF. **Overview of Internet Standards (STD) in Numeric Order**. IETF Web site **2003–2006**. URL: http://www.rfc-archive.org/standards.php. Last accessed 9/9/2006.

[137] IETF. **Internet Engineering Task Force**. Web Document **2004**. URL: http://www.ietf.org/. Last accessed 9/9/2006.

[138] IETF Standards. **Overview of IETF Standards in numeric order with their statuses.** IETF Web site URL: http://www.rfc-archive.org/standards.php. Last accessed 7/7/2006.

[139] IMAMURA T., DILLAWAY B. AND SIMON E. **XML Encryption Syntax**

and Processing. W3C Web site **2002**. URL: http://www.w3.org/TR/xmlenc-core/. Last accessed 9/9/2006.

[140] IRI. **IRI**. W3C Web site URL: http://www.w3.org/International/iri-edit/. Last accessed 23/7/2006.

[141] ISWC06. **The 5th International Semantic Web Conference**. Web site. **2006**. URL: http://iswc2006.semanticweb.org/.

[142] JACOBSON I., BOOCH G. AND RUMBAUGH J. *The Unified Software Development Process*. Addison-Wesley **1999**.

[143] JECKLE M. AND WILDE E. **Identical Principles, Higher Layers: Modeling Web Services as Protocol Stack**. In *XML Europe 2004, Amsterdam* **2004**. URL: http://dret.net/netdret/docs/wilde-xmleurope2004.pdf. Last accessed 9/9/2006.

[144] JUN-FENG S., WEI-MING Z., WEI-DONG X., GUO-HUI L. AND ZHENNING X. **Ontology-Based Information Retrieval Model for the Semantic Web**. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service* **2005**. ISBN 0-7695-2274-2, pp. 152–155.

[145] KALFOGLOU Y., ALANI H., SCHORLEMMER M. AND WALTON C. **On the Emergent Semantic Web and Overlooked Issues**. *Lecture Notes in Computer Science: The Semantic Web ISWC 2004*, vol. 3298/2004 **2004**: pp. 576–590.

[146] KESSLER G.C. **An Overview of TCP/IP Protocols and the Internet**. Paper originally submitted to the InterNIC and posted on their Gopher site on 5 Aug. 1994. Current document is a continually updated version of original paper. **2004**. URL: http://www.garykessler.net/library/tcpip.html#arch. Last accessed 9/9/2006.

[147] KIFER M., BRUIJN J., BOLEY H. AND FENSEL D. **A Realistic Architecture for the Semantic Web**. *Lecture Notes in Computer Science*, vol. 3791 **2005**: pp. 17 – 29. doi:DOI10.100711580072_3.

[148] LEE J., UPADHYAYA S.J., RAO H.R. AND SHARMAN R. **The semantic e-business vision: Secure knowledge management and the semantic web**. *Communications of the ACM*, vol. 48 **2005**.

[149] LING R. **'One can talk about common manners': the Use of mobile Telephones in Inappropriate Situations**. *Technical report*, Themes in mobile telephony: Final report of the COST 248 Home and Work Group. **1997**.

[150] LIPPITT G.L. *Visualizing Change: Model Building and the Change Process*. University Associates, Inc. **1973**. ISBN 0-88390-125-0.

[151] LU S., DONG M. AND FOTOUHI F. **The Semantic Web: opportunities and challenges for next-generation Web applications**. *Information Research*, vol. 7(4) **2002**. URL: `http://informationr.net/ir/7-4/paper134.html`. Last accessed 13/9/2006.

[152] LYMAN P. AND VARIAN H.R. **How Much Information, 2003**. How Much Information Web site **2003**. URL: `http://www.sims.berkeley.edu/how-much-info-2003`. Last accessed 13/9/2006.

[153] MAEDCHE A., MOTIK B. AND STOJANOVIC L. **Managing multiple and distributed ontologies on the Semantic Web**. *The VLDB Journal*, vol. 12(4) **2003**: pp. 286–302. ISSN 1066-8888. doi: http://dx.doi.org/10.1007/s00778-003-0102-4.

