UNIVERSITY OF WEST LONDON
The Career University

UWL REPOSITORY

repository.uwl.ac.uk

Model driven validation approach for enterprise architecture and motivation extensions

Essien, Joe (2015) Model driven validation approach for enterprise architecture and motivation extensions. Doctoral thesis, University of West London.

This is the Accepted Version of the final output.

UWL repository link: https://repository.uwl.ac.uk/id/eprint/1269/

Alternative formats: If you require this document in an alternative format, please contact: open.research@uwl.ac.uk

Copyright:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy: If you believe that this document breaches copyright, please contact us at open.research@uwl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

UNIVERSITY OF WEST LONDON
SCHOOL OF COMPUTING AND TECHNOLOGY

MODEL DRIVEN VALIDATION APPROACH FOR ENTERPRISE ARCHITECTURE AND MOTIVATION EXTENSIONS

By

Joe Essien

A thesis submitted in fulfilment of the requirement for the degree of

Doctor of Philosophy in Computer Science

April, 2015

Dedication

This work is dedicated to God whom without His blessings, protection and faithfulness, this work would not have been possible (Mathew 19:26).

Acknowledgements

I would like to extend my heartfelt and sincere thanks and appreciation to my thesis supervisors, Dr Samia Oussena and Professor Peter Komisarczuk whom I was extremely fortunate to have as mentors and advisers throughout the duration of this work. Samia was an exceptional blessing in every way conceivable; supportive, encouraging and allowing me to learn grow in understanding the concepts that are relevant to this research, focusing my thought process within the domain of the research while at the same time exposing me to various concepts of Enterprise Architecture, institutions, bodies that broadened my knowledge concerning Enterprise Architecture. All the while Peter backed me up and responded instantly whenever I need advice, guidance and support. He acted veraciously to resolve challenges I encountered during the cause of this work, extending his immense expertise and experience to resonate on this research. At all times both supervisors directed me towards new trends in information systems and encouraged me tremendously as best as possible towards persevering and completing this work.

I would also like to thank my dear sister Eme Asuquo who provided moral support and advice that enabled me to cope with the various external voracious challenges I faced during the course of this work.

Special thanks to Maria Pennells who responded at all times very promptly and effectively to ensure that I am well guided through the requirements of the UWL for the research; reaching out whenever necessary to advice and smoothen situations that could have hindered this work. I would also like to thank Professor Balbir Barn of the Middlesex University, London who was involved with my case study on Laptop Loan Scheme for endorsing my use of the case study with support from my supervisor Samia, enabling me to expatiate on its principles and apply it to this work. My gratitude goes also to the British Computer Society who through their various seminars broadened my knowledge on Enterprise Architecture and modelling techniques.

Lastly, I thank God for his grace and gift of all these wonderful people who believed in me and committed their time, intellect and experience to guide me through this research work from beginning till completion. I remain ever appreciative to God and them.

Copyright Statement

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods for the purpose of profit or commercial gains without the prior written permission of the author. Permission is granted in the case of quotations or citing for use in critical reviews. Certain other non-commercial uses are permitted by copyright law.

Copyrights © Joe Essien, 2015

OVERVIEW OF CONTENTS

DETAILED TABLE OF CONTENT

List of Figures

Figures for Case Study A: – UME-LLS

Figures for Case Study B: – UWL-SIP

List of Tables

Common Abbreviations

ACF	Architecture Content Framework
AD	Architecture Description
ADM	Architecture Development Method
BDD	Behaviour Driven Development
BDMCS	Behaviour Driven Modelling Constraints Specification
BPEL	Business Process Execution Language
BPM	Business Process Modelling
BPMN	Business Process Model and Notation
COBIT	Control Objectives for Information and related Technology
DoDAF	Department of Defence Architecture Framework
DYA	Dynamic Architecture
EA	Enterprise Architecture
EAF	Enterprise Architecture Framework
EAM	Enterprise Architecture Maturity
EAML	Enterprise Architecture Modelling Language
EAVEM	Enterprise Architecture Validation Extension Metamodel
FEAF	Federal Enterprise Architecture Framework
GEAF	Gartner Enterprise Architecture Framework
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electrotechnical Commission
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
ITA	Information Technology Architecture
KAOS	Knowledge Acquisition in autOmated Specification
MDA	Model Driven Architecture
ODP	Open Distributed Processing
OMG	Object Management Group
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RM-ODP	Reference Model for Open Distributed Processing
SEAM	Systematic Enterprise Architecture Method
SPARQL	SPARQL Protocol and RDF Query Language
TDD	Test Driven Development
TDM	Test Driven Modelling
TOGAF	The Open Group Architecture Framework
URI	Uniform Resource Identifier
VEM	Validation Extension Metamodel
XML	Extensible Markup Language
XSD	XML Schema Definition
ZF	Zachman Framework

ABSTRACT

As the endorsement of Enterprise Architecture (EA) modelling continues to grow in diversity and complexity, management of its schema, artefacts, semantics and relationships has become an important business concern. To maintain agility and flexibility within competitive markets, organizations have also been compelled to explore ways of adjusting proactively to innovations, changes and complex events also by use of EA concepts to model business processes and strategies. Thus the need to ensure appropriate validation of EA taxonomies has been considered severally as an essential requirement for these processes in order to exert business motivation; relate information systems to technological infrastructure. However, since many taxonomies deployed today use widespread and disparate modelling methodologies, the possibility to adopt a generic validation approach remains a challenge. The proliferation of EA methodologies and perspectives has also led to intricacies in the formalization and validation of EA constructs as models often times have variant schematic interpretations. Thus, disparate implementations and inconsistent simulation of alignment between business architectures and heterogeneous application systems is common within the EA domain (Jonkers et al., 2003).

In this research, the Model Driven Validation Approach (MDVA) is introduced. MDVA allows modelling of EA with validation attributes, formalization of the validation concepts and transformation of model artefacts to ontologies. The transformation simplifies querying based on motivation and constraints. As the extended methodology is grounded on the semiotics of existing tools, validation is executed using ubiquitous query language. The major contributions of this work are the extension of a metamodel of Business Layer of an EAF with Validation Element and the development of EAF model to ontology transformation Approach. With this innovation, domain-driven design and object-oriented analysis concepts are applied to achieve EAF model's validation using ontology querying methodology. Additionally, the MDVA facilitates the traceability of EA artefacts using ontology graph patterns.

1 INTRODUCTION

A popular definition for Enterprise Architecture is that which is provided by the Institute of Electrical and Electronics Engineers (IEEE) and states that it is “the organization of a system embodied in its components, relationships to each other, environment, the principle guiding its design and evolution" (IEEE, 2000). Many other definitions of EA are given in several works. The core of these definitions presents EA as coherent principles, methods and models used in the design and realisation of organisational structure, business processes, information systems and infrastructure (Lankhorst, 2013). These definitions also provide the basis for specification of important characteristic of EA as a means to provide holistic visualization of the enterprise through models. This extends to include specification for the relevant aspects of the business as it applies to each stakeholder. As a result of the many perceptions of the capability of EA and the relevance, various reactions and criticisms are made regarding its intention, criteria for success and implementation. While some maintain that an effective Enterprise Architecture model must provide the insight needed to align business functions with technological requirements, others insist that it must also facilitate the transition from corporate strategy to daily operations (Tang et al., 2006). However, these multi-dimensional interests, non-standardization of definitions and principles of EA has led to the adoption of heterogeneous approaches and modelling techniques in many organisations today (Fischer et al., 2010; Schekkerman, 2003; Sessions, 2007; Stanley & Uden, 2013). Literal analysis of these EA patterns show many disparate taxonomies, understood by each stakeholder from a different perspective (Weston & Defee, 2004). Yet the definition of compositions and dependencies that entwine these different views are extremely complex in some cases (Winter, 2007). Though the design and implementation of EA models currently specify predominantly perspectives and alignment of goals, issues regarding validation and harmonization criteria are ambiguous or presented in a rudimentary way (Stanley & Uden, 2013). The postulations do not consider the behavioural attributes of the components that comprise a model as a concept that should be subjected to validation. Many authors believe that EA models are not reusable and are designed to actualize a specific goal after which it is archived or at best used as a reference provenance (Stanley & Uden, 2013; Rudawitz, 2003). Thus model validation has not been properly defined and most definitions tend to specify the expected results of implementing an EA against predefined goals or established levels of maturity. Thus the accomplishment of this benchmark is often regarded as the realization of EA initiatives within the organization. However, this analogy has been deemed incomplete and unsatisfactory as affirmed in many past and recent surveys (Jorgensen, et al., 2008; Brame and Barlow, 2010; Bloch et al., 2011; GENECA Research Report, 2011; Logica, 2014). Even so with the emergence of cloud computing, big data and other service oriented technologies, validation of Enterprise Architecture Framework is becoming even more relevant and a necessity for many practitioners (Chapurlat & Braesch, 2008).

As modelling of enterprise continues to influence the way many organisations represent their business strategies and technologies, there is a commensurate growth in knowledge base with resolute lessons gained. This is despite criticism that Enterprise Architecture development should have started by the elaboration of an agreed architecture representation language in order to avoid contemporary perilous proposals (Kang et al, 2010). Other practitioners argue that since the advent of EA, a lot of prominence has continually been placed distinctively on business process modelling and information technology infrastructure with less emphasis on alignment and formalisation (Bakhshadeh et al., 2014). Approaches such as top-down and bottom-up have been proffered with fastidious ambience of best practises for EA development (Carla & Sousa, 2005; Kulkarni, et al., 2013). However, the development of criteria and methods for evaluating architectures have been given less importance when compared with the development of various architectures and modelling methodologies (Khoury, 2007). Without an extensible and comprehensive validation method, it is difficult for enterprise to evaluate the usefulness of architectures as complex architectures are intricate and difficult to understand by stakeholders. Supporters of this view acknowledge also that meaningful semantics can provide the basis for interrogation of constructs (McShane & Nirenburg, 2013) if provided with the models.

In recognition of this need, the use of maturity matrices to benchmark as-is against to-be scenarios has been prevalent in many cases (Gartner, 2013; Weston & Defee, 2004; Lakhrouit et al., 2014) as a means to validate EA. While this has been deemed sufficient in certain situations, it has also been argued that it is inadequate as it is subjective and not based on constraints that constitute the model artefacts (Beznosov, 2000; Carla & Sousa, 2005; Fischer et al., 2010). In consideration of the significant success in the field of EA modelling, many authors have been traversed to suggest the need for schematization and distinction between aspects of visualization in order to aid validation (Salmans et al., 2010; Venkatraman et al., 2010). With increasing collaboration of enterprise concerns through cloud computing technologies, big data and case based reasoning, the significance of EA modelling and validation have continued to point towards methods that can segregate domain knowledge from the operational knowledge; or facilitate the analysis of the domain structures through formalised decomposition and systematic integration such as denoted by use of ontologies. While the use of ontology is not entirely new (Jan & Dietz, 2006; Wache et al., 2001), the concepts especially as it relates to schematization and resource description frameworks present a chasm which can allow the development of a different approach for validation of EA models (McGuinness, 2002; Chapurlat, et al., 2008; Almeida & Guizzardi, 2013). Thus this research presents EA model validation from a perspective different from the commonly used maturity matrices, balanced scorecard and reference models and espouses the use of interrogative constructs on ontology derived from model artefacts to confirm that the EA model meets the intrinsic goals defined by their motivation.

1.1 Evolution of Problem and Derivation of Research Question

In many organizations, EA patterns exist that encapsulate business concerns such as strategic planning, processes, integration and compliance to regulatory stipulations (Weston & Defee, 2004). However, literal analysis of these patterns shows disparate architectures that are understood by each stakeholder from different perspectives. The connections and dependencies that exist amongst the prevalent EAF views and models have been described in many cases as intricate and complex. In many implementations, validation of EA has been confined to the use of maturity matrices, balanced scorecards and reference models. Though these approaches have been explored extensively, collaborative works have indicated that the approaches do not provide the sustainable capability needed to validate EA Framework (EAF), models or provide the descriptive semantics for the components with respect to their role in actualizing business behaviour, motivational goals or traceability (Klein & Gagliardi, 2010). A common weakness to these approaches for EAF validation is that the models created are not usually at the same level of abstraction. With the lower levels of abstraction being more detailed than the higher levels of abstraction, there are either disjunctions or multiple conjunctions between the artefacts. The variation in levels of metaphors and semantics creates inconsistencies between the taxonomies and layers in the model thus making it difficult to assign commensurate weighs of measures, specifically for methods such as levels of maturities.

The broadness in terms of scope, proliferation of concepts, use of heterogeneous and sporadic modelling approaches have been suggested as the basis for the complex and spasmodic challenges that exist today in EA validation.(Sessions, 2007; Lankhorst, 2013). In an effort to streamline these complexities, many EA practitioners primarily recognise four facets of EA, described as business, application, information, and technology perspectives (Goethals, 2003; Iyer & Gottlieb, 2004; Carla et al., 2005; Urbaczewski & Mrdalj, 2006; Winter, 2007; Capgemini, 2013). However, the determination of what should constitute their categorization has been discursive. Often homogeneous artefacts deploy varied or sometimes indistinct information components to conceptualise similar paradigms for process management and change (Jorgensen et al., 2008). For instance, many EA architects have augmented these perspectives to include strategy, organization, culture, data, integration, security, infrastructure, solutions and more (Schekkerman, 2003; Winter & Fischer, 2007). Undaunted by this apparent stretch of EA composition, many critics contend the completeness of this categorisation and proffer distinct deviations consisting of Information Technology (IT), Information Strategy, Organisational Policies, Principles, Information System, Infrastructure and Implementation (Venkatraman et al., 2010). Still it is not uncommon to observe other elaborations created by fusions of these categories so that more convoluted perspectives of the abstractions are aggregated as architectures.

Without the harmonization of EA concepts but with continuous systematic omissions and inclusions of components as elements of EAF, attempts to integrate with various existing technologies and adaptation to multi-facet domains, an immense challenge confounding enterprise architects is that

none of the frameworks can be asserted as generic and capable of satisfying most aspects of the enterprise identified collectively (Danesh & Yu, 2014). Very significant in this aggregation is that there is no categorization of any EA perspective that reflects on need for validation of EA model, evaluation or testing as a concept that should be encapsulated within the EAF. EA methodologies that predicate verifiable patterns for business behaviour are either lean or docile in specifying an approach for validating the model artefacts against motivation. While many practitioners contemplate on the benefits of validation, most EAF modellers are not clear on how this can be achieved nor assert definitively that it is unnecessary (Engelsman et al., 2011). Be that as it may, almost all enterprise architects know that at some point in their EA project endeavour, their prototypes would have to be proven or justified especially when subjected to broadened contexts, challenges of change management, risk assessments, integration, business dynamism and regulatory compliance. This is because complexities in EA taxonomy have serious consequences which are closely related to the numerous failures in IT projects as affirmed in many past and recent surveys (Jorgensen, et al., 2008; Brame and Barlow, 2010; Bloch et al., 2011; GENECA Research Report, 2011; Logica, 2014).

In view of orthodox postulations that Enterprise Architecture can offer the insight needed to balance requirements for facilitating synergy from corporate strategy to recurrent operations, EA models continue to be developed using divergent Enterprise Architecture Modelling Language. This puts high cognitive strains on modellers as the correlation and interpretation of artefacts and notations often become difficult to understand especially to the non-technical users of the system. Lack of formalization makes the semantics of the models inconsistent and superfluous. Therefore models produced in many organizations make meaning only within those organizations. The implication of this is that as the importance of the adoption of EA continues to grow, these already identified complexities are amplified as they are not often reviewed with definitive progression with the advancements in modelling tools. Consequently, these anomalies make it impossible to develop or adopt a generic approach for validation of many artefacts in the framework. Varieties of Enterprise Architecture Modelling Language adopt their own unique notations, symbols and protocols. This impairs interrelation accross the various domains. The dependence on domain specific modelling notations implies that across the various enterprise models, there exist inconsistencies, gaps and overlaps that make validation superfluous in some instances and unreliable in other scenarios. Therefore a need for amalgamation of heterogenous metamodels through a common unified ontology is often considered hypothetical for addressing these concerns.

The issue of whether model validation procedures can enhance the accuracy and relevance of information reported to management in terms of its ability to promote circumstantiated and rationalised decisions have been widely debated. The relevance of this altercation is based on concerns regarding the appropriateness of principles ascribed for modelling as well as the integrity of the modelling tools used in creation of the models (Da Xu, 2011). Emphasized is the need to incorporate a means that can result in accurate representation of intrinsic instantiation of metamodel.

Equally important is the accession of prerequisite aspects that would enable the precipitation of predefined goals; views and different perspectives that can clearly be visualized and communicated comprehensively to all stakeholders and, the relationships that exist between the various artefacts that can accurately be appropriated for traceability. With current trends whereby system designs are built with model orientations, complex business cases are increasingly simplified to generic tripartite notations and triples with consideration for testing. Thus a starting point for EA model validation in engaging with this innovation has been identified as the need to formalize EA concepts and constructs such that consistency can be maintained for the variant and widespread paradigm of EA domiciliation (Lankhorst, 2013). A unified or homogenous description of artefacts that constitute a model, meta-model, meta-meta-model, and framework would allow standardised interpretation and annotation of EA artefacts thus facilitate validation. In the same way, exclusive and cohesive definition of the relationship between the various elements of EA would delineate peculiar characteristics of that association and facilitate artefact traceability.

Considering the significance of models within enterprise architectures, it is incontrovertible that the requirement to encapsulate validation into all aspects of EA modelling would be a compelling necessity. Evidence suggests that with time the requirement to present the effects of change on integrated views of the enterprise along all phases of modelling would be mandatory (Chen, et al., 2008). For instance, in recognition of this exigency many regulatory bodies have of recent heightened awareness of model risk with expectations that enterprises identify key models that need periodic validation in order to confirm their accuracy. This will not only ensure consistent alignment between business functions and information systems, it will also bring a common understanding between business stakeholders, developers and the users. Accordingly, this suggests that the existing validation approaches can still be improved upon perhaps by the adoption of a methodology that can transform a model to a common platform, viewable and testable through unified schemas as attainable with resource description framework. Fundamental to this postulation is the inference that since most EAFs can be transformed to ontology, the means for validating ontology schemas provided by their service resources can be extended and adapted to validate the EA models even if they are created with heterogeneous Enterprise Architecture Modelling Language.

Based on theoretical principles for change management, enterprise systems and goal evaluation, this research advances a novel multi-conceptual perspective for addressing challenges of EA validation. The methodology which is original and unique produces as outcome artefacts that represent assessment archetypes for metamodel, models and traceability. The proposal extends the traditional perception of EA validation beyond the level of balance scorecards, maturity matrices and reference models to proffer an advanced metaphoric approach that generates new understanding of the levels of validation in relation with motivation. It argues that modelling of EAF with motivational attributes as opposed to the traditional embellishment of structural classes and relationships is critical for ensuring the actualization of business goals and intrinsic values of the enterprise. The methodology articulates

in details the workflow for EA validation implementation, the query approach, and the factors that influence the outcome and results. Furthermore, the convergence of illustrative perspectives in this work brings innovation to EA model management, change management and formalization concepts to bear on a subject which traditionally has been of predominant concern to EA practitioners.

With regard to these challenges associated with EA, this work is limited in its scope in order to maintain cognitive tenacity for the foundational aspects of modelling that affect validation. Though models created with heterogeneous semantics can be applied with the methodology presented in this work, focus is placed on models created for the business layer of the EAF. The major rationale for setting this confine is that the business layer extends to other multifarious levels of the EA taxonomy thus the effective validation of this base abstract would encrust validation into the entire framework. It would also allow the case scenarios to be modelled based on specific viewpoints of business concerns, constraints and goals defined from motivation. The requirements for the modelling of the business layer are considered integral to the framework entities and cascades though the multiple levels of abstraction. Accordingly, these reasons form the justification for this research and provide the grounds for derivation of the research strategy, aims and questions.

1.2 Research Strategies, Aims and Objectives

The aim of this research is to extend an Enterprise Architecture Modelling Language by introducing capabilities that allow validation to be performed on instantiated models based on motivation from divergent user perspectives. To achieve this aim, it is intended that the models would be formalised based on definite views and perspective. Though many presumptions have been put forward to argue that formalization permits the systematization, refinement, and methodological clarification of models (Zimmerman et al., 2010), the association between these presumptions and various assertions are still subject for speculations. What is however common amongst these presumptions is that formalization makes possible the identification and formulation of unresolved problems (Bicchierai et al., 2013). Thus leveraging on the grounds of these presumptions, there is no doubt that formalising and enhancing model with validation extensions would add clarity in the presentation of its taxonomy, terms and goals that need to be achieved. It will also establish traceability and more transparency in visualising the effect of change thus exposing gaps and overlaps.

The research aims to adopt an idealization concept to ascertain the values of the EAF proposition. A derivative axiom that represents the outcome of the transformed model would be factorized to create resource description graphs and schemas that can be validated using natural query language semantics. To execute the query semantics, validation metrics are proposed and used with the validation element extended from an open and independent enterprise architecture modelling language. The metrics which in itself spans most of the specification for model validation as defined by Control Objectives for Information and related Technology (COBIT, 2014) consists of five validity

criteria to support goals realization, model traceability, motivation assessment, business behaviour analysis and perspective visualization. These validation metrics are applied at the business layer of EA on aspects that encompass critical qualities of model validation. One major advantage of this approach is that it compliments constraints validation specified by motivation. Additionally, the nonparametric validation approach can be expressed as contingency tables and triples thus playing an important role in validating the EA model ensuring congruity and consistency.

Thus the main objectives addressed in this work are as follows;

i. To extend an Enterprise Architecture Modelling Language with validation capability. This entails the addition of a modelling element for validation and the relationship of the element with other artefacts. In modelling therefore, the extension allows the encapsulation of motivation within the Business Architecture of EA and facilitates formalization and alignment of goals with the business strategy.

ii. To provide a methodology for model transformation to ontology description schema with capability for validation using query language semantics and ideas from domain-driven design and object-oriented analysis. It is envisaged that this approach would provide a means for describing a cycle of testing interactions and would allow the description of behavioural requirements of an enterprise system in a way that is comprehensible to all stakeholders.

iii. The source code of the extended modelling tool would be released as Open-Source for further research works and for use by the software community. Feedback would also be gained on its effectiveness.

iv. To develop validation metrics for testing Enterprise Architecture artefacts. This adds agility to the organization's EA modelling processes.

v. To enhance the traceability capability for Enterprise Architecture artefacts through Resource Description Framework Graphs. This exposes gaps and omissions in the model's taxonomy and enables decision making regarding the quality of the EAF. Additionally information regarding how to improve the taxonomy is made more explicit.

In addition to collaborations with research work in this field, a number of published research papers have ensued from this work. These are presented in Appendix B.

1.3 Research Question and Subquestions

Based on the aims of the research stated in the preceding section, the study is grounded on the following research question and subquestions.

> *How can validation be incorporated into Enterprise Architecture models so as to enhance realization of the organization's motivational goals?*

To achieve this, the question is subdivided into four main subquestions that require knowledge regarding Enterprise Architecture Framework and models, Enterprise Architecture Modelling Languages, Enterprise Architecture model validation and techniques and Ontology Schematization.

i. Research subquestion regarding Enterprise Architecture Framework and models

What are those indicatives or drivers that can be encapsulated into the EAF and models to ensure correlation with its motivation and facilitation of validation?

The first challenge of this research is addressed by the first subquestion as it delves into the various multifaceted perspectives of EAF and models. Several researches have shown that there is no common agreement on which structural distinctive artefact types and dependencies characterize each of the EAF layers (Beznosov, 2000; Bittler et al., 2005; Carla et al.,2005; Chen et al, 2008; Halttunen et al., 2005, Lankhorst, 2013). Consequently, many frameworks for modelling do not clearly map EA layers, derivative viewpoints and aspects to motivation (Lankhorst, 2013; Bredemeyer, 2013; Winter & Fischer, 2007; Ylimaki, 2008). Many EAF have gaps and overlaps amongst the various layers of abstraction. There are divergent views for similar aspects of the EAF (Lankhorst, 2013; Venkatraman et al., 2010; Urbaczewski & Mrdalj, 2006). Relationship between aspects and layers is ambiguous (Braun et al, 2005) with no emphasis focused on validation of EA goal (Klein & Gagliardi, 2010). The frameworks have very specific scope and purpose thus not amenable to change thus they apply to explicit applications or development methodology. They are generally weighted towards planning and analysis with little or no emphasis on the framework validation and change management (Jonathan et al., 2008). Sequel to all these inferences from experts, authors and researchers, it is reasoned that many EAF may not be capable of viewing the enterprise in its entirety (Sessions, 2007; Rudawitz, 2003; Schekkerman, 2004; Lankhorst, 2013). A selection of widely used EA taxonomies is discussed extensively in chapter 2.2. As it is widely agreed that many of the frameworks are limited and incapable of creating actionable, extended EA that address today's rapidly evolving complex modelling requirements (Bloch, et al., 2011; Danesh, & Yu, 2014), a choice of a generic EAF that can be extended for validation is desired. Its justification is discussed in section 4.4 with a robust and critical debate on the methodological choices which underpin the research. Reflections on limitations are also presented.

ii. Research subquestion regarding Enterprise Architecture Modelling Languages

How can an Enterprise Architecture Modelling Language be extended with additional constructs to incorporate validation capabilities?

The second subquestion is concerned with challenges that constrain EA modelling languages. Enterprise Architecture Modelling Language is a high level structured lexicon that aims at representing objects, characteristics, properties of frameworks and design. Over the decades, there has been proliferation of modelling languages as a means of presenting visual images of design concepts (Chen, 2008). Enterprise Architecture Modelling Languages (EAML) provides this means for handling the complexity of modern information-intensive enterprises (Lankhorst, 2013). It enhances communication between different stakeholders and ways to express business concerns articulately. However, most Enterprise Architecture Modelling Languages are not open source and do not provide the flexibility that allow modification of the construct with extensions (Lankhorst, 2013, Fischer et al., 2010). Thus, today many organizations use heterogeneous set of modelling languages to express their organisational structure, business processes, information systems and infrastructure. This is due to the limitations inherent in most contemporary EAML as none is capable of modelling all these aspects of EA with consistent notations and semantics (Danesh and Yu, 2014; Da Xu, 2011; Engelsman, et al., 2011). Connections and dependencies that exist among the different viewpoints of EA models can be extremely complex in many cases with no consideration for validation of created models. Thus complexities exist between the metamodels and their instantiations across the heterogeneous framework. Chapter 4.0 provides a description of these modelling languages with a correlation given in chapter 4.3. The analogy affirms that business behaviour and components modelled with these disparate Enterprise Architecture Modelling Languages do not share homogeneous annotations, semantics and relationship thus streamlining validation in a standardized formalized way is difficult. To enhance the quality of enterprise architecture models, there is need to bring together all these information from segregated domains and adopt a formalised approach that is understood by all stakeholders.

iii. Research subquestion regarding Enterprise Architecture model validation and techniques

What queries can be developed and deployed to validate EA model, its motivation as well as demonstrate consistency, alignment and traceability?

Though many Enterprise Architecture Modelling Language exist that present knowledge encapsulation of their respective architectural domains, none support the direct validation of high-level models with their instantiations through a common vocabulary (Dietz, 2006; Calì et al., 2012). Consequently with sporadic researches that seek to amalgamate models through common semantics, the validation of

motivation across models and viewpoints in a coherent, interconnected and consistent manner is often prematurely adjudged. Perhaps this is because it has been suggested that the issue of validation of model is not feasible and possibly pointless. In the contrary the intensified advancement of maturity matrices as means to establish that models meet their intrinsic goals seam to denegate this perception. A comparative evaluation therefore of domain-driven design and object-oriented analysis suggests that specification of validation at the time of ascertaining the values of the proposition and design of the models can lead to executable specifications in a way that allows traceability to be established. Hence challenges associated with Enterprise Architecture validation and assessment are discussed more extensively in section 3.1, EA validation techniques presented in section 3.2 and correlation of these techniques as applied to EAF presented in section 3.3. The theoretical principles for model validation are explained in chapter 5 while the formalization of these theories for EAML validation extension in discussed in chapter 6.

iv. Research subquestion regarding Ontology Schematization

Is there a formalization approach that can be adopted to transform an EA model extended with validation constraints into an ontology so as to allow querying and traceability validation?

The final challenge is associated with ontology schematization. Ontologies are used to capture knowledge about some domain of interest. It describes the concepts in the domain, the relationships that hold between those concepts (Horridge, 2009) and provides an explicit specification of conceptualisation including descriptions of the assumptions regarding both the domain structure and the terms used to describe the domain (Su, 2002). Thus ontologies are central to semantic as they allow harmonization of terms and relationship. However, the adoption of multiple strategies for mapping incongruent information as in disparate models, relational schemas and metamodels to ontology often times lead to problem of anomalies in their interpretation and greater complexity in the semantic interoperability. As ontologies and schematisation play a central role in the development of the semantic (Noy, 2012), the family of description logic embedded in ontology designs present specific functionalities that allow answering complex queries, particularly conjunctive queries over RDFS and RDF graphs commensurate with large instance sets generated by models (Rosati & Almatelli, 2010). Additionally, query answering within ontology abstractions allows the execution of these constructs and filtering for traceability. The constructs represent search conditions with respect to the intentional constraints and goals of the motivation. The theoretical principle for model validation is grounded in chapter 5 and this lays the foundation applied in the transformation of models to ontology in chapter 7.

1.4 Research Hypothesis

In consideration of the research question, a hypothesis is developed. The hypothesis is founded on the background that extensive empirical researches and validation methodologies have been engaged in

an attempt to validate EA models. The hypothesis explores a new dimension in these efforts of which the research will either confirm or refute. Thus the primary hypothesis under test is;

> By incorporating validation attributes as an extension into Enterprise Architecture models, it is possible to validate the elements of the model in relation to the constraints define in its associated motivation.

To enable thorough discourse and testing of the hypothesis proposed, the following sub distinctions are derived to allow successive and methodological description of the approach;

i. The adoption of an independent Enterprise Architecture Modelling Language with an open source, cross-platform tool and editor. While this attribute would not only facilitate the extension of its functionality, it would also support the description, analysis and visualisation of architecture within and across business domains. Additionally, the capability to model different perspectives associated with each stakeholder can be explored. Ideally the modelling language should also be able to capture the varied features that characterise most EAF layers.

ii. The quality of models can be improved if constraints are encapsulated into the taxonomy to ensure that validation is achieved. Because constraint specifications can be applied to evaluate the model formulated as a set of conditions or predicates using rules based on scenarios or sometimes symbolic algorithms, modelling with constraints has continued to be dominant as a modern approach for concept formalization. With regards to enterprise modelling, a key open issue is how to come up with a set of models for the enterprise that are amenable to rigorous analysis and simulation. Specifically for effective articulation of the properties of the system-to-be, the design of business, information and technology layers require a means for assessments. A number of formalisms for assessing qualities such as performance, reliability and conformity have been proposed for EA (Kulkarni et al., 2013). However, these proposals in practice are constrained as they do not allow the modeller to express uncertainty with respect to the design of the considered system (Johnson et al., 2014). However, in contemporary businesses, the high rate of change in the environment often compounds uncertainties about future characteristics of the system. So significant are these effects that ignoring them usually results in serious problems. Thus invariant conditions that must hold for the system-to-be modelled or queries over objects described in such a model require ultimately constraints within the architecture framework in order to allow its validation. Such propositions have been asserted by several authors, confirming that using constraints derived from a broad range of

motivation specified in requirements and use of principles underlying the assessments and goals is capable of leading to significant increase in the quality of the model (Moss 2007). Be that as it may, there is a mixed body of literature that suggests that application of constraints can best be used in combination with other techniques (Apt, 2003; Xu, 2014). This work is grounded on these premises and leverages its hypothesis on the preposition.

iii. A Metamodel transformed into ontology with associated motivation constraints can allow querying and systematic validation to be executed on the generated resource description framework schema. Ontologies have been used in several research domains to offer the means to describe and represent concepts of information sources. Several approaches and repositories that store ontology schemas, triples and their instances have been proposed (Kumar, 2013; Boury-Brisset, 2003; Heflin et al., 2013). Consequently, defining a query language to support ontology-based repository though has become a challenge for the EA community, formalising the transformation from model to ontology provides a way to define the semantics of validation proposed. A prototype of the implementation using a case study is developed in this research.

iv. A case for formalization of Enterprise Architecture models and transformation to ontology could lead to the integration of models created from different viewpoints through a common vocabulary. This would harmonise the variant perspectives into a whole, expose gaps, overlaps and ensure comprehensive validation of the consolidated taxonomy. Based on an in-depth evaluation of existing approaches to knowledge acquisition and monitoring cognitive processing in semantically complex domains (Frederiksen et al., 2013), models have been decomposed and transformed to ontologies so as to support the integration task. The use of mappings between models and ontologies combined with the establishment of unified descriptions of the relationships allows validation through interoperability and traceability (Calvanese et al., 2002).

1.5 Research Methodology

Pragmatic approach to scientific research involves the use of a method that appears to be best suited to the research problem. To decipher this best suited approach, a consideration of the artefacts, techniques, procedures, limitations typically associated with the method and complementariness play a critical role in determining a choice of methodology. While Quantitative, Qualitative and Natural Science Research methods are commonly adopted in scientific researches, the one approach that specifically addresses gaps that exist in many academic researches, particularly in the management and information systems disciplines is the Design science Research method. The Design Science

Research (DSR) is an enhanced complementary methodology of the more prevalent behavioural science research paradigm as it produces clear contributions to knowledge base in the form of constructs, models, methods, and instantiations (March and Smith, 1995). This attribute contrasts with many other research methods particularly tithe Natural Science Research method which though appropriate for the study of existing and evolving phenomena are deficient in the study of problems that require creative, inventive, and innovative solutions. It is also opposed to Explanatory Science Research method as the academic objectives are of a more pragmatic nature (Van Aken, 2005).

For this major consideration, Design Science Research is particularly suitable for the investigation and understanding of Enterprise Architecture and Motivation as it involves the design of novel or innovative artifacts and the analysis of the use and/or performance of such artifacts to improve and understand the behaviour of aspects of Information Systems (Kuechler & Vaishnavi, 2008). Therefore, Design Science Research is deployed in this work as researches in these disciplines are often seen as a quest for knowledge and improvement of human performance. According to Van Aken, the main goal of design science research is to develop knowledge that can be used to design solutions for identified field problem. This is further emphasized by Gregor and Hevner as the main purpose of design science research is to achieve knowledge and understanding of a problem domain by building and application of a designed artefact (Gregor and Hevner, 2013).

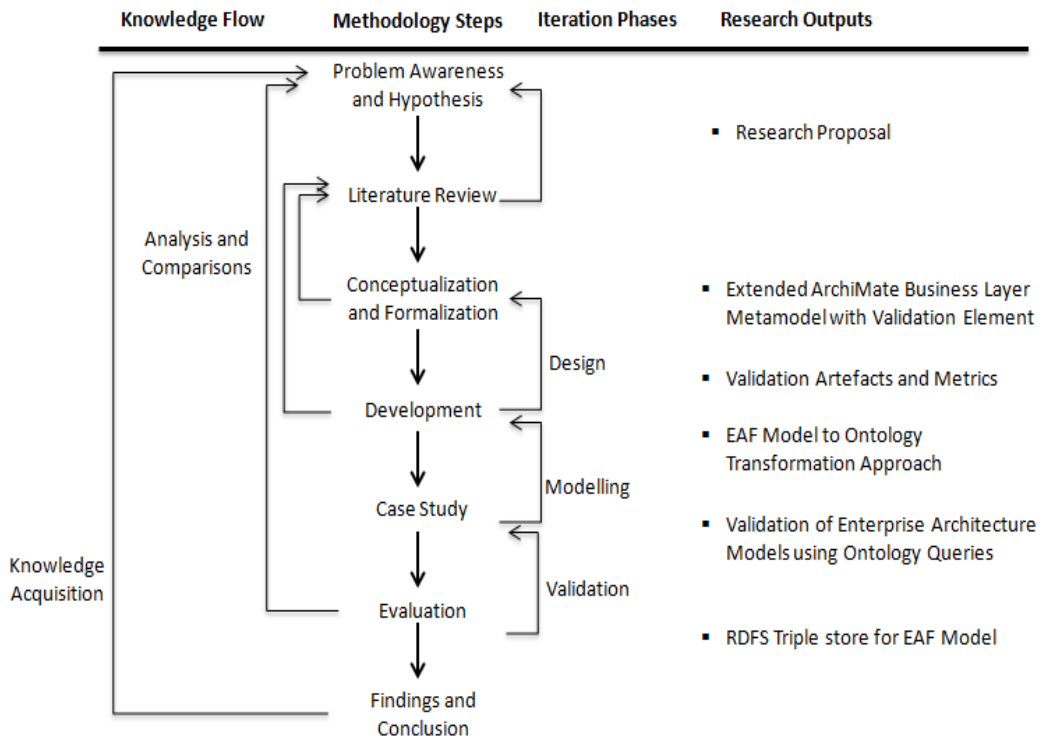

Figure 1-1: Research methodology overview

Figure 1-1 brings this research into a clear perspective and identifies the various activities that constitute the conceptual reviews, methodology development, findings and closure. In consideration

of the three cycle view of design science research guidelines proffered by Hevner (Hevner, 2007), and contemporary approaches proffered by other researchers (Alvarez et al., 2008) the following design cycle is construed;

i. Design as an Artefact – Production of viable artefact in the form of a construct, a model, a method, or an instantiation. This is achieved by extension of the Archi construct.
ii. Problem Relevance - Development of technology-based solutions to important and relevant business problems. This is achieved by transformation of models to ontology.
iii. Design Evaluation - The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods. This is achieved by use of two case studies.
iv. Research contributions - Effective design-science research provides clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies. This is expressed by the outcome of the research.

1.5.1 Structural Review of Literature

Applying the concepts of DSR in this research, three phases are identified. The first phase delves into structural review of EA literature and elaborates on decomposition methodologies to enable the identification of current concepts and relationships for the development of validation techniques. It also expounds on the extensibility of modelling tool with validation components and constructs of the primordial source code required for enhancing the capabilities of the modelling language. Layers of EAF are identified in conjunction with their design and functional components in selected domains. This also sets the background and presents descriptions of components from the domain and specific notations that specify the intrinsic nature of EA artifacts. The section determines associations, motivation, business behaviour and constraints.

Review of existing methodology includes composition and structure of model designs, aspects of the domain and functionalities. It focuses on identifying validation metrics for validation and looks at provisions provided by the Control Objectives for Information and related Technology (COBIT, 2014). According to Alvarez et al (2008), this is the most important factor for experimental research validity in EA. Results analysis and discussions are based on this substantiation. The review also studies comparisons and explores the mapping of the metamodel to instantiations, mapping of the model components to respective validation metrics to be applied. It also identifies ontology methods through which the validation metrics can be mapped and implemented on the model components, to classes and slots of the ontology and how validation can be achieved.

1.5.2 Development and Critique of Knowledge

The second phase of the research deals mainly with the design of experimental artefacts and is a two-stage activity namely; structural which deals with design constructs; and strategic which is concerned with manipulation and control of validation logic. These stages are connected with the research plan to explicate the research problem. The design construct entails the extension of an EA modelling language with validation elements semantic. Annotated models are created in the first stage to allow validation of motivational abstractions and to derive taxonomies for traceability across the Business and Motivation layers. Transition of the model to a Resource Description Framework Schema (RDFS) is implemented. This entails the modularization and analysis of the current state or requisite and gradual but systematic redesign to encompass components, relationship, constraints and generation of test data. Extensive logical paradigms are created to explain ideologies using universal modelling languages and other tools supported by the domain.

Within this phase, the analysis and implementation of two case studies are considered. To exemplify the decomposition of the model with the new extended semantic; the University of West London Student Internship Project (UWL-SIP) case study and the University of Middle England (not actual name) Laptop Loan Scheme (UME-LLS) case study are used. Test procedures are developed using BDD concepts and test scenarios applied to the domain. The codification and semantics are developed, implemented. Input and results are collected and collated as data. Analysis and extraction of testable business behaviour and scenarios are carried out. The business behaviours are also decomposed to identify artefacts and relationships for incorporation into model instances. The case studies exemplify pragmatic application of the transition process; from the creation of the models at the business layer and motivation extension to the transformation into autonomous ontology for querying. The information model, organisational model, functional, service and process models are also designed. Business behaviour from the perspective of a stakeholder, the relationship to various artefacts in the EA with derivation of test basis for validation is developed.

Unlike conventional development testing which can be iterative, EA validation is terminated once the determinate metrics outputs a result that asserts the state of conformance or non-conformance of a specified motivation. Typical exit criteria consist of thoroughness measures through conformity, traceability of dependency, functionality, reliability through maintenance of integrity and availability of artefacts and services (Alvarez et al, 2008). These attributes are synonymous with the validation elements defined in this work. Though in principle, EA validation can never be conclusive as enterprise concerns are not static and are constantly evolving with new innovations and trends, outcome would confirm that validation of EA can be deemed adequate if there are least possible artefacts in the model such that the concerns of the stakeholder are completely realised.

1.5.3 Analytical Methods

The research assumes the constructivist view, progressing iteratively to build on knowledge gained from one stage to another through evaluation of the models, queries, results and findings thus epistemologically validating the hypothesis of the research. It determines if by decomposition of EA business behaviour and design of the model with validation annotations, the components of the model instance can be validated and traceability achieved through ontology mapping and query. It contests the supposition that EAF validation is intricate due to existing inadequacies and gaps and emphasizes ontology transformation and formalisation as the integrant that facilitates consistency, congruency of notations and common semantics for substantiation. It advances that validation can be applied in these circumstances to allow maintenance of a common vision shared by the business and IT, thus enhancing continuous Business/IT Alignment. Some practitioners have suggested that building business case for any architectural solution requires solutions that can be mapped to business and organizational drivers (Schekkerman, 2004). As a high success rate has been achieved in producing quality software products sequel to systematic testing methodology and agile development approaches, the success of this research would affirm that this sort of cohesion and collaboration which leads to modelling for effectiveness does exist within the domain of enterprise architecture.

1.5.4 Evaluation Method

The evaluation of the methodology and contributions of this research involves the adoption of two case studies that relate very specifically to the methods and outcome of this work. Modelling and motivation criteria are used to plan the appropriate testing technique for assessing the artefacts and conclusions of the outcome of the research against the research goals and objectives. This design science research adopted in this work produced artefacts from the extended Archi construct, validation elements, model to ontology workflow transformation process and the semantics for querying the ontology. In terms of validation of these artefacts, the following steps have been undertaken:

i. Formalisation of concepts for the enterprise architecture modelling extension.
ii. Determination of layer of EAF abstraction and artefacts that need to be evaluated.
iii. Establishment of mapping metaphors and triples for transformation to ontology.
iv. Determination of how the ontology will be queried and traceability achieved.
v. Use of two case studies to exemplify and test the hypothesis.
vi. Comparison of the expected and actual behaviour of the query results
vii. Examination of the quantitative results collected in form of RDFS and graphs to ensure that the appropriate patterns are obtained.

In terms of evaluation of the research itself, a strategic framework for evaluation was formulated. The strategic framework is based on several valuable principles of the Information systems Design Theories (ISDT) explained in section 5.4.1 and serves the purpose of evaluating the research

outcomes as well as improving the understanding of unstated evaluation implications in the case studies adopted. Drawing upon the above principles, a strategic framework is formulated by choosing prominent alternatives that describe *when* evaluation takes place, *what* is actually evaluated, and *how* it is evaluated (Venable, 2010). *What* is evaluated is extended to include the granularity which specifies (a) whether the individual artefact was retrieved, (a) whether the business function which involved the artefact was completed, and (c) whether the completed task had a valuable impact on the associated goal or motivation. The framework is applied on the three distinctive validation levels specified in section 5.2 and grounded by principles for model validation rules also defined in section5.2.

1.5.5 Limitations of Design Science Research

Some distinctiveness of the Design Science Research from other methods in terms of its focus and trajectory also constitute its limitations in some ways. Of these differences, five are most prominent. Firstly, DSR emphasizes the domain in which the design activity will take place, placing a premium on innovativeness within a specific context. This contrasts with most other research methods which emphasize increased understanding of design methods often independent of the domain. Secondly, the domains of study for DSR have typically been the information and computing technologies as opposed to a broader scope of subjects. Thirdly, DSR has a closer affinity to disciplines such as computer science, software engineering, and organization science rather than with other cognitive science and professional fields. This implies that internal design cycles are at the centre of design science research projects. Fourthly, Design Science Research is motivated by the desire to improve the environment by the introduction of new and unique artifacts and the processes for building these artifacts. Thus the applicable domain consists of people, enterprise and technical systems that need to interact and work towards predefined goals. Thus a limitation is that the iterations of DSR would need to initiate a procedural context that must not only provide the requirements for the research but also define acceptance criteria for the final evaluation of the research outcome. Finally, Design science draws from a vast knowledge base of scientific theories and methods that provide the foundations by which the research is grounded. Thus all models created have limitations and are subject to the validity of their underlying theory and assumptions.