[154] MATHEUS C.J., KOKAR M.M., BACLAWSKI K. AND LETKOWSKI J.J. **An Application of Semantic Web Technologies to Situation Awareness**. *Lecture Notes in Computer Science: The Semantic Web ISWC 2005*, vol. 3729/2005 **2005**: pp. 944–958. doi: 10.1007/11574620.

[155] MAYRHAUSER A.V. **Measurement-based Modeling**. *Lecture Notes in Computer Science - Experimental Software Engineering Issues: Critical Assessment and Future Directions.*, vol. 06 **1992**: pp. 183–187.

[156] MCGUINNESS D.L. **Question Answering on the Semantic Web**.

IEEE INTELLIGENT SYSTEMS, vol. Vol. 19, No. 1 **2004**: pp. 82–85.

[157] MCGUINNESS D.L., FIKES R., HENDLER J. AND STEIN L.A. **DAML+OIL: An Ontology Language for the Semantic Web**. *IEEE Intelligent Systems* **2002**: pp. 72–80.

[158] MCGUINNESS D.L. AND VAN HARMELEN F. **OWL Web Ontology Language Overview**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-owl-features-20040210/`. Last accessed 13/9/2006.

[159] MCKINNON A. AND MCKINNON L. *XML*. ISBN: 0-619-03514-5. Web Warrior Series, Thomson Course Technology **2003**.

[160] MCKNIGHT D.H., CHOUDHURY V. AND KACMAR C. **Trust in e-commerce vendors: a two-stage model**. In *Proceedings of the twenty-first international conference on Information systems (ICIS '00), Brisbane, Queensland, Australia*, ISBN:ICIS2000-X. Association for Information Systems, Atlanta, GA, USA **2000**, pp. 532–536.

[161] MEDVIDOVIC N. AND TAYLOR R.N. **Separating fact from fiction in software architecture**. In *ISAW '98: Proceedings of the third international workshop on Software architecture*. ACM Press, New York, NY, USA **1998**. ISBN 1-58113-081-3, pp. 105–108. doi:http://doi.acm.org/10.1145/288408.288435.

[162] MEINRATH S.D. **Creating the Global Internet: The Formastion of International Computer Communications Networks**. Web site **2006**. URL: `http://www.saschameinrath.com/writings/CreatingTheGlobalInternet.doc`. Lastaccessed 12/12/2006.

[163] MELNIK S. AND DECKER S. **A Layered Approach to Information Modeling and Interoperability on the Web**. In *ECDL, 2000 - www-db.stanford.edu* **2000**. URL: `http://www-db.stanford.edu/~melnik/pub/sw00/`. Last accessed 13/9/2006.

[164] Meraka. **The Meraka Institute (The African Advanced Institute**

for Information and Communication Technology). Meraka Web site URL: http://www.meraka.org.za/. Last accessed 10/11/2006.

[165] MERTZ D. **XML Matters: Comparing W3C XML Schemas and Document Type Definitions (DTDs)**. IBM Developer Works **2001**. URL: http://www-106.ibm.com/developerworks/library/x-matters7.html. Last accessed 13/9/2006.

[166] MIKA P. **Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks**. *Journal of Web Semantics*, vol. 3 **2005**. URL: http://www.websemanticsjournal.org/ps/pub/2005-20. Last accessed 13/9/2006.

[167] MIKA P. **The Semantic Web Challenge**. The Semantic Web Challenge web site **2006**. URL: http://challenge.semanticweb.org/. Last accessed 13/9/2006.

[168] MILES M.B. AND HUBERMAN A.M. *Qualitative Data Analysis*. SAGE Publications, Thousand Oaks, London **1994**. ISBN 0-8039-4653-8.

[169] MILLER E. **The W3Cs Semantic Web Activity: An Update**. *IEEE Intelligent Systems*, vol. 1094-7167/04 **2004**: p. 95.