1.6 Research Contributions

The thesis contributes principally to the validation of enterprise architecture models in the following ways:

RC1: Extension of metamodel of Business Layer of EAF with Validation Element: The ArchiMate EAML is extended with validation capabilities by copulating additional constructs within the Archi. This provides a capability for expressing metamodels and models in a form that allow motivation aspects to be associated with business architecture artefacts. The models

encapsulated with validation elements for constraints and motivation is transformed to ontology description schema with that virtue thus allowing validation to be performed using query semantics.

RC2: Development of EAF model to ontology Transformation Approach: This involves modelling of EA from varied perspectives and with the extended validation element. Validation of a single model is supported by this approach. This also facilitates the composition of multiple models or heterogeneous models created with multifarious EA tools into a single ontology thus improve validation of the EAF in a more comprehensive and holistic manner. Furthermore, the clarification of traceability for the Enterprise Architecture artefacts through the use of ontology filters and logical reasoners allow dependencies and effect of change to be more apparent. The approach also facilitates clarity in the presentation of EA model in terms of goals that are required and business artefacts that constitute the processes needed to achieve those goals. In effect, it aids alignment of business strategy with goals as well as identification of gaps and overlaps in processes.

RC3: Application of domain-driven design and object-oriented analysis concepts in the Validation of EAF models: Domain-driven design and object-oriented analysis concepts are formalised to develop the semantics that describe the queries for the EA validation. The construct is expressed in simple user stories associated with business behaviour of the models created. Domain-driven design and object-oriented analysis concepts applied in a formal way are used to specify the query for the constraints specified by motivation. The queries adapted using SPARQL semantics allow interrogation of the RDFS to obtain results that can be compared against the associated goals, establish traceability and ensure alignment. By implementation of these concepts, language semantics is built with preconditions and post conditions. Evaluating the result yields three outcomes; (I) values that allow comparison to ascertain if the tested goal is realized, (ii) component traceability and (iii) reusable artefacts for further testing on subsequent validation iterations.

RC4: Validation of Enterprise Architecture Models using Ontology querying methodology: Contemporary approaches that have been preferred as a means of validating EAF and models have been maturity matrices, balanced scorecards and reference models. These approaches which are based on qualitative evaluation are very subjective as they are often susceptible to many inhibitions such as user bias, levels of respondent's discernment and sometime organizational intricacies. This research presents a contribution that is logical, objective and targeted based on input and output artefacts that must adhere to set constraints and business rules.

RC5: RDFS Triple store for EAF Model: EA models transformed to ontology provides the capability to create a unified store house for triple patterns, conjunctions, disjunctions, and RDFS. Triple stores are incrementally developed with each transformation to ontology and can enhance deep querying and traceability within the EAF. This also enables the development of regression testing of EAF models thus improve the overall quality of the framework.

RC6: Model-Driven Validation Approach: Ultimately, a Model-Driven Validation Approach (MDVA) is contributed. MDVA validates a model iteratively by testing primarily elements and attributes of the model against goals and constraints in its motivation extension. The MDVA improves the quality and design of the model through goals to component association while simplifying the traceability process. The validation scenarios for MDVA describe the behaviour and attributes of the component to be validated in order to realize set motivation goal. Granted that the methodology is adhered to, it also ensures better conformance to user Goals. The MDVA workflow specifies both the behavioural and the structural attributes of the EA components. With MDVA, validation themes are defined by a set of motivational goal specifying the components to be tested in the model. The validation metrics specify what types of test are to be carried out on the components and the expected results while traceability is established by exerting associations and relationships.

1.7 Thesis Outline

The chapters that constitute this thesis and their chronology are shown in Figure 1-2. The summary of intentions is given in Chapter 1 and includes Challenges of Enterprise Architecture And Validation, Research Aims and Objectives, Research Question and Subquestions, Research Hypothesis, Motivation, Research Justification and Rationale, Research Contributions and Research Methodology. The next three chapters 2,3 and 4 grounds the themes in their academic context through literature reviews to present background information, analogies for EAF, validation techniques and EA Modelling Languages respectively. Following these reviews, Chapters 5 and 7 grounds the research question within the considered rationale for the study taking into consideration theoretical principles for model validation and formalization of theories for EAML validation extension. These chapters present a critical review of the whole conceptual and methodological framework for EA modelling. Chapter 7 continues directly to the methodology and presents an articulate description of ontology, metamodel and model transformation. In the narratives presented in this chapter, the rationale for the contributions proposed is well grounded. At this stage, the research work process and timeline are also evaluated to ascertain that there is a clear structural succession. The chapter defines and presents the methodology proposed, extensions made and the transformation to ontology. Chapter 8 demonstrates the approach proposed and exemplifies the principles in two case studies, where decomposition,

formalization, modelling, transformation and querying are effectuated. Finally Chapter 9 evaluates the research and presents findings. These are evaluated and benchmarked with proposed hypothesis. It also concludes the research and presents opportunities and areas for further investigation.

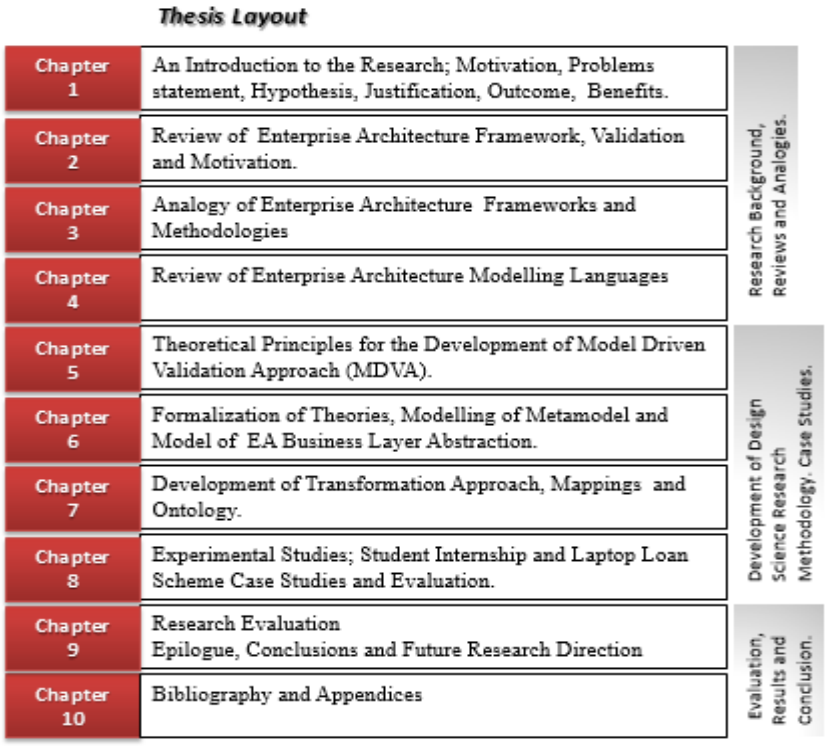

Figure 1-2: Thesis structure and chapter description

1.8 Summary

The thesis deals with one of the various challenges of EA validation. The complexities and problems created by use of heterogeneous modelling languages and domain specification in EAF is a reality that if not addressed, can only compound the understanding of its ideologies. It has been suggested that the solution to these challenges may be attained by the introduction of validation and testing techniques for EAF and its models (Chapurlat & Braesch, 2008). The need for formalization is also considered by many authors as fundamental so as to maintain standards (Gangemi et al., 2006). This research is scoped within the Business Layer of the EAF so as to address these challenges from the roots. The rationale for this also is based on the fact that almost all EA modelling languages consider in their denotations the modelling of the business processes. The variant and diverse modelling languages deployed today somewhat attempt to describe metaphors, components, relationship and artefacts that constitute this layer hence most of the constructs that extend to other layers of abstraction are inherited from this layer. Other layers such as Information, Data and Technological infrastructure would need the same kind of attention as an area for further research.

2 CRITIQUE OF KNOWLEDGE ON ENTERPRISE ARCHITECTURE FRAMEWORKS AND MOTIVATION

This section begins with a general introduction to Enterprise Architecture and presents theoretical foundations of the work with respect to definitions, scope, purpose and reasons for need to validate EA. Commonly uses EAF are selected for articulation and the rationale for selection is presented. This is followed by a literature review of the considered EA frameworks and validation techniques applicable with EA frameworks. The selected frameworks are subsequently discussed as needed to ground the contributions of the research within the context of these already existing methodologies. It is worth stating that though an adept attempt is made to describe the basic principles and in many cases structure of these frameworks, the overviews presented in this work cannot replace the original extensive elaborations provided in documentations about the frameworks. Accordingly, the discourse for each methodology consists of two subsections. The first section presents an outline of the Enterprise Architecture Frameworks, asserts the foundation of the taxonomy and how their instantiated models are deployed to support business behaviour within organisations. This is followed by the second section with states how validation is carried out for the given EAFs. Analogies and challenges associated with the selected frameworks are proffered to pave the way for the presentation of the Model Driven Validation Approach proposed in the subsequent chapters. Following these articulations is a presentation of empirical discourse on the role of motivation within enterprise architecture. The reviews presented in this section are also collated and compared. This section is associated with theoretical principles which pertain to the right subject group where the contributions of this work extend.

2.1 Definitions of Enterprise Architecture

The proliferation of Information System and its wide spread use is a prevalent anomaly that constitute major decisions in many organisations. Although the concept of Enterprise Architecture (EA) has not been well defined and agreed upon within many organisations, EA has continued to be developed to support information system development and enterprise engineering. While there are significant differences in most EAs in terms of content and nature, most are also incomplete as most taxonomies represent specific concerns and process aspects of the enterprise for which they are intended. Consequently, definitions of EA vary also considerably depending on their purpose and domain. What the definitions seek to express in a rudimentary sense therefore is a means to explain how their symbiotic embracement of information technology transforms their enterprise with respect to their organizational culture, vision, structures, business objectives, business processes, roles, behaviours and the relationship with each other. This leads also to many disconcerted interpretations. Thus in an attempt to introduce specifications or scope the content of their information and knowledge automation, some organizations refer to their process of development of methodical information technology specifications, models, guidelines and notations, as Information Technology Architecture (ITA), Information Systems Architecture (ISA), Enterprise Information Systems Architecture (EISA)

and the alike. But there are clear distinctions between ITA, ISA, EISA and the alike with EA. The IEEE 1471 -2000/ISO/IEC 42010 standard defines EA as the organization of a system embodied in its components, relationships to each other, environment, the principle guiding its design and evolution (IEEE, 2000).

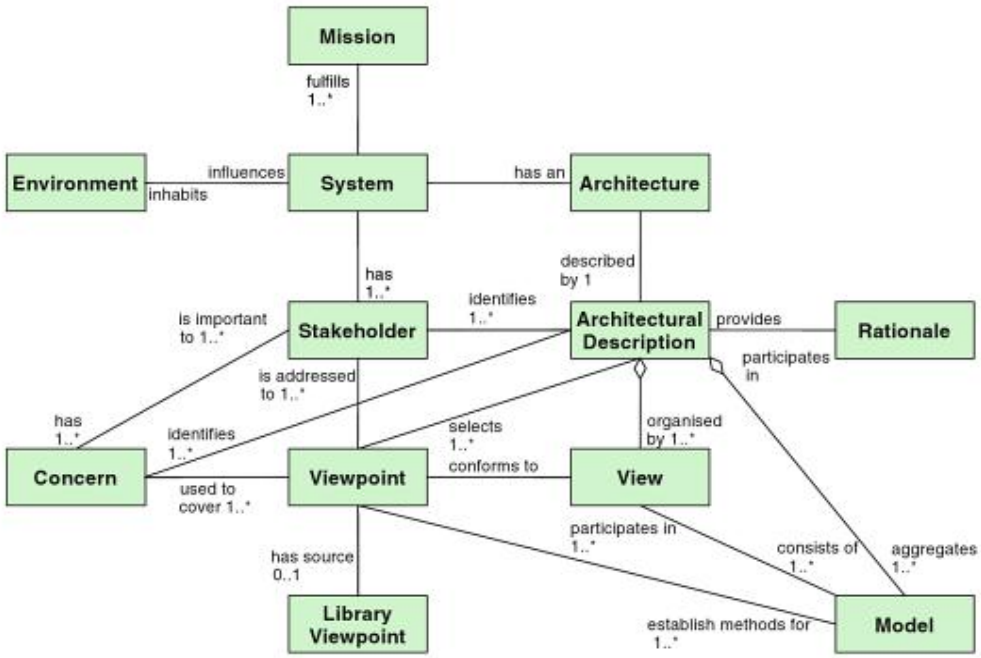

Figure 2-1: Conceptual model of Architecture description (IEEE, 2000)

The standard as depicted in Figure 2-1 provides a conceptual framework and a set of definitions for key terms such as acquirer, architect, architecture description, models, life cycle, system, system stakeholder, concerns, mission, context, views and viewpoints. Even the substantive specification of "architecture" by IEEE 1471 tends to focus mainly on software intensive systems and composite systems in the context of computing. Though one can compare this definition to many types of frameworks, it does not attempt to establish the specifics nor standardize the process of developing the architecture. It does not also include recommendations for modelling languages, methodologies and validation. However, IEEE1471 provides a number of valuable concepts and terms of reference which reflect the generally accepted trends in practice for architecture descriptions. It is important to note that architectural descriptions that are compliant with IEEE 1471 can be deployed to satisfy the requirements of many other standards. In this regard, Lankhorst, a renowned author defines EA as a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure (Lankhorst, 2013). Though this definition is more coherent and all embracing, one would notice that there is no reference to the *"realization of goals and motivations"* that drives the organization in the first place. Consequently, many practitioners have scoped this broad description and offer their own interpretations that fit their jurisdiction. Thus some well known definitions of EA are as follows;

In the case of Enterprise, the definition of architecture would be;
"that set of descriptive representations (models) that are relevant for describing an enterprise such that it can be produced to management's requirements (quality) and maintained over a period of its useful life (change)" (Zachman, 1996).

"EA is about understanding all of the different elements that go to make up the enterprise and how those elements interrelate" (TOG, 2013).

"EA is a strategic information asset which defines the business mission, the information necessary to perform the mission, the technologies necessary to perform the mission, and the transitional processes for implementing new technologies in response to the changing mission needs" (USA Federal CIO Council).

"EA is the holistic expression of an organisation's key business, information, application and technology strategies and their impact on business functions and processes. The approach looks at business processes, the structure of organisation and what type of technology is used to conduct these business processes" (Mega group Inc, 2013).

Gartner defines enterprise architecture as;

"The grand design or overall concept employed in creating a system, as in the architecture of a city or a customer information system; also an abstraction or design of a system, its structure, components and how they interrelate." (Gartner, 2013)

Also, by application of EA in another dimension, Gartner defines EA yet again as;

"EA is a family of guidelines (concepts, policies, principles, rules, patterns, interfaces and standards) to use when building a new IT capability."

All these broad definitions from renown authorities affirm that Enterprise Architecture is widely accepted as an essential mechanism for ensuring agility, consistency, compliance and efficiency in organizations. Though there is no common agreement on its composition, artifact types and dependencies, most authors agree that it consists of business, application, information and technology perspectives. The business perspective refers to the processes and standards by which the business operates on a daily basis; the application perspective connotes the interactions among the processes and organizational standards; the information perspective depicts and classifies the raw data that the

organization requires for efficient operations while the technology perspective embodies the organization's hardware, operating systems, programming, and networking solutions. Since information systems remain consistently dynamic, there is a change management perspective which presents views of fundamental construct of analysis though not often acknowledged distinctively as such. Therefore within this research, there is a need to specify what EA will imply and the scope. This is stated as;

> *"EA consist of coherent principles, methods, and models used in the design and expression of the organisational structure, business processes, information and relationship with each other so as to realize the high-level goals and policies of the organizations (value) through low-level implementations of systems and technology."*

The rationale for this contemporary definition is that there is need to include terms that specify the realization of high level goals and policies within EA deliverables as this is absent or very inarticulate in many other definitions. To achieve this, validation must be part of the intentions of EA as many organizations strive to successfully transit to corporations that utilize information technology strategically. EA is a key driver that can facilitate the actualisation of this change by decomposing and then aggregating business processes and strategies into models and layers of abstraction (Sessions, 2007). These layers are then integrated into a framework that specifies the behaviour, attributes and relationships between the components and the layers. Depending on the domain, some organisations focus the modelling of their EAF towards products, services, processes and applications such as SEAM (Urbaczewski, 2006), while others specify standards to guide the principles of the design such as GEAF, TOGAF and DoDAF. However, there has been no common agreement on which architectural layers, which artefact types and which dependencies constitute the core of enterprise architecture (Carla, 2005, Weston & Defee, 2004). In consideration therefore of one framework against the other, overlaps, inconsistencies and gaps are commonly identified. Most EA modeling techniques have disparities in their semantics which result in complexities in the implementation of their models.

Though many EA practitioners recognize four facets of EAF and suggest that it comprises of Business, Information, Application and Technology architecture (Salmans et al., 2010), others in their review identified eight perspectives in which EA alignment can be achieved (Venkatraman et al., 2010). In a bid to broaden EA framework perhaps to achieve more comprehensiveness (an inspirational inclination which many enterprise architects and authors have great proclivity to), many other practitioners have created fusions of perspectives comprising of combinations of pairs of adjacent perspectives. This is well documented in the literature of Coleman and Papp (2006). In many efforts to define EAF with descriptions that specify perspective, harmonization and alignments, the issues of validation are completely ignored or at best remain rudimentary. The postulations do not consider the behavioural attributes of the model's components as a process that should undergo test. Most authors

believe EA models are not reusable and are designed to actualize a specific goal after which it is archived or at best used as a reference compendium.

Taking all these definitions into account, many authors argue that none of the Enterprise Architecture Frameworks can completely view the enterprise in its entirety as comprising of business objectives, business processes, roles, organizational structures, organizational behaviours, information, software applications, computer systems and the relationships between these various entities (Chen, 2008). Though efforts continue to be made towards standardization (TOG, 2013; OMG, 2013), many frameworks are still specific in scope and purpose and apply to specific domains, generally weighted towards planning and business process analysis without commensurate emphasis on validation and change management. The following sections present a more analytic review of the purpose, relevance and structure of many of these contemporary EAF.

2.1.1 Model, Metamodel, Framework and Enterprise Architecture

Central to the theme of this work are concepts relating to model, metamodel and framework hence a clear and precise description is presented to preclude ambiguity in the application of the terms. A model simply refers to a collection of related components within a domain that is instantiated from a metamodel with the aim to explicitly provide functionality wholly or in part for the actualization of specific goals. In this regard, a model must highlight the properties of the metamodel and must conform to its boundaries and constraints. Therefore, models describe the logical business functions or capabilities, business processes, human roles and actors, the physical organization structures, data flows and data stores, business applications and platform applications, hardware and communications infrastructure of a case domain.

A metamodel on the other hand consists of explicit description of constructs and constraints of a specific domain. Though metamodels have also been described as comprising of a formalized specification of domain-specific notations which adhere to strict rule set for developing EA (Gudas & Lopata, 2007), metamodel consistently represents relevant artifacts of enterprise architecture both from a business perspective and from an Information Systems perspective. Thus it can be said that while models provide the reasoning about the systems being designed, metamodels specify the language for expressing these models.

In contrast to models and metamodels, a framework defines how to create and use enterprise architecture. It specifies the principles and practices for creating and using the architecture description of a system by segregating the architects' description into domains, layers and views. To enable the documentation of views, a framework may consist of metamodels and models with artefacts that specify a tripartite structure to guide its configuration. These are often expressed as descriptions of the architecture from several viewpoints composed of entities and relationships; methods for designing

the architecture defined by its objectives, inputs, phases and outputs; and guidance on the organizational structure, actors, policies, skills and experience.

2.1.2 Relevance of Enterprise Architecture

The current trend in organisations is a renewed focus on Business Process Management (BPM). BPM allows businesses to adapt promptly to critical changes in their business process strategies in combination with technology through implement, orchestration and execution. The ability to trace business strategy straight through to execution is ensured by alignment and traceability within Enterprise Architectures. Thus a vital role of EA is to provide the methods that ensure delivery of growth oriented projects for this business process management. To achieve this, there is need for a means that facilitates the substantiation of the various artefacts that make up the business processes, IT strategies and motivation. Building business skills into the IT organisation and IT skills into the business process is crucial. Also the need for measuring service levels and performance must also be emphasised. Leading organizations use a business strategy driven architecture approach that focuses on translating the key components of the business strategy into a future state vision and an architecture road map they can implement.

Therefore Enterprise Architecture provides this means to model this integration with other strategic planning disciplines and ensure that the long-term vision of the business is preserved as the enterprise builds new business capabilities and improves on old ones. Enterprise Architecture is designed to ensure alignment between the business and IT strategies, operating model, guiding principles, the software development projects and service delivery. By taking a global, enterprise-wide, perspective across all the business services, business processes, information, applications and technology, Enterprise Architecture ensures the enterprise motivation are envisioned in a holistic way across all endeavors and are deployed efficiently.

2.1.3 Rationale for Enterprise Architecture Validation

Many practitioners favor the practice of Enterprise Architecture for many reasons. As an analysis tool, it provides the capabilities for abstraction and modelling all levels and perspective of the enterprise's concerns (Lankhorst, 2013). While other authors agree that EA is a planning tool that translates strategic thinking into architecture roadmap of future development and integration (Greefhorst & Proper, 2011), there is no doubt that EA also assists in the analysis and explicit plotting of the key relationships and dependencies between the business services, business processes, applications and technology. As a framework that supports decision-making, selection and justification of strategic development options, it provides support for designing industry best practice approaches, guidelines, and reference models (Wan et al., 2013).

Therefore it can be said that the rationale for validating EA is based on the enormous benefits it yields to organizations. One of such major benefits is the provision of alignment methodology for essentially bridging the gaps between business strategy and IT delivery. It does this simply by furnishing business managers with a non-technical overview of the supported enterprise operations (Sessions, 2007). The need to ensure that EA can be validated is also related to change management obligations to provide a framework for synchronizing and coordinating development activities across multiple development initiatives (Coleman and Papp, 2006). Effective management of change is dependent on a clear understanding of its impact and outcome of proficient validation methodologies of EA. As a governance tool, validation ensures that the holistic architectural design is consistent with the enterprise's blueprint of principles, standards, patterns, policies, guidelines and reference models.

It has been suggested that lack of focus on enterprise requirements, common direction and synergies is a characteristic reason for the existence of gaps in architecture and leads to complex, fragile and costly interfaces between applications (Roth et al., 2013). Thus the capability to adjust rapidly and adapt to new business situations can only be assured with efficient and strategic understanding of the impacts of the artefacts and components across model entities that realize the target Enterprise Architecture. With the need for alignment between IT and business, inability to validate EA in order to respond to challenges driven by business changes can led to vague visualization of the current and future target EA vision.

Another rationale for validating EA is that it ensures transparency and objectivity in modelling enterprise architectures. In addition to the reduction of model complexity, validation ensures increased reuse of existing artefacts. Validation allows comprehensive testing of the models thus expose the impacts of change. It also enables the stratification and dissemination of only required and critical knowledge relevant to the deployed solutions exposing inharmonious integrations, incompatibilities and adverse interoperability.

2.2 Review of Enterprise Architecture Frameworks and Methodologies

A number of contemporary architectural frameworks are in use today, which portends to solve specific needs or concerns of the enterprise. Though some frameworks overlap, address similar views or may even be deficient in several aspects, they provide a means to implement and integrate the building blocks within the organisation (Schekkerman, 2004). Several comparisons and analysis have been made between enterprise architecture frameworks by Iyer et al (2004), based on support and design; Chen (2008), based on mappings and relationships, Tang et al (2006), based on high level goals, inputs and outcomes and Schekkerman (2004) based on complexity and added value. In this research and in consideration of the EA definition stated, a different analytic view for EA models is presented based on the composition, structure and validation capabilities. In order to establish a common ground for the determination a suitable EAF for extension of its construct, a rationalization of preferences is provided.

2.2.1 Rationale for Selection of EAF

The major objective of Enterprise Architecture is to provide architectural principles, frameworks, methodologies, processes, tools, knowledge base and techniques that can support the mission of the enterprise. These vestures are also expected to facilitate the alignment of artefacts, ensure traceability of relationships, localization, harmonization of interactions and visualization with perspectives in order to make the entire enterprise more productive and efficient. In selecting EAFs for review, certain criteria were considered. These include the capability of the methodology to identify the steps necessary to produce each deliverable of EA development or evolution. This is critical as the practitioner needs to be able to easily determine and execute steps necessary to produce a selected goal or motivation. The consideration here is that the methodology should simplify the EA development and evolution process. For frameworks that deploy a variety of modelling tools, compatibility is important. Such methodologies should be broad in scope of coverage in order to be able to support current techniques and technology such that new processes, methods and repositories. This should be proven pragmatically as complete, concise, and proficient in supporting perspective visualization without complexities.

Of relevance are methodologies that have capabilities to validate its models. This is critical as this criterion would conform to applicable theories that can be grounded with this research. As enterprise architectures are expected to be adaptable to change, certainly the extent of that adaptability need validation. This may be possible if the methodology can be formally defined with semantics that is unique, consistent with relationships that describe the underlying taxonomy. This also implies that the EAF should be customizable so as to meet specific standards and practices of the enterprise.

In consideration of the myriad of EAFs in use today, preliminary reviews were carried out on some widely used methodologies including the Zachman Framework (ZF), The Open Group Architecture Framework (TOGAF), Gartner Enterprise Architecture Framework (GEAF), Federal Enterprise Architecture Framework (FEAF), Generic Enterprise Reference Architecture and Methodology (GERAM), Systemic Enterprise Architecture Methods (SEAM), Dynamic Architecture (DyA), Integrated Architecture Framework(IAF), ISO's RM-ODP, ISO/IEC/IEEE Standards, Department of Defence Architecture Framework (DoDAF) and Treasury Enterprise Architecture Framework (TEAF). Their structures were considered as a guide for the selection of EAF for review in this work. The structure also extended to include Enterprise Architecture Management which is the act, manner, and practice of leading the enterprise to improve its architectural environment as well as of obtaining and controlling resources to support enterprise architecting activities. This criterion is clearly reflective in the professional development of the framework and acceptability within the EA community. The precursory reviews indicated that integration principles and methodology, input, tools and techniques, output of integrating structures of few of these frameworks could actually represent a functional, efficient and harmonious enterprise structural environment. These were the ZF, TOGAF, GEAF, FEAF, DoDAF and SEAM. ISO/IEC/IEEE Standards was considered additionally

because it specifies the recommended practice for architectural description of intensive systems. These selected EAF or methodologies addressed holistically and more appropriately the elements of strategy, modelling, the overall EA process, methods and techniques, standards and tools that enable the coordination and delivery of the various elements that constitute the Enterprise Architecture within the organization with consideration for goals or motivation. The following sections describe in details these selected EAFs.

2.2.2 Zachman Framework

The Zachman Framework (ZF) considered as one of the pioneering models in EA domain is based around the principles of classical architecture that establish a set of perspectives for describing complex enterprise systems (Zachman, 2008). The framework originally referred to as the Information Systems Architecture (ISA) for EA is unlimitedly generic. It can be used to classify broad descriptive representations of EAs thus facilitate the analysis of relative architectural compositions. One major characteristic of the concept of the framework set forth by Zachman is its recursive capability which enhances a top-down analysis of phenomenon being modelled (Martin, & Robertson, 2000; Martin et al., 2004). A framework is deemed recursive if it consists of frame's descriptors which constrain the scope for abstraction, allows a consistent treatment of artifacts within that frame and links to other sub-frames (Delgado, 2014).

Thus the Zachman Framework as a recursive logical structure classifies and organizes the descriptive representations of an Enterprise and is significant to the management of the enterprise as well as to the development of the enterprise's systems (Zachman, 1996). Although ZF is an application of framework concepts to enterprises, the framework itself is a generic and logical structure for descriptive representations of models, design artifacts of complex object. It is also independent of any processes or tools that can be used for the description of its artefacts. For this reason, the framework is helpful for sorting out very complex and disparate technologies, methodologies and significant concerns of the enterprise. A very comprehensive and descriptive analysis of the ZF is presented by several articles by Zachman himself (Zachman, 2008; Zachman, 2002) and many other authors (Beznosov, 2000; Goethals, 2003; Lankhorst, 2013; Venkatraman & Henderson, 2010).

It has been asserted that though the ZF denotes a semantic structure, it does not provide an admissible guidance about its implementation processes (methodologies) or tools. As the framework attempts to analyze entire enterprise segments, this results in complex overtures and considerable implications; posing limitations when boundaries are drawn beyond jurisdictional control. Consequently, the instantiated model is difficult to declare or arbitrate. Many practitioners have noted that if the engineering design principles of the ZF are not observed in relation to the primitive cell models, the realization of the engineering design objectives of alignment, integration, reusability, interoperability, flexibility and efficiency becomes a fallacy (Urbaczewski, 2006). The ZF does not provide any standardization on sequence, process, and implementation or testing, rather it focuses on ensuring that

all views are established for system completeness (Lankhorst, 2013). The implementation of the framework reveals that there can be potentially huge inconsistencies as no explicit compliance rules are mandatory. It is also not authored by professionals in Information Technology (Noran, 2003).

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*
Objective/Scope (contextual) *Role: Planner*	List of things important in the business	List of Business Processes	List of Business Locations	List of important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) *Role: Owner*	Conceptual Data/ Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) *Role:Designer*	Logical Data Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technology Model (physical) *Role:Builder*	Physical Data/Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Reprentation (out of context) *Role: Programmer*	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Speculation
Functioning Enterprise *Role: User*	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

Figure 2-2: The Zachman Framework (Zachman, 2008)

Similarly, application of the ZF is an extensive and difficult exercise due the large number of cells and complex detailing within the cells. While some of the cells can be modelled using some standard and well structured techniques, other cells cannot. Realistically, the modelling of some cells in the ZF still remain an open research problem and in particular, well defined modelling language for modelling the technical infrastructures are almost non existence. Therefore, it can be said that while the ZF provides a means for organizing architectural artifacts such as design documents, specifications and models, ZF does not contain concepts that relate to process or methodology hence validation cannot be applied. There are many issues that are critical to EA modelling that Zachman does not address such as step-by-step process for creating an architecture or guidance in assessing architecture's appropriateness or usability. Additionally, the relationships between the different cells that make up the framework are completely ignored. As heterogeneous modelling techniques are used to populate each of the cells including sub details within the cells, it is impossible to adopt common or even identify similarities across the cells in order to allow the mapping of relationships. Thus the fundamental basis of the ZF is the segregation of the enterprise into isolated units.

Rather than promote the development of multiple views of EA based on stakeholders concerns, the ZF assumes that there can be only six discrete viewpoints achievable with six roles namely planner, owner, designer, builder, programmer and user (Figure 2-2). With non-specification of hierarchical levels across the rows that distinguish the viewpoints, symmetry or alignment cannot be realized. The ZF presents contemporary concerns such as security, governance, validation, artefact orientation and change management. Since enterprises evolve, these deficiencies make the ZF distinctively

incapacitated as a prescriptive framework. It has been maintained that though the ZF is fashionable and a conviction of conglomerate affinity, it is founded on a subjective, untested observation thus lacks scientific foundation (Beznosov, 2000; Goethals, 2003; Lankhorst, 2013).

2.2.3 The Open Group Architecture Framework

The Open Group Architecture Framework (TOGAF) created and maintained by The Open Group (TOG) is built based on an earlier framework known as Technical Architecture Framework for Information Management (TAFIM) originally devised by the U.S. Defence Department in 1995. Over the years, several versions of TOGAF have evolved making it increasingly comprehensive and adaptable. Due to this maturation of TOGAF in terms of structural composition, reliance on modularised and standardised existing proven technologies, it is most widely accepted as an approach for designing, planning, implementation, and governance of enterprise information architecture. TOGAF is modelled at four levels to encompass Business, Application, Data, and Technology aspects of EA (TOG, 2012). The core TOGAF contains descriptions of an Architecture Development Method (ADM) and is related to other techniques specified in its Architecture Content Framework (ACF), Enterprise Continuum (EC), TOGAF Reference Models and a Capability Framework amongst other enhancements. A more detailed description and updates on TOGAF are available at their portals.

To a large extent, the ADM describes a method for developing and managing the lifecycle of enterprise architecture, and forms the core of TOGAF (TOG, 2012). It integrates elements of TOGAF specified by the ACF, EC and other obtainable architectural assets to meet the business and IT needs of an organization. While the Enterprise Continuum provides a framework and context which supports the leverage of relevant architecture assets executed in the ADM, the Enterprise Continuum facilitates the categorization of the architectural source material, repositories, reference models and standards within the industry. Consequently, it has been asserted that the architecture design of TOGAF can be a technically complex process (Winter & Fischer, 2007). Though this has been partly contested comparatively, other views actually favour this interrelation and maintain that it essentially demystifies the architecture development process as the framework is loosely coupled with parts that are often well synchronized in terms of terminology and scope (Halttunen, 2005; Hoogervorst, 2004). For this reason also, Jan and Dietz (2008), considers TOGAF as a great starting point for building strategy design and increasing awareness within organizations thus acknowledge the need for enterprise-wide investments in the architecture.

However, there are also pitfalls associated with TOGAF. One of such is the attempt during implementation to execute every phase, deliver each artefact and create all repositories defined by TOGAF. Though this in itself may not be wrong, TOGAF strictly emphasizes as key for success, though not explicitly, the need to make selections, and tailor the framework to the context at hand so as to optimize the generation of real business value. Another pitfall is that in many domains, TOGAF has been adjudged to be very technical-minded and focused on delivery of models. Although

architects need models, technology, instruments, languages, and deliverables to effectively communicate with stakeholders, TOGAF is not wholly specific with respect to generation of documentations. In fact, it provides very little in the way of prescriptive document templates (Sessions 2007; Schneider et al., 2013).

TOGAF allows partial completion of phases. Facets of design in TOGAF may be circumvented, combined, reordered, or reshaped depending on the requirements of the viewpoint (Sessions, 2007). Consequently it is not uncommon to see two different TOGAF-certified consultants end up using two very different processes even when working with the same organization. Because TOGAF is very flexible about the actual archetypes generated, the final architecture might be useable, deficient, or even apathetic. This is sequel to the fact that TOGAF merely describes *how* to generate an enterprise architecture, not necessarily how to generate an unobjectionable enterprise architecture. However this limitation is not restricted to TOGAF alone.

With respect to validation, though TOGAF is incorporated with the ACF as means that expresses content metamodel to provide a definition for all the types of building blocks that may exist within architecture, it has been suggested that the ACF is not flexible enough to adjust to the different contexts found in the organizations. The ACF represents the whole enterprise and that is too much information. In order to obtain effective communication with the stakeholders and participants, the architecture contents should be presented in views that address the particular concerns of each interest group. Thus it has been argued that the ACF is inadequate as a means for validating, measuring and communicating the impact of TOGAF implementation.

This notwithstanding, TOGAF is an attempt towards standardization of best practices and a common language for practicing architects. It does to a large extent adhere to the IEEE Standards 1471-2000 and its contextual usage covers a formal description of a system, a detailed plan of the system at component level and a guide to implementation. Also, it has a good structure of components, their inter-relationships, principles and governing guidelines. As a result of these advantages, TOGAF has a high affinity and acceptability amongst practitioners as it strikes a balance between promoting formal accepted concepts and terminology already familiar to majority of system architects.

2.2.4 Gartner Enterprise Architecture Framework

The Gartner Enterprise Architecture Framework (GEAF) is a practice carried out by one of the known IT research and consulting organizations, Gartner. Gartner believes that enterprise architecture is about bringing together three constituents: business owners, information specialists, the technology implementers to share a common vision of driving business value (Gartner, 2013). The Gartner EA process model, first developed in 1996 focuses on desired goals and how current resources of the organization relate to the desired goals. The focal point is on the destination and the most timely and pragmatic strategy to apply in order to get there.

Though GEAF is widely used by Gartner in many of their enterprise's IT projects, it has been considered by many practitioners as enterprise architectural best practice and not a methodology (Hoogervorst, 2004). This is also acknowledged within the precepts of the approach itself by Gartner. For instance in identifying the drawbacks for effective actualization of EA benefits and possible reduction of risks, Gartner Inc (2013) stresses as key critical success factor the need to engage with the right people and communication. Specifically it emphasizes that "selecting the wrong person as lead enterprise architect, and not engaging business people through effective communication are two of the biggest pitfalls organisations face when trying to establish effective enterprise architecture (EA) programme" (Gartner, 2013). Additionally, contrary to what many may expect pragmatically, Gartner is completely against evaluating the current state of EA first. Gartner reasons that establishing the future state of EA is paramount as the business context focused first on future state provides prescriptive guidance required to accelerate delivery of EA value; an edge which establishing the current state of EA first does not provide. Therefore for GEAF, the future state analysis constitutes the reason for changes while the current state or as-is is necessary only to provide an initial baseline to compare against the future state. Through this approach, the GEAF aims principally to bridge the gap between business strategy and technology implementation by steering the current state to sync with the future state through a logical approach (Gartner, 2013).

Gartner (2013) asserts that the concept deployed by GEAF to bridge this gap is the EA Process Model flow. It is a basic cycle which assesses the future versus current states of the EA. This is usually followed by gap analysis between the two to proffer recommendations. Gartner claims that the approach is cyclic thus should be applied iteratively on the many phases of the development and directed towards process development, migration, governance, organizational and management of sub-processes. However, according to Lankhorst (2013), a major limitation of this concept is that it relies on being able to predict extensively the business needs of the future, composite relationships and the basics of technical conundrum of the future. If the exact future-state, technical complexities and business requirements could be anticipated in advance, perhaps all that would be needed is a design and not architecture (Perkins, 2003; Carla & Sousa, 2005). However, enterprise architecture describes much more than a state as proffered by Gartner. It subsumes requirements at a much higher level including relationship between motivation and business strategy, information and applications, technologies and capabilities (Stanley & Uden, 2013).

Gartner contends that the framework is intended to broadly influence and support investment decisions and organizational change. Thus the GEAF is channelled towards proper organisation, scoping of resources and execution of processes with consistent communication of goals and accomplishments. However, it has been argued that the practice relies too heavily on reference material and continual update of infrastructure documentation (Kaisler et al., 2005) rather than establishing functionalities identifying dysfunctions, duplications, complexities and dependencies.

As a framework, many processes in particular those related to structural composition are often outsourced from other methodologies. Thus by itself, many see the GEAF as providing only paltry descriptive information about information and technological concepts (Morganwalp & Sage, 2004). Though GARNER asserts that their practice is timeless as it is continually being augmented with each client's experience, the current Gartner methodology was not solidified until 2006 after the Gartner/Meta merger (Gartner, 2013). For this same reason, the GEAF is deemed as an ongoing practice for defining best procedures; creation and maintenance of EA based on past experiences to harness formalistic synergy. In summary therefore, the GEAF is about strategy not about engineering and void of a standardised step-by-step process. As it is focused on the destination defined as future state, any architectural activity that is extraneous or contravenes this focus is irrelevant in the scheme.

2.2.5 Federal Enterprise Architecture Framework

The Federal Enterprise Architecture Framework (FEAF) was developed and published by the US Federal Chief Information Officers (CIO) Council in response to industry trend and the Clinger-Cohen Act, 1996, which required Federal Agency CIOs to develop, maintain, and facilitate integrated systems architectures to guide the development of large, complex systems (FEAF, 2006). The principal objective of FEAF is to systematize and promote sharing of Federal information for the entire US Federal Government.

	Data Architecture (entities = what)	**Applications Architecture (activities = how)**	**Technology Architecture (locations = where)**
Planner's View Objectives/Scope	List of Business Objects	List of Business Procesess	List of Business Locations
Owners'sView Enterprise Model	Semantic Model	Business Process Model	Business Logistics System
Desiger's View Information Systems Model	Logical Data Model	Application Architecture	System Geographic Deployment Architecture
Builder's View Technology Model	Physical Data Model	System Design	Technology Architecture
Subcontractor's View Detailed Specifications	Data Definition "Library or Encyclopedia"	Programs "Supporting software components (i.e., Operating Systems)"	Network Architecture

Figure 2-3: Views and Architectures of the FEAF

The FEA perspective on EA is that an enterprise should consist of segments defined as a major line of business functionality or organizational unit. For organizational units, their depth includes not just the technical, but also the business and the data architectures. FEAF defines two types of segments, core mission-area segments and business-services segments. A core mission-area segment is one that is central to the concerns or purpose of a particular political boundary within the enterprise while a business-services segment is one that is foundational to most other organizations.

When compared with all contemporary methodologies, it comprises of a comprehensive taxonomy like the ZF and an architectural process, like TOGAF. This is even more evident when the FEAF is presented in the form of a matrix (Figure 2-3). The FEAF shows clear collaboration with the ZF on three of the six columns (*what, how and where*) while the remaining three columns (*who, when* and *why)* are not considered (Figure 2-2). Though these three collaborations correlate with the three significant aspects of the ZF, unlike the ZF, the constraints of each perspective are additive. In other words, the constraints of higher rows affect the rows below though the reverse is not necessarily true. As the FEAF is additive, there is a risk of making illogical suppositions if all cells are not modelled.

With FEAF, the concept of slivers and slices as a portion of a cell or of several cells is important to realise the segment architecture approach as this provides a way to relate the segmentation of the federal enterprise to understandable parts without losing the definition of the overall integration. In consideration of traceability, it has been argued that discrepancies are prevalent if cells are not made explicit throughout the taxonomy, other slivers in the same cell may not relate to or integrate with the previous slivers unless by chance, or unless steps are taken to pre-integrate following efforts (Sessions, 2007). The architectural segments are developed individually within structured guidelines, with each segment considered to be its own enterprise within the Federal Enterprise.

Five FEA reference models are set to establish standardisation of a common language in the FEAF. These consist of a set of interrelated references designed to promote cross-agency analysis and the identification of duplicative investments, gaps, and opportunities for collaboration but not the models. Collectively, it has been claimed that the reference architecture describes important elements of the FEA in a common and consistent way that facilitates communication, cooperation, and collaboration across political boundaries. As a major arguable principle of FEAF is the unification of the various EA initiatives of agencies of the US Federal Government, it attempts to provide a single standardised, common and ubiquitous platform for sharing of information and collaboration. However, when the five FEA reference models are juxtaposed with validation, it has been pointed out that the FEAF is too flexible. As it allows individual federal agencies to use methods of choice, varied work products, and tools to define their own EAF (Urbaczewski, 2006), validation of the EAF in a consistent way is distinctively impossible and impracticable. Consequently, the FEAF and its Reference models are evolutional and cannot be applied comprehensively for many other domains.

2.2.6 Systemic Enterprise Architecture Methods

Systemic Enterprise Architecture Methods (SEAM) is a family of methods for strategic thinking, Business / IT alignment, and requirements engineering. The originality of SEAM is embodied in its ability to integrate generic system thinking principles with discipline-specific methods (Wegmann, 2002). In contrast with other frameworks, SEAM has the capability to relate different disciplines through common systemic principles thus represent systematically business, organizational and IT

concepts through a commonly shared modelling ontology. This advantage leverages its susceptibility to acquire specific knowledge by use of shared vocabulary and heuristics of each integrated discipline (Regev et al., 2013).

The family of SEAM methods are comprehensively explained in the works of Golnam (2013). Each method is a specialization of a generic approach applied to a specific set of disciplines. SEAM is typically applied to define the scope of an SOA project, to assess the outsourcing and organizational strategies but the choice of method depends of the problem to solve. For example, SEAM for Business is typically applied to define a company's business plan. However, SEAM is best suited as a method that can be adopted to analyze and design strategies at the business, the inter-company, the company and the IT system levels. Though it is held that SEAM can be quickly deployed and is specifically applicable in the requirements and scoping phases of projects, this has not been extensively corroborated (Schneider, et al., 2013).

The structure of SEAM can be described as being a hierarchy of systems. Though it provides tools for reasoning about alignment between business and IT through the description of organization's motivation, it is considered as a pragmatic tool for communicating about projects and strategies. Preferred by Gartner for the implementation of its practice, SEAM enterprise models facilitates the representation of as-is and to-be scenarios. One major advantage of SEAM is its flexibility which allows different designers to build and analyze the enterprise model through views that represent the part of the model relevant for them.