[170] MIT. **Massachusetts Institute of Technology (MIT)**. MIT Web site **2004**. URL: http://web.mit.edu/. Last accessed 13/9/2006.

[171] MOATS R. **RFC2141: URN Syntax**. IETF Web site **1997**. URL: http://www.ietf.org/rfc/rfc2141.txt. Last accessed 13/9/2006.

[172] MOUTON J. *How to succeed in your Master's and Doctoral Studies, A South African Guide and Resource Book*. Van Schaik Publishers, South Africa **2004**. ISBN 0-627-02484-X.

[173] MULLET K. AND SANO D. **Designing Visual Interfaces**. *SIGCHI Bulletin*, vol. 28(2) **1996**: pp. 82–83. ISSN 0736-6906. doi:http://doi.acm.org/10.1145/226650.570118.

[174] MYERS M.D. **Qualitative Research in Information Systems**. *MIS Quarterly*, vol. 21 **1997**: pp. 241–242. URL: `http://www.misq.org/discovery/MISQD_isworld/index.html`. Last accessed 16/9/2006.

[175] NCI Thesaurus. **NCI - US National Cancer Institute's Center for Bioinformatics 2005**. URL: `http://whitepapers.techrepublic.com.com/abstract.aspx?scid=442&docid=120560`. Last accessed 31/10/2006.

[176] NEJDL W., OLMEDILLA D. AND WINSLETT M. **PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web**. *Lecture Notes in Computer Science: Secure Data Management*, vol. 3178/2004 **2004**: pp. 118–132.

[177] NOTTELMANN H. AND FUHR N. **Learning probabilistic Datalog rules for information classification and transformation**. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM)* **2001**.

[178] NUTT G.J. *Centralized and Distributed Operating Systems*. Prentice-Hall International Editions **1992**. ISBN 0131223267.

[179] OBERLE D., STAAB S., STUDER R. AND VOLZ R. **Supporting application development in the semantic web**. *ACM Trans. Inter. Tech.*, vol. 5(2) **2005**: pp. 328–358. ISSN 1533-5399. doi: http://doi.acm.org/10.1145/1064340.1064342.

[180] OBERLE D., STAAB S., STUDER R. AND VOLZ R. **Supporting Application Development in the Semantic Web**. *ACM Transactions on Internet Technology (TOIT)*, vol. Volume 5 , Issue 2 **2005**: pp. 328 – 358. doi:http://doi.acm.org/10.1145/1064340.1064342.

[181] OIVO M. **Multiple Viewpoints of Software Models.** *Lecture Notes in Computer Science - Experimental Software Engineering Issues: Critical Assessment and Future Directions.*, vol. 06 **1992**: pp. 180–182.

[182] Ontoknowledge. **On-to-Knowledge: European IST Project**.

Web site. URL: http://www.ontoknowledge.org. Last accessed 21/3/2006.

[183] OTEC. **Data, Information, Knowledge, and Wisdom**. OTEC Web site **2006**. URL: http://otec.uoregon.edu/data-wisdom.htm. Last accessed 16/9/2006.

[184] PALMER S.B. **The Semantic Web: An Introduction**. W3C Web site **2001**. URL: http://infomesh.net/2001/swintro/. Last accessed 16/9/2006.

[185] PAN J.Z. AND HORROCKS I. **Metamodeling Architecture of Web Ontology Languages**. In *International Semantic Web Working Symposium (SWWS)* **2001**. URL: http://www.semanticweb.org/SWWS/program/full/paper11.pdf. Last accessed 16/9/2006.

[186] PARNAS D. **On the Criteria To Be Used in Decomposing Systems into Modules**. *Communications of the ACM*, vol. 15 **1972**: pp. 1053 – 1058.

[187] PARNAS D.L. **Designing Software for Ease of Extension and Contraction**. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*. IEEE Press **1978**, pp. 264–277.