However SEAM has been criticised as being concentric on functionality analysis with emphasis on cost and security while other dimensions such as technology, business behaviour, knowledge and information management are largely ignored (Schneider et al., 2013). Additionally as SEAM places emphasis on the properties of these built functional models and not on the expertise and process for modelling, the use of divergent tools for modelling its architecture in this context actually perpetuates the exponentiation of complexities.

However, SEAM has a major originality which is of relevance to this work. This is identified as modelling of ontology. The ontology features of SEAM are systemic and systematic because of the importance of it attributes to system-related concepts such as explicit definition for concepts, the boundaries of the systems and the life cycle of the systems. With the systematic implementation of ontologies, similar concepts represented in business and application models can be transmuted in short iterations and cascaded across relevant levels on the ontology. The ontology is deemed complete when the model elements are aligned to the ontology artefacts and when all the represented ontology classes are related to specific goals. For this reason SEAM is preferred for early requirement engineering phases of EA that are associated with ontologies.

Compared to many other frameworks and methods, the SEAM brings an elaborate analysis of the environment based on RM-ODP approach. It attempts to provide a systematic ontology for system modelling with the ability to integrate the whole of EA coherently. A point of view in favor of the SEAM approach is that the same concepts and principles can be leveraged to model business, operational and IT aspects simultaneously. Thus contextual modelling of processes seamlessly interrelates with modelling of behaviour, segment and goal. On the whole, the models are more comprehensible and role associations are more explicit.

In practice, SEAM is used to scope projects. When the business processes have been modelled, they are usually transformed into BPMN with tools able to generate BPEL. Though Enterprise Architecture Frameworks are structured in hierarchies that allow analysis across different aspects and layers, SEAM does not place significant emphasis on technology in its taxonomy (Wegmann, 2002). In the well cited article and work presented by the renown originator of SEAM, Wegmann (2002), there is no discussion on how models created with the SEAM can be validated except for a passive reference that since the SEAM methodology is iterative, in adapting the model to represent changes within the organisation, validation and testing can be achieved with real people against the hypothesis made in the model. In a prospective case study of Dahalin et al., (2010) where an enterprise architecture methodology for business-it alignment is implemented from adopter and developer perspectives using the SEAM methodology, SEAM validation is effected by determining the magnitude of relationships that exist between constructs and formulation of intensity indices for each construct based on questionnaire instrument. This is analogous to the use of balance scorecard method.

In an attempt to validate the output artefacts of the SEAM, in a recent work of Golnam which included Wegmann (Golnam et al., 2014) a problem structuring method (PSM) called "Value Map" is introduced. Value Map was designed to be an extension to the Supplier Adopter Relationship Diagram in the Systemic Enterprise Architecture Method (SEAM) and aimed to assist in understanding, analysis and design of value creation and capture in service systems. To validate the usefulness of the Value Map in SEAM, an empirical study was also conducted to demonstrate that the Value Map can help business practitioners in understanding and analyzing customer value, customer value creation, and the value capture processes. However, the work clearly contradicted its aim as it emphasized that the Value Map does not validate the model artefacts after all but provides only a graphical representation of value creation and capture concepts.

In another recent attempt to validate the models created by SEAM, Wegmann again with Popescu, explored a means to apply the Physics of Notations Theory (PoNT) to evaluate the visual notation of the Systemic Enterprise Architecture Methodology (Popescu & Wegmann, 2014). The PoNT is a systemic method that is applied to model business and IT requirements in a way that allows evaluation of how effective the modelling languages are for communicating their intended messages using a set of nine principles defined in the Physics of Notations Theory (Moody, 2009). Wegmann believes that

as PoNT helps designers evaluate the notation of modelling languages and provides guidelines for improving it, the principle can be extended to the SEAM models in order to make the SEAM notation more cognitively effective. However the limitation of the PoNT is that it focuses on the physical (perceptual) properties of notations rather than their logical (semantic) properties. This is identified by Moody (2009). Wegmann (Popescu & Wegmann, 2014) also acknowledged and flawed this proposal. Despite the specific recommendations for improvement of each of the nine PoNT principles proffered as antidote, similar to the Value Map, the effectiveness of this approach remains largely rudimentary and has not been tested outside the confines of the works of Wegmann.

2.2.7 ISO/IEC/IEEE Standards

The ISO/IEC/IEEE 42010:2011, systems and software engineering architecture description are the latest edition of the original IEEE Standard 1471:2000, Recommended Practice for Architectural Description of Software-intensive Systems and a replacement for the IEEE 1471:2000. It is identical to the ISO standard approved in July, 2011. The new standard, designated ISO/IEC/IEEE 42010:2011, Systems and software engineering architecture description, are available from IEEE and ISO.

In March 2006, IEEE 1471 was adopted as an ISO standard. It was published in July 2007 as ISO/IEC 42010:2007. Its text was identical to IEEE 1471:2000.ISO/IEC/IEEE 42010:2011 and replaces ISO/IEC 42010:2007 and IEEE Standard 1471:2000. The ISO/IEC/IEEE Standard specifies conformance requirements on contents of Architecture Descriptions of Systems, Architecture Frameworks, Architecture Description Languages and Architecture Viewpoints. It defines architecture as “the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” (ISO/IEC 42010, 2007). ISO/IEC/IEEE 42010 is based upon a conceptual model or metamodel of the terms and concepts pertaining to Architecture Description (AD). The conceptual model is presented in the Standard using UML class diagrams to represent classes of entities and their relationships. More and extensive descriptions of the ISO/IEC/IEEE Standard can be found on their IEEE portal (IEEE, 2000).

The Architecture Description (AD) presented in Figure 2-4 is an artefact that expresses the architecture. ADs are used to express, analyze and compare Architectures, and often as blueprints for planning and construction. It depicts the contents of an AD and the relations between those content items when applying the standard to produce an Architecture Description. The Standard also specifies requirements on the AD and may take the form of a document, a set of models or a model repository.

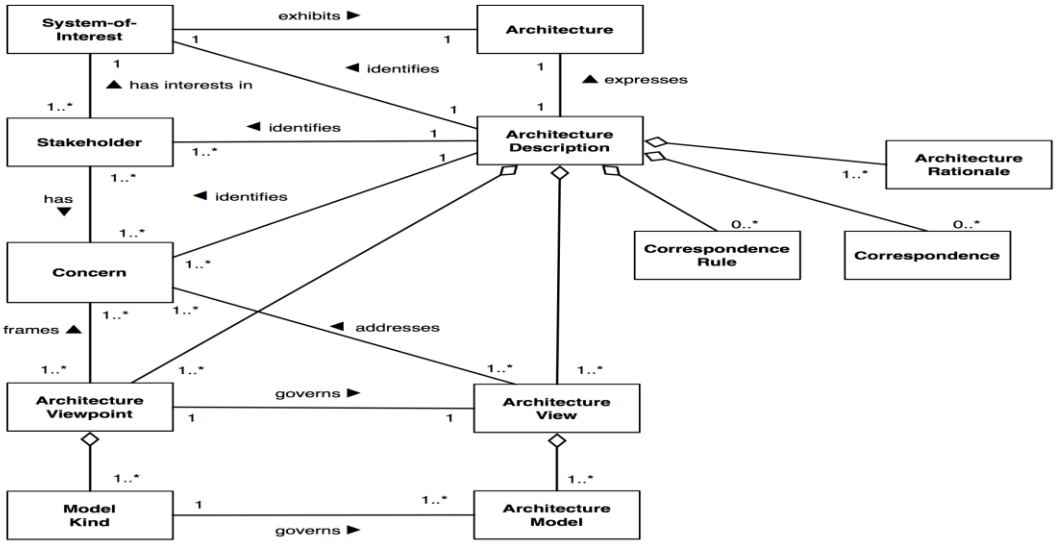

Figure 2-4: IEEE Architecture Description (Source: IEEE, 2000)

One important specification of the ISO/IEC/IEEE Standard which is relevant to validation of models as expressed in this work is the role of Concerns as outlined by the AD. The Standard defines Stakeholders as individuals, groups or organizations holding Concerns for the System of Interest. An Architecture Viewpoint within the AD represents a set of conventions for constructing, interpreting, using and analyzing a type of Concern. Specifically, Concern are addressed in Architecture View and governed by Architecture Viewpoint. The Architecture of the System of Interest from the perspective of one or more Stakeholders addresses also specific Concerns using the conventions established by its viewpoint.

In summary, ISO/IEC/IEEE is critical as a means to express specific sets of standards that many EAF can adhere to in terms of definitions and structure of what an EA should comprise of and what notations should mean. Therefore, granted that ISO/IEC/IEEE essentially influences the taxonomy of many EA, it is privileged to do so due to its generic disposition. As it does not specify a step by step approach for modelling an EA nor the intrinsic nature of how Concern and other motivation should relate to the core, in contrast with other methodologies, it is seldom classified as a methodology. The reason therefore why ISO/IEC/IEEE is important in EA is in order to adhere to principles geared towards standardization and presentation of a common understanding in comparable aspects of many taxonomies.

2.2.8 Department of Defence Architecture Framework

The Department of Defence Architecture Framework (DoDAF) is an architecture framework for the United States Department of Defence (DoD). Organized by viewpoints, it consists of a large number of systems architecture frameworks and provides visualization infrastructure for the development and documentation of all major U.S. DoD weapons and information technology systems. While DoDAF is clearly aimed at military systems, it has broad applicability across the private, public and voluntary

sectors worldwide (Schekkerman, 2003). DoDAF specifically defines concepts and models usable in DoD's six core processes for Joint Capabilities Integration and Development Systems (JCIDS), Planning, Programming, Budgeting and Execution (PPBE), Defence Acquisition System (DAS), Systems Engineering (SE), Operational Planning (OPLAN) and Capability Portfolio Management (CPM). The foundation ontology for the meta-model in DoDAF is defined by the DoDAF Meta-Model (DM2) and consists of conceptual data models, logical data models and physical exchange specifications. This underpins the DoDAF framework and defines the types of modelling elements that can be used in each view and the relationships between them (DoD, 2013). Consequently, the views of DoDAF distinctively define artifacts for visualizing, understanding, and assimilating the broad scope and complexities of an architecture description through tabular, structural, behavioural, ontological, pictorial, graphical, probabilistic and conceptual means (DoD, 2013). This makes DoDAF specifically suited to large systems with complex integration and interoperability challenges as it provides common denominator for understanding, comparing and integrating architectures across organizational and multinational boundaries.

In consideration of the approach adopted by DM2 in the validation of its models, DM2 establishes and defines vocabulary constraints for linguistic context and description for DoDAF models as applied to the six core processes. It specifies the semantics and format for federated EA data exchange between architecture development, analysis tools and architecture databases across the DoD Enterprise Architecture Community of Interest (COI). Furthermore, DM2 supports discovery and lucidity of EA data using DM2 categories of information and precise semantics augmented with linguistic traceability (Dryer, et al., 2007). Consequently it is widely acclaimed that though DM2 provides a basis for semantic precision in architectural descriptions and supports heterogeneous architectural description integration and analysis, it does not substantiate the model's artefacts. In practice, DoDAF deploys very substantial levels of details. There is no clean separation between the planning and development stages and, as a result, there tends to be substantial duplication of effort between development and planning. Though DoDAF has a respectable pedigree, many practitioners do not have a clear understanding of its scope, including how the models can formalized, levels of interoperability and what types of validation or reference architecture can be applied.

2.3 Summary of Comparison of Enterprise Architecture Frameworks

The framework proposed by Zachman (2008), the ZF identifies a descriptive model for every column in the framework. One of the strengths of the Zachman Framework is that it can serve as a classification scheme for information entities. However, an important observation is that this classification scheme is not adapted to the recent advancement in technological and information trends. For instance it does not provide a basis for classifying business to business (B2B) integration initiatives. In particular cloud computing or big data technologies cannot be classified by ZF. While acknowledging that B2B initiatives do exist especially with the mergence and acquisitions of organisations, the role the EAF in the overall process of the ZF, and the relationship between aspects

and views cannot be represented. It is often suggested that in comparison with other EAF, the Zachman Framework is rather focussed on functional requirements rather than on non-functional ones (Bahill et al., 2006; Urbaczewski & Mrdalj, 2006). Thus new modelling concepts associated with technological innovations and business strategy may be difficult to implement with the ZF when compared to TOGAF for instance.

Many of the existing EAF and methodologies are derived from each other and have similarities. For instance, TOGAF emerged from Technical Architecture Framework for Information Management (TAFIM) and Integrated Architecture Framework (IAF); DoDAF from Command, Control, Computers, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR); Recent version of TOGAF 9 has been is redesigned to include most of the concepts of ZF; GEAF is a blend of many EAF such as ZF, SEAM and TOGAF as it is a practice. SEAM is based on Reference Model for Open Distributed Processing (RM-ODP). FEAF, DoDAF are for the US government departments so there is a common architecture for integration. With many of the EAF, there is a deliberate attempt to achieve conformity with each other and many of the specifications of ISO standards as reference and guide.

Given the genealogy of these EAF, in terms of completeness and prominence, the TOGAF and Zachman Framework are generally considered to be the most comprehensible and comprehensive framework. Many other frameworks do not specify in a clear manner the core definition of their taxonomy, consequently incongruous representations of diverse or similar viewpoints are modelled (Urbaczewski & Mrdalj, 2006). It is sometimes unclear for example whether the information system architecture should only show a model of the data and process, or should also depict application with it, used when and where. In addition, different perspectives reveal different constraints, but it is not clear if constraints should for example only be propagated top-down as in the Zachman Framework, or bottom-up, as in SEAM. Whether the business models should be adapted to other layers if they are not realisable under other structures is questionable. Zachman (2002) noted amongst these myriad of EAF that the sources of legacy frustration arise from fundamental architectural description deficiencies and that rows 4, 5 and 6 models of his ZF were seldom built to specifications. However, the business architecture is prominently addressed in many of the EAF indicating similarity with the work artefacts situated in the top three rows that correspond to the Zachman framework while the activities in successive rows are scantily presented.

It is also observed that some EAFs such as FEAF adopt the concepts of the Architecture Markup Languages (AML) in modelling viewpoints. This allows specifically the transformation of models with a specific format into models with another format (Dashofy et al., 2002). The advantage is that these EAFs are able to blend the concepts of AML as a means to correlate between types of work products and data requirements. Additionally, two architecture descriptions built with different view and perspective structures such as in the FEAF can view congruent perspectives mutually to

understand and compare same work products. One benefit of the AML is that as it is not specific to any platform; it can also be adopted with the ZF. However for other EAFs, it may conversely become a disadvantage in some circumstances due to issues of compliance. In all, one of the biggest drawback of AML adaptation as a concept for modelling in EA is that it is lacking in the area of adequate functionalities that support process representation.

Many practitioners agree that the power of architecture descriptions lay exactly in making the right abstractions, without folding views into other views (Winter & Fischer, 2007). However, many frameworks do not argue why the chosen views have been selected as such the soundness of the foundations of these frameworks may be unclear. The motivation and capabilities across layers can work correctly with a framework depending on the understanding of the framework and of the importance of all parts of the artefacts. Based on the analysis as presented in this work, many separations of aspects are ambiguous and sometimes disputed amongst practitioners. Consequently it may be argued that the underpinnings of many frameworks are need for formalization of guidelines or best practises as practiced by Gartner.

2.4 Fundamentals of Enterprise Architecture Motivation and Modelling

Motivation Model is an enterprise architecture concept that facilitates the identification of aspects that aid the actualization of business strategy through graphical representation and relationship between the factors of the business plans and intentions. At the centre of motivation model are schemas and structures for developing, communicating, and managing business plans in an organized manner (OMG, 2013). The Business Rules Group (2010) states specifically that the Business Motivation Model should perform all of the following:

- identify factors that motivate the determination of business plans;
- identify and define the elements of business plans and
- indicate how all these factors and elements correlate.

Thus the main elements of motivation model can be specified as *Ends* represented as *Why* in the ZF to define goals and objectives; *Means*, represented as *How* in the ZF to define strategies and tactics; *Directives* represented as *What* in the ZF to specify rules and policies; *Influencers* represented as *Who* in ZF to specify drivers for change and *Assessment* partially represented as *When* in the ZF to specify strengths, weaknesses, opportunities and threats. In most implementations, elements of the Motivation model are developed from a business perspective and stakeholder's viewpoint with the aim to develop a business model for the elements of the motivation. In this manner, the motivation becomes the foundation for activities, connecting system solutions firmly to their business model artefacts.

Notably, amongst the many EAF discussed so far, the ArchiMate Modelling Language demonstrates the concepts of motivation modelling very distinctly. In ArchiMate, Motivational models are used to explain the reasons that underlie the design or change in the enterprise architecture, Figure 2.5. It also influences, guides, and constrains the design of the model by use of artefacts that represent goals, principles and requirements (TOG, 2012).

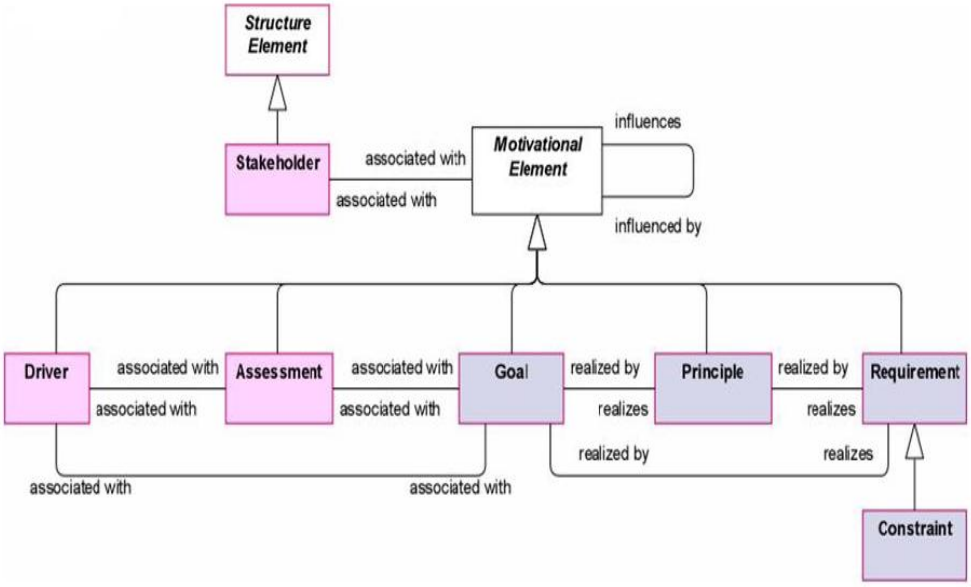

Figure 2-5: ArchiMate Motivation Extension Metamodel (Source: TOG, 2012)

While Goals represent some desired result or end that a stakeholder wants to achieve, Principles and Requirements represent desired properties of solutions or means to realize the goals. In addition, Requirements also specify formal statements of need, expressed by stakeholders (TOG, 2012). Thus it has been suggested that Motivation model is a blueprint design that can support a range of EA methodological approaches (Quartel et al., 2009). Implementation of the Motivation model results in a set of concepts that act as a checklist of factors to be considered in the architecture, a standard vocabulary and a flexible model that supports artefact development processes.

Methods such as TOGAF, SEAM and ZF acknowledge the importance of motivation modelling in the development of EAs. Motivation modelling support is needed to specify, document, communicate and reason about goals and requirements (Wegmann, 2002). In addition, motivation modelling techniques provide a way to describe structured requirements lists and use cases. Contrary to the significance placed on motivation, many other modelling techniques for EA such as GEAF have focused on products, services, processes and applications with little support available for modelling the underlying motivation of EA (Lopez, 2000; Ylimaki, 2008; Engelsman & Wieringa, 2014). In terms of stakeholder concerns and the high-level goals that address motivation, many techniques are also indistinctive (Urbaczewski & Mrdalj, 2006).

For many organisations, Enterprise Architecture is often used as a blueprint to deal with change. Many of the methodologies for EA discussed so far are driven by motivation to represent knowledge about information, processes and the use of technology in a concise but comprehensible manner. Therefore it can be said that understanding motivation is critical to achieving this objective, ensuring success in implementing EA initiatives, management of business processes and adaptation to changing business environment.

2.4.1 Homogeneity of Enterprise Architecture, Motivation and Goals

Though in principle, motivational conceptions model the fundamental assumptions that inspire the design of enterprise architecture, the ability to actualise set goals is considered as one of the organization's key catalyst for modelling motivation (Engelsman & Wieringa, 2014). Given that motivation influences, guides, and constrains models (TOGAF, 2013), the aptitude to conduct an architecture assessment is critical in ensuring that set goals are achieved. Thus the most common goals for implementing EAF have been identified as a desire to improve a specific business process, support a major business opportunity, facilitate organizational change, strengthen consolidation process or management restructure (Kappelman, 2009). Benefits resulting from motivation include the provision of a clear benchmark for evaluating proposed EA changes and identification of how the changes will fit into the existing information, application and technology architectures. Assessment of models based on its motivation highlights and identifies major gaps, overlaps or inconsistencies in the architecture and may uncover detrimental impacts on the architecture (Simon et al., 2014).

Advances in Motivation modelling have largely been focus on why enterprises run their businesses in a certain way and other underlying principle such as how they react to change (Braun and Winter, 2005; Gustas, 2010; Dietz, 2008, Carla et al., 2005). Business Motivation model often include approaches for motivation specification such as vocabulary for governance, concepts that define business drivers, assessments, business policies, strategies, tactics and goals. While policy governs the course of action and implicitly supports the end-to-end processes, drivers influence (regulations, competitions, environment, etc) and have impact on the business (Lapouchnian, 2005). This widely accepted specification methodology for extrapolation of business concerns offer a rich support for implementing the reaction to business impact, business processes, business rules and organization responsibilities. As it also forms the basis for logical design of a repository for storage of motivation models for the enterprise (Hoogervorst, 2004; Lankhorst, 2013; Schekkerman, 2004), there is no doubt therefore that the impact of motivation on enterprise modelling brings the business rationale within business scope, focusing on (a) specific deliverables (goals and objectives), (b) how it intends to achieve them (its strategies and tactics), (c) what will govern the approach (business policies) and (d) its business behaviour (assessments of the impacts of drivers). It also provides a high-level structure that supports fundamental associations with three aspects of business models namely Business Processes, Business Rules and Organization Roles. The differences between these specifications are that while goal defines the broad primary outcome, objective addresses the

measurable steps taken to achieve a goal. Strategy on the other hand specifies the approach adopted to achieve a goal while tactic specifies the tool used in pursuit of an objective associated with a strategy.

In many literatures and EA implementations such as the Zachman Framework and TOGAF, there exist sophisticated declarative mechanism for specifying goals and other model's motivations (Lankhorst, 2013). These specifications are related to business concern and standardised high-level logic for business behaviour. Varieties of EA motivations covering early literature until 2013 have also been identified (Ross et al., 2006; Salmans et al., 2010; Golnam, A. (2013). These include Business-IT alignment, Cost reduction, Standardization, Consolidation, Governance, Agility and Risk management all identified as internal motivation. While Business-IT alignment aims at bringing business requirements in line with IT implementation, Cost reduction is targeted at mainly reducing IT-related and also business process related costs. In consideration of Standardization and Consolidation, removing complexity from the architecture so as to improve cost and project delivery such as time-to-market is desired. Whereas Management and Governance is purposed towards improving decision making processes for business and IT consortiums, Agility is actuated towards improving both process as well as IT flexibility so as to facilitate adjustment to new market situations. Other miscellaneous motivations aim to support business functions such as business continuity management.

Motivation can also be categories as being external. External motivations have been identified as Regulatory compliance and Competiveness in many literatures. For many considerations, Regulatory compliance is adopted to fulfil various regulatory requirements while competiveness motivation focuses on the acquisition of business edge and astuteness. Diverse as motivation may seem and in all references, there is the need to be able to clearly assess its realization in order to ensure conformity with set criterion and to identify adaptability with the dynamic changes in the Information Technology terrain.

2.4.2 Conceptual Coherence of Modelling and Ontologies

In many references, the concepts of modelling, metamodels and ontologies have often been used without real reflection on their characteristics and their relationship to one another (Hofferer, 2007). As this is a fundamental presumption, the need to elaborate on these concepts is necessary in this work in order to establish the basis for their combined use for achieving semantic interoperability of business processes and applicability in EA validation. Often, common business requirements demand an integrated view on the collection of extant information resources processed by the heterogeneous information systems within the enterprise. The need for metamodels, models and ontologies are a prerequisite for the interoperability of this collection of resources.

Models are created using a modelling language to transmute a distinctive peculiarity of a metamodel. This implies that a model must conform to its metamodel. Following this backdrop, ontology explicitly expresses the semantics of the modeling concepts whose syntax is defined by the

metamodel (Kappel et al., 2006). Thus the most fundamental intention of ontology is simply to describe reality represented by a metamodel as an explicit specification of a conceptualization. The definition of "specialization" within ontology is often extended to include formalization, implying that ontologies can be automated or shared as a consensus within the domain of application. Thus while an ontology is a set of logical axioms designed to provide for the intended meaning of a vocabulary, it formally defines the relations among terms in a metamodel using a set of inference rules. The inference rules are deployed in this study to effect validation of the model as the joint use of metamodeling and ontologies allow for explicit description of knowledge for a complex domain (Hofferer, 2007). Ontologies stabilize (formalize) the description of a business domain while metamodels allow common deep characteristics to be specified (Kappel et al., 2006). In the context of enterprise architecture modeling and validation presented in this work, models are used for the creation of process models while ontologies are a means to provide the vocabulary of the empirical domain to facilitate validation.

Metamodelling is closely related to ontologies as both are often used to describe and analyze the relations between concepts. While ontologies are used to articulate succinct entities within a specified domain of discourse by utilizing a grammar for vocabulary, metamodelling provides an explicit description comprising of formalized specification of the domain-specific notations, constructs and strict set of rules of how a domain-specific model is built. The grammar of ontology usually deploys a formal construct to specify well-formed statement, assertion, query, etc. Thus a valid metamodel may be transformed to an ontology, but not all ontologies are modelled explicitly as metamodels.

However, due to divergent simulation and modelling tools supporting different types of modelling, most models constructed are not interoperable within ontology. Thus, it is a commonly acknowledged axiom that as specifications and applications of meta-models and models grow in complexity, a single formalism or definition would be unsuitable for generic application on all parts of a complex enterprise system.

3 REVIEW OF VALIDATION TECHNIQUES AND CONSTRAINTS

In order to correlate the discussed frameworks and validation techniques, the definition of enterprise architecture framework stated in section 2.1 as it refers to this work is used. This is reiterated as;

> "*EA consist of coherent principles, methods, and models used in the design and expression of the organisational structure, business processes, information and relationship with each other so as to realize the high-level goals and policies of the organizations (value) through low-level implementations of systems and technology."*

EA, in our definition specifies a methodology for accessing, organizing, validating and displaying information. The definition specifies four key elements of enterprise architecture namely;

- A description of the method by which EA is realised.
- A definition of artefacts that the framework should comprise of.
- A description of the structure of the architecture or framework.
- A description of validation capability of the taxonomy.

The analogy presented takes into consideration these key elements in comparing, correlating and identifying differences and similarities in the various volition techniques.

3.1 Review of Enterprise Architecture Validation Techniques

Background studies of the evolution of EA techniques and collaborations reveal a major impediment that plagues its advancement. Simply stated, the more EAF, tools and methodologies are broadened, the more complex it tends to become. Even with the drive in which EA has been embraced; today no single practice is capable of satisfying all necessary aspects of the enterprise identified collectively (Fischer et al., 2010; Sessions, 2007; Lankhorst, 2013). Attempts to mix and match rather, has resulted in EAFs with inconsistent semantics and weak ontology. Huge IT projects still fail to deliver expected goals even with the adoption of EA within the organizations. This raises the question; can EAF itself and its models be validated to ensure that there is harmonization with motivation and business processes? Notwithstanding this impediment, many EAF continue to evolve without giving much consideration to how the models can be validated or integrated as in the case of models created with heterogonous modelling methodologies. Regardless of the gaps, overlaps, inconsistencies as acknowledged in many literatures and practices, issues regarding validation of EA are not given the diligent consideration it requires.

Several workshops have been held to reflect and emphasize the need to incorporate validation techniques into EAF modelling (Klein & Gagliardi, 2010). However, this has been met with little or no responses due to the complexities involved in the initiative. Rather there has continued to be a proliferation of more complex reference models, balance scorecards and indeterminate weigh indices for maturity matrices. This is not surprising as evidence shows that early frameworks such as the popular Zachman Framework, Generalized Enterprise Reference Architecture and Methodology, Federal EA Framework, The Open Group Architecture Framework did not consider in their taxonomy a means to validate their models from inception (Fischer et al., 2010; Urbaczewski & Mrdalj, 2006). The definitions for EA used today as a guide to the practise fluctuate about components and relationships with few tacitly stipulating alignments and many reticent regarding how created models can be validated.

3.1.1 Maturity Matrices

Various levels for control of systems maturity have been proposed in EA. In some cases this necessitated outright extension of the frameworks such as in the TOG consortium with the introduction of Architecture Content Framework (ACF) and others by insertion of principles that facilitate validation such as in FEAF; where assessment frameworks with reference models are used. While some of these approaches have been in fact effective considerably, others have been adjudged complex, permeable and inapplicable in many scenarios (Hailpern & Tarr, 2006). One of such is the Maturity Matrix (MM). Maturity Matrix has been adapted in many EA implementations with autonyms such as Dynamic Architecture Maturity Matrix (DyA MM), Capability Maturity Matrix (CMM), Risk Maturity Matrix (RMM), Test Maturity Matrix (TMM). It is used as an instrument for assessing the level of Enterprise Architecture development in organizations. Often, it is a list of key areas that represent different dimension within the EA. Many organisations stretch maturity matrix indices to include all aspects of EA concerns on the premise that though the enterprise is syndicated, it is nevertheless stratified at different levels. The benchmark is based on key drivers (Fraser et al., 2002; Ylimaki, 2008). The use of sophisticated text analytic techniques as well as content-specific rules, to extract and weigh deliverables against expected outcomes with MM has enabled automated categorization of decisions. It has also enhanced the process of improving the accuracy and consistency of information within the EAF. Certain maturity techniques such as CMM and EAM apply a rigorous methodology that employs many attributes across key dimensions such as vision, viability, validity and value analyzed with a weighted algorithm.

However, one major disadvantage of the maturity matrices is that the prioritisation of the key evaluation criteria related to specific identified pivotal concerns of the business is absolutely subjective (Coleman, 2006). The graduation along the levels of maturity scale as a means of determining progression can be elusive. Occasionally, management compromise logical accomplishment of high valued goals and adherence to constraints in favour of the resolution of immediate propositions. Also, a productive iteration with a high strategic sequential value may be

placed on hold while its resources are assigned to a minor concern in order to boost the result of scale of progression on the maturity matrix. Indeed, it has been argued that the assertion of the level of maturity of the EA is often based on cognitive opinions gathered through hypothetical compendium thus it is possible that the questionnaires may not always be well understood by the respondents. Consequently the questioner's preferences may influence the outcome in given circumstances (Klein & Gagliardi, 2010). This bias often rescinds the validity of the outcome of maturity matrices.

3.1.2 Reference Models

Several literatures and journal have indicated a general ambiguity about the application of the term "Reference Model" and "Reference Architecture". Be that as it may and to place this review in perspective, a brief distinction is given between the two for the purpose of clarity. A Reference Model serves as the taxonomy that establishes a common structure for communication for *specific instances* of business behaviour while Reference Architecture is a *proven architectural template* that specifies taxonomy for the enterprise domain of interests. Thus Reference Models serves as the common communication platform that enables total participation while Reference Architectures on the other hand are the architectural guides which can be reused to expedite architectural designs. While Reference Models and Reference Architectures serve different purposes, a Reference Model is needed to adopt the right architecture template in appropriate contexts. Consequently, a Reference Model is a conceptual framework used as a blueprint for information systems development and is the subject of this section.

Many enterprises use the Reference Model as an abstract framework consisting of interlinked set of clearly defined concepts to encourage clear communication between EAF. The reference model represents a complete set of the component parts of the EAF as specified from the business functions to system components, and is used as a frame of reference to communicate ideas clearly among components and an indication of their relationship. Specifically a Reference Model creates the standards for both the objects that constitute the model and the relationships to one another. In so doing, it enhances communication between collaborators by decomposing the entities of the EAF taxonomy and creating clear roles and responsibilities. Applied to validation of EA, a Reference Model describes a set of business measurements needed to create a balanced scorecard. Each measurement is assigned to specific business roles that allow allocation of responsibilities for production of quality output. By decomposing an interest or EA concern into basic concepts, a Reference Model may be used to examine multiple alternate solutions to a phenomenon.

Though Reference Models are preferred by many EA practitioners as a methodology for assessing enterprise maturity and used in GEAF, TOGAF and FEAF methodologies, there is no doubt that Reference models are not comprehensive enough as means of validating EA models due to the fact that a Reference Model describes the type or kind of entities that may occur in an environment but not the particular entities that actually do occur in a specific environment. Additionally, since the list of

entity types or constraints defined by Reference Model basically adheres to some Reference Architecture, it cannot provide enough information to serve as a reference metrics for the entire framework in many cases. Thus to be useful, a Reference Model should include a clear description of the problem that it solves, and the goals and concerns of the stakeholders who need the solutions.

Finally, the usefulness of a Reference Model is limited as it often makes assumptions about the business and technology platforms deployed in a particular enterprise domain. A Reference Model typically is intended to promote understanding of a set of concerns and design specifications, not specific solutions for those problems. Although an effective validation approach needs to support the process of envisaging and simulating a variety of pragmatic test scenarios specified by motivation, this is not provided for by the Reference Model.

3.1.3 Architecture Content Framework

TOGAF is an example of architecture with content categorization. As one of the most popular frameworks in EA, it provides a uniform representation for diagrams to describe its enterprise architectures models using ArchiMate. ArchiMate Enterprise Architecture Modelling Language is developed to support TOGAF Architecture Development Method (ADM) and offers an integrated architectural approach that describes and visualizes the different architecture domains (TOG, 2012). This encompasses both the underlying relations and their dependencies (TOG, 2013). In response to the need for validation and testing the effectiveness of EAF, TOG introduced version 2.0 of ArchiMate with an extension incorporated with tools; first to model motivation and secondly to assess the Architecture Content Framework (ACF). Motivational concepts are used to model the intentions and reasons that underlie the design or change of the enterprise architecture. Motivations influence, guide, and constrain the design thus allow validation to be performed on the model (TOG, 2013).

TOG claims that the ACF defines the various models that describe a generic EA as its coverage includes EA artefacts and definition, processes, standards and guidelines for artefact development and the associated modelling notations that enable common understanding and collaboration. But specifically, the very core of ACF is a concept that defines a set of content specification that is coherent with the four major dimensions of its cognate modelling language ArchiMate; namely business, application, information and technology with selection and customization driven by motivation. While many other EAF continue to use maturity matrix as the practical assessment instrument for identifying gaps between business vision and business capabilities, the ACF is a significant innovation of TOGAF designed to provide a structured metamodel for architectural artifacts with support for checklist of architectural outputs. TOG claims that the ACF appropriated with consistent architecture building blocks, allows for better integration of architectural work products and provides a detailed open standard for describing architectures (Chapurlat & Braesch, 2008). However this has not been exhaustively proven as the assessment methodology is not integrated with ArchiMate Core itself. In addition to the ACF, Maturity Matrices discussed earlier still

play an important role in TOGAF to identify the level of compliance between business vision and business capabilities.

3.1.4 The Balanced Scorecard

The Balance Scorecard is a strategic planning and management system that is used extensively in business, industry and government to align business activities to the vision and strategy of the organization. The objectives of Balance Scorecard can also be extended to include the improvement of internal and external communications and monitoring of organizational performance against strategic goals. Though the Balance Scorecard provides a framework that facilitates performance measurements, its ability to help planners identify what should be done and measured has been contested (Kaplan & Norton, 2001). Research has also shown that the Balanced Scorecard is preferred specifically as a performance measurement framework for adding strategic non-financial performance measures to traditional financial metrics (Abdullah et al., 2013). It is suitable for this purpose as it provides feedback around both the internal business processes and external outcomes needed to improve strategic performance and results. In most implementations, the Balanced Scorecard is categorised into four perspectives to present learning and growth, business process, customer view and strategy mapping. Therefore the development of its measurement metrics is also done by analyzing collected data relative to each of these perspectives (Wongrassamee et al., 2003). In this respect, Balanced Scorecard assists organizations to clarify their financial vision, strategy and helps to translate them into action.

Nevertheless, the Balance Scorecard is prone to many limitations. In practise, several assumptions are made in the evaluation of its process and outcome. For instance it is common to assume that everyone understands the terminologies used; that the organization's strategy has been correctly formulated by management and that the business plan is the right one. However, it has been proven that these assumptions are not always flawless (Abdullah et al., 2013). Another limitation is that of the expected large number of participants in order to ensure that all areas are represented. The need for this is based often on a conscious attempt to try to meet the objectives of every participant's expectation and to ensure that their extensive knowledge is recognized. In many exertions of the approach, it has been affirmed that the Balanced Scorecard can be very subjective; based strictly on qualitative scrutinization. For this reason, the Balanced Scorecard has been perceived as unsuitable for validation of model as it does not relate model artefact, relationship and motivation to validation distinctly.

The balanced Scorecard in most cases is internally focused and ignores developments of the external business environment. It selectively focuses on shareholders and customers and fails to consider the various activities within the EAF from different viewpoints. The Balanced Scorecard, most notably adopted for balancing financial and non-financial metrics, lauded by many as the answer to most of both corporate and non-profit organizations' management issues, critics warn that not enough research has taken place over a long period of time to validate its efficacy even with the balancing of

financial and non-financial statistics. It is also contended that when validating scenarios such as presented with traceability within the EA domain and relationships amongst model artefacts the balanced scorecard method does not make sense (Van Grembergen & Saull, 2001). Therefore in many situations, it has even been suggested that if the scorecard fails to include financial and non-financial objectives, it loses its value as a strategic tool. The balanced scorecard must be continuously updated to reflect changes in the organisation. This requires time, resources and labour which could act as a limitation for smaller organizations without commensurate visible added value.

3.1.5 DoDAF Capability Test Methodology Approach

To enhance the effectiveness and efficiency of the DoDAF through capability assessment and evaluation, innovative enterprise initiatives were undertaken within the Department of Defence (DoD). A key competence that specified the enterprise-level Capability Test Methodology (CTM) was developed to deliver joint capability assessments and evaluations across the acquisition life cycle of DoDAF by the Joint Test and Evaluation Methodology (JTEM). Primarily, the endeavour purposed to identify gaps, seams, and overlaps related to testing in a joint environment of DoDAF. Intensive documentation with respect to process anomaly in policy, organizational or resource application, changes outside the test scope is a crucial part of this approach.

To ensure proper analysis and implementation of the DoDAF strategies, the JTEM identified DoDAF limitations and causal dependencies for further methods and process development; performed operational assessment of the impact of external DoDAF issues; validated DoDAF issues and formulated recommendations. The JTEM also vetted program level findings and made their recommendations through DoDAF community of interest governance bodies (Dryer et al., 2007).

A key component of the JTEM approach therefore involved the incorporation and refinement of CTM-related DoDAF data models and representations that best support test evaluation at a joint mission level. In order to enhance DoDAF's ability to support capability assessments supporting joint missions, JTEM developed executable product recommendations and extensions for DoDAF 1.5, as well as a capability evaluation Metamodel (CEM) to provide DoDAF schema enhancements. These enhanced DoDAF and model's support for the CTM's evaluation approach by incorporating measurement at metamodels, task, and mission performance levels.

However, there had been limitations identified with the extended DoDAF which inhibits the potential enhancements to DoDAF and auxiliary governance (Dryer et al., 2007). Individually DoDAF defined models are deemed to be deficient in their taxonomies as they adopted unsuitable CTM templates to describe essential CTM concepts including joint mission concepts, measurement metrics for metamodel and model performance, task performance, and goals actualization levels (Dryer et al., 2007). Another critical deficiency identified is poor integration of test and evaluation measures in relevant DoDAF model and the CTM test plan test matrix. Though DoDAF artefacts are found to be

relevant when creating the CTM's Joint Mission Environment (JME), discrepancies are noted between the DoD artefacts and model design methods (Dryer et al., 2007). Gaps are also identified when comparing the CTM's evaluation business rule structures, referenced as the Capability Evaluation Metamodel (CEM), and DoDAF's data model, referenced as the Core Architecture Data Model (CADM).

In order to provide conceptual consistency and an underlying business rule structure for the CTM, an ontology approach was deployed. Ontology in this context defines the explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist within area of interest and the relationships that hold among them (Bakhshadeh et al., 2014). In consonance with this characterization, the ontology supporting the CTM evaluation thread incorporates a JTEM lexicon and capability evaluation metamodel (CEM) to provide underlying conceptual definitions and relationships for the CTM. This proven approach has similarities with the Model-Driven Validation Approach (MDVA) adopted in this work except that the MDVA is inclusive and formalized to be effectuated with open frameworks rather than DoDAF which is strictly exclusively. The JTEM lexicon defined for DoDAF is a cross-domain dictionary of CTM relevant to DoD terminology and definitions. In addition, the CEM provides a conceptual model to relate key CTM test and evaluation lexicon concepts, including capability, models, motivation, task, and various types of constraint measures. Capability ensures the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks (DoD, 2013). As a result of the adoption of this methodology, future DoDAF artefact compositions supporting the CTM's Joint Operational Context for Test descriptions and Capability Test Evaluation Designs have the potential to significantly enhance capability test and evaluation within the DoD joint capability planning process.

3.1.6 Ontology-based Evaluation and Validation

Though the need for an approach for evaluation of ontology development emerged since 1994 and has grown steadily ever since (Gangemi et al., 2006), no global and comprehensive approach for the concern has been proposed to date. As it is anticipated that ontologies would be a crucial components in the leverage of other technologies such as cloud computing, big data and change management, the concepts of development of semantics able to cope with interconnectivity of semantics has also continued to arouse significant interest. The lack of well understood and shared notions of ontology evaluation and validation has also significantly slowed down the transition of ontology from esoteric symbolic structures to reliable enterprise postulate. Several studies conducted to present a formal approach for ontology evaluation and validation identified three main types of measures. These are categorised as structural measures typical to the ontologies presented as graphs; functional measures related to the intended application of the ontology and of its components; and usability profiling measures which specify the level of annotation of the considered ontology (Gangemi et al., 2006). The application of ontologies to conceptualise model schema using query language for evaluation instead

of reference models or maturity matrices ensures the retention of the domain-specific quality of the model. The satisfaction of domain-specific requirements of the model represents the particular motivation within the domain of interest.

In practice, ontology is evaluated as a diagnostic task based on ontology descriptions (McGuinness & Van, 2004) for models. Though in this work a model to ontology transformation is proposed for EA, there has been no specific literature that claims this methodology of profiling EA models against motivation. Description of models for ontology validation makes explicit knowledge about artefacts that are critical to the validation of the ontology such as roles and functions of the considered ontology. Parameters for the descriptions typically denote the quality of the ontology and are composed according to preferential hierarchy. In validating ontologies, issues considered during development include (a) the capability of the ontological categories to be grouped according to some criteria; (b) the relationship between the ontology category elements and (c) formalization of the visual model in a formal notation that is understandable to the stakeholder. This enables the process of identifying variances especially between versions of the model by comparing structures, objects and compliances and use of heuristic methods which extends the rules thus deduce new conclusions from imperfect or incomplete information.

One of the established methods of evaluating quality of artifacts in otology is to develop a quality model usually done during the early stages of the ontology development, and serve as guidance throughout the project (Di Maio, 2011). This is synonymous to our approach where quality model is derived from motivation. Quality Models are developed upfront, and used as target parameters throughout the development, evaluation and testing. It contains patterns of qualitative and quantitative measurements of various aspects; in the case of EA this is identified as goals and constraints. The quality of ontology is sometimes measured across two dimensions: its accuracy and its comprehensiveness (Vazifedoost et al., 2007). Almost the entire range of standard testing techniques used in programming consistency integrity, validation, redundancy can be applied to test the validity of ontology. A good summary of quality evaluation criteria for ontology can be found in work of Stvilia (2007).

3.2 Challenges and Critical Success Factors with Existing Validation Techniques

Over the past few decades, EA has gained substantial awareness amongst practitioners and academics. This has been in accordance with the conviction that better understanding of the dynamism in enterprises and the business environment can be significantly enhanced with the practice of EA's principles. However, research on EA has mainly been focused on the development and modelling of artefacts, while quality aspects of the models have gained less attention (Ylimaki, 2008). This section delves into these challenges of EA validation and some Critical Success Factors (CSF) that can enable alignment between the business vision, business requirements and information systems. EA is generally conceived as an approach that can identify the important components of an organization and

their collaboration with a disposition to actualize desired business objectives (Hoogervorst 2004; Kaisler et al., 2005). With the extended dimensions of EA, most initiatives continue to be focused on the development and modelling of EA (Zachman, 1987; TOG, 2013; Lankhorst, 2013; Halttunen et al., 2005; Pulkkinen & Hirvonen, 2005), while the quality and assessment aspects have only recently gained attention, especially in the form of maturity models and assessments (U.S. Department of Commerce). The maturity models are usually based on qualitative analysis (Fraser et al., 2002; Chrissis et al., 2003), for reasons of simplicity. The maturity of the EA refers to the organization's capability to manage the development, implementation and maintenance of architecture that consists of various viewpoints (Van der Raadt et al., 2004). Typically, the viewpoints considered include business, information, systems, and technical architecture. This is exemplified with the FEAF, DoDAF and SEAM. The idea of these maturity models is to gradually assess the evolution of the EA from as-is state to to-be state and from higher level of abstraction to a detailed level of actualization. This is the most declarative means of presenting the quality of EA. Despite this, questions to determine what a high quality means in the context of EA have been asked, with no empirical studies to address the questions. In this research, it is postulated that a high quality EA must conform to the agreed, understood business requirements, motivation and governance processes guiding EA design through model-driven validation technique.

In addition, the concept of critical success factor (CSF) has been considered as desired attributes in ascertaining the quality of EA model to indicate those issues that must be done exceedingly well in order to succeed (Tari, 2005). This is sequel to the fact that in order to confirm that favourable outcome are achieved in specified key index, the status of performance in each area of specification need to be measured at each milestone against expected values. While the idea of CSF has been adopted in many areas of project management, it has also awakened interest for studies in the context of EA. The constraint for use of CSF with EA is that if its measurable indices are not carefully determined and effectuated, it can become or constitute challenges in achieving a high quality level of EA model. A more articulate discourse of these challenges is presented accordingly.

3.2.1 Communicating the Terms and Concepts of EA

Though common and well-defined vocabulary of terms and concepts has been identified by some practitioners (Lankhorst, 2013; Motwani et al., 2005; Ylimaki et al., 2005), there is need to uniquely define and document the key architectural concepts with sources in which the model is based. This is necessary due to challenges that often emerge as a result of poor communication or specification of adopted plans and strategies (Rehkopf & Wybolt, 2003; Industry Advisory Council, 2005). The means of various communication channels and the timing, phases or situations in which the communication relates to the architecture are often not stated (Rudawitz, 2003).

3.2.2 Business Model Driven Approach

The primary approach to the development of most EA is through consideration of the business processes while adopting a model driven approach. This is envisioned to establish that EA initiatives are traceable to the business strategy and alignment between business and IT (Schekkerman, 2004; Van der et al., 2004). The challenge here is how to determine that the business strategy and related business requirements are taken into account in the design of the architectural model. The techniques for recognition and documentation of the business requirements for the architecture are paramount to the appropriate specification of the framework, definition of views and levels of abstraction (Ylimäki, 2008).

3.2.3 Establishment of Architecture Process for Methodology

This involves the application of appropriate processes for the design of EAF. The challenge of identification of an adaptable analysis to be adopted, constructs, constraints and theories expedient for modelling predefined viewpoints of an EA has been identified as a problem that inhibits the validation of models and associated artefacts in the works of Morganwalp and Sage (2004), Rudawitz (2003), Stanley and Uden (2013). In many instances, there is lack of guidance for the architectural decision making and documentation process. This includes documentation of the support for reuse of the processes, instructions, models or other artifacts (Kaisler et al., 2005). As variant visualization techniques, modelling languages and support tools have been adopted over the years in many large conglomerates and governments to model the EA (Chief Information Officers Council, 2001; Industry Advisory Council, 2005; Perkins, 2003; Kaisler et al., 2005; Lam, 2005), the issue of business and Information Technology alignment continue to be one of the most relevant concerns in these organisations. Therefore, since Enterprise Architecture and Information Technology have different and distinct governance approaches, many practitioners have proposed and emphasize that there is need to establish a common frame of reference in any methodology and processes adopted (Vincente et al., 2013).

3.2.4 Enterprise Architecture Models and Artifacts

As the models and artefacts are valuable in the communication of the architecture to the various stakeholders, it is important that their definition and documentation be defined extensively enough to convey the appropriate meaning to all stakeholders. Models provide a coherent and concise picture of the enterprise (NASCIO, 2003; van der Raadt et al., 2004; Kaisler et al., 2005; Lankhorst, 2013). Models need to be communicated to relevant stakeholder in a clear and comprehensible manner indicating the relevant views, composite artefacts and dependencies. The models should also address the current situation (as-is descriptions) and the future situation (to-be descriptions) (Industry Advisory Council, 2005) in conformity to the architecture principles and standards (van der Raadt et al., 2004).

3.2.5 Enterprise Architecture Traceability

One of the responsibilities of the Enterprise Architect is to provide complete traceability from requirements analysis and design artefacts, through to implementation and deployment. The formal definition of traceability usually refers to the ability to link requirements to stakeholders' rationales and progressively to corresponding design artifacts, code, and test cases. Thus Traceability is intended to support numerous EA activities such as change impact analysis, compliance verification, constraints testing and requirements validation. However in EA, Traceability often means different things to different people. Some practitioners refer to enterprise model traceability as prove for alignment to business goals; end-to-end traceability to business requirements and processes; a matrix that maps systems functions back to operational activities; reference across artefacts such as services, business processes and architecture; a footprint between a technical component and a business goal. Traceability has also been used to imply identification of associations between artifacts from business and IT strategy to solution development and delivery. However, despite these divergent perceptions of the bounds of Traceability , there is a general concession that by adopting traceability between IT and business inherent in enterprise architecture, it is possible to evaluate the IT portfolio against operational performance and business needs to determine areas where misalignment is occurring and change needs to take place. Unfortunately the practice of constructing and maintaining traceability especially in the form of a matrix is that it is very arduous and over time the traces tend to erode into an inaccurate state unless date/time stamped or versioned.

3.2.6 Enterprise Architecture Governance

Governance and management have been given various explanations depending on the context. Though in general, governance denotes the management and organizational aspects of architecture (van der Raadt et al., 2005), it can also infer the principles that guide an organization to make decisions, set priorities, allocate resources, designate accountability, and manage its architectural processes (Baker & Janiszewski, 2005). Some key questions related to EA Governance are as follows: Is the architecture governance structure defined, documented and complied? Are the roles, responsibilities and authorizations defined, documented and complied? (Industry Advisory Council, 2005); Are the processes, activities or tasks (such as definition of the architecture policy, principles or architecture compliance strategy) defined and documented (Control Objectives for Information and related Technology, 2000; van der Raadt et al., 2005)? Thus there is a challenge in maintaining effective governance processes and activities, identification of risks and management that is needed for validation especially when adopting the maturity matrices approach. The extent of integration of the EA governance processes to the organization’s business management processes, such as strategy refinement process is also an issue often underestimated. This is because EA development is usually conducted through projects and project management skills but does not essentially include validation as a crucial concern in order to assure the success of the project (Ashmore et al., 2004).

3.2.7 Organizational Culture

While developing an EA, the organizational culture should also be taken into consideration aiming at good organizational and cultural fit (Lam, 2005). In many cases cultural changes are inevitable especially in the development and adoption of EA. Organisational culture includes aspects such as attitudes towards changes by stakeholders, in the communication environment, technological innovations and economic dynamics. In particular, when performing a qualitative evaluation such as with maturity matrices, interviewees and respondents bias can influence their response. The organization culture, particularly the organizational structure, has serious impact on the success of an EA. Thus there is need for attitude towards EA to be focused on as an approach that can guide the business and IT decision making processes rather than as an auditing or controlling mechanism (Rudawitz, 2003; van der Raadt et al., 2005). A trusting organizational culture facilitates open communication, interdepartmental interactions, objective evaluation and criticism which help to improve the overall EAF (Rudawitz, 2003; van der Raadt et al., 2005).

3.2.8 Assessment, Evaluation Criteria and Scope

Engagement in EA assessment and evaluation is often considered as part of the EA governance (Morganwalp & Sage, 2004). EA evaluation is challenging because its effects and consequences are often realized much later during the life cycle of the organization's endeavours. Issues in EA evaluation, planning and implementation have been identified as concerning the models and artifacts; processes, maturity strategies, value, goals and principles; business-IT alignment, effectiveness, completeness and correctness of the EA. Other issues include utilization and usage of architectures; people's competency and skills; work environment including culture, leadership and structure. Thus defining the scope of the purpose of EA artefact assessment and evaluation has also been a challenge (Curran, 2005; Industry Advisory Council, 2005; Morganwalp & Sage, 2004; National Association of State Chief Information Officers, 2003). Determining the evaluation process and criteria, how and when the evaluation is conducted has been an issue (Bredemeyer Consulting, 2000). In most cases, validation platform, empirical data, metrics and tools are not formalized making the entire process not only complex and intricate (Erder & Pureur, 2003), but difficult for generic, formalised or iterative approaches to be developed.

3.3 Comparison of Enterprise Architecture Validation Techniques

The success and quality of EA is influenced by several interrelated factors. While the challenges faced in validating EA seem to be dependent on commitment and communication through a common language, it also appears that if the EA objectives are well defined to support the business objectives, it would be easier to gain both the top management commitment and the organizational subscription. The potential CSFs for EA can provide a selection of important issues to be taken into consideration in EA model validation initiative though this varies from one organization to the other. The comparison as presented in Table 1 affirms the conclusions in the works of Klein & Gagliardi, (2010)

that qualitative maturity matrices are still predominantly used as the main approach for verification of EA; whether in form of reference models or balanced scorecard with clear recognition that there is no validation approach adopted for any existing EA model artefacts.

It has also been argued that a reference model does not attempt to describe all things. A reference model is best used to clarify elements within specified dimensions, an environment or a problem domain. To be useful, a reference model should include a clear description of the problem that it solves, and the concerns of the stakeholders who need to see the problem solved. Applied to EA, a reference model's usefulness is limited as it often makes assumptions regarding the technology or platforms deployed in the particular EA environment. A good example is the Reference Model engaged with the FEAF which are typically intended to promote the understanding of the class of problems associated with the FEAF and not specific solutions or validation techniques to decipher those problems.

Table 1: Collation of EAF, EAML and Validation Techniques

EAF	*Enterprise Architecture Modelling Languages*	*EA Validation Technique Adopted*
ZF	Inherited from Heterogeneous ML	Inherited from Heterogeneous ML Heterogeneous Domain Specific Maturity Matrix
TOGAF	ArchiMate	Architecture Content Framework (ACF) Heterogeneous Domain Specific Maturity Matrix
GEAF	Practice	Balanced Scorecard
FEAF		FEAF Assessment Framework 2.0 Five Reference Models for Business, Components, Technical, Data and Performance
SEAM		Heterogeneous Domain Specific Maturity Matrix
ISO/IEC/IEEE	None (Standards)	None
DoDAF	IDEF (Integrated Definition Languages)	Reference Models

Many frameworks use the CSFs as a checklist for balanced scorecard. Though these may achieve some levels of comprehensiveness when initiating EA, it still potentially amounts to benchmarking of expected functionalities or outcome of desired process and not the validation of the models or artefacts of the EAF. At best, the CFSs can help to determine targets for which EA evaluation criteria, metrics and methods could be developed with respect to business behaviour from different perspectives of the EAF. Despite the various methodologies in EA modelling, validation of Enterprise Architecture Framework continue to be acknowledged as an aspect of EA that require serious attention though very little work on how this can be carried out has been proposed. Model validation still has not been properly defined as most EAF are either very generic or domain specific. As many tools can be used to depict structural components as in ZF, it is difficult to implement validation as the diverse tools used cannot offer a consistent component description for relating all the objects across the different layers of the framework.

4 ARCHITECTURE MODELLING LANGUAGES

In order to enhance visualization, improve efficiency, assure quality as well as achieve alignment of business processes with IT, enterprise modelling has gained significant consideration especially in the alcoves of Architecture Descriptions. This is not surprising as EA models provide structures that are often deployed for planning, designing, simulation and management of change as the business evolves. The role of Modelling Language (ML) has been to provide a high level abstraction language capable of representing these structures, their characteristics and properties. Over the decades, there has been proliferation of several MLs as a means to present visual images of design concepts (Chen et al., 2008). Modelling has been applied to various dimensions of enterprise such as management, quality, engineering, software, technology and human resources, oftentimes adapted to domain or specific for purpose. For this reason also, several techniques for modelling the enterprise have been developed such as Active Knowledge Modelling (AKM) (Lillehagen & Krogstie, 2008), Design & Engineering Methodology for Organizations (DEMO) (Jan & Dietz, 1999), Dynamic Enterprise Modelling (Heinz-Dieter Knoll et al., 2003), Enterprise Modelling Methodology/Open Distributed Processing (EMM/ODP) (Veryard et al., 1994), ArchiMate (TOG, 2013), Extended Enterprise Modelling Language (Krogstie, 2008), Integrated Enterprise Modelling (IEM) (Mertins, 2006), Multi-Perspective Enterprise Modelling (MEMO) (Frank, 2002). For process modelling, languages include Business Process Model and Notation (BPMN), Computer Integrated Manufacturing Open System Architecture (CIMOSA), Integrated DEFinition for Process Description (IDEF) and Unified Modelling Language (UML). The list is extensive. In many cases, these languages address specific modelling concerns, thus define and use concepts that suite the domain under consideration. The diversity of the Modelling Languages has also culminated in heterogeneity in their definition of semantics. Most definition of concepts is considered ambiguous. Comparison of models created or even integration of these models have been difficult. The need for a coherent description of architectures in the face of these disparities has of recent become even more critical as the importance of ascertaining traceability and establishing congruency among different models has been desired.

Just like the EAF, there is currently no existing architecture description language that can fully enable integrated enterprise modeling in its entirety. Therefore in this research, focus is placed on the review of Enterprise Architecture Modelling Languages (EAML) that adheres to the threshold of EA as defined in this thesis and encapsulates the modelling of at least by perspectives, business process, information systems, technological infrastructure and the relationship between associated components. To enable some form of analogy, a justification of the selection of EAL is given by specifying characteristics that the EAM should exhibit in section 4.1. A description of some commonly used Enterprise Architecture Modelling Language is presented in section 4.2 and a comparison of the reviewed Enterprise Architecture Modelling Languages is given in 4.3. This section forms a rationale for selection of an Enterprise Architecture Modelling Language for extension to achieve the hypothesis postulated in this work.

4.1 Reflection on the Choice of EAML for Exposition

In adopting the description of Enterprise Modelling as concerned with the representation of the organisational structure and the behaviour of business for the efficient analysis, engineering and optimisation of its operations (Greefhorst & Proper, 2011) complexities in planning and designing are encountered. There is need for a deep understanding of the company's current situation and advancing trends in information technology. With the increasing importance of electronic commerce, more and more companies are rethinking the way they do business, including the redesign of both internal and cross-organizational business processes (Beznosov, 2000). In consideration of this, perhaps the main challenge results from the complexity and diversity of the tasks involved. While enterprise modelling of systems, analysis and redesign of corporate strategy and structure are complex tasks on their own, their harmonization is required in order to provide for information systems that are consistent with strategic and organizational guidelines (Prasse, 1998). To meet this challenge, enterprise models have been introduced on various levels of abstraction. Software systems supplemented with models of business processes and conceptual data models have helped system designers in understanding systems. However, while these have been beneficial in deployment of particular systems, in other utilizations such as in EA, they have been limited.

To promote the establishment of common semantics for Enterprise Architecture Modelling Language, a number of consortia have provided high levels of enterprise models that emphasize roles and responsibilities of stakeholders within certain domains. However, these have remained superficial. The general purpose modelling languages such as UML, BPML and EEML, though allows for modelling a wide range of domains especially as a passable foundation for software development, do not provide concepts and graphical representations that are appropriate for enterprise architecture (Lankhorst, 2013). In addition to the generic specification that the EAML must be structural and graphical, with the capability to use named symbols that represent concepts, lines that connect the symbols to represent relationships and various other graphical notations that represent constraints, the selection of EAML for analysis in this work is based on their capability to adopt the following techniques which allow for applicability and comparison.

i. Firstly, the subscribed Enterprise Architecture Modelling Language is distinguished by its scope and the central role of integrating multiple layers of the taxonomy. The Enterprise Architecture Modelling Language must be able to represent concepts in the domain of activity, covering data, events, business behaviour, service, systems, constraints, viewpoints, infrastructure and motivation.

ii. Secondly, the Enterprise Architecture Modelling Language must contain minimal number of elements that are easy to articulate and understand by the users so that it can be applied consistently and interpreted across the enterprise in a uniform and coherent manner.

iii. Thirdly, the core set of primitive elements must be formalized and segregated without overlap in interpretation so that the representation can support reasoning at various levels of abstraction or in detail.

iv. Fourthly, the Enterprise Architecture Modelling Language should aim at the integration of the partial models that represent particular views in the enterprise. Integration implies that semantic relationships between partial models should be expressible. This is a prerequisite which ensures referential integrity between different models and traceability on the taxonomy.

v. Finally, the language should provide reusable and adaptable concepts. Reusability can take place on different levels including design patterns and generic reference models for divergent domains. In order to support the construction of queries for validation, the language description should be sufficiently formalized with language semantics that can be mapped onto ontologies.

4.2 Overview of Enterprise Architecture Modelling Languages

Based on the rationalization and justifications presented in section 4.1, the following Enterprise Architecture Modelling Languages are reviewed and critiqued to determine the EAML that is best suited for extension for purpose of this research;

a) Multi-Perspective Enterprise Modelling (MEMO) by International Federation of Automatic Control /International Federation of Information Processing (IFAC/IFIP)
b) UML by OMG
c) ArchiMate, a technical standard from TOG based on concepts of the IEEE 1471 standard.
d) IDEF by DoDAF
e) DEMO by Enterprise Engineering Institute
f) I* from iStarwiki.org

4.2.1 Multi-Perspective Enterprise Modelling

Multi Perspective Enterprise Modelling (MEMO) is a method for enterprise modelling that offers a set of specialized visual modelling languages together with a process model, techniques and heuristics that supports problem specific analysis and design (Frank, 2002). The MEMO group of languages allow modelling diverse correlated aspects of the enterprise. MEMO models serve a dual purpose. The first purpose of MEMO is to facilitate the development of integrated information systems which aligns with the corporate strategy of the organization. The second purpose is for these abstractions to

be used as the justification of an enterprise metamodel. The derivative instantiation allow for a tentative representation of all key facets of the enterprise strategy, organizational structure, business processes, business entities, business rules and events.

MEMO offers three exclusive languages to further the schematisation of its models. The strategy modelling language (MEMO-SML) includes notable concepts from strategic planning, such as portfolio analysis and value chains. The organization modelling language (MEMO-OrgML) serves to model the organizational structure in terms of business processes and resources. To cater for the specification of information as a basis of database design or software development, MEMO also includes an object oriented modelling language (MEMO-OML) (Frank, 2002). Both MEMO-SML and MEMO-OrgML include concepts that guide the user with the assessment of resources so as to advance the analysis of the organisations competitive position.

As an introduction to more detailed abstractions, and as a medium to foster cross-disciplinary discourses, MEMO offers a generic conceptual framework that corresponds to common abstractions of business firms (Frank, 2001). By differentiating language into three perspectives namely strategy, organization and information system, each perspective is further stratified into four common aspects covering structure, process, resources, and goals.

To control the model's integrity and provide means of navigating through the views of an enterprise model on various levels of abstraction, MEMO is collocated with a development environment, called MEMO Centre. One of the advantages of MEMO is that all component editors for graphical notations are combined with textual editors and browsers. MEMO Centre is implemented as a constructionist learning tool and can run on many platforms available in virtual machines. The components are integrated through a common object model. The integration with a corresponding object model is accomplished by ascriptions to classes and associated services respectively.

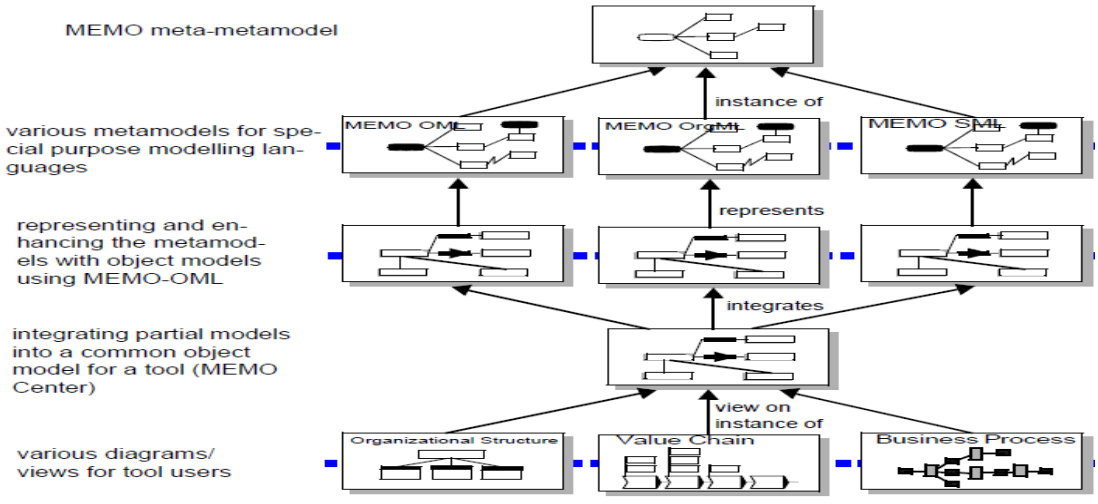

Figure 4-1: Architecture of MEMO Modelling Languages (Extract from Frank, 2002)

All languages within MEMO are specified with a meta-language which connotes a common meta-metamodel. Frank offers a detailed description of this connotations and a comparison with other meta-metamodels in his acclaimed work on MEMO (Frank, 1998). Every metamodel is designed as an object model specified in the MEMO-OML in order to prepare for the development of a tool environment.

However, MEMO has some limitations. Although an object model representing the language is usually very similar to the corresponding metamodel, the mapping is cumbersome in MEMO. This is because MEMO-OML requires a wide endorsement of concepts for multiple inheritances to be performed when compared to single inheritance, aggregation, delegation and services of other metamodels (Frank, 2002). The suggested justification for this is that it allows for a more articulate reconstruction of metamodel with a more exquisite and appropriate concepts. Despite this, adept users of MEMO have lauded its capability to assert practicable version control as well as its capability to facilitate the integration of various object models easily into one common object model (Figure 4-1). This background it is claimed enables systematization and composition of a conceptual foundation for an integrated modelling environment Frank, 2002).

4.2.2 Unified Modelling Language

Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of software engineering (ISO/IEC 19501:2005). The term "unified" applies to the unification of many critical prior existing and competing object-oriented languages developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in the nineties (Hamilton, 1999). It was adopted by the Object Management Group (OMG) in 1997, and has been managed by this organization ever since. Since year 2000 when UML was accepted by the International Organization for Standardization (ISO) as industry standard for modelling software-intensive systems, it has continued to evolve in versions and scope.

The Unified Modelling Language is a fusion of techniques from data modelling (entity relationship diagrams), business modelling (work flows), object modelling and component modelling. It is used with all processes in software development life cycle and across different implementation technologies. UML includes a set of graphic notation techniques for creating visual models of mainly object-oriented software-intensive systems. Its notation is derived from the unification of notations for describing a set of objects, their relationships, Object Modelling Techniques (OMT) and use case methodologies. As a modelling notation, the influence of the OMT notation is most dominant. With UML, modelling is usually abstracted at a much lower level of design detail. Concepts from many other Object Oriented (OO) methods have also been loosely integrated with UML with the intent that UML would support all OO methods. Many other contributions and approaches favour the many versions of UML including techniques such as OO Structured Design (OOSD) notation, timing analysis, data analysis and state charts. As a result, UML is useful in a variety of engineering

problems, from single process, single user applications to concurrent, distributed systems. Consequently, though UML is extensive in coverage, it is also over laden.

An overview of UML diagrams shows that it is often used to represent static and dynamic views of system models. Static or structural view emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams while the dynamic or behavioural view emphasizes the dynamic behaviour of the system. The later is demonstrated by depiction of collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams. A categorization of hierarchy of the UML is as shown in Figure 4-2.

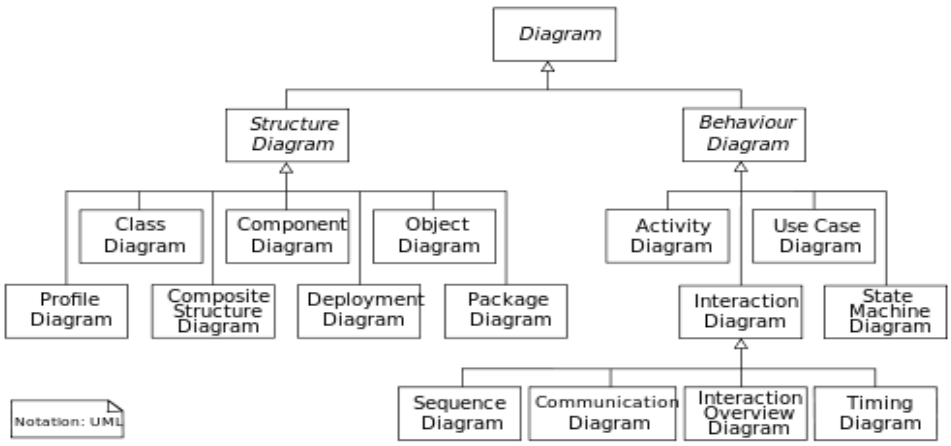

Figure 4-2: An Overview and hierarchy of UML diagrams

Despite the comprehensiveness of UML, it also has several inhibitions that make it unsuitable for use in modelling EA. Though many UML tools support some of the new features of UML 2.x, there is no test suite to objectively test compliance with its specifications (Alhumaidan & Zafar, 2014). It is a well acknowledged fact that UML does not restrict UML element types to a certain diagram type. In general, every UML element may appear on almost all types of diagrams as such not suitable for presentation of architectural layers and aspects (Khoury, 2007). Although, UML is widely recognized and uses modelling principles, it is frequently criticized for standards bloating as it contains many diagrams and constructs that are redundant or infrequently used (Gusta, 2010). In practice, users often draw diagrams with the symbols provided but without the meanings those symbols are intended to provide. Thus there is linguistic incoherence as its standards have been cited as being ambiguous and inconsistent. A capability of UML and implementation language mismatch is typical of the notational systems. This problem is particularly pronounced as the UML does not adhere to orthodox object-oriented doctrine (Chen et al., 2008). The direct one-to-one correspondence of annotation across layers of the architecture used by UML is ineffective and not coherent when applied to modelling of EA. For these reasons, UML has been criticized for being extremely complex compared to other tool and unsuitable for modelling EAF (Larman, 2012).

4.2.3 ArchiMate Modelling Language

In view of the shortcomings of TOGAF, TOG extended its definition of Architecture to include a formal description of a system, organized in a way that supports reasoning about the structural and behavioral properties and its evolution (TOG, 2012). To provide a uniform representation for diagrams that describe the components and the building blocks of the new architecture, the ArchiMate enterprise architecture modeling language was developed. The ArchiMate presented a unified architectural approach that visualizes and describes the diverse architecture domains and their underlying abstractions, relationship and dependencies. This implies that the ArchiMate is a more scalable Enterprise Architecture Modelling Language when compared with other EAML for several reasons. Firstly, it incorporates a concept of service orientation consisting of organizational principles that correspond to business, application, infrastructure and motivation (Lankhorst, 2013). Secondly, its architecture framework is lightweight and comprehensive, structured by architectural domains, layers, and aspects. TOG (2012) maintains that the rationale for this structure is to provide a graphical language for the representation of enterprise architectures over time. Thirdly, the ArchiMate specification adheres to the Open Group Standard of TOGAF and IEEE standards. The rationale for this is to strike a balance not only between the particularities of the individual architecture domain but also to provide a generic set of architecture concepts which describe concepts of divergent levels of architectural specialization. However TOG emphasize that while ArchiMate can be used for many other enterprise architecture modeling tasks, the most important design restriction on ArchiMate is to maintain compaction. Unlike many other modelling languages, such as UML 2.0 which attempts to domicile too many requirements of multi-purpose uses, ArchiMate has been constrained to suffice for concepts that are applicable specifically for EA.

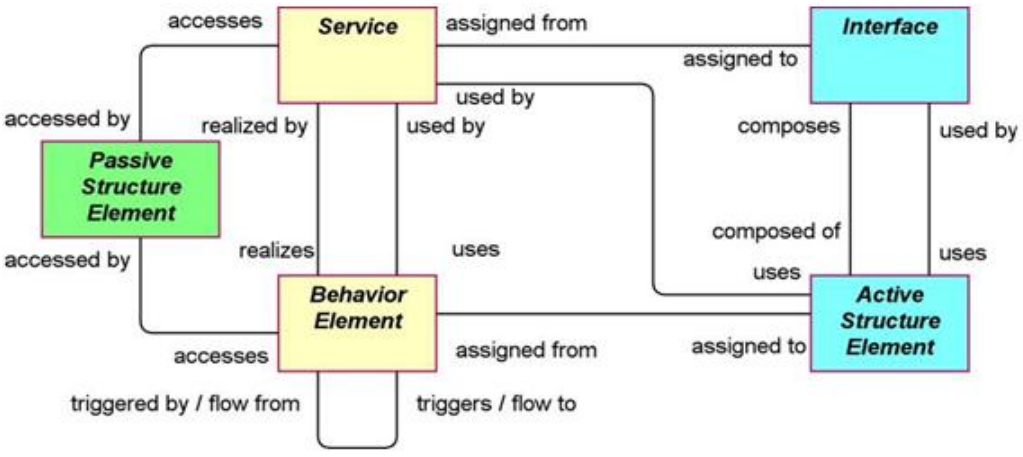

Figure 4-3: The Core Concepts of ArchiMate Generic Metamodel (Source: TOG 2012)

The core language of ArchiMate consists of three main types of elements specified as *active structure* elements, *behavior* elements, and *passive structure* elements. An active structure element is defined as an entity that is capable of performing behavior, a behavior element is defined as a unit of activity

performed by one or more active structure elements and a passive structure element is defined as an object on which behavior is performed (TOG 2012). Figure 4-3 depicts generic metamodel of the core concepts of ArchiMate with aggregations, composition, specialization, interactions and collaborations.

The ArchiMate Enterprise Architecture Modelling Language also defines three main dimensions of specializations based on the core concepts presented in Figure 4-4. These are the *Business Layer* which offers products and services to external customers and realized through business processes; the *Application Layer* which supports the business layer with application services and realized by software systems and the *Technology Layer* which offers infrastructure services required to execute applications and realized by systems infrastructure. Thus, while Service is the externally visible behavior of the providing system, it is accessible through interfaces and constitutes the external view on the active structural aspect.

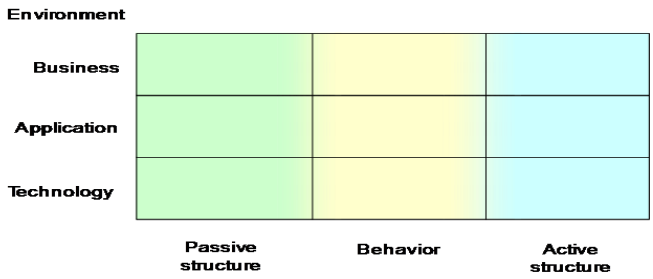

Figure 4-4: ArchiMate Architectural Framework (Source: TOG 2012)

ArchiMate can also be described as consisting of aspects and layers organized as a framework of nine cells. However, TOG maintains that these classifications are generic with no clear and strict demarcation of boundaries between cells as illustrated in Figure 4-4. The structure of the framework allows for modeling of the enterprise from different viewpoints, where the position within a cell or multiple adjacent cells highlights the concerns of the stakeholder (TOG, 2012).

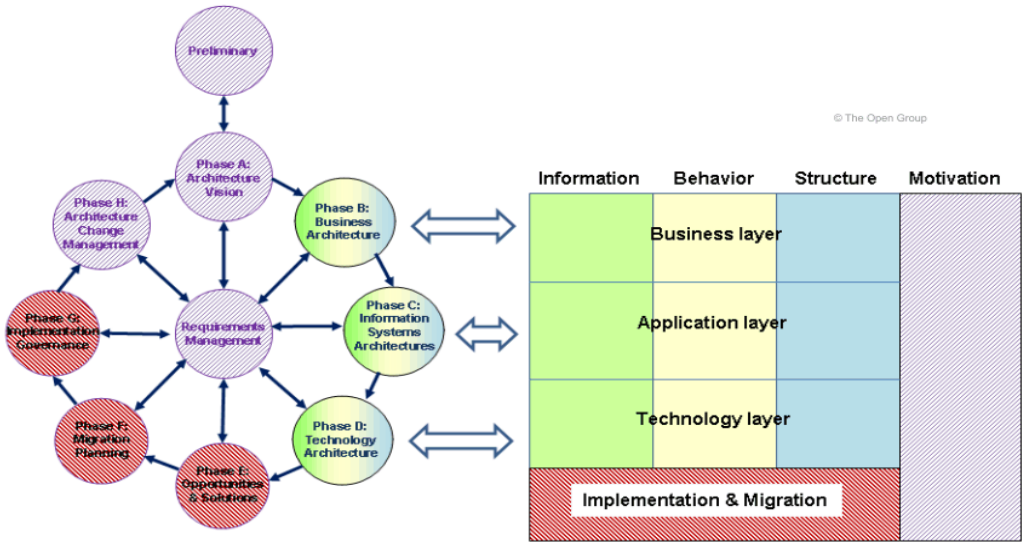

Figure 4-5: Correspondence between ArchiMate and Extensions with TOGAF (Source: TOG, 2013)

The ArchiMate Enterprise Architecture Modelling Language provides a vendor-independent set of concepts, including a graphical representation that facilitates the creation of a consistent, integrated model of EA. The structure of the core ArchiMate language closely corresponds with the three main architectures as addressed in the TOGAF ADM (TOG 2013) enabling an objective mapping between TOGAF views and the ArchiMate viewpoints as depicted in figure 4-5.

Many critics maintain that some of the viewpoints that are defined in TOGAF cannot easily be mapped onto ArchiMate viewpoints. However, TOG maintains that ArchiMate has analysis techniques which still support such concepts as addressed in viewpoints even if the mapping may not be one-to-one. The hypothesis that an architecture modelling language should provide a means to handle the complexity of modern information-intensive enterprises (Lankhorst, 2013) and should enhance communication between different stakeholders seems to be met by ArchiMate. ArchiMate Enterprise Architecture Modelling Language has gained considerable acceptance by practitioners due to its extensibility capability, segregation of architectural artefacts into aspects, viewpoints and pragmatic layers of abstraction. These qualities have enhanced extensively processes for formalization and alignment between the business processes, their supporting applications and technical infrastructure.

4.2.4 Integrated Definition Languages

Integrated Definition (IDEF) is a collection of modelling methods applied to describe operations in an enterprise (Menzel & Mayer, 2006). Created by the United States Air Force and now maintained by Knowledge Based Systems (KBS), IDEF was initially developed for the manufacturing environment. Through its evolution IDEF methods have been adapted for a broader use including software development. Occasionally used in conjunction with gap analysis, IDEF methods are used to generate graphical representations of diverse systems, analysis of models and to facilitate the transition amongst models.

Currently, there are sixteen methods of IDEF implementation ((Menzel & Mayer, 2006). Each designed to capture a specific type of information through modelling processes. Table 2 lists the IDEF methods known to date including IDEF0 through IDEF4 which are most commonly used. The IDEF0 methods are used mostly to model engineering functions of an enterprise. It creates a graphical model that shows what controls a function, activities performed, what resources are used in carrying out the activities, produced artefacts and relationships that exist with other functions.

IDEF0 is mainly an engineering technique best suited for performing and managing needs analysis, benefits analysis, requirements definition, functional analysis, systems design, maintenance, and baselines for continuous improvement of systems (Imran et al., 2010). IDEF0 diagrams are organized in hierarchical structures which disintegrate the problem domain to greater level of descriptive details

and refinement. Thus it is unsuitable for creation of metamodels which represents a more concise description of EA required at a higher level of abstraction.

Table 2: List of IDEF Methods

IDEF METHODS			
IDEF0	Function Modelling	**IDEF7**	Information System Auditing
IDEF1	Information Modelling	**IDEF8**	User Interface Modelling
IDEF1X	Data Modelling	**IDEF9**	Scenario-Driven IS Design
IDEF2	Simulation Model Design	**IDEF10**	Implementation Architecture Modelling
IDEF3	Process Description Capture	**IDEF11**	Information Artefact Modelling
IDEF4	Object-Oriented Design	**IDEF12**	Organization Modelling
IDEF5	Ontology Description Capture	**IDEF13**	Three Schema Mapping Design
IDEF6	Design Rationale Capture	**IDEF14**	Network Design

IDEF0 is disadvantaged in many ways as an EAML that can be used to model EAF. Though IDEF0 provides a comprehensiveness and expressiveness in graphics and represents a wide variety of business, it is best suited for manufacturing and other types of engineering operations systems where knowledge-based system approach is required (Kim et al., 2003). Instead of presenting diverse views of the enterprise architecture, IDEF0 emphasises on the hierarchical exposition of details. It has been observed as a limitation that IDEF0 models can become so concise that it becomes incomprehensible (Imran et al., 2010). Difficulties often arise in communicating across the various IDEF methods, domain concerns and between different domain experts. The limitations of using IDEF0 model for socio-technical aspects has been clearly acknowledged as it does not allow model of motivation and poor at establishing traceability (Imran et al., 2010).

The IDEF1 language was created to allow a neutral description of data structures while IDEF2 was originally intended as a user interface modelling method. Integrated DEFinition Methods (IDEF3) was intended to address many aspects of enterprise modelling (function, data, process, object-oriented design, and ontology). However, the lack of a methodology to support the structuring of descriptions of views has been a major shortcoming of the IDEF systems ((Menzel & Mayer, 2006).

4.2.5 Design & Engineering Methodology for Organizations

Design and Engineering Methodology for Organizations (DEMO) is an enterprise modelling methodology for transaction modelling, analysing and representing business processes. Developed at the Delft University of Technology by Jan Dietz and others in the early 1990s, it is inspired from the Language/Action Perspective (LAP) (Winograd, 2014). DEMO originally stood for "Dynamic Essential Modelling of Organizations" and is a methodology for designing, organizing and linking

organizations with the central concept being communicative action. The DEMO methodology is based on the following principles:

- Organization consists of people with authority and responsibility to act and negotiate.
- Rational modelling of business processes and information systems to achieve uniformity.
- Information Protection models for concerned stakeholders.
- Segregation and composition of Information to fit specific user's requirements.

DEMO is further developed and supported by the Enterprise Engineering Institute and consists of basic pattern of a business transaction. The three phases of DEMO are; the action generation phase during which facts are requested; the action execution which abstracts the required fact and the fact generation phase, which involves the evaluation and rendering of the results.

DEMO assumes that an organization consists of three integrated layers (Jan and Dietz 2008) expressed as the B-organization, I-organization and D-organization. The B-organization or business layer according DEMO is the essence of the organization. This includes software which supports the business processes. The B-Organization also is segregated into three perspectives or levels of abstraction to define essential business systems, informational systems and documental data. The I-organization defines five related models for the organization (Jan & Dietz, 2008) which are harnessed to produce a series of graphical renditions. These are;

- The interaction model – Communication Diagrams (CD)
- The process model – Process Diagrams (PD)
- The action model – Transaction Diagrams (TD)
- The fact model – Fact Diagrams (FD)
- The interstriction model – Action Diagrams (AD)

However, the DEMO like other EAML is also fraught with constraints. Though the methodology provides a comprehensive understanding of the business processes of the enterprise, the actors involved, it is unclear about pragmatics aspects of the transaction, such as the conversation structure and the intentions generated for each view (Kecheng, 2001). Also it has been argued that despite that the DEMO provides a coherent understanding of communication, information, action and organization of an enterprise, it does not provide a comprehensive annotation to describe and relate the artefacts of an EAF. The scope of DEMO is contained within Information Systems Engineering and Business Systems Engineering and not the entire enterprise.

4.2.6 I*

The I star (I*) framework is a modelling language that is suited for the initial stages of system modelling. It aids in the analysis of problem domain and allows modelling of both as-is and to-be scenarios. The I* is an approach originally developed for modelling and reasoning about organizational environments and their information systems. Its key composition is based on heterogeneous actors with different goals and communal dependencies. Consequently, the EAML is intentional actor and intentional goal oriented corresponding to the *WHO* and *WHY* of the Zachman Framework though specifically not *WHAT*, *WHERE* and *HOW*. It has been observed that the I* is comparable to the UML Use Case approach which deals with functional goals and actors but contrasts with KAOS approach which covers goals with less intentionality of actors (Lapouchnian, 2005).

The configuration of the I* Enterprise Architecture Modelling Language allows expression of dependencies among actors with four descriptive elements namely goal, soft goal, task and resource. The intentional actor constitutes the central concept in I*. Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments. Actors depend on each other for goals to be achieved, tasks to be performed and resources to be shared. By depending on others, an actor may be able to achieve goals that are difficult or impossible to achieve individually. On the other hand, an actor can become vulnerable if the dependant actors fail to deliver. Actors are strategic in I* in the sense that they are concerned about opportunities and vulnerabilities, and seek rearrangement of their environments that better serves their interests through restructuring intentional relationships (Lapouchnian, 2005).

The framework of the I* consists of two main modelling components namely the Strategic Dependency model (SD) and the Strategic Rationale model (SR). A Strategic Dependency model describes a network of dependent relationships among various actors in an organisational context. The actor is usually identified within the context of the model with the actor correlated with tasks that depend on the actor. An SD model consists of a set of nodes and links connecting the actors. Nodes represent actors and each link represents a dependency between two actors. The SR model allows modelling of the reasons associated with each actor and their dependencies, and provides information about how actors achieve their goals and soft goals. For the I*, a model is relevant only if it includes elements considered critical enough to impact the results of a goal.

While both SD models and SR models are used in the development of software use cases, the SR model differs from the SD model in that the SR model provides a more detailed level of modelling by extricating the tasks of the actors to model internal and intentional relationships. Intentional elements (goals, soft goals, tasks, resources) appear in the SR model not only as external dependencies, but also as internal elements linked by means-ends relationships and task-decompositions. The *means-end links* provide understanding about *why* an actor would engage in some tasks, pursue a goal, needs a resource, or wants a soft goal. The *task-decomposition links* provide a hierarchical description of

intentional elements that make up a routine. Instantiated model of I* describes stakeholder interests and concerns, and how they might be addressed by different configurations of systems and environments.

Despite these advantages, there are also concerns regarding the I* which makes it not quite adaptable to modelling EAF. Firstly, the I* is an agent-oriented modelling framework that is best suited for requirements engineering, business process reengineering, organizational impact analysis, and software process modelling (Lapouchnian, 2005) rather than the entire EA abstractions including data and infrastructure. Secondly, though the I* models offer a number of levels of analysis, in terms of ability, workability, viability and believability, it is specifically to promote the early understanding of the organizational relationships. Thirdly, I* relies heavily on the modelling of the business domain using Use Cases developed from organizational models. This renders the establishment of relationship between the functional requirements of the intended system and the organizational intended goals complex as both are defined within the organization abstraction. For instance, within EA, it is a common requirement that a function can depend on a service directly or a data object without any need for actor interaction. It would render the model ludicrous if for instance a data object is considered to be an actor as impressed by I*.

Similarly, as seen in its pragmatic disposition, the I* is an ideal language for expressing actors, tasks, resources, goals in a constructivist susceptibility, therefore it would be very generic and flaccid in coverage and definition of the different dimensions of EA based on contextual specifications. This limitation is deeply inherent in the semantics of I* as models developed at the early stage help primarily to create understanding why a new system is needed while models developed at the late requirements phase serve specifically to forecast the new system configurations. Thus the desired processes and evaluations are based on alignment to functional and non-functional needs of the actor and every other concern is inconsequential.

4.3 Comparative Analysis of Enterprise Architecture Modelling Languages

The analysis of Enterprise Architecture Modelling Language identifies wide differences in almost all specifications of measure in terms of their approach on particular parts of the enterprise modelling activity. Though the representation of modelling language as taxonomy is intended to enable comparability in terms of enterprise architecture modelling cycle coverage, capability, extensibility and enterprise information rendition, this has not achieved parity that allows generalization in terms of a focus or common practice. The anticipation that this concise analysis will help to harmonize the results of enterprise modelling as well as the terminology used, both which are relevant in the subject on enterprise integration and validation has been sporadic. Therefore the table of comparison presented in Table 3 augments other evaluations in this regard and provides a guide in determining their suitability in the context of this work.