[188] PARNAS D.L., CLEMENTS P.C. AND WEISS D.M. **The Modular Structure of Complex Systems**. In *ICSE '84: Proceedings of the 7th International Conference on Software Engineering*. IEEE Press **1984**. ISBN 0-8186-0528-6, pp. 408–417.

[189] PATEL-SCHNEIDER P. AND SIMEON J. **The Yin/Yang web: XML syntax and RDF semantics**. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*. ACM Press, New York, NY, USA **2002**. ISBN 1-58113-449-5, pp. 443–453. doi:http://0-doi.acm.org.oasis.unisa.ac.za:80/10.1145/511446.511504.

[190] PATEL-SCHNEIDER P.F. **A Revised Architecture for Semantic Web Reasoning**. *Lecture Notes in Computer Science*, vol. 3703 **2005**: pp. 32 – 36. doi:http://dx.doi.org/10.1007/11552222_3.

[191] PATEL-SCHNEIDER P.F. AND FENSEL D. **Layering the Semantic Web: Problems and Directions**. In *Proceedings of The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy*, vol. 2342 / 2002. Springer-Verlag GmbH **2002**, p. 16.

[192] PATEL-SCHNEIDER P.F., HAYES P. AND HORROCKS I. **OWL Web Ontology Language Semantics and Abstract Syntax**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`. Last accessed 16/9/2006.

[193] PERRY D.E. AND WOLF A.L. **Foundations for the Study of Software Architecture**. *ACM SIGSOFT SOFTWARE ENGINEERING NOTES*, vol. 17(4) **1992**: p. 40.

[194] PLOEG J. **Identifying the best research design to fit the question. Part 2: qualitative designs**. *Evidence-Based Nursing*, vol. 2 **1999**: pp. 36–37. URL: `http://ebn.bmj.com/cgi/content/full/2/2/36`.

[195] POPESCU-ZELETIN R. **Implementing the ISO-OSI reference model**. In *SIGCOMM '83: Proceedings of the eighth symposium on Data communications*. ACM Press **1983**. ISBN 0-89791-113-X, pp. 56–66.

[196] PRESSMAN R.S. *Software Engineering: A Practitioner's Approach*. sixth edition. McGraw-Hill International **2005**. ISBN 0-07-285318-2.

[197] Proof General. **Proof General Web site** URL: `http://proofgeneral.inf.ed.ac.uk/`. Last accessed 26/9/2006.

[198] RICHARDSON M., AGRAWAL R. AND DOMINGOS P. **Trust Management for the Semantic Web**. *Technical report*, IBM Almaden Research Center **2003**. URL: `http://www.cs.washington.edu/homes/pedrod/papers/iswc03.pdf`.

[199] ROY J. AND RAMANUJAN A. **XML: Datas Universal Language**. *IT Pro* **2000**: p. 32.

[200] SADOSKI D. **Client/Server Software Architectures–An Overview**. Carnegie Mellon Software Engineering Institute (SEI) Web

site **1997**. URL: http://www.sei.cmu.edu/str/descriptions/clientserver.html. Last accessed 21/9/2006.

[201] SADOSKI D. AND COMELLA-DORDA S. **Three Tier Software Architectures**. Carnegie Mellon Software Engineering Institute (SEI) Web site **2000**. URL: http://www.sei.cmu.edu/str/descriptions/threetier_body.html. Last accessed 21/9/2006.

[202] SCHACH S.R. *Introduction to object-oriented analysis and design with UML and the Unified Process*. Irwin McGraw-Hill **2004**. ISBN 0072826460.

[203] SCHRAEFEL M.C., SHADBOLT N.R., GIBBINS N., HARRIS S. AND GLASER H. **CS AKTive space: representing computer science in the semantic web**. In *Proceedings of the 13th international conference on World Wide Web, New York, NY, USA* **2004**, pp. 384 – 392. doi:http://doi.acm.org/10.1145/988672.988724.