The Unified Modelling Language adopted by Object Management Group (OMG) is a standardized, general-purpose modelling language in the field of software engineering and includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems (OMG, 2013). Though it combines techniques from data modelling (entity relationship diagrams), business modelling (work flows) and object modelling (Barra et al., 2004), it lacks the versatility of ArchiMate to visualize the entire enterprise as defined by IEEE (2008), Lankhorst (2013) and DoD (2013). UML is focused on definition of system structure and properties and has no built-in testing constructs for behaviour (Baker et al., 2004). The UML Test profile currently proposed is at a much lower level of abstract based on Testing and Test Control Notation Version 3 (TTCN3) and JUNIT than required in business behaviour validation at EA higher abstraction.

Table 3: Collation of Enterprise Architecture Modelling Languages

FEATURES	Aspects	Architectural layering	Motivation/ Constraint	Relationships/ Traceability/ Mapping	Language Extensibility	Functional Composition	Language/ Heterogeneity	Views/ Viewpoints	EA/ Domain/ Framework
EAML									
MEMO	Structure Process Resource Goals	Strategy Organisation Information	None	Fixed invariant bindings	Not open-source thus cannot be extended	No explicit and concise annotations.	Memo-OrgML, Memo-SML, Memo OML	Memo Centre	GERAM
UML	Conceptual/ Logging, Synchronization, Security, Distribution.	No support for architectural layers	No support for reasons underlying design	Variable bindings with specific semantics	Not open-source thus cannot be extended	Linguistic incoherence and ambiguity standards	Data, Object Business and Component modelling	None	MDA
ARCHIMATE	Passive Behaviour Active	Consists of Business Application Technology	Extended with Motivation and Goals	Explicit objective Connectors with distinct semantics	Extensible as it is Java open source. Language code available.	Segregated functionality artefacts with unambiguous specifications	Singular extensible semantics for all modelling	Allow several views and viewpoint abstraction	TOGAF
IDEF0	Who performs; What performed; Which resource; What produced; Sequences	No support for layers. Hierarchical disposition of details	Not Constraint or driven by motivation	Communications hampered across methods & levels. Inconsistencies with mappings	Not open-source thus cannot be extended	Supported by methods composite features	Distinct methods defined for purpose	Hierarchical system views	DoDAF
DEMO	Fact Acquisition Fact Execution Fact Generation	B-Organization I- Organization D-Organization	Not Constraint or driven by motivation	Annotations not comprehensive	Not open-source. Cannot be extended	Communication Process Transaction Fact and Action Diagrams	Communicative actions	None	Enterprise Engineering
I*	Intentional actor (WHO)/ Intentional goals(WHY)	None	None	Dependencies amongst actor	Not open-source thus cannot be extended	Strategic Dependency/ Strategic Rationale	Singular restrictive semantics for all modelling	As-is To-be	Scientific Conceptions

One major advantage of MEMO is that, though it offers a process model as well as heuristics and techniques to guide with the design of enterprise models, unlike other general purpose modelling languages such as the UML, MEMO languages permits the use of more perceptive representations of different perspectives of an enterprise. It achieves this by providing a specialized semantics for various purposes such as organizational analysis and design, strategic planning, information analysis and software development. It also supports the structuring of a problem domain according to established professional principles. Compared to other methods for enterprise modelling, like ARIS (Kruppke et al., 2006); deployed in modelling business processes analysis and management of Computer Integrated Manufacturing Open System Architecture (CIMOSA), the MEMO modelling

languages are more mature in terms of completeness and precision (Neaga & Harding, 2005). With regards to the critical importance of standardization and the more extensive use of UML when compared to MEMO, it has been argued that perhaps MEMO is unnecessary. However, originators of MEMO contend that UML suffers from a number of shortcomings which are detailed in the works of Prasse (1998) up to the works of Alhumaidan & Zafar (2014) such as clarity of notations, completeness and correctness of language descriptions.

From a methodological point of view, the evaluation of modelling languages is a delicate task. This is due to the fact that requirements cannot be specified in a comprehensive way. Some of the features depend on subjective preferences which may vary from user to user and over time. Therefore it is impossible to optimize a modelling language straight off. Instead a language has to be evaluated by prospective functionality against the purpose it is designed to serve. Within several projects, though MEMO, ArchiMate, UML, IDEF, DEMO and I* modelling languages have been evaluated by users with similar and different backgrounds, it has been suggested that the result of the evaluation is ambiguous especially with the determination that deductions obtained from people with similar backgrounds have been substantially inconsistent (Prasse, 1998; Almeida & Guizzardi, 2013).

In conclusion, there is no doubt from this analogy that multi-perspective enterprise model is a favourable subject of knowledge elicitation with EAML. Thus in consideration of the key competence required for exposition of the extensibility of EAML, the ArchiMate which provides this capability is adopted to create the required extensions in order to demonstrate the conceptual foundations of the proposed knowledge management and infusion. To this end, the extended Enterprise Architecture Modelling Language would be incorporated with the capability to validate the model as hypothesized in this work and to provide a blueprint for transformation to ontology, the alignment of business process with motivation through traceability as well as serve as a useful reference models for change management. The subsequent sections therefore look at more closely the rationale for adoption of EAML and justification for the selection. The aim is to provide the groundings for the extension of the EAML, which aspect and constructs needs to be extended, how the validation artefacts will be integrated, what metrics are required for the validation, how validation will be related to motivation and how the model will be transformed and queried.

4.4 Rationale for Adoption of EAF and EAML for Validation Extension

To implement validation on model artefacts, a way to extend the EAML while maintaining consistency and integrity of the code base is needed. One of the contributions of this research is to be able to derive this extension. Thus it is of critical importance that a methodological choice of an EAML which underpins the research is identified. Amongst all the Enterprise Architecture Modelling Language evaluated, only ArchiMate stands out offering this flexibility of extensibility with an open source code available to enable the execution of the extended upgrade. To ascertain if this choice is

appropriate, a further comparison is carried out to rationalise and determine the suitability of the ArchiMate for this work. This is benchmarked in figure 4-6

Rationalising TOGAF & ArchiMate

TOGAF and ArchiMate	Other EAF and ML
Well developed EAF (ADM) and Language (ArchiMate).	Generic or Domain Specific. Any tool may be used to depict structural components.
Offers formal descriptions of components.	Does not offer consistent component description as it depend on the tool used.
Comprehensible and consistent graphical language for representation of EA models.	Graphical language for representation is inconsistent as it depends on tool or ML.
Extensible open source code with the capability for language extension.	Closed sourced code with no capability to extend the language. E.g. UML, Junit.
ACF supports ADM and checklist of Architectural output.	Maturity matrices use in most instances for assessment.
Highlights shortfall between the baseline Architecture and the Target Architecture	Highlights shortfall between the expected output and current Output.

Figure 4-6: Justification of selection of EAF and EAML

The rationale for adopting TOGAF and ArchiMate as a platform for incorporating validation is based on several factors. Firstly the ArchiMate offers formal descriptions of components and supports reasoning about the structural and behavioural properties of the enterprise. This is specified very distinctly in the TOGAF ADM. Secondly, ArchiMate provides a comprehensible graphical language for representation of EA models. Thirdly the extensible open source code of ArchiMate provides the capability to define relationships and annotations required for validation and supporting semantics. This advantage is augmented with the ability to specify definitions for new artefacts as well as establish relations between concepts, between different layers or aspects of the architecture. Fourthly, metamodels defined for the motivational elements such as Stakeholder, Goal, Constraint, Driver and Assessment can be modelled within the extension and related to the core components at the Business Layer to establish associations with the information, business and structural aspects of the EAF. This is of critical importance as it enables introspective insight into the alignment between the business processes, their supporting applications and the technical infrastructure.

While ArchiMate does not claim to completely cover all aspects of EA modeling, it supports specific extension of modularity for new concepts, relationships and attributes. This facilitates tool additions and methodologies which enhances and supports the overall ArchiMate language. TOG demonstrated this extensibility by the addition of the *Motivation extension* and the *Implementation and Migration extension* (TOG, 2012) in its version 2.0. Whereas the core concepts of ArchiMate focus on describing the architecture of systems, the elements which provide the context or reason underlying the design and operation of the enterprise is covered by this motivation extension. The motivational aspects equates to the “Why” column of the Zachman framework and addresses the way the enterprise architecture is aligned to its goals.

Because this research intends to extend ArchiMate with validation capability, it is worth noting that TOGAF has already incorporated in the Architecture Content Framework (ACF) a basic premise for validation. However, it is acknowledged by most practitioners, authors and even TOG that there exist representations of highlights of shortfall between the baseline architecture and the target architecture in the ACF. ACF identifies items that have been deliberately omitted, accidentally left out or yet to be defined (Chapurlat & Braesch, 2008) within the model but does not validate the model against motivation. Therefore as validation with ACF attempts to confirm that the architecture supports all of the essential information processing needs of the organization but does not validate the model against motivation and constraints, this research is justified; not only as a new innovation which extends EAML to allow creation of models with validation capability executable through ontology query but also as a leverage that broadens the provisions of TOGAF, ADM and ACF.

5 EA VALIDATION: ENABLING THEORIES AND PRINCIPLES

The research deals with model validation based on behaviour specification and motivation of EA models. Though the research leverages on several concepts that relate to modelling and validation, it is inspired by Behaviour Driven Development (BDD) concepts in the articulation of the test scenarios. The rationale for this is based on the need to analyse the specifications of business behaviour defined predominantly at the Business Layer of the EA framework. The distinction of the methodology from BDD is signified in the transformation of this behavioural specification and artifacts to ontology and the design of test basis based on motivation specification. Three theoretical principles guide this process. These are (a) Specification of model validation views, (b) Validation of the rule on the metamodel instance and (c) Evaluation of results with motivational goal.

5.1 Principles for Artefact Verification and Validation

There are many theories that relate to verification and validation of a model and its development process. Of particular interest to this work are two concepts of the theories described in a paradigm which shows common ways to view this diligence within the EA domain. One way uses a simple view and the other uses a complex view. In reviewed works using both of these ways, it has been concluded that the simple way more clearly illuminates model verification and validation (Chapurlat & Braesch, 2008). For this reason, the simple view is adopted in this work. The paradigm of the simple way considers the simplified version of the model development process as consisting of the problem entity. This comprises of motivations, constraints, principles, policies or phenomena to be modelled. This is supported by the conceptual model which is the logical or transcribed representation of the problem entity developed for a specific case.

5.1.1 Conceptual Model Validity Theory

The conceptual model is developed through the analysis and modelling phase. Inferences about the problem entity are obtained by conducting tests on the model in the validation phase. Relating the model validation and verification to this simplified version of the modelling process, firstly, Conceptual model validation theory is defined as the determination that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is reasonable for the intended purpose of the model (Sargent, 2005). Secondly, Conceptual model validation theory affirms that the model's representation of the problem entity and the model's structure, logic and relationships are realistic for the intended purpose of the model. The theories and assumptions underlying the principles emphasize testing using analysis and testing methods on the entity data. Examples of these principles are linearity of assumptions, independence of data, and artefacts traceability. Additionally every model instantiation of the Metamodel needs to be evaluated to determine if they are reasonable and correct for the intended motivation of the metamodel. This includes also determining if the appropriate detail and aggregate relationships have been used for the

model's intended purpose, and if appropriate structure, logic, constraints and relationships have been used. The primary validation techniques used for these evaluations are visage validation and traces. Visage validation entails visual evaluation of the conceptual model to determine if it is correct and reasonable for its purpose. The use of traces is the tracking of entities through each model and the overall metamodel to determine if the logic is correct and if the necessary accuracy is maintained.

Closely associated with this theory are the Operational validation and the Data validity theories. The Operational Validation theory determines that the model's output behaviour has sufficient accuracy for the model's intended purpose (motivation) over the domain of the model's intended applicability. The Data Validity theory conversely asserts that the data necessary for model building, model evaluation, traceability, and executing the model's tests in order to validate the model are adequate and correct.

5.1.2 Data Validity theory

In consideration of the Data Validity theory, data are needed for three purposes: for building the conceptual model, for validating the model, and for performing simulations with the validated model. In EA model validation, data is of more relevance only for the first two purposes. To build a conceptual model, sufficient data on the problem entity is required to develop concise metamodel that adequately represents the logical business behaviour of the entity, its specified motivation and to test the model's underlying assumptions. Additionally, behavioural data are needed on the problem entity to be used in the operational validity phases of comparing the problem entity's motivation and goals with the model's behaviour. High model confidence and validity is difficult to attain if sufficient operational behaviour data are not available. Also if data transformations are made, such as disaggregation, formalization or ontology mappings, they would be prone to errors if operational data is unreliable. Unfortunately, there is not much that can be done to ensure that data are accurate. The best that can be achieved is to develop good procedures for collecting and maintaining data, test the collected data using techniques such as internal consistency checks and development of a data repository for large data consortium.

5.2 Principles for Specifying Model Validation Rules

A Validation rule is a criterion or constraint used in the process of EA model validation. It is carried out after the model has been developed. This is a synonymous with formal verification where the behaviour of the model is determined to be as intended by its motivation. The Validation rule still used by many practitioners checks model design, artefacts, attributes and relationship definitions to established limits on what constitutes validity. In the context of EA, formal verification specifies this act of proving or disproving the correctness of intended model design and logic underlying a metamodel and model with respect to specified motivation or constraints. The theoretical principles for validation rules specify two distinct levels of validation applicable in EA. These are the active and

passive levels. These rules applied in EA enhance the quality of its artefacts and produce a more unified validation methodology applicable across several layers of the architecture. The classification into levels introduces parallelism in the validation process as thorough objective tests can be performed. Although the graphic representation of information for the levels use direct contiguous relationships, the levels of abstraction are also useful across the derivative model instances. Consequently, the two levels are also complementary and best suited for EA models.

5.2.1 Active Validation Level

An active level of testing evaluates appropriateness and relevance of the model and is based on explicit validation elements. In the implementation, this is targeted at the resource description framework schema. Thus the Active level of validation is focused on the relevance of the artefact as defined by the scope of the EAF under test and involves the following steps:

i. Identification of the business behaviour that the artefact is expected to exhibit.
ii. Conceptualization of input data based on the behaviour specified.
iii. Determination of expected result based on the behaviour specified.
iv. The execution of the test case.
v. The comparison of actual and expected result.

The Active Level of model validation is focused on the functionality of the artefact as defined by the scope of specified validation metrics and component under test. It captures the business behaviour that the model artefact is expected to exhibit stated in the form of scenarios. The input data is conceptualized based on the constraints, behaviour and goals specified. On execution of a scenario, the actual result based on the behaviour specified is determined and compared with expected motivation. The active level of testing can be extended to include annotations that aggregate the rules of the model as well as associated with motivation and constraints.

5.2.2 Passive Validation Level

A passive level of testing on the other hand adds substantiation to the model and examines interface and relationships of the model to determine traceability. In the implementation of this study, this is targeted at resource description framework graphs generated with reasoners. The level of formality and documentation necessary for passive level of validation varies depending on the complexity of the model. In general, validation documentation are necessary as this can be used to verify the adequacy of a given validation suite. They can also be used to repeat validation cases for the sake of traceability and interoperability of the models. Passive level of valuation examines this collaboration between several models and interactions with different parts of the Business Architecture and consists of the following levels of model collaboration:

i. The interface of business artefacts with motivational goals from view abstractions.
ii. The interface of business artefacts with constraints to enforce cohesiveness.
iii. Interactions on a view from views within the same viewpoints.
iv. Collaboration from views within the same viewpoint to achieve same goals.

Consideration of both levels of validation is important in the determination of validation rules and assessment of the integrity of the EA. The rules facilitate the codification of query constructs and semantics which identify artefacts that constitute overlaps and gaps within the model. Both levels can also institute strategic guidelines and a framework for referencing as a way of standardizing and formalizing the modelling processes.

5.3 Theoretical Principles for Goal Evaluation

Goal evaluation is recognized as an essential skill set for practitioners in service-related fields including computing, management and information systems. Recently, the increased need for evaluation of enterprise architecture has been driven primarily by need to improve efficiency and reliability of its models. However, many enterprise models are deficient of the necessary grounding that would facilitate evaluation of their artefacts. Like any good methodology for analyzing changes in information systems, evaluation requires clear definition of intended motivation and goals, identification of measurable indicators of success, and formulation of procedures for achievement of the goals. This differs from other method of effecting evaluation in a few ways:

i. It requires clear traceability from as-is to to-be architecture by specification of the artefacts needed for the goals to be realized.

ii. It requires articulation of the underlying assumptions and constraints which need to be tested and measured.

iii. It changes the way of rationalizing about the motivation from current architecture to future architecture.

In consideration of the fundamental principle of EA validation which is bound by occurrence of change and ascertaining the validity and impact of that change from as-is to to-be architecture; from current architecture to future architecture; from business to technology architecture, the Theory of change becomes relevant. Theory of Change provides a roadmap needed to accomplish the stated transformation. It focused not just on generating knowledge about whether an approach or concept is effective, but also on explaining the methods that are applied to achieve the effectiveness (Coryn et al., 2011). Theory of Change as a concept has not only strong roots in a number of disciplines including environmental and organizational psychology, sociology and political science, but has also been increasingly connected with information systems, systems engineering and enterprise architecture. Within industrial-organizational psychology, Cummings and Worley (2014) noted that approaches to organizational development are frequently based on more or less explicit assumptions

about the processes through which organizations change, and the interventions needed to effect the change. Within EA evaluation practice, the key reason complex models are difficult to evaluate is that the assumptions that inspire them are poorly articulated. Coryn et al., (2011) argued that stakeholders of complex EAF typically are unclear about how the change process will unfold and therefore place little attention on the early and mid-term changes needed to effect pragmatic validation.

5.3.1 Building Blocks of Theory of Change

The Theory of Change defines all building blocks required to realize motivational goals. The set of building blocks are connected and specify the outcomes, results, constraints and assumptions. This composition is depicted on a map known as a change framework as a graphic representation of the change process. Built around the change framework, the Theory of Change describes the types of triggers that bring about the outcomes depicted in the pathway of a change map. Each outcome in the pathway of change is associated with assumptions, a trigger and constraints, revealing a trace of activities that forms a complex web required to realize the goals. The assumptions are supported by scenarios from case studies which strengthen the hypothesis about the plausibility, feasibility and testability of the methodology and the likelihood that stated motivations will be accomplished. The case studies used in this research are intended for this purpose. Articulated effectively, the roadmap is comprehensive and can easily be deployed in the validation of EA models.

5.3.2 Applying the Theory of Change

In applying the Theory of Change to EA modelling, an important first step in the roadmap is identifying the value of the proposition and outcomes. Once the value of the proposition and goals are identified, constraints that must be applied in order to realise the goals are identified. In practise such constraints are depicted as requirements on the Theory of Change pathway underneath the associated goals. These requirements act as preconditions towards the realization of the motivational goals. The process of identifying preconditions continues, drilling down the pathway by posing fundamental questions such as: “What needs to be in place for the goal to be achieved?” and “Are these preconditions adequate for the goals to be achieved?” By these means, the theory of change is evolved and enhanced.

5.3.3 Success of the Theory of Change

The success of the Theory of Change lies in its ability to demonstrate progress on the achievement of motivational outcomes and goals. Evidence of success confirms the theory and indicates that the initiative is effective. Therefore, the outcome in a Theory of Change usually is coupled with indicators that guide and facilitate measurements. Indicators drive the outcomes making the outcomes understandable in concrete, observable and measurable terms. The relationship of indicator to outcome however can be complex in some cases. In this work, this is clarified with the use of simple assertive post-conditions defined in the “Then” clause of the adapted BDD concept. In this

implementation, every goal on the outcomes pathway has at least one indicator designated optionally on a priority scale.

5.3.4 Evaluation and Monitoring Theory of Change

As the origins of Theory of Change lie in the field of evaluation and monitoring, developments over the years have ensured that Theory of Change continues to be an invaluable method to conduct evaluations of many different types of projects and organizations. Often advancing theory-based evaluation questions helps to focus evaluation efforts on pivotal concerns. Monitoring questions can be adopted to formulate the right indicators from among many available. Empirical studies has shown that instinctive replication or scaling an intervention hardly ever works (Taplin et al., 2013). An important task for monitoring and evaluation is to gather enough knowledge and understanding so as to be able to predict how an initiative and set of events might work in a divergent situation or adjustments that needs to be affected in order to get a similar or better results. There is also the need to combine evidence from a number of case studies in order to build a stronger articulation of what is taking place, how it is progressing and, most importantly how the context influences the initiative.

5.3.5 Comparison and Rationale for adoption of Theory of Change

Practitioners have developed logical models and logical frameworks as strategies and tools to plan and evaluate enterprise architecture models (Funnell & Rogers, 2011). While these models well articulate the goals and resources of the enterprise, they give less focus to the complex structure, attributes, business behaviour and relationships that underlie changes from as-is to to-be in EA modelling. Thus, while logic models and logic frameworks have developed an Implementation Theory behind their work, they lack an underlying Theory of Change (Funnell & Rogers, 2011). Theory of Change also contrasts with logic models and logic frameworks as it starts with a participatory process which clearly defines desired outcomes and assumptions. In this work, this is specified by use of the VPEC-T (Green, 2007) concepts. Theory of Change begins by first working out EA goals or desired impact and working backwards on outcome pathways, rather than engaging in conventional forward oriented reasoning. Though many organizations, including the United States Agency for International Development have used a logical framework and companion Scorecard as evaluation tools for EA (Taplin et al., 2013) the logical framework is often complementary and adaptable to a Theory of Change-based monitoring and evaluation system. This is because the logical framework just like the logical model is limited as they do not show causal connections between conditions that need to change in order to meet the motivational goals. The added value of Theory of Change lies in its ability to reveal the conceptual model, including the causal relationships between and among outcomes, the relationships of activities to outcomes, and of outcomes to indicators. Overall, having a Theory of Change helps make explicit the assumptions upon which the logical framework is based.

5.4 Theoretical Foundations for Enterprise Systems and Structures

Enterprise Architecture is increasingly represented as a discipline that is concerned with the design, construction and use of artifacts based on information technology (IT) and theories. The underlying observation for theories is that they are created to explain observations and help predict new ones. These observations need to be measured, collected, and validated. In the case of theoretical foundations for validation of EA, more information is needed from research but establishing the list of observations regarding EA can provide a good starting point. At the highest level, the basic premise of Enterprise Architecture is expressed in its theoretical foundation. EA hypothesis is expressed as **"***t*he structure of both intentional and unintentional relationships among enterprise systems which has a direct and measurable influence on the rate of potential change and the organizational cost of operating and maintaining those systems" (Hevner, 2007). This theory demands that a definition for "enterprise system" and a method for describing the "structure" of an enterprise with respect to those systems and the description of the "relationships" between components should be created. Clearly enterprise system should include socio cultural systems, information technology systems, workflow systems, and governance systems (Goethals, 2003; Lankhorst, 2013). The EA theory suggests that the relationships between these systems are important as it influences the rate of potential change. The EA theory also demands that the rate of potential change should be validated, and that the organizational cost should be described. To do the latter, there is need to develop a clear idea of what is involved in operating and maintaining each of the included systems.

The theory is also fairly unbounded leaving critical questions that require answers. Can the meaning of "system" be clearly and concisely defined such that two architects independently examining the same enterprise would develop the same list of systems? What are the types of relationships among systems and how can relationships be differentiated? Can attributes of relationships be distinctively defined? Does it apply to one system or a subset of systems? Or can it only be truly understood to apply to the complete system-of-systems that is, in effect, a complete description of the enterprise? What standard methods can we develop for identifying all of the relevant systems of an enterprise effectively for the purpose of understanding the architecture of the enterprise in its entirety? Clear theoretical foundations of Enterprise Architecture are thus needed as answering these questions can be difficult to exact. Firstly, a list of valid observations that require explanation and understanding is needed. Secondly, a simple reusable method for conducting research in the area and a consistent way to count and categorize systems across different types of domains and enterprise is required. Lastly, evidence of the cause and effect of making changes is necessary. This ensures a solid understanding of the value of the changes for the theoretical foundation. Thus the rationale for grounding the theoretical foundation for EA validation as presented in this research is a requirement expressed

through the strategy placed on expanding the capabilities of the methodology through targeted, specific and managed changes to the metamodel. These theories explain why Enterprise Architecture, as a field in computing is effectual and can deliver the expected effects.

5.4.1 Information Systems Design Theories

Information System (IS) is defined as a field of research concerned with the effective design, delivery, use and impact of information technology in organizations and society (Gregor, 2006). Information Systems is concerned with the design of artefacts and their use in human-machine domains and involves theories and practices to achieve these goals (Martin, et al., 2004; Gregor, 2006). The goal-oriented perspective of IS has created a rising interest in designing theories within the information system community (Goldkuhl, 2004) as it enables the enactment of principles from best practices at operational, management or strategic levels. In general, five types of theories can be distinguished in relation to Information Systems: (I) analytical and descriptive theory, (ii) theory for understanding, (iii) prediction theory, (iv) explanatory and predictive theory, and (v) theory for design and action (Gregor, 2006). IS Design Theory (ISDT), which is related to this research is considered part of the theory for design and action and is concerned with how to design the artefact (design product) and the design process (method being used to realize the product) (Kourouthanassis, 2006; Walls et al., 2004). The design product is composed of (I) the meta-requirements used to deal with a class of problems or goals to which the theory applies (Siponen, 2006), (ii) meta-design principles, which describes a class of artefacts hypothesized to meet the meta-requirements, (iii) kernel theories which are relevant theories derived from natural or social sciences governing design requirements, and (iv) testable design product hypotheses, which are used to validate the match between the artefact outcome and the meta-design. By addressing all these elements in conjunction with each other, IS design theory is often thought of as a complete package of guidance for designers facing particular sets of circumstances. However, IS design theories are also regarded as normative theories. That is, they are prescriptive and evaluative, rather than solely descriptive, explanatory, or predictive. Because IS design theories are intended to give guidance to developers, they must not only pass scientific tests of explanatory or predictive power, they must also pass the tests in practice. Its primary contribution is to formalize, justify, and extend the traditional IS practice of labelling system types (e.g., DSS, ESS, and EIS), describing their characteristic features, and prescribing an effective development approach. The value of an IS design theory is to reduce developer's uncertainty by restricting the range of allowable system features and development activities to a more manageable set, thereby increasing the reliability of development and the likelihood of success (Markus, et al., 2002).

Other practitioners extend aspects of ISDT to include the design process (Walls et al., 2004; Siponen, 2006) and comprises of methodologies, guidelines, principles and tools that are used in the

development of the artefacts (Gregor, 2006). This advances the design process by restricting available options, and thus reducing developers' uncertainty and leading to better development results (Markus et al., 2002). Furthermore, this IS design theory allows researchers to generate testable research hypotheses that can be empirically validated using both positivistic and interpretive research methods (Siponen, 2006; Markus et al., 2002). This relates to the hypothesis proffered in this research as IS design theory is drawn on three interconnected elements similar to this work, namely: (I) a set of user's requirements and constraints, (ii) a set principles for selecting system features and perspectives, and (iii) a set of principles deemed effective for guiding and validating the design process. IS design theory is also based primarily on a theory referred to as kernel theory. This theory provides much more practical implementation methods to practitioners (Gregor, 2002; Markus et al., 2002). Many researchers have already used this aspect of ISDT proposed by Walls et al. (2004) for emerging technologies (Kourouthanassis, 2006; Siponen, 2006). This research follows this trend and applies an ISDT for the validation of EA models.

5.4.2 Relating Kernel Theories to Information Technology

Kernel theory enables formulation of empirically testable predictions that relate to the ISDT and other outcomes such as alignment of system requirements. Given the divergent nature of EA validation, three theories that relate with the Kernel theory are applicable. These are theories that pertain to Business Process Management (BPM), IT business value and impacts, and IT diffusion. As reflective in EA validation and conformable with BPM, business process defines a set of interrelated activities that have definable inputs such that when executed, result in an output that adds value to the enterprise. Consequently, Business Process Management (BPM) draws on business strategies and aims at improving organizational performance in terms of cost, quality, service, and speed (Ulbrich, 2006). There appears then to be a concession that EA's validation should be measured based on goals with regard to specified business concerns by adopting a standardized metrics that gauge and assess the assignment of values to deliverables. However, various literatures on the subject have noted that attributing value to EA can be fraught with complexities. Thus the analytical grounds of the kernel theories though not all inclusive or exhaustive, considerably relate Information systems, business management and ontologies to EA validation.

IT business value and impacts, another dimension of the kernel theory expresses the real potential of validation as may be applied to EA. In support of this aspect of the theory, Sarker and Lee (2002) state that "IT is the central object of redesign in the redesign process and the transformational effects of IT investments; that this needs to be explored by taking into consideration the enterprise IT strategy, IT management capability, external environment and industry factors." Three reference

disciplines provide particularly promising sources for underpinning kernel theories for the design of EA artifacts. These are ontologies, computer science and systems theory. Of these three, ontologies form a significant source of kernel theories as it provides a number of ideas and metaphors that support Enterprise Architectures and validation. Accordingly, kernel theories are explicitly applicable to this work as the transformation of models to ontologies implies that the axioms of the kernel theory can be applied for EA validation. A number of influential design theories for EA, business management and ontologies have also been motivated by the need to make designs easy to maintain, modify and change. Many of the design theories of ISDT make use of the idea that change can be more easily accomplished if limited to a single section or modularised as proposed by Kernel Theory.

5.4.3 Implications for theoretical foundations on EA Validation

The review of the different theoretical foundations for supporting the design and adaptability of EA artifacts leads to some conclusions as to how design theories for Information System Design Theories (ISDT) should be specified with validation. Walls et al. (1992) in their formulation of ISDT components affirms that it is pertinent to specify in a theory what states of a system will be covered. The annotations presented demonstrates that these component are directly associated with changing EA artifacts, and almost certainly express some form of uncertainty over their life cycle and state. It has been argued that as ISDT is improved, the exponent of the theory categorically also reflects on the degree of change they anticipate for their designed artifacts. A further interesting conclusion can be drawn by careful study of the nature of the changes that are in concordance with these theories especially with regards to metamodels explained in section 2.2.1. The theories confirm that it is not only system states that can change but also the basic structure of the system itself. One way of conceptualising these broad directions in which information systems change is to think of an IS schema or model (structure and functions of the system) in addition to the IS states that the system can occupy at different times. When thinking of the way in which an information system changes, we can think of changes both to (i) its model/schema (its basic form and functional capacities), and (ii) its state and relationships (i.e. the changes as it moves from one state to another over time). Though a system's model is related to its design and is the subject of design theory in IS, it is also significant to recognise state changes actuated by change in artefact relationships. A system's capability to change its structure requires that the system has a reflective capability traceable through associations.

In this section, the varying degrees to which IS/IT and EA artifacts can be designed and how these artifacts can be viewed as occupying a space within IS design theories along a continuum that exists between artificial, completely designed artifacts has been highlighted. The properties of changeability

that IS/IT artifacts possess are identified as being a consequence of the dynamism of the business environment and strategy. Varying types of changeability have been explored, drawing on work in a number of disparate areas, including Information systems theory, kernel theories and IS/IT design theories. However, it is not claimed that this presented list of theories is exhaustive, although it is envisioned that it has captured the most salient aspects of providing for validation of EA models.

5.5 Behaviour Driven Development as a Modelling Metaphor

Primarily, behaviour-driven development as a modelling metaphor focuses on behavioural specification of a model from a specific viewpoint of a stakeholder. Essentially, it specifies that for each abstract of a model under test, the following activities can be achieved:

- Definition of a test set for the abstract;
- Implement the test set and finally;
- Verification that the implementation of the tested abstraction yields set goals.

This definition is explicit as it allows validation to be carried out in terms of high-level goal specifications leaving out low-level requirement details and dormant model components exertions. By this approach, validating a model allows specific structural test for appropriateness of model taxonomy to be made more specifically rather than depend on inferences from generic maturity matrices. The advantage is that it allows a model to be used for its ability to create value, enhance compliance visibility and change management rather than another metaphoric documentation requirement. Validating a model by concept of Behaviour-Driven modelling ensures that tests of any abstraction are specified in terms of the desired behaviour of the abstracted component. The "desired behaviour" in this case consists of the requirements set by the business, derived from the enterprise motivation concerns and restricted by its constraints. The validation scenarios are phrased declaratively rather than imperatively in a ubiquitous language with no reference to elements of the user interface through which the interactions take place (Mabey, 2008). Following this approach, reference is made to how the desired behaviour of the model abstract should be specified. To realize this, the approach proffers the use of a semi-formal format for behavioural specification borrowed from problem domain specifications of the object-oriented analysis and design to develop a conceptual model that can be used to complete tasks that accomplish specific goal. This method recommends specification of behaviour in terms of concerns and specifies the business value, explicitly written and annotated with constraints (Chelimsky et al., 2010). The problem domain specification in BDD adopts the following steps;

Title

- The problem domain which has a clear, explicit title.

Narrative
A short, introductory section that specifies;

- The primary stakeholder of the problem domain which can be an actor, group or business process that benefits from the model abstract validation.
- Goals derived through the business behaviour
- The components that are to be validated
- The relationship between the components, business behaviour and the goal.

Validation criteria or scenarios
A description of validation test carried out on a specific case. The scenario has the following structure;

- It starts by specifying the initial condition that is assumed to be true at the beginning of the scenario. This consists of a single or several clauses.
- It states which event triggers the start of the scenario.
- It states the input parsed to validate the component attributes, how the input is accepted, and the logic that characterizes the component and response channels.

Acceptance criteria or scenarios
Finally, it states the expected outcome, in one or more clauses.

5.6 Validation Artefacts

To enable testing of the model, validation artefact are identified as test basis and categorized relative to the model components. Specifications from the business behaviour consist of concerns, test scenarios, constraints and expected outcomes. The attributes of the artefacts in some cases extend to include multiplicity definitions for the object and specifies the types of mutuality of properties and relationship annotation with other artefacts. The test scenarios contain data segregation from constraints and goals in relationship with some measurement metrics. Figure 5-1 illustrates this classification. Test scenarios specify procedural data used in a confirmatory way to verify that a given set of constraints to a given business behaviour for given goal produces expected result. Test data is produced in a focused or systematic way adopting the VPEC-T concepts (Green, 2007).

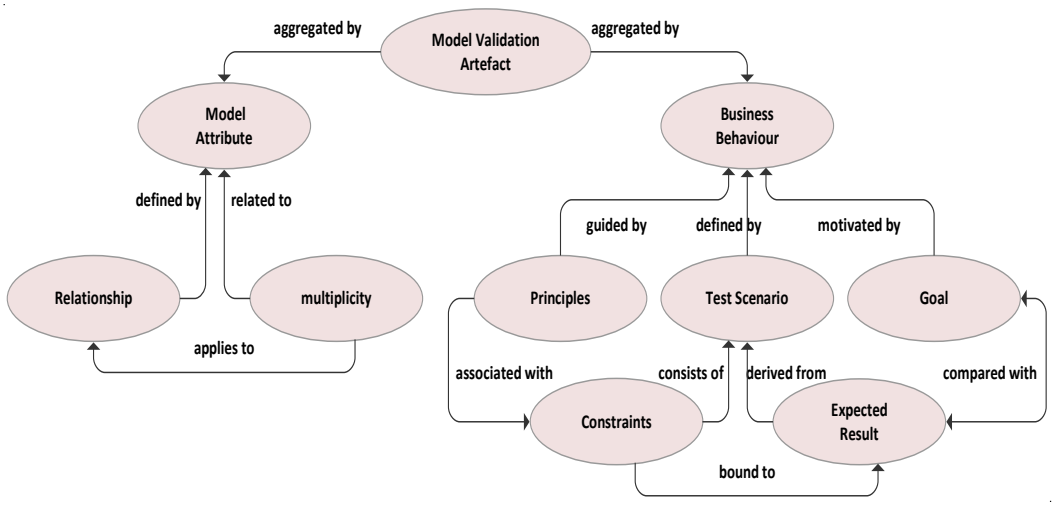

Figure 5-1: Classification of Model Validation Artefact

It has been observed that validation of shared goals across multiple viewpoint cannot be achieved using the approach adopted in this research. Validation can only be done asymmetrically to avoid difficulties in isolating deficiencies to a specific component. At each stage of the validation, the MDVA approach focuses on the behavioural attributes of the artefact, communication between other model components and the maturity of the business behaviour.

5.7 Validation Theme and Elements

The purpose of the Model Driven Validation Approach is to validate viewpoints of a model iteratively, across the three aspects (Information, Business and Structure) of the ArchiMate business layer by testing attributes of the model elements against goals in the motivation Extension. The justification for this approach is that it is detailed, leads to the improvement of the quality and design of the model through goals to component association as well as simplifying the traceability process. The validation scenarios for MDVA represented as BDD features describe the behaviour and attributes of the component to be validated in order to realize set motivation while ensuring better conformance to user goals. The method adopted in the MDVA consists of both the behavioural and the structural attributes of the EA components conforming to the theoretical principles for model validation rules stated in section 5.2. Physical models of business behaviour are created as derivative instances with different stakeholder perspectives for validation. Unlike BDD, test basis created are not extrapolated from the requirements of business specifications but on the business behaviour and attributes of the artifacts that constitute the model instance.

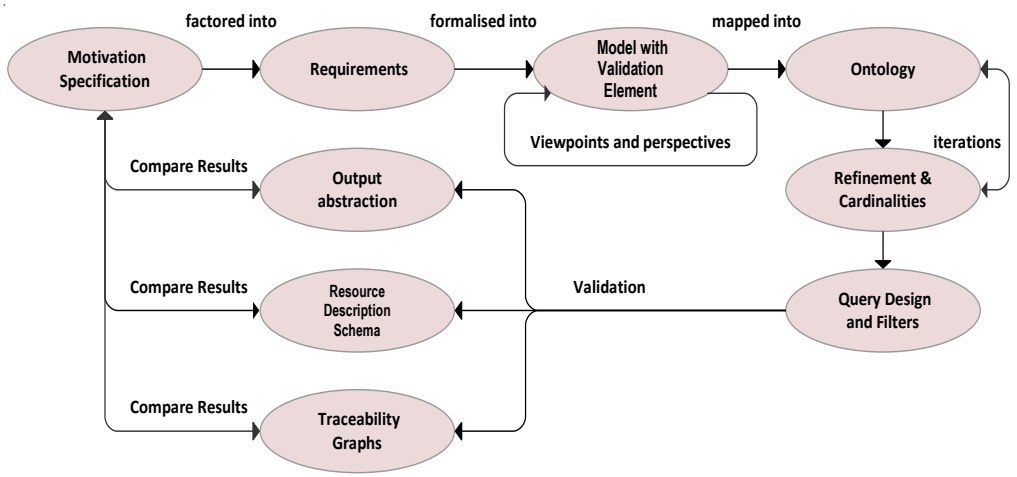

Figure 5-2: Representation of Enterprise Architecture Validation Workflow Cycle

The design of the MDVA is conceptualized from the ArchiMate Motivation Extension by deploying motivational constraints on artefacts of the business layer elements of the core ArchiMate. The methodology iterates correlations of motivational elements over the taxonomy to establish extent and coverage of the business behaviour defined. Through the validation process, gaps and overlapping functionalities are also identified allowing the model to be more concise and purposeful. Figure 5-2 shows an overview of the morphology of the MDVA concept proposed in this research. Validation themes are defined by a set of motivational specifications for the components to be tested in the model. The validation element modelled with the ArchiMate Business core defines the metrics and what types of test are to be carried out on the components and the expected results. Test attributes are specified at this level. Constraints expressed within stakeholders concerns and principles are applied on the selected components to construct the test basis. This stage is iterated to develop the various viewpoints such as process and functional models that would be mapped into the ontology as stipulated by the Theory of Change. The development of the ontology is also an iterative process that allows for refinement of domains, ranges and cardinalities. By implementation of the BDD concept, language semantics is built with the precondition and post conditions adduced to form the queries. Traceability amongst the other components is established by exerting associated relationships. Evaluating the result yields three outcomes;

- Output abstractions that allow comparison to ascertain if the tested goal is realized,
- Resource Description Framework Schema that provides the construct to executing queries
- And Resource Description Framework graphs that facilitate traceability.

To enable the identification and rationalization of the validation metrics for the EA models, study of standards specified by bodies such as COBIT, IEEE, ISO and TOG were conducted and cultured with model driven composites. Several validation metrics and elements were analysed in relation to their characterization and congruity. Through heterogeneous sampling method, five of the elements which exhibited properties that reflect enterprise architecture maturities were selected. These elements are

redefined in context with this work to give it distinctive meaning, relationship and representation. These are Goal Realization, Perspective Visualisation, Behaviour Analogy, Model Traceability and Constraint Assessment.

Model Traceability *VE (MTV)* – Traceability validation determines whether the artefact required for the actualization of business behaviour is available and can be traced to motivation specification.

Perspective Visualization *VE (PVV)* - This refers to the assertion that the perceptibility of the accessed artefacts are in consonance within specified privileges, roles and interfaces of a viewpoint.

Business Behaviour Analogy *VE (BAV)* - This validation deals with relationships amongst artefacts and their ability to function as expected in normal and unusual situations when triggered by events.

Constraint Assessment *VE (CAV)* – This is validation to determine whether a component meets some specified constraints stipulated for actualizing a desired business behaviour.

Goal Realization *VE (GRV)* - This validation assesses to what extent the intended business goals are achieved in relation to either the actual validation outcomes or impacts on other components.

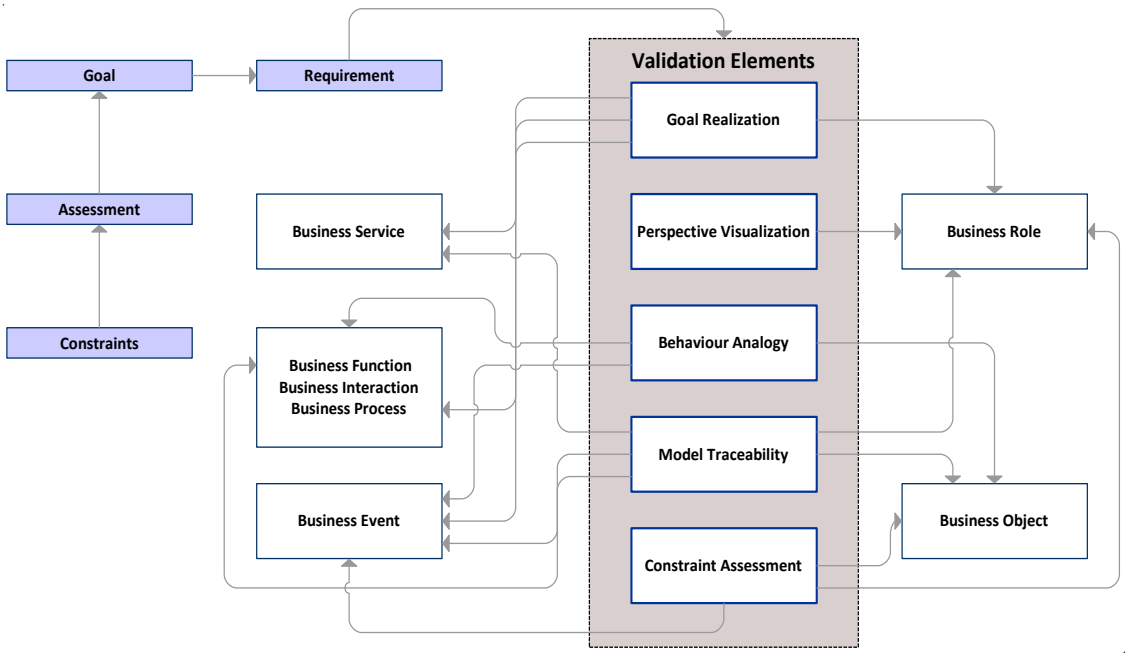

Figure 5-3: Mapping EA Validation Metrics to Motivation and ArchiMate Components

Figure 5-3 shows the validation elements adopted in this research with ArchiMate Business Layer core components mapped into motivation Goal from Constraint. The process is iterative and subsequent cycles refine the model always providing the status of the model and goals achievement. The goals validated on the model are part of the motivation extension of ArchiMate while the components are part of the ArchiMate core. A precursory mapping associates these validation elements with core testable components in the Business Layer of ArchiMate, specifically the Business Event, Business Function, Business Service, Business Interaction, Business Process, Business Role and Business Object artefacts. The Business Behaviour is aggregated by Business Function, Business Interaction and Business Process.

5.8 Conceptual model for the Motivation Driven Validation Approach

The Motivation Driven Validation Approach validates scenarios from specified perspective of model instance derived from its Metamodel. It tests component and relationship against constraints which guide the actualisation of particular goal in the Motivation Extension. The test basis extrapolated are derived from the business behaviour definitions encapsulated with motivational elements such as stakeholder concerns, business drivers, assessments, principles and requirements. The MDVA is realized across the business and motivation layers of ArchiMate. The diagram in Figure 5-4 depicts the concepts. There are three swim lanes of the approach across the Business Layer and Motivation Extension of EAF, modelling with Validation and Ontology transformation. The first section delves into the value of the proposition and decomposes the EAF to its motivation and components.

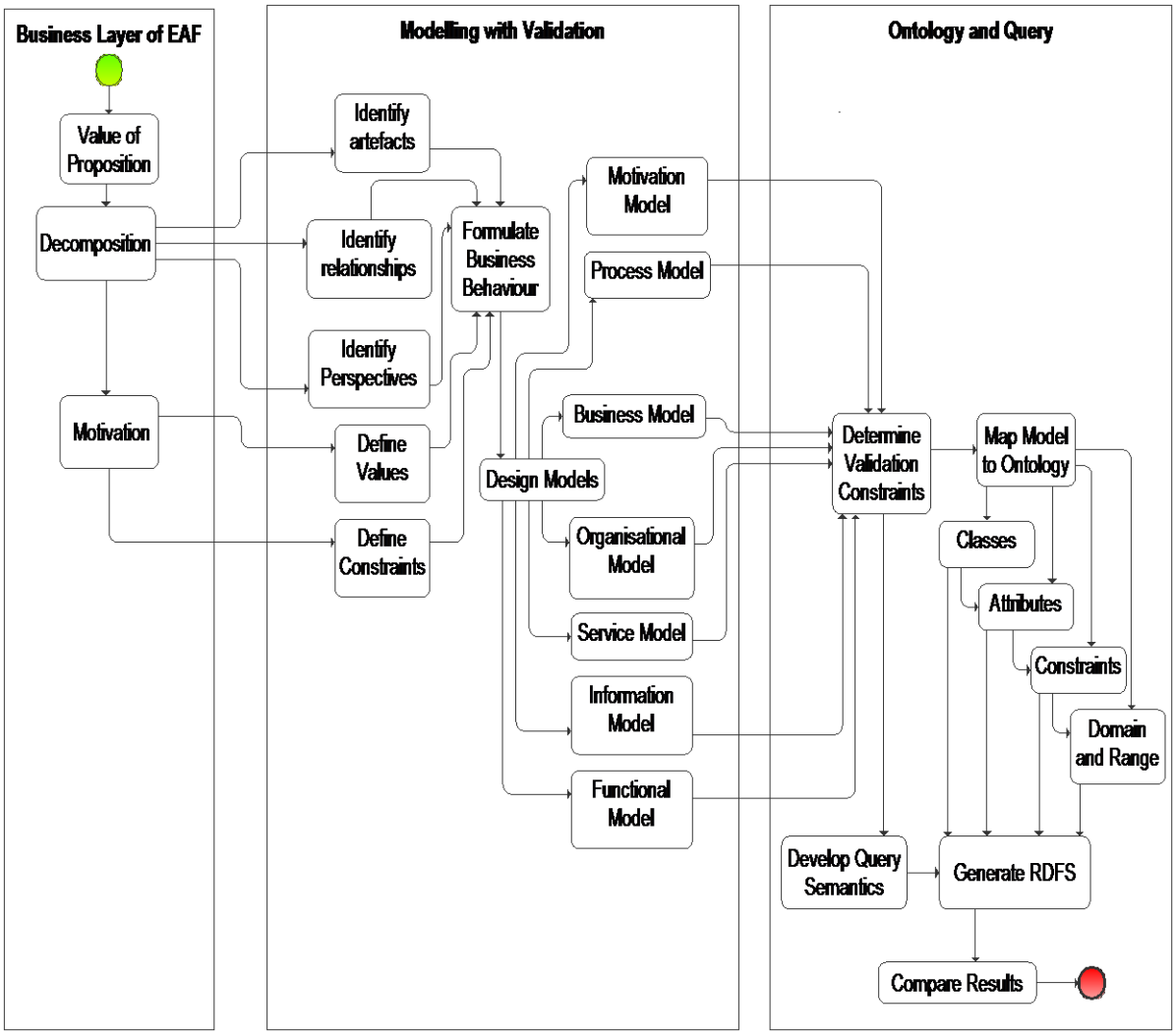

Figure 5-4: Workflow Diagram for the MDVA

The second section depicts the modelling aspects and comprise of metamodels and its instances. Case studies are applied at this level to exemplify the methodology. This is in conformity with the specifications of the evaluation and monitoring Theory of Change. It also illustrates the realization of business behaviour from the Business layer and Motivational constraints. The model instance

extended with validation concepts are analysed, extrapolated and integrated from specific business behaviour using the VPEC-T (Green et al., 2007) concepts. Validation metrics are realised from the identified business behaviour while the business behaviour is refined to define the models. The third group of activities handles the ontology creation and all aspects that relate to mapping, building of query semantic and generation of the resource description framework. The model instances created with validation constraints are mapped into its equivalence class, properties, domain and ranges in the ontology framework. This is transformed into Resource Description Framework Schema (RDFS) and traceability graphs. The resultant triple constructs of the RDFS are then executed with a series of queries to validate the model against motivation goals. A complete description of the steps required for the adoption of this workflow is given is Appendix D.

5.9 Model Driven Engineering and Description Logic for Ontologies

Model Driven Engineering (MDE) is an emerging technique for development of models. It advocates the use of models and Description Logic (DL) to symbolize significant design decisions in information systems and software development projects. According to the Model Driven Architecture (OMG, 2008), DL provides a framework for annotation of these models. Though there are several significant disparities between DL and metamodeling languages as noted in many literature, it is generally acknowledged that they share a generic set of fundamental concepts such as classification of artefacts or elements into layers, components and classes; the establishment of association between these elements using properties and the specialization of classes and properties into groups, perspectives, domains and ranges.

Description Logic (DL) is of particular interest in this work and is defined as a family of logic languages that are especially suitable to model knowledge in a domain in terms of concepts and roles (Franz et al., 2003). The main characteristic of DLs is their reasoning capabilities. By creating a mapping between a metamodeling language and the DL, two important benefits are obtained:

- A formal and unambiguous definition of the metamodeling concepts that is independent of a specific model repository is obtained. Such a definition is necessary in order to ensure interoperability of metamodeling language.

- The use of existing reasoning tools to analyze and validate metamodels and to detect problems as described is facilitated.

The use of Description Logics in ontology languages in the context of metamodeling has been suggested in the past. Parreiras et al., (2007) presented discussions on the advantages of integrating metamodeling and ontology languages. Their postulations led to the introduction of the OntoDSL language as a means to define new domain specific languages. Dragan et al. (2007) elaborated on the use of UML diagrams to construct ontologies. Wang et al (2006) suggested a partial mapping of Meta Object Facility (MOF) to OWL for consistency checking. The proposal presented in this work is

based on the OWL Web ontology Language with the semantics rooted in Description Logics. This is the standard recommendation from the W3C aimed to improve machine interoperability of web content.

Though the MDE can consist of up to three levels that facilitate the description of taxonomy transformation, in this work two levels are considered as this is sufficient and adequate for model manipulations. The first level is the meta-model (source) which provides the language for describing models and constitutes the basis for the transformation. The second level is the model (target) and an instantiation of the metamodel. The target conforms to the source with some semantic requirements imposed on the target models. In many implementations, transformation of the source or target is expressed by rules defined using ontology. Web Ontology Language (OWL) reasoning is a widely studied topic of research, and many tools have been developed in this context (for instance, the Protégé (Stanford University,2013) and the OWL reasoners Hermit, Jena, Fact++, Racer, among others). OWL reasoning ranges from ontology consistence validation to ontology based inference (i.e. derivation from ontology axioms). Validation of transformed models is a wider topic of research. Model validation involves ontology consistence testing and ontology-based inference. The rationale for this is that the use of OWL provides a suitable framework for validation of model entities and properties. Also the relationship between logic modelling and ontologies is a well advanced and pragmatic.

6 MODELLING LANGUAGE EXTENSION PROTOTYPES

In this section, discussions on the paradigms that relate to the extension of an EA modelling language are presented. Model validation and verification is related to simplified rendition of a formalized taxonomy created with the extended EAML. Formalization of Enterprise Architecture concepts is an area which has continued to constitute a major obstacle in understanding the principles that guide its implementation as ubiquitous use of terms such as models, meta-models, meta-meta-models, frameworks in the description of EA taxonomies and the relationship between the various artefacts has not been exclusive or consistent (Lankhorst, 2013). Consequently variant interpretations of schemas, conflicting methodologies, disparate implementation have ensued. Incongruent simulation of alignment between dynamic business architectures, heterogeneous application systems and validation techniques has been prevalent as well (Jonkers et al., 2003). The divergent and widespread application of EA within enterprises makes it even more challenging to adopt a generic formalized approach in which models can be interpreted or verified (Martin et al., 2004). The unavailability of a unified EA modelling language able to describe a wide range of Information Technology domains compounds these challenges leading to the multifarious contrivance of EA perspectives. This section presents a formalization of concepts towards addressing validation concerns of EA and extends EAML with validation capabilities with definition of the artefacts that constitute the archetypes.

6.1 Prototype Ambiguity

Though several publications have referred to the practice of Enterprise Architecture and associated terminologies such as patterns, segments, governance, perspective, views, viewpoint, etc, research has shown that many concede to its ambiguity. A very good example is given in a concise glossary presented in an article published by the California Technology Agency EA 1.1 on Enterprise Architecture Glossary by Set (CTA, 2011). The fact that there is no common perception of the prototypes and ideologies behind concepts is undisputed. Comparative surveys have been carried out to identify possible dimensions of EA based on timelines of relevant literature, author's background, structural dimensions, differentiation between aspects, motivations, contributions and the elucidation of axioms and terminologies. Depositions from these studies indicate that increasing number of IT practioners and authors use the term EA and its associated phraseology explicitely in their practise and publications to expound strategies that are either restrictive in order to demonstrate their domain requisites, or extended to encompass architectural understanding for all forms of EA ramifications (Braun et al.,2005). These types of inferences constitute irreconcileable extremities that is prevailent currently. Often, there is limited significance in relationtionship between background hypothesis and pragmatic requirement. Considering the maturity and the focus of contributions towards EA, most of the approaches postulated are still evolving especially in terms of applicability, making formalization subject to persistent variations. Frameworks and modelling are often surmised by differentiation depending on the proclivity of the practioner. With majority of presumptions being generic, it would

seem pertitent that enterprise should evolve techniques for validating the models that drive their business strategy in order to ensure that its motivation and goals can be realized.

However, while business views are identified in many EA proposals, business strategy modelling from the perspective of motivation and business drivers are often overlooked (Lankhorst, 2013). Thus IT solutions cannot be traced back to business strategy in a clear and unambiguous manner. The intention to formalize the validation of an extended metamodel for the MDVA aims to establish such process. Therefore formalization of validation extension for MDVA is the rationalization of known validation strategies with precise semantics enabling its model-level usage to provide strategic awareness of EA and propose a conceptual relationship towards Enterprise Architecture models artefact exposition.

Preliminary studies for this work took into cognizance approaches that refer to validation strategy formalization. Reviews affirmed that many adopted prototypes focused on specific domains. Thus generic alignment approaches which formalize conjunctions between business and IT for metamodels are not adequately articulated to address alignment in a comprehensive manner. Formalization is therefore not attained due to methodological ambiguity and divergences that exist within prototypes. Thus, the extent of formalization differ depending on the purpose of the design from motivation to direct EA model, maintenance of metamodel or even the abstract meta-metamodel. As such instantiations do not establish meaningful traceability as expressed by their metamodels and frameworks. Therefore, EAF needs to be formalized in order to enable transformation of semantics and principles from domain specific constructs to unambiguous descriptions of their concepts.

6.2 Characterization of the Validation Extension

As identified by many practitioners, an important and common dilemma that plagues the disparate methods adopted currently for validating and assessing EAF is how to systematically seek information within the framework (Chapurlat & Braesch, 2008). Evidence suggests that the most ambivalent information is that which pertains to the rationale for the design decisions taken for any model (Dutoit et al., 2006). The design rationale confers the justification behind the nature of the validation approach and the reasoning that goes into determining the artefacts of the model that should be validated. Thus capturing design characteristics enables its retrieval, enhances significantly the artefacts integrity and increases the effectiveness of the validation approach. Similarly, the design rationale also supports traceability and promotes collaboration amongst design artefacts, exposes differing EA viewpoints and facilitates integration of model abstractions. For Validation Extension for Metamodels, the design decision which delineates the validation artefacts involves contemplation of the following capabilities;

Capture: The ability to capture information through the process of mental filters or guides using the VPEC-T (Green, 2007) thinking framework would prevent loss in translation from business needs to IT solutions. VPEC-T is used when analyzing the expectations of multiple parties having different views of a system in which they all have an interest in common, but have different priorities and

different responsibilities. System here represents a set of interacting or interdependent entities, real or abstract that form an integrated whole.

Formalization: The proclivity to transform motivation into desired requirements in a formalised way is a justification for extending the metamodel. There are many ways in which metamodel argumentation may be structured. Common thought preferences include the use of schemas having different stakeholder perspectives, constraints and principles, goals representation and key drivers.

Relationship Retrieval: This disposition provides a process for artefact traceability and collaboration. The association between artefacts in this way elicits knowledge encapsulation upon transition across domains.

To exert the characterisation of validation artefacts within EAF, a stereotype is formalized as an extension of the ArchiMate modelling language at the Business Layer. The design decision aims to transverse all artefacts of the Business layer enabling the extended viewpoints to be precise and independent of each other. This augmentative formalization of ArchiMate allows concepts and constructs that span the entire schema. For the relationship retrieval, existing navigation or query-based systems can be integrated such as adapted from BDD and Simple Protocol and RDF Query language (SPARQL). This makes the conception well suited for sharing of knowledge through ontology and collaboration of dependencies.

6.3 Extension of the ArchiMate Business Layer Metamodel with Validation Elements

The Validation extension presents an extension of a generic business layer of EA with embedded artefacts for validating its usability and important specifications for key performance indicators, business behaviour, perspectives and their relationships. To create a model-driven taxonomy that spans the business and motivation layers for validation, a schema designed to relate the motivation elements with core ArchiMate is presented in Figure 6-1. It depicts the conceptualization from a stakeholder's perspective and the transformation of concerns with principles and constraints through Assessment to Goal and Requirement.

Within the ArchiMate Business Layer, the Validation Element is represented as high-level information artefact. Goal on the metamodel extension is associated through Requirement and Composite Motivation to Business Behaviour elements through the ambience provided by the Validation Element. This sub classification allows further query relation to be distinctively applied to the business processes and business function to ascertain the artefacts integrity and effectuality respectively. Requirements which specify the Goals defined in Motivation appropriate a theme to be adopted by the evaluation iteration process. The query structure and semantics of the Validation Element allows criteria specified by constraints to be tested for instance against Business Objects, Business Role and Business Event.

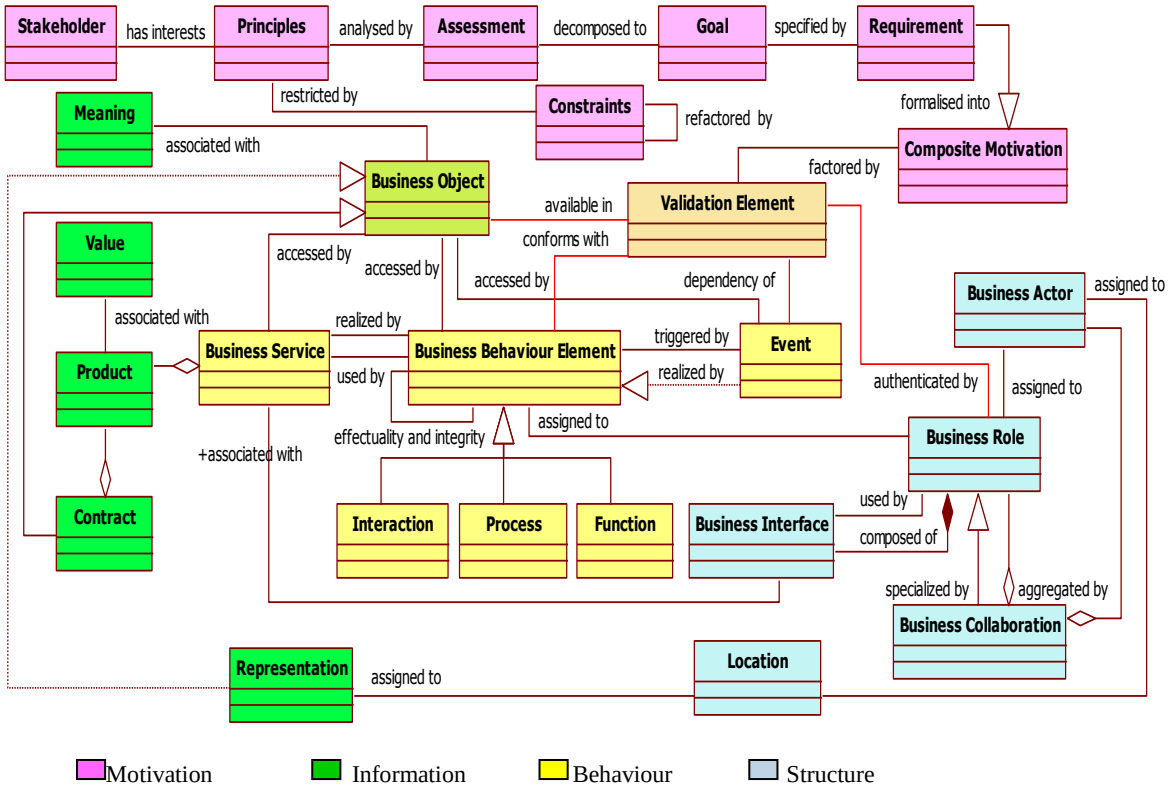

Figure 6-1: ArchiMate Business Layer MM extension with Validation Elements

Further, the metamodel presented in Figure 6-1 depicts a high-level conceptual construct that abstracts information, associates business behaviour with organisational structure and aggregated relationships. Business viewpoints are derived by analyzing Business Roles which are composed of primarily Interface, Collaboration and Actor. However, the Information System Architecture and the Technology Architecture of EA are not covered in this metamodel. This is deliberate as the intention of this work is to espouse the alignment between the business strategy and motivation. The following therefore describe the key elements that denote the metamodel in relation to the extension.

6.3.1 Validation Element

The Validation Element extends the metamodel and represents a high-level business component. The Composite Motivation links the ArchiMate Motivation Extension to the ArchiMate Core though the Validation Element. This is aggregated with validation element attributes which substantiate the business behaviour of the model. This aggregation allows directed queries related to the three aspects of the EAF namely information, behaviour and structure. Requirement specifies the Goals defined in Motivation and appropriates a theme to be validated through Composite Motivation. The query structure and semantics of the Validation elements allows criteria specified by constraints to be tested against Business Objects (Information aspect); Business Function, Process, Services, Interaction and Business Events (Behaviour aspect) and Business Role (Structural aspect). Business viewpoints may be derived from any of the aspects including the Stakeholder element of Motivation.

6.3.2 Constraint

Constraint specifies conditions which the metamodel must adhere by. In an instantiation or model, constraints define the test conditions which bind motivation with the business strategy. Specifically this may be a restriction on the effectuality of the composite behaviour of the model, conformity of business elements, availability of business objects, dependent events, etc. Thus assessment of the metamodel is restricted by constraints directly. Constraints are also bound by principles which are normative guidelines that guide the design of all possible solutions in a given context such as data storage and consistency.

6.3.3 Assessment

Assessment defines the result of an evaluation of a constraint. In this context, assessment exposes impacts of specified criteria on metamodel artefacts thus enabling test procedures to be developed to validate the artefact and related components. It may also be deployed as analytics by adjusting existing criteria or setting new ones in order to evaluate the effect on motivation, the composite behaviour of the metamodel and the enterprise architecture as a whole.

6.3.4 Goal

Goal is an end state that a stakeholder intends to achieve (TOGAF, 2013). However in context of EA metamodel, a goal is defined as the propensity for the taxonomy to produce model instances that encapsulates the meaning and intent of the enterprise. Example would be the ability of the metamodel to affirm the principles guiding its design and specified in its constraints. The aggregated desire of the stakeholder in this context is to assert that the morphology of the metamodel and produced instances yield the desired intrinsic values that constitute the motivation of the enterprise. Consequently, Goals in this case are expressed using qualitative words such as Business Object is Available for access; Business Element Conforms with; Business Event Dependencies are met by; Business Role is Authenticated for; etc. At a lower level of abstraction of the metamodel given by instances, concrete objectives can be used to annotate both the quantitative and homogenous measures of a goal which is essential for the description of the desired state and values of a case.

6.3.5 Requirement

Requirement in this context is a functional business behaviour that the metamodel must be able to perform. It identifies the peculiar attributes, capabilities, characteristics and relationships of the metamodel which describe the desired motivation of the stakeholder. Requirement is an important input into the verification process of the business layer as it offers traceability to goals. Requirement also identifies the business elements that need to be validated and the extent of validation needed. The term “business behaviour” in this context is used in its generic connotation to refer to a group of functionally related elements within a metamodel. It may associate with other active, structural, behavioural or passive elements such as a business role, business event, business interaction, business

events, business process or business object. Therefore Requirement specification is used to extract the properties of business behaviour needed to realise the metamodel's motivation archetype. Accordingly, Requirement represents the means for realization of the enterprise motivation through validation.

6.3.6 Composite Motivation

Composite Motivation of the metamodel is composed of the intentions of the enterprise defined in the requirements, goals and constraints. The sources of these intentions are specified within assessments. Composite Motivation (CM) aggregates the theme for validation and relates with the core elements of the business layer. CM consists of the validation suites and specifies the validation technique, data and approach that establish a viewpoint. Certain factors have been identified as capable of influencing Composite Motivation. These can be internally or externally driven. While at a lower level of abstraction, internal drivers such as customer satisfaction, compliance to legislation, or profitability can influence CM and can be assessed using SWOT analysis, the intrinsic motivation collated in consonance with the metamodel are the external drivers and have been identified as constraints, principles, requirements and goals discussed in previous sections. These external drivers are of utmost relevance to the approach proposed in this work.

6.3.7 Validation Element Attributes

The core of this work is concentric about the Validation Element. The Validation Element (VE) provides the logic, semantics and links to the ontology needed to validate the core business layer of the enterprise. The annotations attributed to the validation of the metamodel are essentially transformed into ontologies in order to allow the description and analysis of the relations between artefacts and composite motivation. The metamodel transformation to ontology expresses these annotations and constructs allowing validation semantics to be interjected in a systematic and logical way. The semantics also provide the basis for which the construct can be query through formalized statements and assertions. With this approach, it is found that when constraints are embedded into test scripts, the resultant construct provides a creditable and consistent morphology for validating the ontology against motivation. This assertiveness is further determined as the distinctive predicate of the construct are explicit descriptions of the artefact. Additionally this culminates into compositions that constraint the metamodel with attributes and mappings which adhere to distinctive and formalized business rule set. A business rule set in this context is a statement that defines or constrains certain aspects of the metamodel and serves as a guideline in determining its behaviour.

6.3.8 Viewpoint

A viewpoint shapes the context of the metamodel with the validation element as viewed from a particular perspective. A number of standard viewpoints for modelling motivational aspects have been defined (Johnson, et al., 2014). Each of these viewpoints presents a different perspective on modelling

the motivation of the EA focusing on defined abstractions of the metamodel. In this research, each viewpoint is an excerpt of business behaviour in relationship to a specific business role, and encapsulates related requirements as extrapolated from a validation theme. The rationale for adopting this approach is to ensure that validation is focused on expected intrinsic values for which the metamodel is designed to realise. Viewpoints can be presented from perspectives of stakeholders. A Stakeholder can be an individual, team, or organisation that has interests or concerns relative to the system. As stakeholders are often influenced by their particular exigencies in many observed enterprise, best practices require that concerns are designed around these exigencies and interests with consideration given to the architecture description of the enterprise. These should also be associated with goals, the present state and future operability of the system in relation to the enterprise goals. Extradition of unnecessary information is also important when conceptualizing the Stakeholder's specific views of architecture. Therefore a view in this proposition maintains distinctiveness by expression of exactitude in perception and description of annotations for visualised stakeholder concerns. Our definition of concepts will be based on the viewpoints of different stakeholders. The enterprise under consideration is the university enterprise information systems expressed in two case studies. As other elements which constitute the metamodel are stereotype and are well explicated in definitions of TOGAF (TOG, 2013), no further explanation is required.

6.4 Logical Conception of the Motivation Extension

The metamodel in Figure 6-2 captures the relationship between Motivation, Business Validation Elements and ArchiMate Core. It shows the modelling of an Assessment as containing nil to many constraints though in general at least one constraint is associated with some assessments. The proposal also accommodates many constraints to many assessments to derive requirements. Likewise Principles can also precipitate goals which can be refined to sub goals as determined by Requirements.

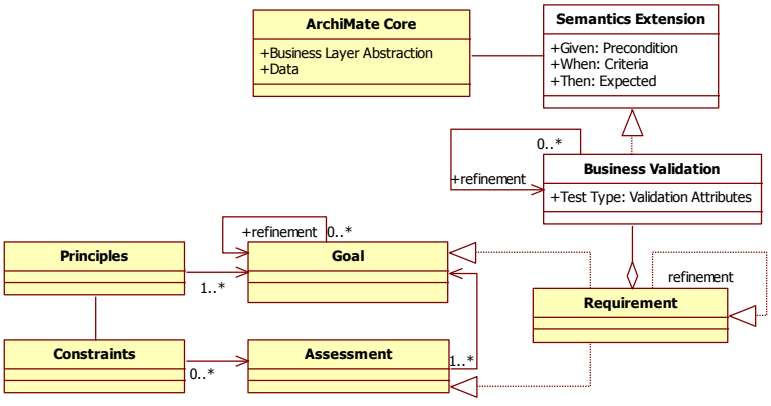

Figure 6-2: Relationship between Motivation, Business Validation Elements and Core

Only Requirement associates with the ArchiMate Core through the Business Validation class as adapted and appropriated by the Enterprise Architecture Modelling Language (ArchiMate) standards.

Though an annotation can be inherited directly, it can also be transitive. In consideration of this requisite which complies with the Theoretical Principles for Goal Evaluation presented in section 5.3, the validation extension attributes inherit the ArchiMate annotations extending the Enterprise Architecture Modelling Language definition and artefacts for validation.

6.5 Domain Specific Modelling with Enterprise Architecture Metamodel Validation

Domain Specific Modelling Languages has been used extensively in Enterprise Architecture to make explicit viewpoints of business concerns. Domain Specific Modelling Language (DSML), a concept stemmed from Domain Specific Language (Lochmann, 2006) is a language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain (van et al., 2000). DSML is therefore adopted in this research as a graphical and textual language to offer visualisation, specify annotations, document abstractions to express the constructs of the extended EA metamodel. This also allows expression, validation, and modification of business behaviour specific to a variety of domains. Empirical studies have also confirmed that DSMLs are superior to more general purpose languages in all cognitive dimensions as quality, productivity, reliability, maintainability, re-usability, flexibility can be enhanced (Lochmann, 2009). Though it has been argued that a disadvantage of DSLs is that its development is costly, onerous and requires both domain and language development expertise, the use of meta-tools has greatly reduced this limitation due to the possibility to reconstruct the base language. This is also part of the reason that favours the application of model-driven approach for the extension of metamodel for validation.

Therefore the extension development process proposed in this study is comprised of four phases. These are the Domain analysis, Extension Design, Implementation and Evaluation. As ample discussions in previous chapters have elaborated on many domains including the ArchiMate which will be extended, further elucidations are focused on the Extension Design, Implementation and Evaluation phases as these are the essential phases for extending a DSL. In the meta-modelling for language extension, the abstract and concrete syntaxes are described using semantics. One way of extending operational semantics is by generating code from the new language to existing code base thus produce an executable version. The mapping between abstract and concrete syntax; and between abstract syntax and semantics may be described using Model Transformations (Sintek & Decker, 2002). To enable the development of the validation extension, a language re-use mechanism is used to create a profile that is generic thus customize the base language with constructs that are specific to the extended element. Thus the extended meta- language inherits the semantics and abstracts with concrete syntaxes of the base language with enhanced capability from constructs which define the extension. A significant advantage regarding this extension is that the validation extension definition and development approach, presented is independent of the chosen Enterprise Architecture Modelling Language. To exemplify the approach and implementation, the rationalized ArchiMate language is adopted.

6.6 Designing the Validation Extension Metamodel

This section provides the extended code for the new element extended using the ArchiMate model. The Archi construct is extended with the artefact Validation Element and associated Relationships at the Business layer. The Eclipse version 3.8 with the open source code of the Archi projects plug-ins found at http://archi.cetis.ac.uk/developer/model-new-relation.html is used as shown in Figure 6-3.

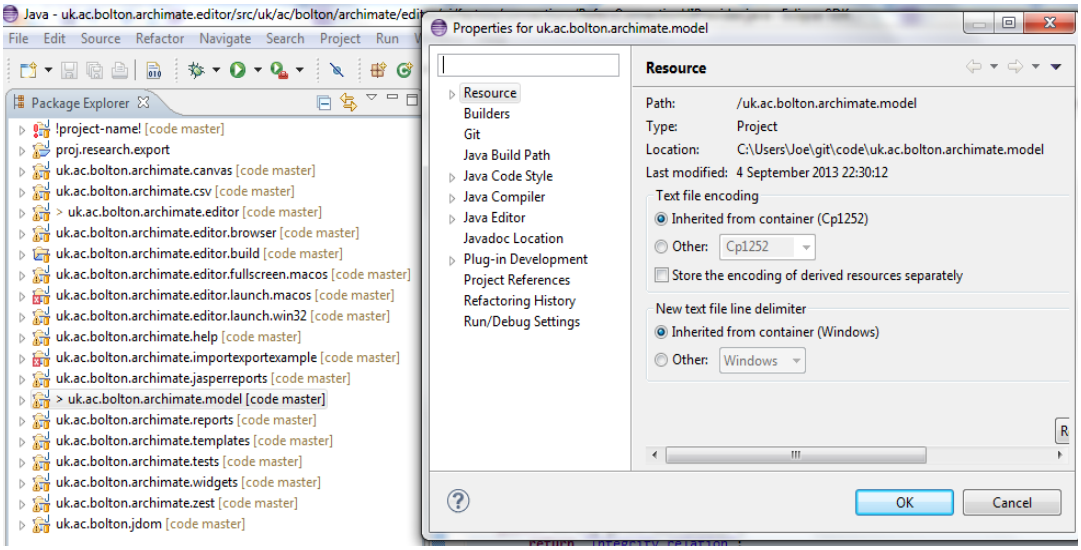

Figure 6-3: Eclipse 3.8 with Archi Plug-ins

6.6.1 Creating the Validation Element in the ArchiMate ML

In Eclipse, the archimate.ecore file in the "model" folder of the "uk.ac.bolton.archimate.model" plug-in is opened. In the tree of the Ecore editor, the "model" node is selected and then a "New Child" selected. A new blank EClass node is created at the bottom of the tree.

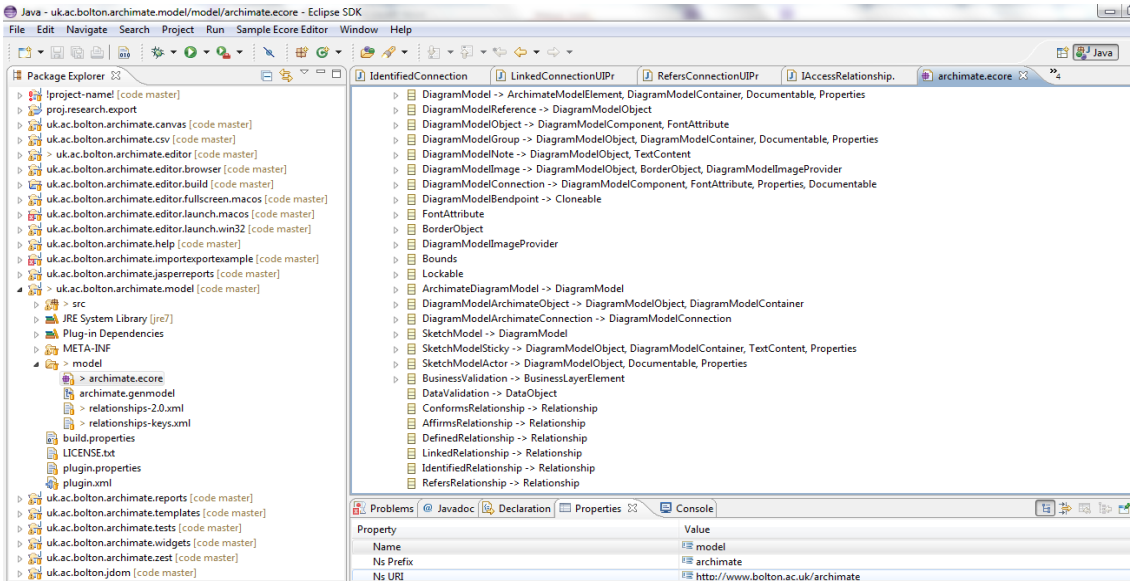

Figure 6-4: Creation of Business Validation as a Business layer Element

The property of the new blank node is edited, renamed “BusinessValidation” and made to inherit the Business layer Element using the dialog box as BusinessLayerElement -> ArchimateElement, see Figure 6-4.

6.6.2 Adding Business Validation to the relationships rules file

As the ArchiMate language defines a strict set of rules for relationships between elements, the relevant relationships allowed between the new Business Validation and the others need to be declared. This is defined in the "relationships-2.0.xml" file found in the "model" folder. This is an XML file that is designed to be reasonably human readable at the expense of prolixity. An element is declared by its name in a "source" tag and allowable target elements are declared as "target" elements. Allowed relationships are set in the "relations" attribute. These are key letters like "o" and "c" and "f" and are defined in the "relationships-keys.xml" file. For example, "a" represents the "AccessRelationship".

The "target" XML elements is added to the other "source" elements in the XML file so that relationship rules can be declared from these elements to the new element. This code is shown in figure 6-5.

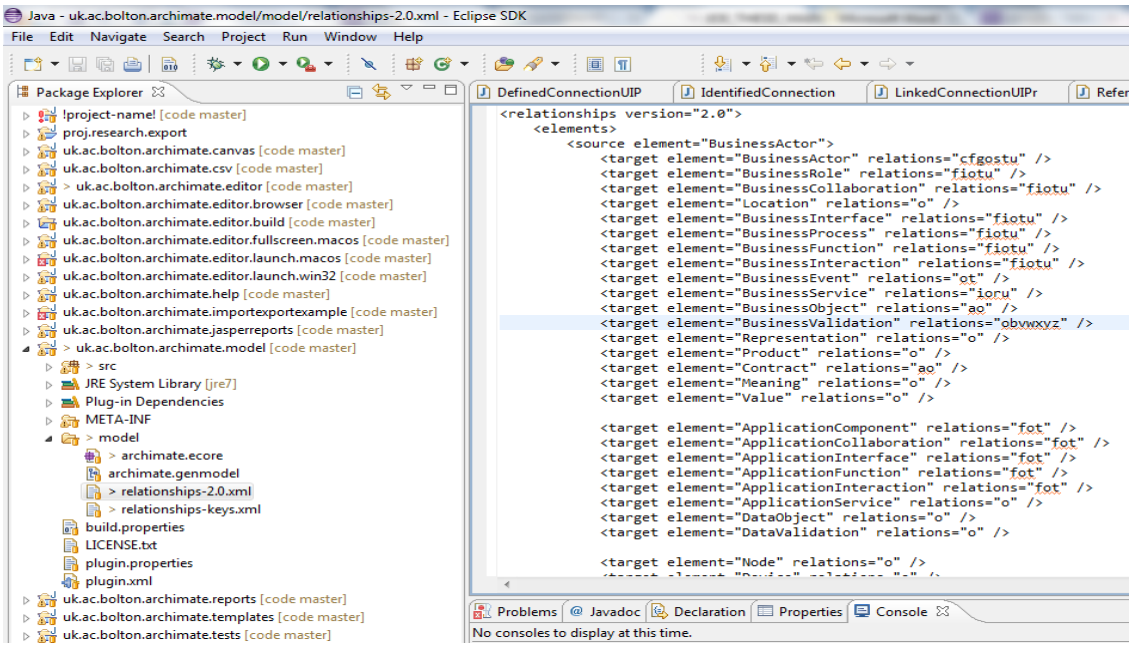

Figure 6-5: Declaration of Relationship for the Business Validation Element

6.6.3 Adding the element to the ArchiMateModelUtils.java file

Having created the model element and its relationship rules, it is added to the list of elements that will appear in the User Interface. The "ArchiMateModelUtils.java" found in the "uk.ac.bolton.archimate.model.util" package is opened and the getBusinessClasses() method by inserting the following line amongst the others:

IArchimatePackage.eINSTANCE.getBusinessValidation(),

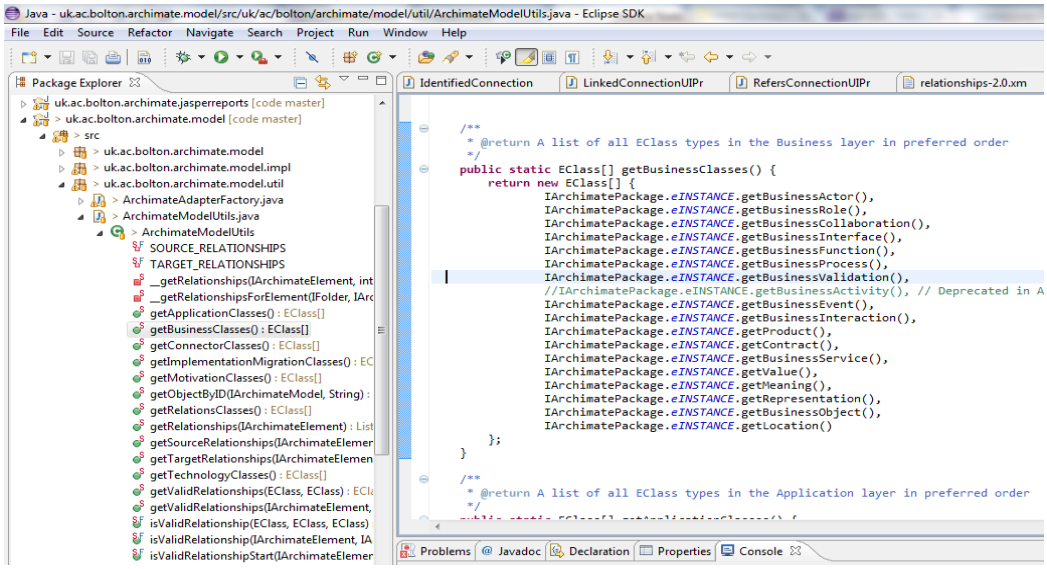

Figure 6-6: Adding Business Validation Element to the list of elements in the User Interface

6.6.4 Definition of the User Interface for the Business Validation Element

Having added BusinessValidation element to the model plug-in, the "uk.ac.bolton.archimate.editor" plug-in is edited to provide some code to create the UI for the new element. For this, a 16x16 icon for the element is created and saved as "business-validation-16.png" and copied to the "img/archimate" folder in the "uk.ac.bolton.archimate.editor" plug-in. The image reference file is declared to the icon in the "IArchimateImages.java" file in the "uk.ac.bolton.archimate.editor.ui" package:

```
String ICON_BUSINESS_ELEMENT_16 = ARCHIMATE_IMGPATH + "business-validation-16.png";
```

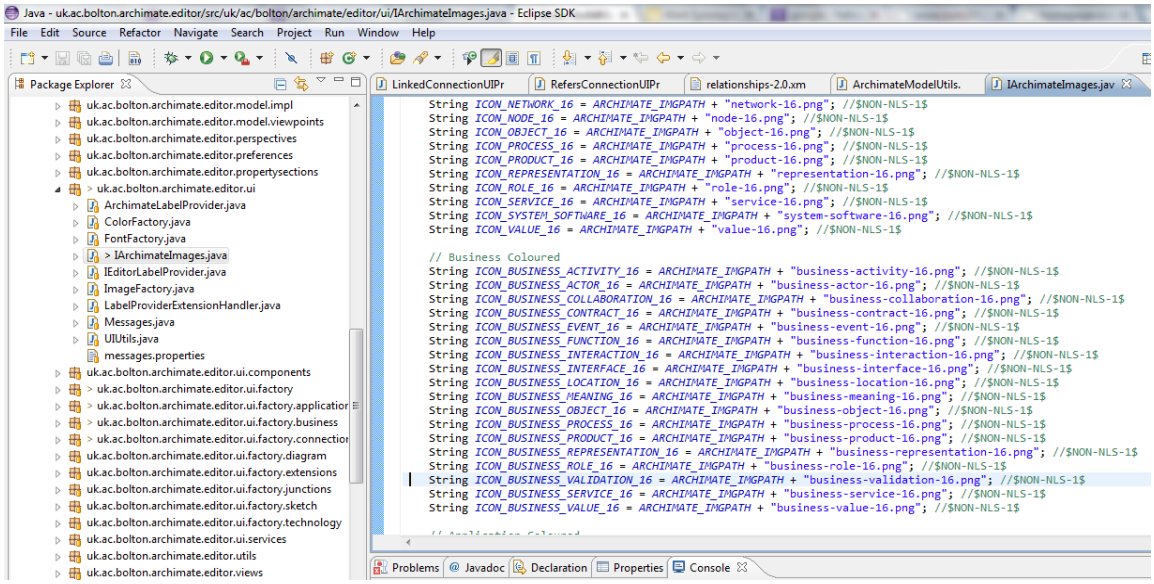

Figure 6-7: Definition of the UI Icon for the Business Validation Element

6.6.5 Adding a GEF Figure and GEF Edit Part

The Graphical Eclipse Framework (GEF) figure class for the Business Validation element is created in the "uk.ac.bolton.archimate.editor.diagram.figures.business" package. The constructor method in this class is edited so that the new icon file is used. This is shown in Figure 6-8.