[204] SCHRAEFEL M.C., SMITH D.A., OWENS A., RUSSELL A., HARRIS C. AND WILSON M. **The evolving mSpace platform: leveraging the semantic web on the trail of the memex**. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. ACM Press, New York, NJ, USA **2005**. ISBN 1-59593-168-6, pp. 174–183. doi:http://doi.acm.org/10.1145/1083356.1083391.

[205] SEI. **Software Engineering Institute 2005**. URL: http://www.sei.cmu.edu/architecture/. Last accessed 21/9/2006.

[206] Semantic Web Challenge. Website URL: http://challenge.semanticweb.org/. Last accessed 26/9/ 2006.

[207] SemTech06. **Semantic Web Technology Conference**. Web site **2006**. URL: http://www.semantic-conference.com/.

[208] SGML. **Overview of SGML Resources**. W3C Web site URL: http://www.w3.org/MarkUp/SGML/. Last accessed 31/10/2006.

[209] SGML and HTML. **On SGML and HTML**. W3C Web site URL: http://www.w3.org/TR/html4/intro/sgmltut.html. Last accessed 31/10/2006.

[210] SGML Introduction. **TEI Guidelines for Electronic Text Encoding and Interchange: A Gentle Introduction to SGML**. Virginia E-text Web site. URL: http://etext.virginia.edu/bin/tei-tocs?div=DIV1&id=SG. Last accessed 31/10/2006.

[211] SHAH U., FININ T., JOSHI A., COST R.S. AND MAYFIELD J. **Information Retrieval on the Semantic Web**. In *Proceedings of the eleventh international conference on Information and knowledge management* **2002**. ISBN 1-58113-492-4, pp. 461–468.

[212] SIIA. Website. URL: http://www.siia.net/. Last accessed 21/9/2006.

[213] SIMON E., MADSEN P. AND ADAMS C. **An Introduction to XML Digital Signatures**. Web site: XML.com **2001**. URL: http://www.xml.com/pub/a/2001/08/08/xmldsig.html. Last accessed 21/9/2006.

[214] SIMPSON H.R. **Layered Architecture(s): Principles and Practice in Concurrent and Distributed Systems**. In *1997 Workshop on Engineering of Computer-Based Systems (ECBS '97)* **1997**, pp. 312–320.

[215] SMITH M.K., WELTY C. AND MCGUINNESS D.L. **OWL Web Ontology Language Guide**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-owl-guide-20040210/. Last accessed 21/9/2006.

[216] SRIVASTAVA R. **XML Schema: Understanding Namespaces**. *Technical report*, Oracle **2004**. URL: http://www.oracle.com/technology/pub/articles/srivastava_namespaces.html. Last accessed 21/9/2006.

[217] SUN. **Swing Package for Java**. Sun Web site http://sun.java.com **2003**. URL: http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html. Last accessed 21/9/2006.

[218] SW Identification Problem. **The Semantic Web Identification Problem**. W3C Web site. URL: http://www.w3c.org/2001/03/identification-problem. Last accessed 5/7/2006.

[219] SWWS06. **SWWS06, the 2006 International Conference on Semantic Web and Web Services**. Web site **2006**. URL: http://www.world-academy-of-science.org/worldcomp06/ws/SWWS/index_html.

[220] TANENBAUM A.S. **Network Protocols**. *ACM Computing Surveys*, vol. 13(4) **1981**: pp. 453–489. ISSN 0360-0300. doi:http://0-doi.acm.org.oasis.unisa.ac.za:80/10.1145/356859.356864.

[221] THOMANS C. AND SHETH A. **On the Expressiveness of the Languages for the Semantic Web - Making a Case for A Little More**. *Technical report*, Large Scale Distributed Information Systems lab, University of Georgia, Athens, GA USA **2005**. URL: http://lsdis.cs.uga.edu/library/download/FLSW01.pdf.

[222] THURAISINGHAM B. **Building Secure Survivable Semantic Webs**. In *14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)*. IEEE **2002**, p. 395.