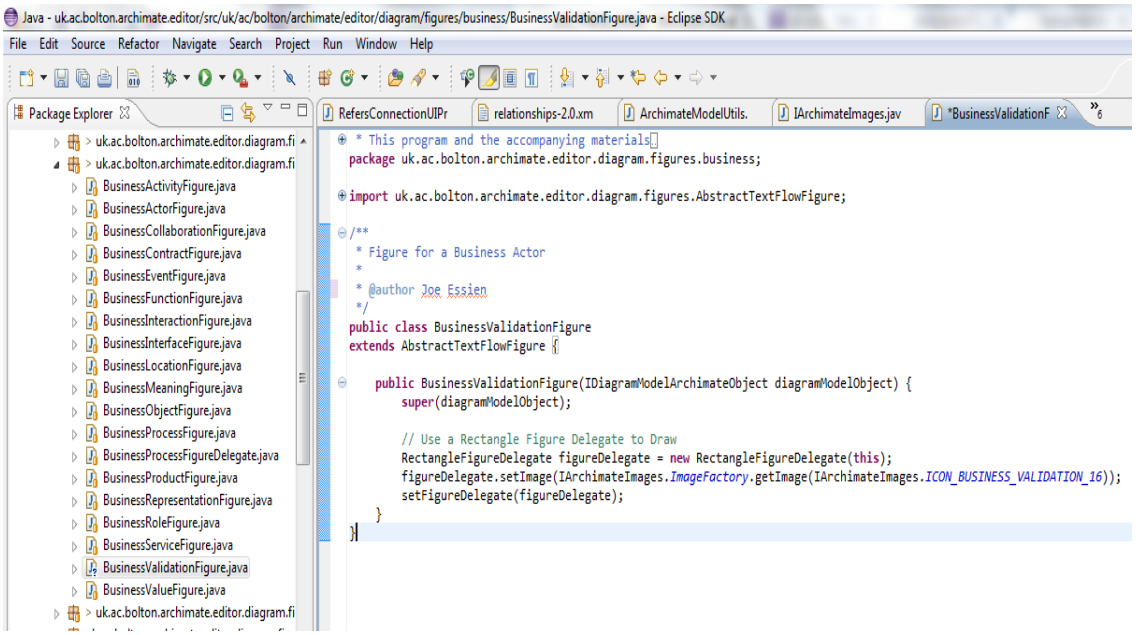

Figure 6-8: Addition of the GEF figure for the Business Validation Element

The GEF Edit Part is added by creating a GEF (Graphical Eclipse Framework) Edit Part class for the element in the "uk.ac.bolton.archimate.editor.diagram.editparts.business" package. The getFigure() method in this class is edited so that the new figure class for “BusinessValidation is returned. The code is as follows;

```
package uk.ac.bolton.archimate.editor.diagram.figures.business;
import uk.ac.bolton.archimate.editor.diagram.figures.AbstractTextFlowFigure;
import uk.ac.bolton.archimate.editor.diagram.figures.RectangleFigureDelegate;
import uk.ac.bolton.archimate.editor.ui.IArchimateImages;
import uk.ac.bolton.archimate.model.IDiagramModelArchimateObject;
/**
 * Figure for a Business Actor
 * @author Joe Essien
 */
public class BusinessValidationFigure
extends AbstractTextFlowFigure {

    public BusinessValidationFigure(IDiagramModelArchimateObject diagramModelObject) {
        super(diagramModelObject);
        // Use a Rectangle Figure Delegate to Draw
        RectangleFigureDelegate figureDelegate = new RectangleFigureDelegate(this);
figureDelegate.setImage(IArchimateImages.ImageFactory.getImage(IArchimateImages.ICON_BUSINESS_VALIDATION_
16));
        setFigureDelegate(figureDelegate);
    }
}
```

6.6.6 Addition of a UI Provider and Registration

The UI Provider class for the Business Validation element is created in the "uk.ac.bolton.archimate.editor.ui.factory.business" package. The class edited to incorporate the classes for the BusinessValidation. The code construct is as below and the UI implementation is shown in Figure 6-9.

```
package uk.ac.bolton.archimate.editor.diagram.editparts.business;

import org.eclipse.draw2d.IFigure;

import uk.ac.bolton.archimate.editor.diagram.editparts.AbstractArchimateEditableTextFlowEditPart;
import uk.ac.bolton.archimate.editor.diagram.figures.business.BusinessValidationFigure;
/**
 * Business Actor Edit Part
 *
 * @author Joe Essien
 */
public class BusinessValidationEditPart
extends AbstractArchimateEditableTextFlowEditPart {

    @Override
    protected IFigure createFigure() {
return new BusinessValidationFigure(getModel());
```

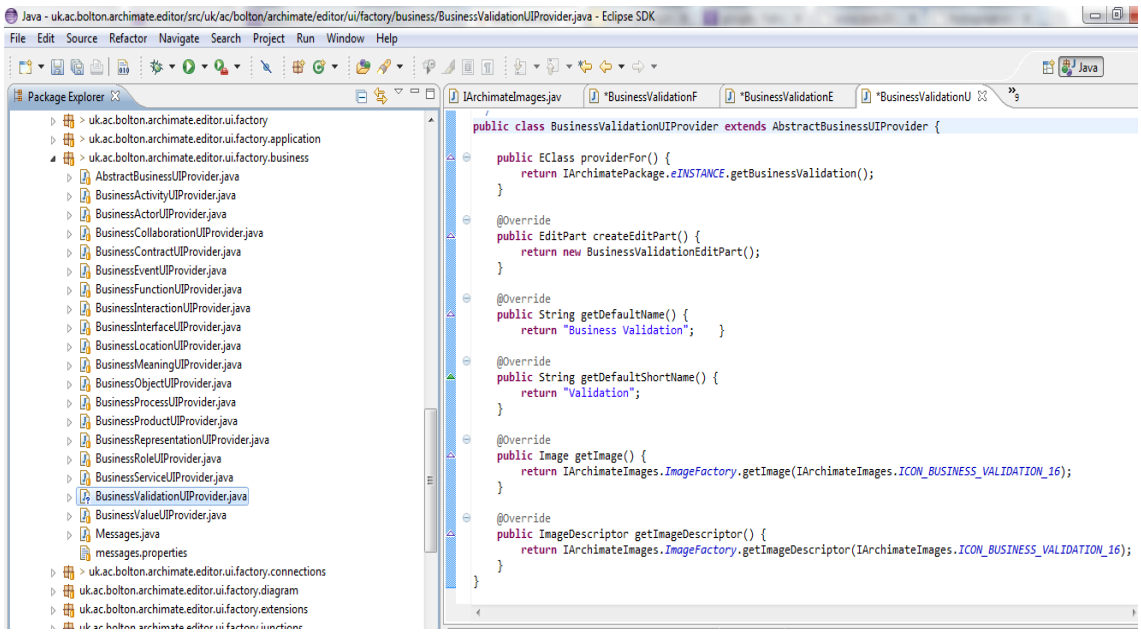

Figure 6-9: Addition of UI Provider for the Business Validation Element

Finally, the UI Provider is registered by editing the "ElementUIFactory.java" class and adding the following line to the ElementUIFactory constructor:

```
registerProvider(new BusinessValidationUIProvider());
```

6.6.7 Executing the Extended Archi from Eclipse

When the extended Archi is executed from Eclipse, the new Business Validation element appears in the context menus on the model tree and in the editor palette. The new element also appears in diagrams with specified relationship rules. This is demonstrated in figure 6-10.

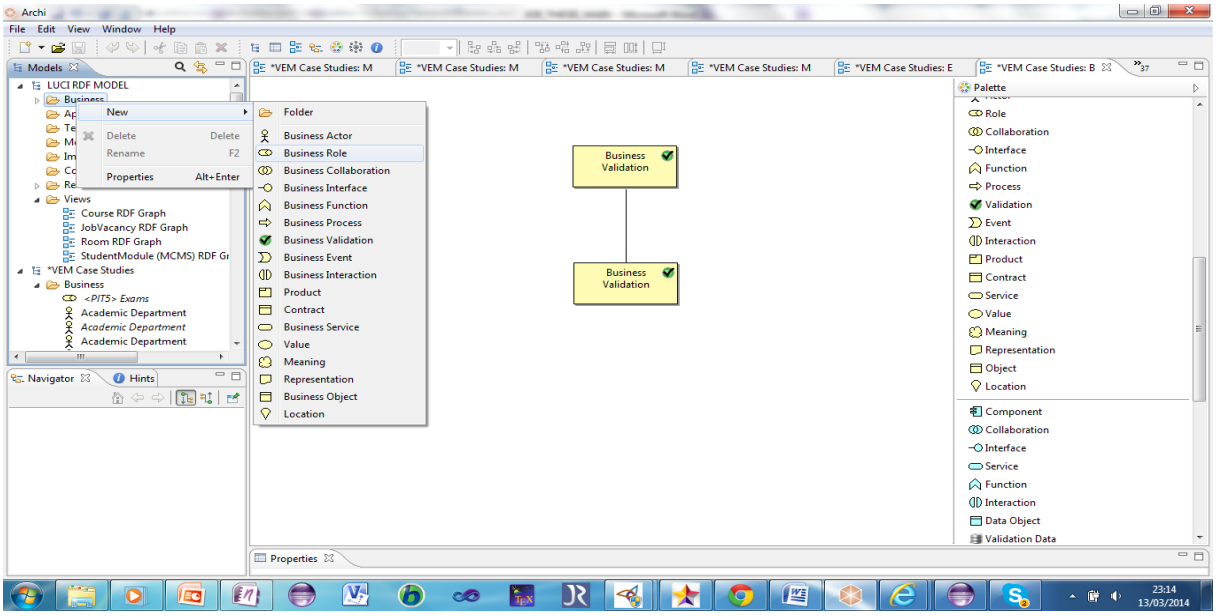

Figure 6-10: Execution of the Extended Archi with the Business Validation Element.

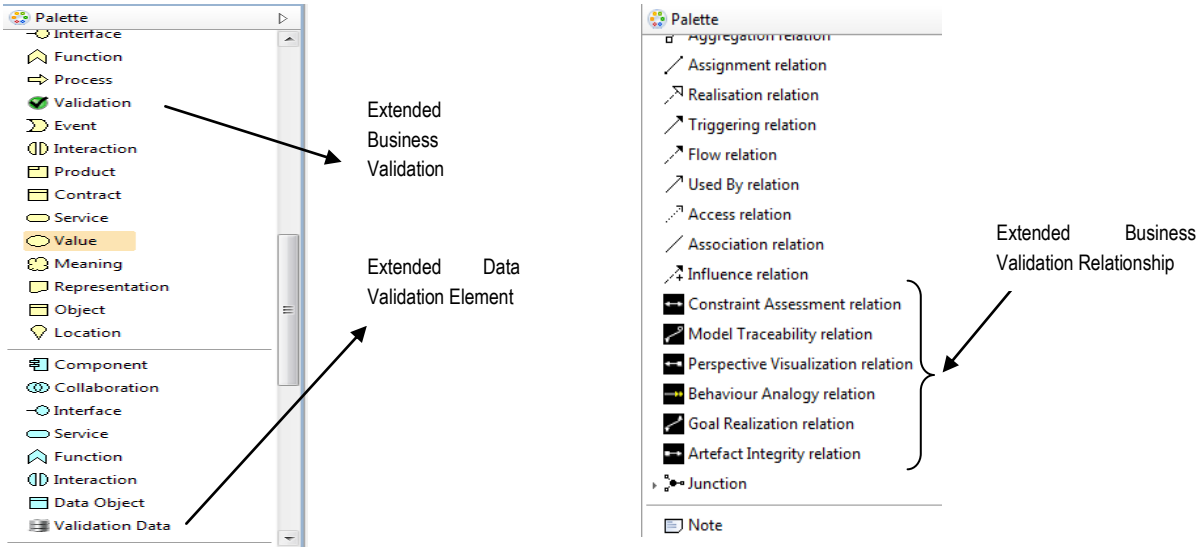

Figure 6-11: Extended tool sets of ArchiMate Enterprise Architecture Modelling Language

The graphical representations have been implemented in the ArchiMate Enterprise Architecture Modelling Language as shown in figure 6-11 on the palette of the graphical editor. The Business Validation and Validation Data artefacts are the analytic component extended from ArchiMate core for the synchronisation of validation and traceability within the metamodel instance.

6.7 Implementing of the Validation Extension Metamodel

In the context of Enterprise Architecture (EA) and in relation to this work, models posses the capability of being transformed using an appropriate modelling tool. A model transformed to its constituent components enable the efficient utilization and understanding of its taxonomy (Braun & Winter, 2005) and presents a description of the architectural artefacts, lists, tables and diagrams (Braun & Winter, 2005). According to the International Standard ISO/IEC/IEEE 42010, these architectural descriptions are the typical work artefacts used to illustrate the architecture. In a generic modelling environment, configurable toolsets exist that support the easy creation of domain-specific models, relationships, description semantics synthesis to ontology (Ledeczi et al., 2001). A retrospect of related works on enterprise architecture shows that within the three generic architectural layers, there is no representation of artefacts that address direct validation of metamodels and models against motivation. Thus differential components that define validation need to be specialized within these taxonomies. The postulations that address this phenomenon need to specify the plenary design of the metamodel, correlating its attributes with specific validation metrics at the strategy abstraction where motivation is factored. This would bridge the gap that exist within the construct of the business layer of EA and would also emphasize through validation the intrinsic values to be validated rather than the derivative docile elements which are merely informative and play no roles in benchmarking the metamodel against motivation. Though a rudimentary presentation of this concept is contained within the Information aspect of the ArchiMate Business Layer of TOGAF (TOG, 2013), an extended metamodel of business layer of EA with artefacts for validation would ensure strict cohesiveness between the core elements of the model and motivation thus ensure extensive and strategic substantiation of the business behaviour, perspectives and their relationships.

The design specification of the extended elements for the MDVA takes into consideration several proposals for solving validation issues and compares their attributes using empirical methods in order to ensure relevance. The outcome from this analysis is a composition of motivation artefacts that are formalized with Requirement and factored into the core ArchiMate through the Validation Element (Figure 6-1). The Requirement is validated against concepts adapted from BDD to relate the validation extension with the MDVA and the validation metrics defined in section 5.8. In this way, the methodology inherits annotation concepts from the extended Enterprise Architecture Modelling Language and can be validated by the proposed ontology method.

7 ONTOLOGY, METAMODEL AND MODEL TRANSFORMATION

Organizations of recent have continued to focus on how to implement effective knowledge management so as to improve their core competitiveness by adopting new opportunities presented by ontology and schematization (Hao et al., 2013). This work adopts a generic method of model construction and functional analysis proposed by TOG and applies the theoretical principles defined in chapter 5 to design the conceptual framework for the transformation of model to ontology. The rationale for this approach is that it supports knowledge management and enables the practical application of validation concepts on the extended EA models. Through this approach, the research also aims to demonstrate that the use of ontologies can facilitate better representation of knowledge management system. The application of knowledge acquisition techniques such as the VPEC-T (Green & Bate, 2007), which involves the extrapolation of business process artefacts and perspectives facilitates modelling and allows the knowledge collocated to be applied in mapping, creation of resource description graphs, filtering of ontology schemas for traceability and querying ontology to ascertain alignment with motivation. The main contribution of this work is therefore grounded on these advancements presented by ontologies schematization systematized with formalization of EA artefacts and validation. As this in a way asserts the altercation that ontologies can provide a means for formalization of domain knowledge (Bakhshadeh et al., 2014), the same knowledge is thus applied for the validation of EA models in order to support and guide the construction of meaningful representations of model triple stores necessary for querying.

7.1 Ontology and Harmonisation of Principles

Though it has been argued that Enterprise Architecture ontologies can contribute to the semantic interoperability between different enterprises architectures, however, the development of enterprise architecture ontology has raised issues concerning how to define enterprise ontology and also the architecture of the ontology itself (Chen et al., 2008). Some research works have been done to define enterprise ontology such as for example TOVE (Fox, 1992) and others (Fernandez-Lopez, 1999; Gruninger et al., 2000). The focus of those works which aimed at a higher level of abstraction delved into the representation of business user's concerns with no direct link to model validation. The main drawbacks of these historical developments in the past can be summarized as follows:

- Insufficient understanding of enterprise architecture concepts and lack of enterprise architecture ontology.
- Absence of a scientific method to justify an enterprise architecture proposal and difficulty to evaluate and compare different architectures.
- Inadequate means to represent and describe enterprise architecture including interoperability between various existing architectures.

- Weak impact of enterprise architecture research in industry and insufficient maturity of standards on enterprise architectures.
- Lack of clear consensus on the interoperability of concepts or on global interoperability framework defining the interoperability domain.
- Insufficient development of reference architectures for supporting preliminary validation of enterprise systems.

Despite these limitations, ontologies have continued to be used to capture knowledge about domains of interest, describe the concepts in the domain and the relationships that hold between those concepts (Rector et al., 2004). This is sequel to the conviction by many practitioners that ontologies provide an explicit specification of conceptualisation including descriptions of the assumptions regarding both the domain structure and the terms used to describe the domain (McShane & Nirenburg, 2013). Further, it is argued that ontologies are central to semantic as they allow harmonization of terms and relationship. In contract to these assertions, within deployments of ontology languages such as the Web Ontology Language (OWL), there is no agreement yet on the nature and the right formalism for defining mappings between ontologies (Ceh et al., 2011). This is despite the fact that OWL for instance is considered a de facto standard for representing and using ontologies. This is not surprising as in a recent discussion on the nature of ontology mappings, though some general aspects of mapping strategies were identified (Kumar et al., 2013), these strategies when used to map congruent information, relational schemas and metamodels to ontology have led to problem of anomalies in their interpretation. The consequence is that often, it adds greater complexity to the semantic interoperability.

However, there are also good reasons for developing ontologies. These include the capability;

- To share common understanding of the structure of information among stakeholder.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyze domain knowledge.

The most critical expediency required for ontology is effective correlative mappings. To enable consistency in mappings, this work adopts the direct mapping principle of model to ontology. This approach accentuates an uncomplicated definition of transformation and provides a basis for distinctive comparison and validation on the ensued Resource Description Framework (RDF). The direct mapping takes as input a relational database derived from metamodel decomposition to generate direct algorithms and graphs. The algorithms are then resolved against base International Resource Identifiers (IRI) to establish references or associations between different business behaviour contained

in the metamodel. This allows values relating to motivation to be queried for the metamodel and its instances. Central to the approach is the extraction of business behaviour defined by the metamodel instance and transformation using a ubiquitous language for the domain driven design. The adopted ubiquitous language is a formalized language which is understood by all members of the enterprise, technical and non-technical and provides a common means of discussing the domain of the metamodel. The similarity this exists between codified specifications of desired behaviour has been compared to the concepts of the Behaviour Driven Development (BDD) (Haring & Ronald, 2011; Chelimsky et al., 2010). But unlike BDD, the proposal proffered in this work differs as it incorporates RDF triples instead of the Gherkin syntax to allow for effective analysis of the metamodel and application of the query patterns thus ascertaining the validation of the intrinsic specifications of the metamodel.

7.2 Principles for Mapping Model to Ontology

The theoretical principle for goal evaluation is discussed in section 5.3. Grounding this principle provides the basis for logical and conceptual exposition of principles for mapping model to ontology. The Theory of Change articulated in the subsection 5.3.2 is adopted. This approach is widely acclaimed to yield dependable result as it is based on a series of critical thinking exercises aimed at providing a comprehensive picture of the early and intermediate variations in a given scenario. Applying this principle, the process of ontology mapping in this approach is delineated and specified as follows;

i. Given a model, identify the various testable artefacts to ascertain the values of the proposition. VPEC-T can be used here as a thinking framework.
ii. Identify the business behaviour associated with the model and associate goals and constraints.
iii. Identify the various motivational, structural, information and functional models.
iv. Decompose the model to class representation annotating its associated attributes and relationships.
v. Identify testable artefacts for the nodes, domains and ranges.
vi. Identify relationship that exists between the nodes and slots. This is required for the development of traceability.
vii. Identify attributes of the node that relate to constraints. This is required for the development of the features and scenarios of the ubiquitous language.

Thus, the result of a mapping process is a set of mapping rules which connect concepts in the transformation to concepts in metamodel or model instance. Approaches from different communities have been proposed in literatures to deal with these sorts of transformation (Sintek & Decker, 2002; Bakhshadeh et al., 2014). Many autonomous information repositories have also described approaches

for a variety of database schema transformation methods. Analysis by Rahm (2011) indicates that these can broadly be classified into schema comparison, schema conformance and merging. Rahm claims that a fundamental operation in the manipulation of schema information is match, which takes two schemas as input, for instance a metamodel and its instance and produces a mapping between elements of the two schemas that correspond semantically to each other. This can also be referred to as metamodel-level match against its instance model-level match taking into consideration properties of the schema elements, such as name, description, artefact types, relationship types, constraints and schema structure. As a complementary method this approach can provide critical insight into the contents and semantics of the metamodel artefacts but in general, it does not offer a means for validation of the underlying motivation of the metamodel. Though no specific literature has been identified that map ontology to EA model's motivation, the originality of this work is to explore that capability while drawing inferences from related areas; and based on theoretical principles and inferences, envisage a solution that fits with the hypothesis of the Model Driven Validation Approach (MDVA).

7.3 Ontology Transformation Metaphor

Several principles have been proffered to demonstrate that metamodels are closely related to ontologies (Gudas and Lopata, 2007). This is perceptible as both are used to describe and analyze the relations between concepts. To augment this assertion, many practitioners have also suggested that annotation of a model with constraints can allow coherent transformation of the model aspects to ontology with formalized specification (Bakhshadeh et al., 2014). The rationale for the juxtaposed constraint is to ensure that the entity adhere to strict rule set. A number of ontology transformation, integration methods and tools exist. Amongst them, are SEMAPHORE (Smartlogic, 2013), PROMPT (Noy, 2004; Choi et al., 2006), Protégé OWL (Horridge, 2009; McGuinness & Van Harmelen, 2004) are few which have working prototypes. These tools support the generation and mergence of ontological elements such as class and attribute names from various sources. While Semaphore automatically applies metadata and classification to improve context traceability, PROMPT provides more automation in merging ontologies. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C). Protégé OWL enables the description of complex concepts with rich set of operators and allows queries to be applied onto its ontology. One major advantage of the OWL which is relevant to this work is its capability to infer logical consequences from a set of asserted facts or axioms by use of Reasoners. The Reasoner has the proficiency to check whether or not all of the statements and definitions in the ontology are mutually consistent and can help to maintain structural hierarchy correctly thus making it useful when dealing with cases with multiple class dependency as in the case of metamodels. For these reasons, OWL is preferred for the generation of ontology for MDVA. However, it is worth noting that while a valid model can always be transformed into ontology, not all ontologies can be explicitly transposed as models. Thus applying validation constraints to an "unstructured" ontology not predicated from an EA model may be non-indicative that a model is necessarily being validated.

7.4 Content Categorization for Model to Ontology Mapping

Content categorization is a link-based classification approach used in isolation or in conjunction with text-based classification to assign artefacts to one or more predefined categories based on their contents (Gyongyi et al., 2006). A number of modelling classification and knowledge management techniques have been applied to content categorization such as nearest neighbour, Support Vector Machine, Voted Classification and Neural Networks (McShane & Nirenburg, 2013). More recently, some preliminary studies have attempted to apply content categorization techniques into merging and mapping of ontologies (Lacher & Groh, 2001). Lacher presented an approach using supervised classification (Rocchio) for ontology mapping while another method referred to as FCA-MERGE, based on the theory of formal concept was proposed by Stumme & Maedche (2001). Though these approaches are veteran, their analysis which stipulates that generation and mergence of ontologies should follow a bottom up approach guided by application-specific instances is still widely practiced of recent. In our approach, this theory is enhanced. While the general implementation of the mapping process identifies class artefacts from top-down perspective, the mapping of the properties follow a bottom-up perspective. The metamodel to ontology elements mapping are determined by similarity in characteristics per pair. Only the combinations with similarity attributes are considered as equivalent. In order to establish definitions of similarity and to support development of accurate mapping, a framework for the mapping is defined. The framework presented in figure 7-1 is adopted as the basis for correspondence assertion for OWL.

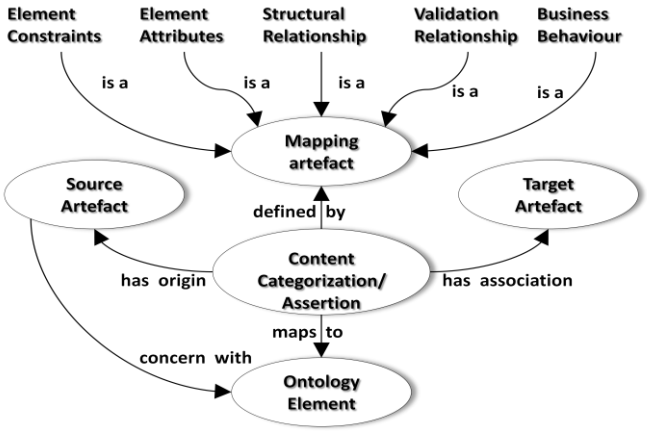

Figure 7-1: Conceptual framework for OWL Mapping

The diagram in Figure 7-1 describes the content categorization as an objectification of the relationship between ontology elements with support for further description of that source and dependent associations. A content categorization is uniquely assigned to the ontology elements. It has also an association in order to provide a way of establishing dependencies and traceability of the artefact within the schema. Definition is also attached to the content categorization in order to establish content and specify how the mappings of the ontology elements are related. The objective of maintaining artefact's derivation is to provide an explanation for the source of its origin.

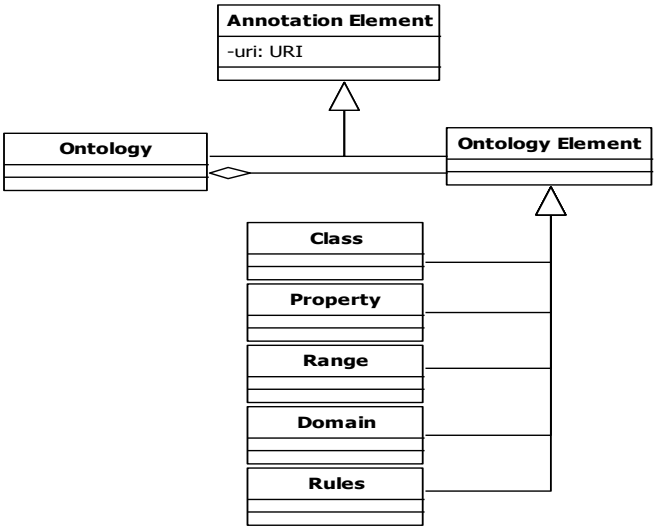

Figure 7-2: UML Profile for OWL Ontology and Rules Extension

For a mapping process, ontology element is the core output of the procedure. The process of mapping ontologies is supported by analysing the extension of concepts defined with the "is a" constructs to derive corresponding intentional descriptions in the ontology. The transformed metamodel for mapping onto ontology is presented in Figure 7-2. It depicts artefacts mapped to ontology class or subclass of the ontology element. The UML profile also delineates Rule for establishing correspondence with constraints inherited from the model. Additionally it specifies representation of the motivation abstraction for business behaviour that must be satisfied by the source model. This is mapped onto Range and Domain subclasses. The metamodel transformation can adhere to one-to-one or one-to-many mappings, all which are transitive for the ontology for maintenance of strict traceability.

7.5 Mapping Formalization Definition

In this implementation, the mapping formalization is restricted to declarative mapping specifications as this is sufficient to define a semantic relation between elements in or across divergent ontologies. Though a number of different semantic relations are available (Choi et al., 2006), used in this proposal is the logical semiotics defined by Kumar and Harding (2013) specified below.

- Equivalence ($\mathcal{I}\equiv$): Equivalence indicates that the connected elements in the ontology represent the same aspect of the metamodel and inherits the same equivalence constraints.

- Composition (Ⓒ): Composition states that the element in the ontology represents more but specific artefacts in the metamodel. Depending on which of the artefact is more specific, the composition relation is defined in the one that aggregates the others.

Assumptions

In consideration of the formalization definitions, it is assumed that the mapping of metamodel does not concern itself with overlaps, gaps and the basic relations that supplement their negative counterpart as these are implicit reasoning. Rather, the mapping seeks to transform the taxonomy objectively with its intrinsic format structure, describing exactly the same set of objects. It is assumed that overlaps can exist within the metamodel especially as it may relate to expression of variant business behaviours. As a variation, a metamodel artefact can also be completely disjoint with other artefacts. In that case the artefact is also presented as disjoint in the ontology. Thus the mapping to ontology strives to maintain the same intentional interpretations as expressed in the metamodel.

Another assumption made by this approach concerns the use of unique names for objects in order to preserve consistencies across the metamodel and the mapped ontologies. Mapping may be organized independently from a metamodel or model to ontology. In the case of many instantiations from the same metamodel, it is also possible to map from such instantiations onto the same ontology as long as they are derived from the same metamodel. All mappings in this transformation are normative and unidirectional. A mapping is consummated when there is a set of identical assertions that consist of a semantic relation between map-able elements from model to derivative ontology. The mapping is irrelevant without the originating metamodel or model as its purpose in this work is to embed queries that can allow the metamodel to be validated.

7.6 UML Mapping Profiles

The UML profile is a visual notation that specifies how the mapping to ontology will be characterised. The objective of the profile is to enable the specification of mappings in a generic sense and independent of any specific mapping language or any specific semantic relation. The UML profile is consistent with the design considerations made for the defined UML profiles for OWL ontologies and rule extensions in Figure 7-2. The Artefacts and Relationship are primary elements moderated from Behaviour Driven validation approach to constitute the map able components of the metamodel. The Equivalence and Composite components are secondary elements used to establish the semantic association for the mapping assertions. By this specification, elements or abstractions are identified from the metamodel through URI descriptors to the ontology as depicted in Figure 7-3. The ontology is aggregated with Properties and Classes. Adopting this autonomous formalisation approach for ontology mappings, the metamodel is consistently transformed to ontology that can be queried.

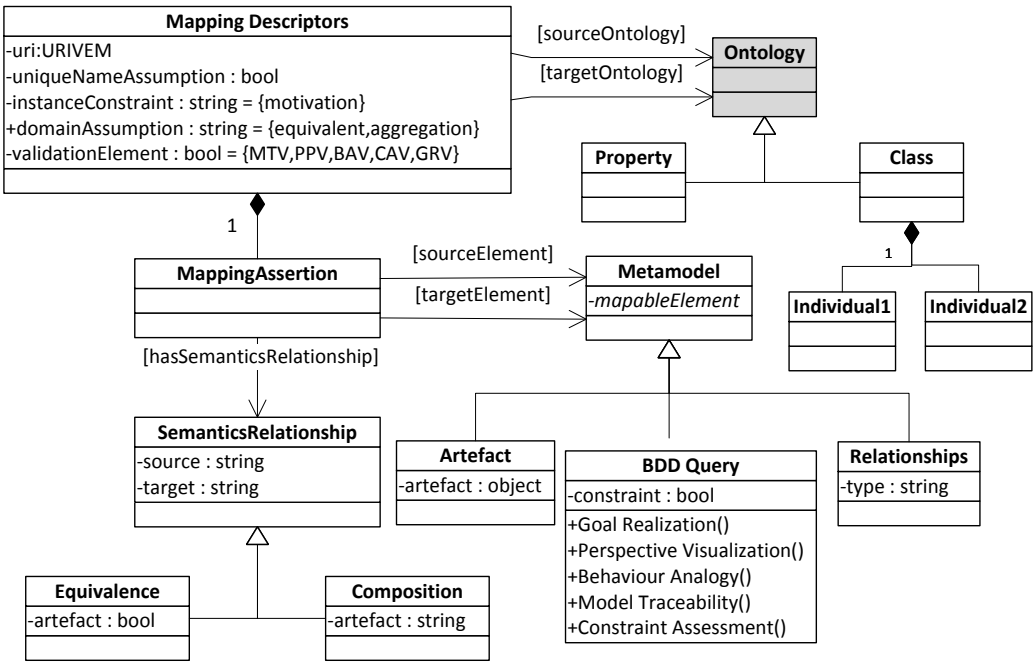

Figure 7-3: UML Profile of Metamodel to Ontology Mappings

Pivotal to the mapping transformation is the class Mapping Descriptor which consists of five attributes namely URIVEM, Name Assumption, Constraint, Domain Assumption and Validation Element. The URI is defined by the attribute uri:URIVEM to allow the unique identification of mapping and reference to it as the primary class object in the ontology. The assumptions about the use of unique names for objects and the preservation of inconsistencies across mapped ontologies are defined through the Boolean attributes uniqueNameAssumption. For the constraints applicable to the instantiation of the metamodel, an attribute instanceConstraint is defined. This attribute inherits specifications from the motivation extension introduced in the methodology. For the assumptions about the domain, an attribute domainAssumption is defined. This attribute may take specific values that describe the relationship between the connected artefacts: equivalence and composition. Mappings are unidirectional from metamodel to ontologies. For validation metrics associated with an instantiation, an attribute validationElement is defined. This attribute allows explicit query of the ontology with specified motivational constraint. Ontology is represented by the class Ontology in the OWL DL metamodel.

Two associations from Mapping Descriptors to Ontology *sourceOntology* and *targetOntology* specify the source and the target ontology of the mapping respectively. Cardinalities on both associations denote that for each mapping instantiation, there is an ontology connected as source and another as target. A mapping consists of a set of mapping assertions, denoted by the aggregation relationship between the two classes *Mapping Descriptors* and *MappingAssertion*. Patterns of mapping notations used in the transformation are as follows;

Direct Mapping

A direct example of visual notation for one-to-one equivalence mapping is as defined in the profile presented in Figure 7-4 for class. A direct mapping also is achieved for property from the metamodel to ontology using an Equivalence relationship as in Figure 7-5.

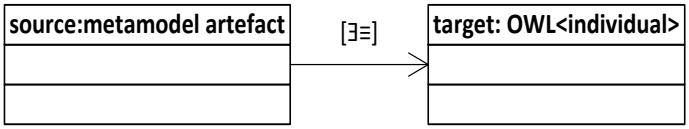

Figure 7-4: Direct Equivalence mapping between metamodel artefact and ontology element.

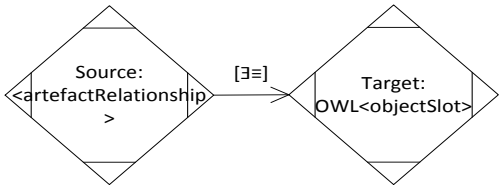

Figure 7-5: Equivalence mapping between Relationship and Ontology Slot

Complex Class Mapping Descriptions

Complex aggregation of related multiple properties from source metamodel to target ontology element is also possible. This can be achieved via a Composition Relationship with the corresponding symbol of the semantic relation. In the first step of the process, related properties are identified without semantic annotations as the dependency does not carry any relation symbol. Stereotypes in the two boxes denote source and target ontology. A grouping construct of the metamodel abstraction or its instantiation is applied to represent mappings as collections of assertions as in Figure 7-6. A complex mapping for properties can also be any relationship from the taxonomy or its instantiation shown in Figure 7-7.

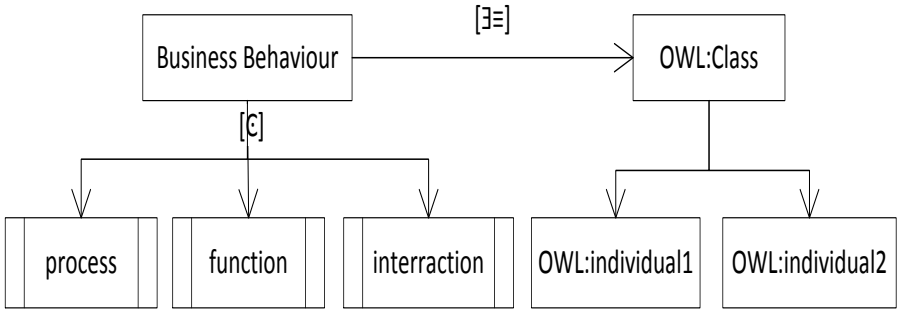

Figure 7-6: Equivalence mapping between business behaviour and complex class descriptions

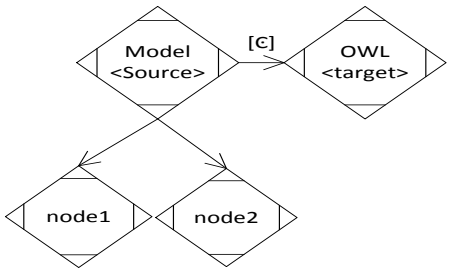

Figure 7-7: Composition mapping between complex Relationship and Ontology Slot

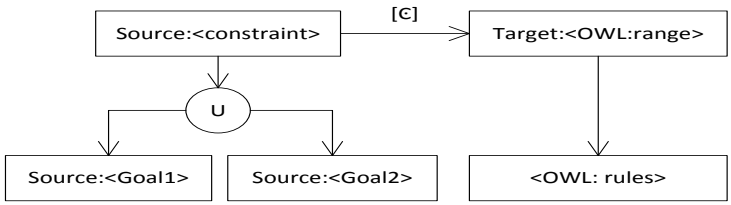

Figure 7-8: Composition mapping between complex motivation and ontology class

Duplex combinations of complex mapping assertion can be achieved as unions as illustrated in the profile in Figure 7-8. The example defines the union of two motivation aspects which aggregates the source constraints in the metamodel. The expression is mapped to Ontology class with a Composite Relationship with specification of range and rules.

7.7 Developing the Reference Description Framework Schema and Query

The transformation of EAF metamodel or a model instance to a Reference Description Framework Schema (RDFS) allows the ontologies to be validated using a ***P****rotocol* ***a****nd* ***R****DF* ***Q****uery* ***L****anguage* (SPARQL). OWL Protégé is used to build the context of coalition operations and to design the collaborative ontology domain. It also enables the capture of artefacts represented in the Business model. Given the objective of collaborative creation and querying of the ontology, it has been advocated that the primary requirements for ontology is to support the methodological process of building an RDFS that can be queried (Huang et al., 2011). In particular, this research provides these functionalities and relates to collaborative business behaviour of the EAF. In order to facilitate the query of ontology, it advocates building of triples and use of enhanced visualization. The result thus is a well documented guideline for the approach of ontology design based validation and description of the different mapping of the EAF artifacts.

While there are no formalised ways of mapping generally acknowledged as standard by practitioners, to ensure that there is consistency in the methodology and to avoid overlaps of artefact mapping, a top-down class bottom-up slots approach is proposed. This approach starts by identifying the uppermost artefacts as the taxonomy and associates all immediate subcomponent as branches ensuring that the maximum association is achieved for each branch as it scales down to the lowest artefact on

the leaf without any overlaps. This approach decomposes the metamodel into RDFS Structural hierarchy.

7.7.1 Model Mapping and Creation of Classes

Though several approaches have been identified for developing ontologies, for instance the single ontology approach, multiple ontologies approach and the hybrid approach (Boury-Brisset.2003), most ontology are relative and attempt to define a common vocabulary for defining concepts across different individuals. In the approach presented in this research a global ontology is defined to provide a shared vocabulary for the specification of the semantics with all information sources related to the global domain ontology. To enable the interoperability and connections of the ontology, query and analysis of metrics for validation, a common approach referred to as Structure Enrichment (Wache et al. 2001) is used with an aberration. The Structure Enrichment consists of logical model blocks that resemble the structure of the original information structure. The aberration proposed consists of additional definition of concepts with meta-annotations that add semantic information to the information source allowing validation and query of an Enterprise Architecture Framework to be carried out. Using this approach, a logical block is developed as in Figure 7-9 with a structural hierarachy for transformation to ontology. A naming syntax is maintained to enforce clarity of the mapping process. Domain applies to specify the Source while the Range specifies the Target. The definition of the syntax for the formalization of the ontology slot is thus;

<target_object UID><metamodel_property>

Where *target-object* refers to the ontology element; *UID* a form of Universal Identifier for the model's artefact and *metamoel_property* the associations of artefact defined by the UID. This ensures the use of appropriate relationship as required throughout the ontology and circumvents a serious limitation of the OWL (OWL does not allow annotations with same identifier throughout the hierachy even though this is very comon in EA models). For instance, OWL will not allow a defininition such as "accessed by" more than one instance within one ontology, thus the "accessed by" property of the "Business Object" to Business Service, Business Event and Business Behaviour would have been impossible to map without this formalization.

In the outline of the Validation Extension Metamodel (VEM) presented in Figure 7-9, two sibblings to the OWL Root are identified. The first is Composite_Motivation to represent a theme of motivation in a business behaviour to be queried. The second is BDD Validation_Elements which encapsulates the core EAF artefacts to be validated. To enable explicit traceability, the properties are defined as functional hierarchies.

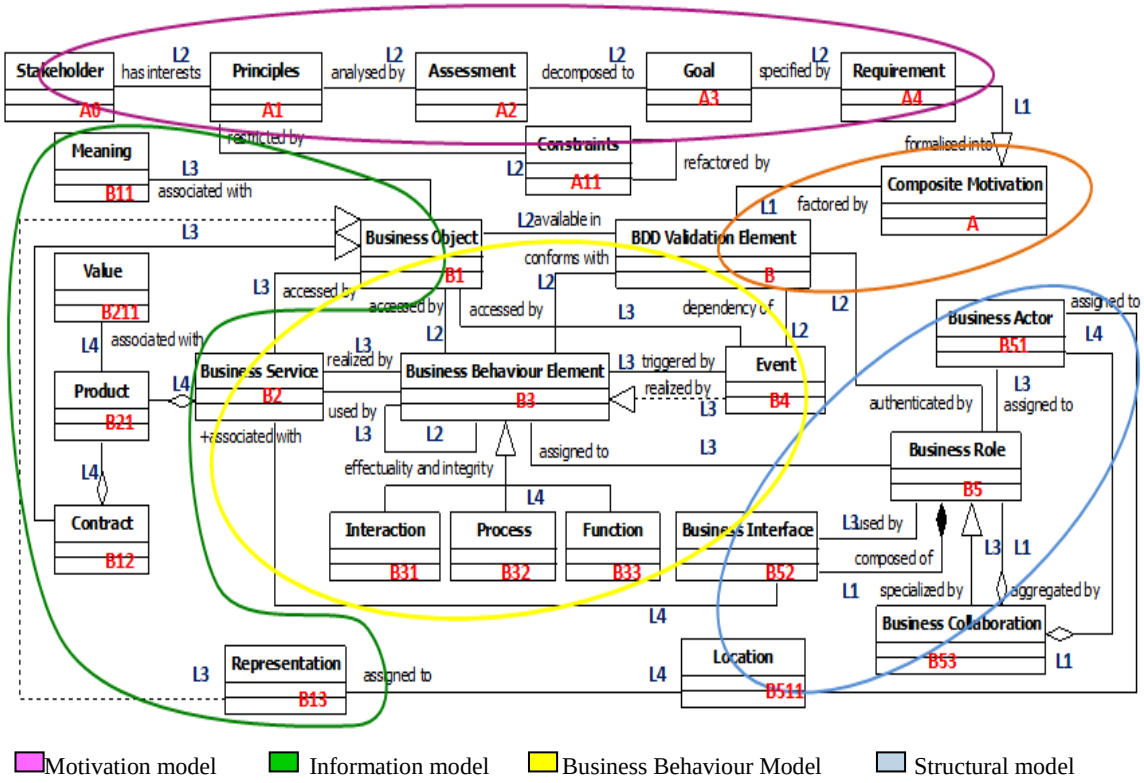

Figure 7-9: Outline classes, property hierarchies for ranges and domains

This ensures that within the ontology, there are appropriate bindings to allow constraints to be applied accordingly on associated artefacts. Thus this Hierarchies of Properties (HoP), Ranges and Domains of applicability are defined as in Figure 7-9. For this metamodel, there are five distinct domains and four HoPs identified. The five domains are distinct and specify motivation, information, validation, business behaviour and structural aspects of the architecture. These five aspects of architecture must be modelled in order to achieve completeness for validation of the EAF. This also ensures full business layer perspective oriented analysis of business behaviours for the composite motivation. Properties link individuals from the domain to individuals from the range and are construed as N-triples in the ontology:

triple ::= subj pred obj '.'
subj ::= IRI_REF BLANK_NODE_LABEL
pred ::= IRI_REF
obj ::= IRI_REF|BLANK_NODE_LABEL|lit

The HoP is annotated as L1, L2, L3 and L4 and the characteristics inheritance of the properties grow sequentially up the scale. Because the relationship between properties is chronological, example L4»L3»L2»L1, this allows transitivity of the properties as it collaborates with the other classes. This is shown in Figure 7-9 and used in the transformation to ontology depicted in Figures 7-11 and 7-12.

Table 4: Mapping of VEM Properties to Ontology Hierarchy

TOP- DOWN Decomposition Analysis			RDF TRIPLES			Hierarchies of Properties					Inverse Properties	Mapping Profile
METAMODEL CLASS			Subject UID	Predicate Description	Object UID	L1	L2	L3	L4	VE	Description	
Composite Motivation			A	factored by	B	L1				GRV		ℭ
Motivation Extension	Stakeholder		A0	has interest	A1		L2			PVV		ℑ≡
	Principles		A1	analyse by	A2		L2			CAV		ℑ≡
	Constraint		A11	restricted by	A1		L2			CAV		ℑ≡
	Assessment		A2	decomposed to	A3		L2			CAV		ℑ≡
	Goal		A3	specialisation of	A4		L2			GRV		ℑ≡
	Requirement		A4	formalised into	A	L1				GRV		ℑ≡
BDD Validation Element			B	available in	B1		L2			MTV		ℭ
BDD Validation			B	dependency of	B4		L2			MTV		ℑ≡
			B	conforms with	B3		L2			BAV		ℑ≡
			B	effectuality	B3		L2			GRV		ℑ≡
			B	integrity	B3		L2			CAV		ℑ≡
			B	authenticated by	B5		L2			PVV		ℑ≡
Information Model	Business Object		B1	accessed by	B2			L3		MTV		ℭ
			B1	accessed by	B3			L3		MTV		ℑ≡
			B1	accessed by	B4			L3		MTV		ℑ≡
			B1	associated with	B11			L3		MTV		ℑ≡
			B1	specialisation of	B12			L3		MTV		ℑ≡
			B1	realized by	B13			L3		MTV		ℑ≡
			B12	aggregated by	B21				L4	MTV		ℑ≡
Business Function / Process Model	Business Service		B2	aggregated by	B21				L4	BAV		ℭ
			B2	used by	B3			L3		BAV	realized by	ℑ≡
			B2	used by	B52				L4	BAV		ℑ≡
	Business Behaviour		B3	realized by	B2			L3		BAV		ℭ
			B3	specialisation of	B31					BAV		ℑ≡
			B3	specialisation of	B32					BAV		ℑ≡
			B3	specialisation of	B33				L4	BAV		ℑ≡
			B3	realized by	B4			L3		BAV		ℑ≡
			B3	assigned to	B5			L3		BAV		ℑ≡
	Event		B4	triggered by	B3			L3		BAV		ℭ
Organisation Model	Business Role		B5	assigned to	B51			L3		PVV		ℭ
			B5	composition of	B52			L3		PVV	used by	ℭ
			B5	specialisation of	B53			L3		PVV	aggregated by	ℭ
	Interface		B52	used by	B5	L1				PVV	composition of	ℭ
	Collaboration		B53	aggregated by	B5	L1				PVV		ℑ≡
			B53	aggregated by	B51	L1				PVV	specialisation of	ℭ
	Actor		B51	assigned to	B511				L4	PVV		ℑ≡

The triples are used to develop the axioms for reasoning for individuals that are members of the class. These specifications as mapped to OWL are presented in Table 4. Each of the sibling class is disjoint and may have a subclass. Each subclass represents an artefact in the metamodel. At this stage, no properties are assigned. The linkages represent only the cascading of groups with no defined annotation. The implementation in OWL shows some corresponding inverse properties defined in Table 4. However the characteristics of these properties are not strictly symmetrical and are not designated as such. The enrichment of the meaning of the property is implemented by use of range and domains maintained at the appropriate hierarchies of property.