[223] THURAISINGHAM B. **Security Issues for the Semantic Web**. In *Proceedings of the 27th Annual International Computer Software and Applications Conference*. IEEE Press **2003**, p. 632.

[224] TYSON J. **How Encryption Works**. Web site **2006**. URL: http://computer.howstuffworks.com/encryption.htm. Last accessed 12/12/2006.

[225] URI. **Uniform Resource Identifier**. IETF Website. URL: http://www.rfc-archive.org/getrfc.php?rfc=2396. Last accessed 11/9/2006.

[226] USCHOLD M. **Where Are the Semantics in the Semantic Web?** *AI Magazine*, vol. 24(3) **2003**: pp. 25–36.

[227] VAN DER MERWE A., KOTZE P. AND CRONJE J. **Selecting a Qualitative Research Approach for Information Systems Research**. In *Proceedings of SACLA2004* **2004**.

[228] VAN HARMELEN F. AND HORROCKS I. **Questions and answers on OIL: The Ontology Inference Layer for the semantic web**. *EEE Intelligent Systems*, vol. 15(6) **2000**: pp. 69–72.

[229] VAN HARMELEN F., PATEL-SCHNEIDER P.F. AND HORROCKS I. **Annotated DAML+OIL (Mar 2001) Ontology Markup**. DAML Web site **2001**. URL: `http://www.daml.org/2001/03/daml+oil-walkthru`. Last accessed 23/9/2006.

[230] VERMA M. **XML security : Implement security layers, Part 1**. Web site **2003**. URL: `http://www-128.ibm.com/developerworks/xml/library/x-seclay1/`. Last accessed 20/12/2006.

[231] W3C. **Resource Description Framework (RDF) Model and Syntax Specification**. W3C Web site **1999**. URL: `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`. Last accessed 24/9/2006.

[232] W3C. **W3C Semantic Web Activity**. W3C Web site **2001**. URL: `http://www.w3.org/2001/sw/`. Last accessed 26/9/2006.

[233] W3C. **XML Schema Part 0: Primer - W3C Recommendation**. W3C Web site **2001**. URL: `http://www.w3.org/TR/xmlschema-0/`. Last accessed 23/9/2006.

[234] W3C. **XML Schema Part 1: Structures - W3C Recommendation**. W3C Web site **2001**. URL: `http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/`. Last accessed 24/9/2006.

[235] W3C. **XML Schema Part 2: Datatypes - W3C Recommendation**. W3C Web site **2001**. URL: `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`. Last accessed 23/9/2006.

[236] W3C. **Information on OWL**. Web site **2004**. URL: `http://www.w3c.oeg/2001/sw/WebOnt/`. Last accessed 24/9/2006.

[237] W3C. **RDF Primer**. W3C Web site **2004**. URL: `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`. Last accessed 26/9/2006.

[238] W3C. **RDF Primer**. W3C Web site **2004**. URL: `http://www.w3.org/TR/rdf-primer/`. Last accessed 24/9/2006.

[239] W3C. **RDF Semantics**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-mt-20040210/. Last accessed 26/9/2006.

[240] W3C. **RDF Test Cases**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/. Last accessed 26/9/2006.

[241] W3C. **RDF Vocabulary Description Language 1.0: RDF Schema**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Last accessed 26/9/2006.

[242] W3C. **RDF/XML Syntax Specification (Revised)**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Last accessed 26/9/2006.

[243] W3C. **Resource Description Framework (RDF): Concepts and Abstract Syntax**. W3C Web site **2004**. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Last accessed 23/9/2006.

[244] W3C. **World Wide Web Consortium Issues RDF and OWL Recommendations**. W3C Web site Press Release **2004**. URL: http://www.w3.org/2004/01/sws-pressrelease. Last accessed 24/9/2006.

[245] W3C. **World Wide Web Consortium (W3C) - Internationalized Resource Identifiers (IRIs)**. W3C Web site **2004**. URL: http://www.w3c.org/Addressing/Activity. Last accessed 24/9/2006.

[246] W3C. **RDF Data Access Use Cases and Requirements**. W3C Web site **2005**. URL: http://www.w3.org/TR/2005/WD-rdf-dawg-uc-20050325/. Last accessed 23/9/2006.

[247] W3C. **RIF Use Cases and Requirements: W3C Working Draft 10 Jul. 2006**. W3C Web site **2006**. URL: http://www.w3.org/TR/rif-ucr/. Last accessed 23/9/2006.

[248] W3C. **SPARQL Query Language for RDF**. W3C Web site **2006**. URL: http://www.w3.org/TR/rdf-sparql-query/. Last accessed 24/9/2006.

[249] W3C. **W3C Technical Architecture Group (TAG)**. W3C Web site **2006**. URL: http://www.w3.org/2001/tag/. Last accessed 4/10/2006.

[250] W3C. **The World Wide Web Consortium (W3C)**. W3C Web site **2006**. URL: http://www.w3.org/. Last accessed 24/9/2006.

[251] W3C. **XML Signature WG**. W3C Web site **2006**. URL: http://www.w3.org/Signature/. Last accessed 24/9/2006.

[252] W3C T. **Findings of the W3C Technical Architecture Group (TAG)**. W3C Web site **2006**. URL: http://www.w3.org/2001/tag/findings. Last accessed 4/10/2006.

[253] W3C T. **TAG Issues List**. W3C Web site **2006**. URL: http://www.w3.org/2001/tag/issues.html?type=1. Last accessed 26/9/2006.

[254] W3C Rules. **W3C Workshop on Rule Languages for Interoperability**. W3C Web site URL: http://www.w3.org/2004/12/rules-ws/. Last accessed 31/10/2006.

[255] W3C Rules WG. **W3C RuleWorking Group**. W3C Web site URL: http://www.w3.org/2005/rules/wg. Last accessed 31/10/2006.

[256] W3C Security Activity. **Security Activity Statement**. W3C Web site **2006**. URL: http://www.w3.org/Security/Activity.html. Last accessed 20/12/2006.

[257] W3C-XML. **W3C Extensible Markup Language (XML)**. Web site **2005**. URL: http://www.w3.org/XML/. Last accessed 24/9/2006.

[258] WEINER L.H. **The roots of structured programming**. In *Papers of the SIGCSE/CSA technical symposium on Computer science education*. ACM Press, New York, NY, USA **1978**, pp. 243–254. doi: http://doi.acm.org/10.1145/990555.990636.

[259] WENTZEL K.D. **Software Engineering Models, Using and Reusing.** *Lecture Notes in Computer Science - Experimental Software Engineering Issues: Critical Assessment and Future Directions.*, vol. 06 **1992**: pp. 191–201.

[260] WWW06. **The WWW Conference**. Web site **2006**. URL: http://www2006.org/tracks/semweb.php.

[261] XML.COM. **O'Reilly XML.com**. Web site **2005**. URL: http://www.xml.com/. Last accessed 26/9/2006.

[262] XU B., WANG P., LU J., LI Y. AND KANG D. **Bridge Ontology and Its Role in Semantic Annotation**. In *Third International Conference on Cyberworlds (CW'04)* **2004**, pp. 329–334.

[263] YAGUE M.I., MANA A., LOPEZ J. AND TROYA T. **Applying the Semantic Web Layers to Access Control**. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03)* **2003**, p. 266.

[264] ZACHMAN J. **The framework for enterprise architecture: background, description and utility**. Zachman International Web site **2003**. URL: http://home.hib.no/AI/data/fag/D056_2003/Zachman%5CZIFA%20Description.pdf. Last accessed 26/9/2006.

[265] ZIMMERMANN H. **OS1 Reference Model - The IS0 Model of Architecture for Open Systems Interconnection**. *IEEE Transactions on Communications*, vol. 71 **1980**: pp. 1334–1340. URL: http://www.comsoc.org/livepubs/50_journals/pdf/RightsManagement_eid=136833.pdf. Last accessed 26/9/2006.

Index