7.7.2 Developing the Validation Extension into Ontology Framework

Benefits of ontology-based approach for validation of EAF have continued to be recognised especially in encoding collaborative understanding of architectural and structural domains from varied perspectives (Horrocks et al., 2014). The specification of ontology consists of a vocabulary of terminologies, each with a definition that specifies a distinctive meaning. Ontologies range from controlled vocabularies to highly expressive domain models (McGuiness, 2002). It integrates data dictionaries designed for human understanding to structured data models suitable for data management, and computational taxonomies. A fundamental distinction between different approaches to ontologies is the manner in which relationships among terms are specified and formalization for automated reasoning. Though ontology classification can be specified in different forms ranging from meta-ontologies, upper-level ontologies to domain-specific knowledge (Boury-Brisset.2003), this research adopts an approach that enables the building of knowledge data store that can be reused in the query of the Enterprise Architecture frameworks while facilitating model alignment with business goals irrespective of the heterogeneity of the modelling language. To provide a flexible extensible medium that can allow collaborative interaction and specification of model artefacts both on functional definition and ontology encapsulation (concepts, attributes, relations) a web-based OWL Protégé is used.

One of the advantages of adopting the design science research in this work is that the simulation of highly descriptive models derived from extensive analysis of motivation, business requirements and process analysis with the VPEC-T (Green et.al, 2007) reasoning and thinking framework has been made possible. The VPEC-T decomposes business conceptions to its elemental components and artefacts based on values, policies, events, contents and trust. This is then formalised into simple triples required for transformation to RDFS. This approach allows the business concept to be mapped to model artefacts and transcribed into classes and slots in the conventional ontology language. The development of the validation extension and transformation to ontology allows the ensued RDFS to be queried using Protocol and RDF Query Language (SPARQL) adopting an adapted Behaviour Driven Development (BDD) concept. The two case studies used in this research are subjected to VPEC-T analysis to translate information from business needs to IT solutions in the form of viewpoints of EA abstractions. The OWL Protégé is used for this transformation as the ontology language and

constitutes the grounding for the SPARQL query. These subsequent screenshots depicts this transformation.

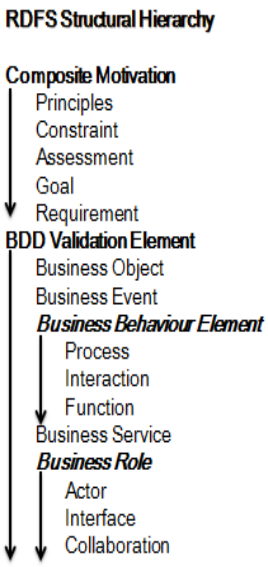

Figure 7-10: Conceptual Class Hierarchy

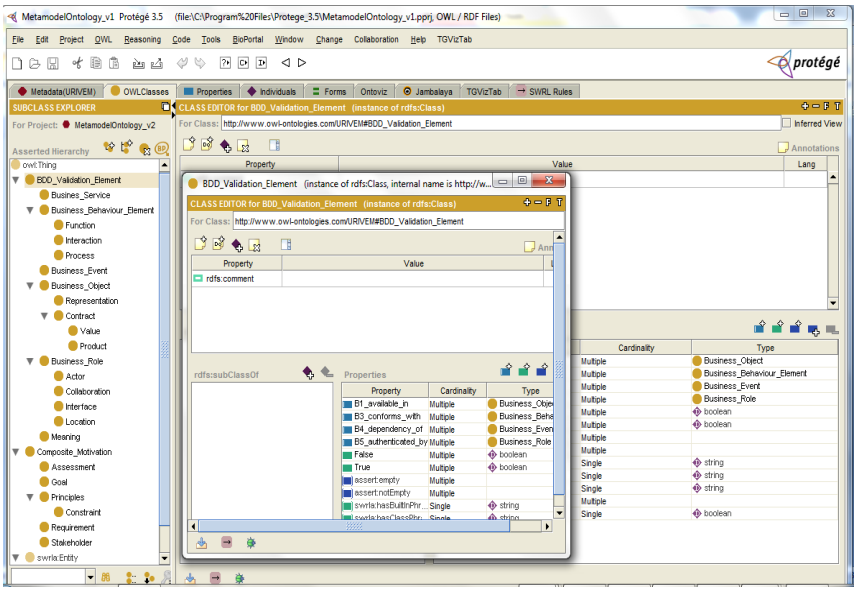

Figure 7-11: OWL Implementation of the Class Hierarchy

In this implementation, the Relationships referred to as Properties or Slots between the various artefacts of the model (Figure 7-10) are identified. These artefacts are mapped to classes and subclasses in the ontology (Figure 7-11). The constraints and validation metrics are defined and are mapped to domain and ranges in the ontology.

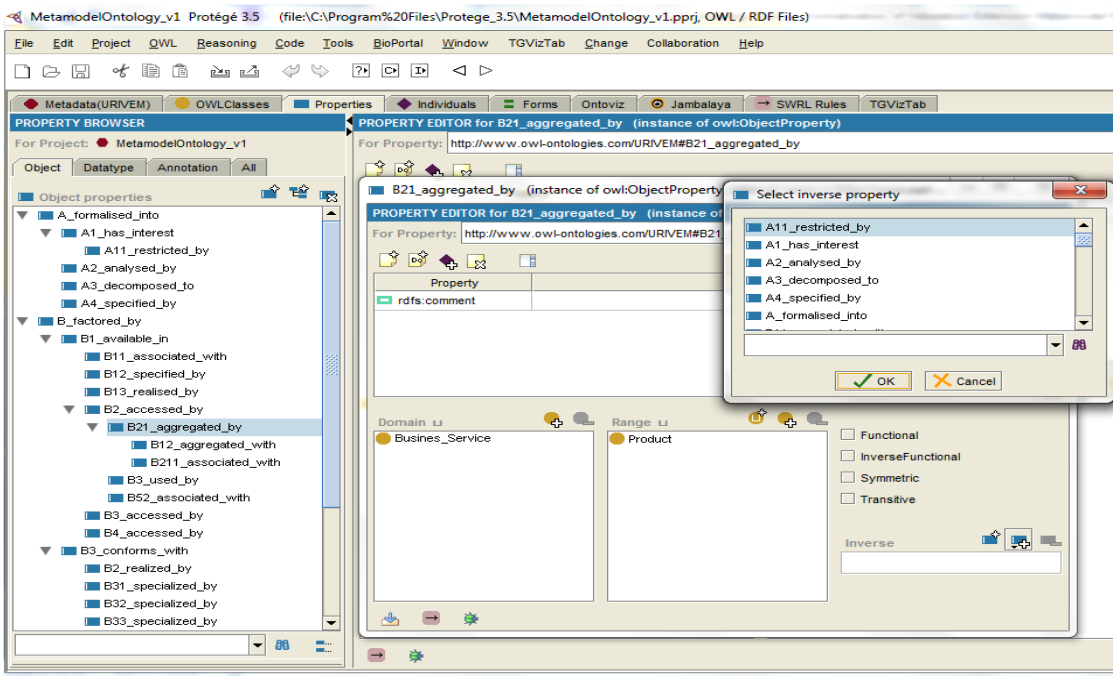

Figure 7-12: Association of properties to domains and ranges in OWL implementation

The limitation of this transformation to ontology currently is that Transitive, Functional, Inverse and Symmetric variants are not considered as they are not synonymous with EAF modelling notations identified in our metamodel. Validation slots are useful as they are used to encapsulate queries by domain and ranges. Domain and Range are constructed to represent the constraints and validation metrics as in Figure 7-12. For development of EAF ontology, each slot is assigned a domain though a slot may not necessary be assigned a range for the reason that not all slots may need to be validated. Slots link individuals from the domain to other individuals from the range. Unlike applied in conventional ontology derivation, a range can be used to set constraints that needs to be validated as well as axioms for querying. The transformation also ensures that all individuals which represent the components belong to domains so as to provide a mechanism to relate objects from the ontology to the business artefacts in the EAF. This in turn ensures that both structural and semantic heterogeneity are resolved allowing concerted queries to be performed on the generated RDFS. One major advantage here is that collaborative mappings can be applied to the ontological components. Additionally, with the extended capability to use specialized ranges provided by web ontology language (OWL) expressions, the level of granularity is increased.

7.7.3 Correlating the Ontology to the Metamodel

A complete ontology transformation of the metamodel is shown with RDF Graph in Figure 7-14. It depicts an extensive filtering criteria which can be used to probe the metamodel exposing the motivation, structure information and business behaviour. For selected properties, the ontology bindings to domains and ranges that relate to the RDF graph are delineated in the mapping scheme in Figure 7-13 implementation.

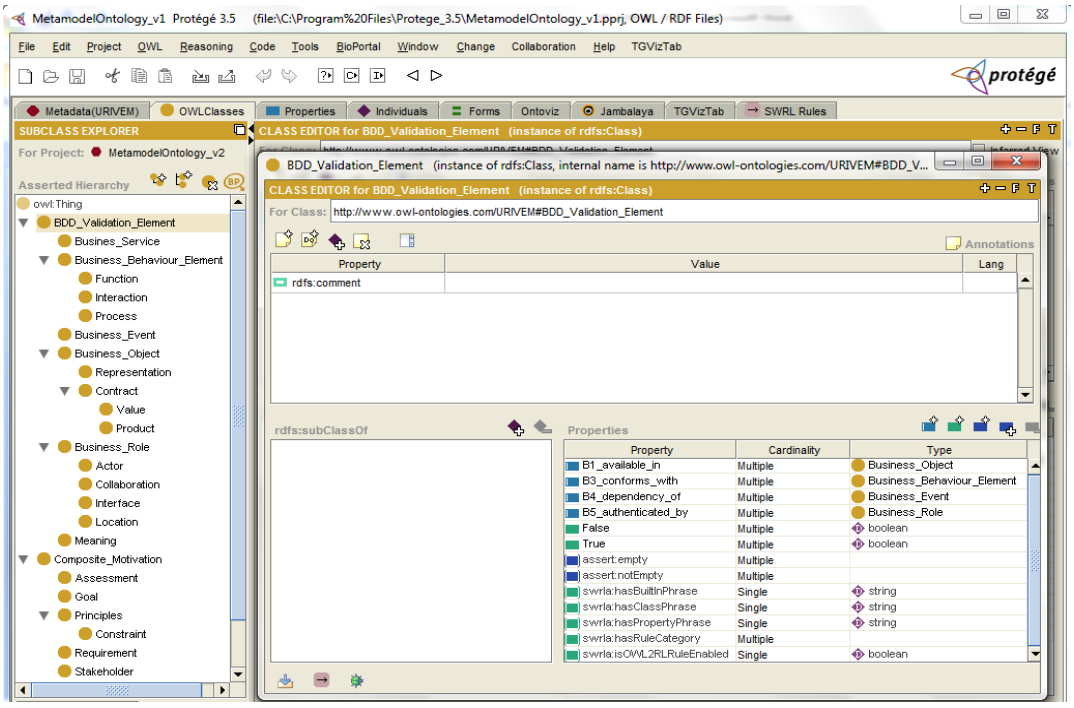

Figure 7-13: RDFS Subclasses, Properties, Cardinality and Type

The diagram in Figure 7-14 also portrays an extensible knowledge representation with elements of a theme for business behaviour from a ubiquitous viewpoint. The vocabulary generated with this ontology forms part of the triplestore that will be queried. With this construct, it is easy to build queries by N-Triples with the subject denoting the class; the predicate denoting traits or aspects of the class and expresses a property between the subject and the object. The object is the motivation associated by transition through the hierarchy of properties.

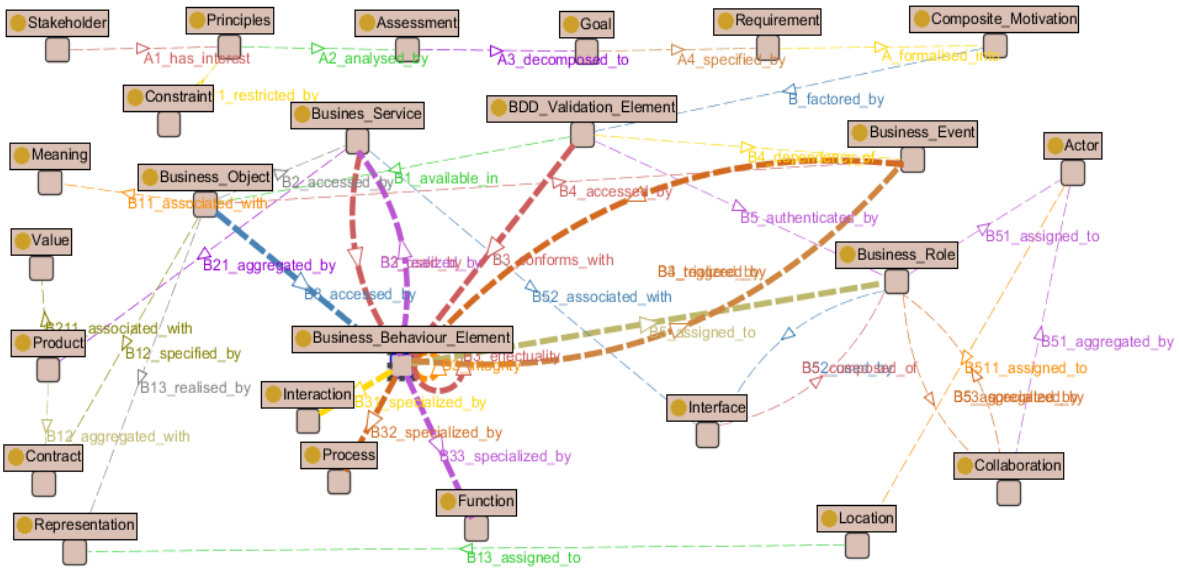

Figure 7-14: RDFS of the ontology showing exact representation of the Metamodel

7.7.4 Querying the Ontology using the Reasoner

The listing of RDFS construct for the metamodel extended with validation is presented in Appendix D. While there are several literatures on querying ontologies, in this section analytic query operations are carried out to demonstrate whether the transformed metamodel can be validated using the triple stores generated with the RDFS. Recently there have been major development initiatives in query processing, access protocols and triple-store technologies. The broad categories include in-memory, native and the non-memory non-native based on the implementation architecture (Rohloff et al., 2007). In-memory triple repository stores the RDF graph in main memory. Though it has been argued that storing everything in main memory introduces complexities especially when complex and large volumes of triples are involved (Wache et al.,2001; Jan and Dietz, 2006), the OWL uses the in-memory stores approach very effectively with its Reasoners to perform abstruse inferences in persistent RDFS stores; which otherwise could have been very difficult to perform. SPARQL has also been used in many implementations including Knowledge Explorer, Open RDF Sesame and Big Data. While OpenRDF Sesame is a de-facto standard framework for processing RDF data and includes parsers, storage solutions triple stores, reasoning and querying, using the SPARQL query language, it also offers a flexible and easy to use Java API that can be connected to many leading RDF storage

solutions. This is primarily because RDF Graphs are a powerful and flexible means of representing all kinds of linked data. In the case of Big Data, it provides an ultra high-performance graph database which supports the RDF data model and provides a standardised way of describing, interchanging, and querying graph data.

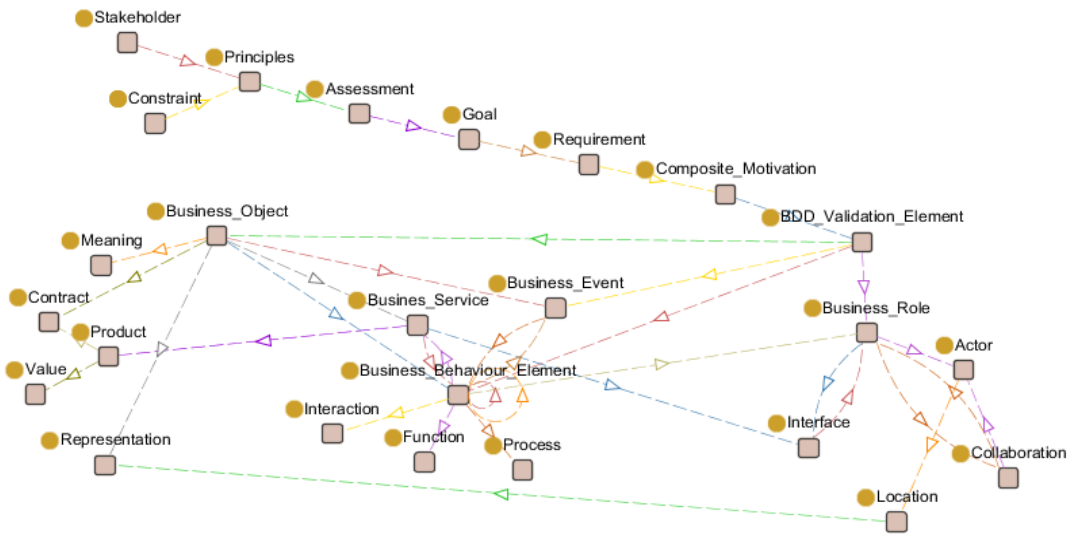

Figure 7-15: Schematic segregation of RDF into motivation, information, behaviour and structure

An example of this interchange is shown in the implementation of the extended validation metamodel in Figure 7-15. The diagram shows a transitive closure on the sub-property and sub-class hierarchies selected on the filter panel. However, the limitation of size of data in storage may affect clarity capability in a persistent storage, precipitating the need for the triple store to have available API.

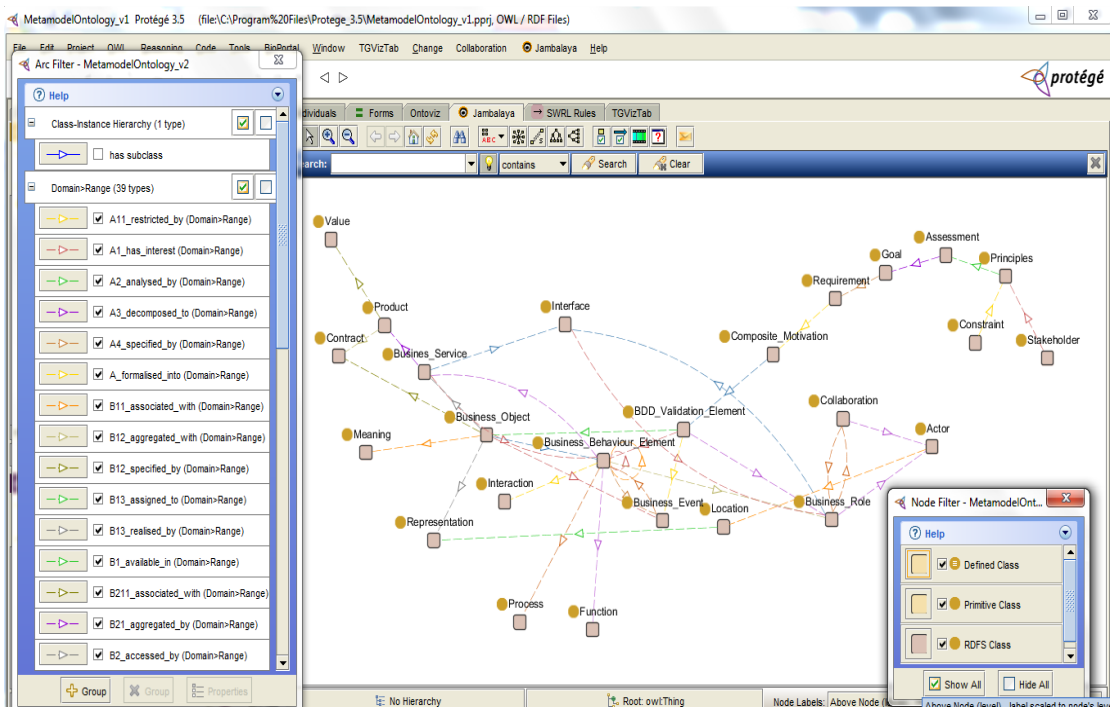

Figure 7-16: Querying the ontology using the Reasoner

7.7.5 Developing the Behaviour Driven Modelling Constraints Specification

The Behaviour Driven Modelling Constraints Specification is a concept inspired from BDD to transform BDD-style scenarios to SPARQL format. It offers an agile query development technique that streamlines collaboration and brings together the various constraints and artefacts that make up a viewpoint. It allows the collaboration of performance acceptance test and model driven test design practices as found in extreme programming to be extended to ontologies. The agile specification of BDD yields a description of iterative cycles with well defined outputs for delivery of practical, testable model that has relevance with motivation.

By adopting this approach, a clear understanding of desired model behaviour through analysis of the business functions, processes, events, interactions, objects and motivation from viewpoints, aspects or stakeholder's perspective is adduced. It extends BDD by writing test cases in a natural language that non-modellers can read. Behaviour Driven Modelling Constraints Specification uses a native language in combination with the ubiquitous language of domain-driven design to describe the purpose and benefit of the RDFS. This allows the testing to be focus on the intrinsic composition of the model artefacts rather than the code and averts metaphoric complexities between the enterprise concerns and the domain semantics. The principle for the approach is grounded on the Theoretical Principles for Model Validation Rules discussed in section 5.2 and is underpinned by a process that encapsulates:

- Establishing the goals of different stakeholders required in the vision of the model.
- Establishing the constraints that restrict the actualization of those goals.
- Segregating those artefacts that are related to the actualization of those goals.
- Descriptive exemplification of the behaviour of model, or parts of the business processes.
- Automating those examples by using BDD constructs to provide test basis for validation.
- Describing the behaviour of the model to help clarify responsibility and specifying the queries such that it can be transformed to SPARQL.
- Describing the constraints of the model to associate motivational goals to requirements and expected outcomes from related artefacts within the model.

Thus Behaviour Driven Modelling Constraints Specification is driven by business motivation. Each artefact of the model provides some aspect of behaviour which, in collaboration with the other artefacts constitutes the business behaviour of the model. For clarity, this principle is exemplified firstly by using the Gherkin syntax. Gherkin is a business readable, domain specific language created especially for behaviour descriptions and provides the ability to remove logic details from behaviour tests. The syntax of Gherkin is as follows;

```
Feature: Some terse yet descriptive text of what is desired
    (In order to realize a named business value
    As an explicit system actor,
    I want to gain some beneficial outcome which furthers the goal)

     Scenario1: Some determinable business situation
          Given some precondition
          And some other condition
          When some action by the actor
          And some other action
          Then some testable outcome is achieved
```

Feature describes some terse yet descriptive text of what is desired. A Feature can consist of many scenarios. Each scenario is an exemplar and is designed to illustrate a specific aspect of the behaviour of the model. The words "Given", "When" and "Then" are often used to help drive out the scenarios while the "And" is not mandatory. Use of contexts, events and outcomes are used to drive model validation at the level of abstraction of the EA model. For instance, the following examples describe an aspect of model artefact composition as behaviour in RDFS:

```
Scenario: New model is empty
Given no artefacts are present
And no Relationships exist
Then the model should be considered empty

Scenario: Model with artefacts Available is not empty
Given a new model
When an artefact is added
And the artefact is constrained by a restriction
Then the model should not be empty.
```

These two examples for instance can also be used to describe the Boolean nature of a model and to ascertain the existence of a constraint. In BDD these examples would often be encapsulated in a single method, with the name of the method being a complete description of the behaviour. Both examples are required for the code to be valuable, and encapsulating them in this way makes it easy to validate the behaviour. For instance as validation, the above examples might become:

```
Class:  ValidateModel(artefact)
def validate_empty_model_is_false(self):
model  = []
assertEqual(bool(model), False)

def validate_populated_module_is_true(self):
   model = []
   model.append('artefact')
   model.append('constraint')
```

The description is intended to be useful if the validation fails, and to provide documentation of the RDFS behaviour. Once the examples have been written they are then run and the code implemented to make them work in the same way as BDD. The examples then become part of the suite of regression tests of the model in SPARQL.

7.7.6 Querying the ontology using SPARQL

A query language in this context refers to a set of interrogative assertives with similar circumspective characteristics with SQL. Query languages usually fall into one or more of the four categories specified as SELECT, AGGREGATE, UPDATE and DELETE (Keith & Schincariol, 2013). In this implementation, the SELECT query category which retrieves a persistent state from one or more entities and filters results is adopted. This may be extended to include the AGGREGATE queries which are variations of the SELECT query and groups results to produce summary data. Together the SELECT and AGGREGATE queries constitute the outcome reports in this implementation as they primarily focus on generating data for comparison. The UPDATE and DELETE queries which conditionally modify or remove parts or entire sets of entities are not considered in this research as the work aims to verify and not make change in the taxonomy. Querying in this context deploys entities and objects written against domain ontologies instead of rows and columns of database.

Queries as a means for EA model validation provides a mechanism through which interactions occur within ontologies. A variety of query languages designed for this purpose includes SeRQL, RDQL and recently SPARQL (Almeida & Guizzardi, 2013). This research adopts SPARQL for querying the EA models transformed to ontology. The rationale for this is that SPARQL is standardised by World Wide Web Consortium (W3C) and supports RDF triple stores (Quilitz et al., 2008). As data formats that support information validation differ depending on forms adopted, there are variant specifications defined in the semantics of SPARQL queries that support these alternatives making them particularly suitable for model validation. For instance while SPARQL 1.1 Protocol for RDF is a protocol defining means for conveying arbitrary SPARQL queries and update requests to a SPARQL service, SPARQL 1.1 Service Description defines a method for discovering specifications and is a vocabulary for describing SPARQL services. Another major advantage of SPARQL is that the RDF triples in both the queried RDF data and the query pattern are interpreted as nodes and properties of the direct graphs corresponding to artefact and relationship in the EA model. This facilitates the harmonization of the resulting query graph and the graph variables (Glimm, 2011). Consequently, this research takes into cognizance the various forms of SPARQL queries which contain a set of triple patterns called basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF stemmed from that subgraph is substituted for the variables. Hence, executing SPARQL queries generally involves graph pattern matching (Huang, 2011). This approach allows queries to be kept in the model as annotation property values at a class level.

Data types for querying the EA instance include the use of literals with the general syntax or an optional data type Internationalized Resource Identifiers (IRI) or prefixed name. The RDF data model itself represents the EA artefacts as graphs consisting of triples with subject, predicate and object and allows SPARQL to make queries that involve artefacts from more than one graph. SPARQL query is executed against the RDF dataset which represents a collection of graphs. A SPARQL query matches different parts of the EA model pattern against different graphs. Thus the RDF graphs and schema are generated as primary test basis for the tests carried out to validate the hypothesis of this research. The construct of the SPARQL query is concentric at the triple level applied on the RDFS. The outcome is extrapolated by filtering out individuals and classes with specific characteristics or properties amongst many other attributes. Finally, the choice of SPARQL as a validation tool in this implementation is grounded on its capability to incorporate logical reasoning, graph patterns along with their conjunctions, disjunctions to support extensible value testing and traceability. This capability is required as it effectively allows the visualization of integrated motivation specifications within queries.

An outcome of a SPARQL query is presented in Figure 7-17 as a result set and a RDF graph. This example illustrates a direct principle of extraction with respect to the metamodel under consideration but a more extensive validation is provided in the case studies where constraint checking is used to validate motivation in conjunction with Protégé-OWL assertions added as annotation properties to the selected classes.

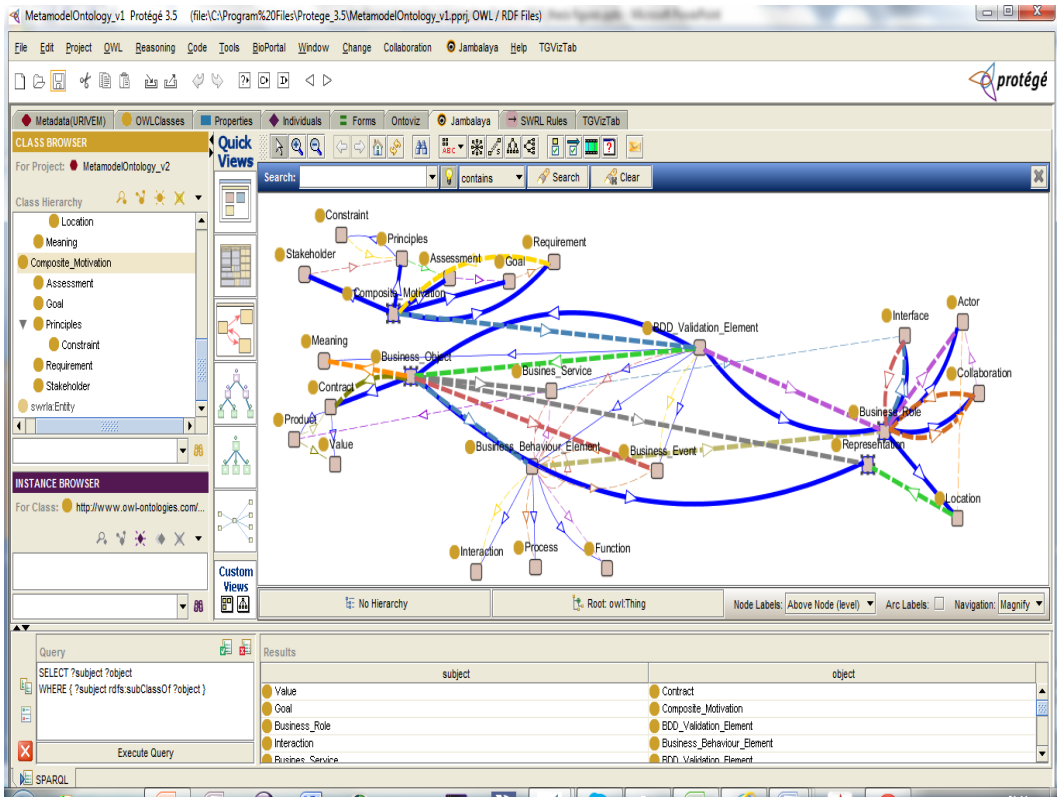

Figure 7-17: Traceability of class and relationships on ontology using SPARQL

SPARQL Query types can also be expressed in terms of Effective Boolean Value (EBV) or as a Filter Evaluation (FE). Here result description is presented in the form of bindings. A binding is a doublet of variable and RDF term represented in a matrix (Sintek et al., 2002). Each solution is represented as one row bound to the corresponding column header. Considering the validation metrics for instance, the queries can be used to evaluate an instance for Goal realization of a class or property of business behaviour analogy. It is also possible to use SPARQL FILTERS to restrict the solutions of a graph pattern match according to a given expression thus implement the Constraint Assessment validation metrics. This is demonstrated in Figure 7-18 where a SPARQL query results in a displayed result of all artefacts in the model with their relationship aggregated as subjects and objects for attributes of the validation elements. Considering the SPARQL query issued as below;

```
SELECT?subject ?object ?cls
WHERE { ?subject rdfs:domain ?object }
ORDER BY ?object
```

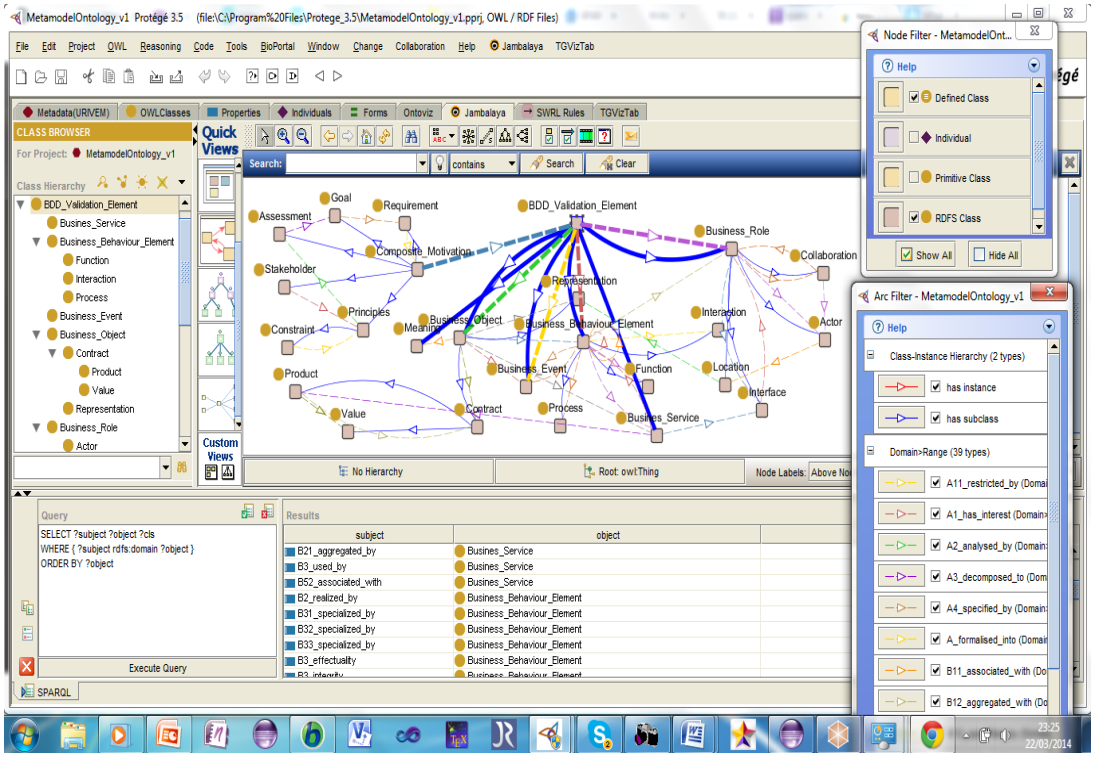

Figure 7-18: SPARQL Query combined with Filters, Cardinality, Artefact and Property

By ordering the result, it allows for a clearer visualization and determines whether the entire artefact have been ascribed with all the desired associations required to confirm the desired Goal Realization; or whether the required or appropriate constraints are applied as in Constraint Assessment specification. The same principles are deployed more extensively in the case studies to demonstrate a variety of validations applied in the validation of the model.

8 EXPERIMENTAL STUDIES

In this chapter, two separate implementations of the Model Driven Validation Approach are presented. The case studies focused on using the extended validation elements of a modelling language to create models that conform to the metamodel of Business Layer and to exert validation through querying the Resource Description Framework (FDF) Schema. The outcome of the queries issued with the taxonomy of the RDF graphs produced are analysed and then subjected to evaluation adopting the supporting theories of this work. The objective of the case study is to provide an in-depth application of the methodology, detailed examination of artefacts and an empirical inquiry that allows the investigation of the hypothesis proposed within a real-life context.

8.1 Case Study A: University of Middle England Laptop Loan Scheme (UME-LLS)

The University of Middle England (UME) is worried about its future. After some analysis, the key reason seems to be that students and the resources used for teaching and learning activities are badly aligned and that some of the modules assume that all students have their own laptops (which are not always the case). As a result of this analysis, UME decides to implement a laptop loan scheme whereby students who do not have their own facilities can loan them from UME at no cost. The scheme raises questions about which modules require laptops, whether the current teaching and learning rooms are suitable for laptop usage, and whether UME has suitable software with appropriate licenses for use by students. In order to implement the laptop scheme, UME decides to use EA. Having used EA to design a migration path, UME wants to determine whether the scheme is consistent with its business goals and to understand how its new architecture operates. A working group is tasked by the UME executive to come up with a description of the information structures relating to student teaching and learning.

8.1.1 Ascertain the values of the UME-LLS Case proposition

Two theories are applied for this exercise; the Data Validation Theory (VDT) and the Information Systems Design Theories (ISDT). The VDT enables the requirements for the conceptual model to be determined by guiding the process for extraction of relevant data on the problem domain. This is crucial for the development of a concise metamodel that satisfactorily represents the logical business behaviour, motivation and pivotal assumptions. Complementing the VDT, the ISDT which is concerned with how to build the artefacts and design process stipulates the composition of the meta-requirements and goals. The principle also directs the annotation of artefact classes that meet the meta-requirements. By integrating the normative and descriptive intentions of these theories in conjunction with the case study, the primary business goal that drives the objective of the case is identified and expressed as;

Loan Laptop to students at no costs for use with modules that require use of Laptop

Having identified this high-level goal of the case study, it is possible to determine the business behaviour and components needed to achieve this goal using the VPEC-T approach. This is achieved by addressing all the elements of these theories in conjunction with each other to decipher a clear understanding of the project's deliverables and the underlying business objectives of the project. In this case, the following sub-goals can be identified as adding integral value to the high-level goal of the proposition.

- Provide suitable teaching and learning rooms for use of the laptops.
- Allow booking of teaching and learning rooms.
- Allow students to loan the laptop from the UME-LLS.
- Provide appropriate software and license for use by the student on the laptops.
- Specify the modules for which the students may use the laptop for learning.

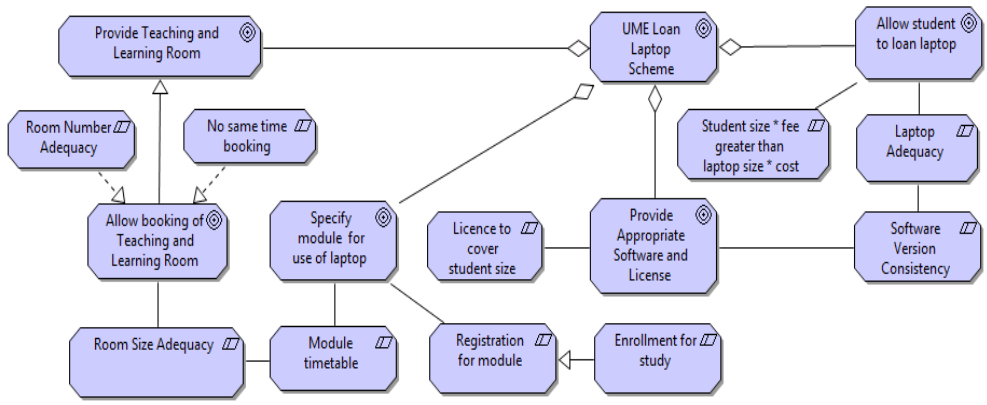

Figure 8-1: Business Goal and Constraint model for the UME-LLS

Using ArchiMate modelling tool, the goals and constraints are modelled as in Figure 8-1. The Business Goal and Constraint model exemplifies the ultimate significance of the scheme which should guarantee the success of the UME Laptop Scheme. The high-level goal is cumulative of four adjoining sub-goals associated with the aggregation attribute. This consolidates the value of the business proposition. Prerequisite constraints are associated with sub goals in this model to aid the assessment and specification of requirements. As noted, this model is specifically defined within the motivation extension abstraction of the ArchiMate. The rationale for this is to correlate those restrictions that are required for validation directly with goals and sub goals. Thus the design of test procedures would ensure that testable attributes are not omitted within the test basis and the ontology created through mapping is as comprehensive as possible.

8.1.2 Process Model for the UME-LLS

Applying the ISDT theory, three major process models are identified with this case study. These are Process for Learning or Teaching Room Allocation; Process for Laptop Procurement and Configuration; and Process for Laptop Request. The general characteristics of these processes are that each one consists of actors, business roles, functions, services and data objects. Each of the data objects consist of requirements that specify the goals to be achieved by the process.

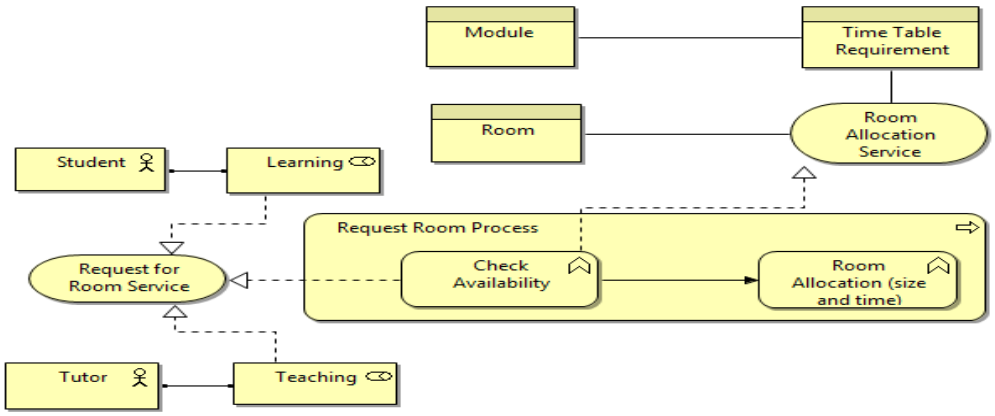

Figure 8-2: Modelling of the Process for Room Allocation for the UME-LLS

The primary outcome of the process model in Figure 8-2 is to allocate a Room to either a tutor for teaching or to a student for learning. The roles of each of these actors are realised through a Request for Room Service which in turn triggers the need to Check Availability of Room. The need is constrained by suitability of room with respect to time and duration of the module, availability and size of the class. The Check Availability function is a component of the Request Room Process which utilizes the Room Allocation Service to realize the need. Data objects available to the Room Allocation Service consist of Timetable associated with Module and Room. These conditions when met direct the Allocate Room function to designate a room accordingly.

The Process for Laptop Procurement and Configuration is a slightly more intricate as it involves more actors and roles. Two sub-processes are identified in the model as procurement and configuration. The outcome for both sub-processes is the Laptop business object. The Laptop procurement Process is managed by the Finance and Procurement Departments taking into consideration the 'Student size fee greater than the Laptop size cost' constraint specified in the Business Goal and Constraint model in Figure 8-1.

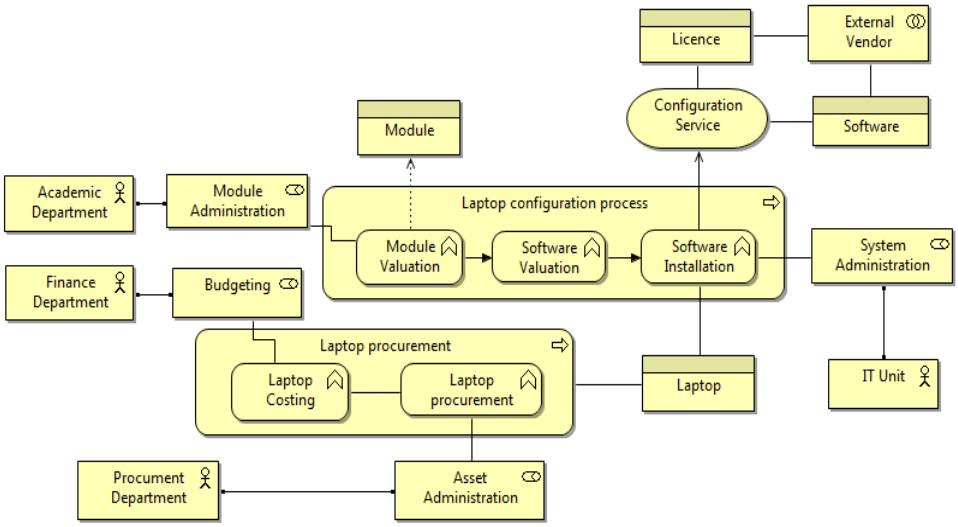

Figure 8-3: Modelling of the Process for Laptop Procurement and Configuration

Once laptops are procured and made available in the Laptop Business object, the IT unit carries out configuration based on the specifications defined by the Module and Software Valuation Functions. This criteria and requirements is a business function determined by the Academic Department. The software installations use Configuration Service provided by external collaboration for the licence and software product. This is depicted in the model in Figure 8-4.

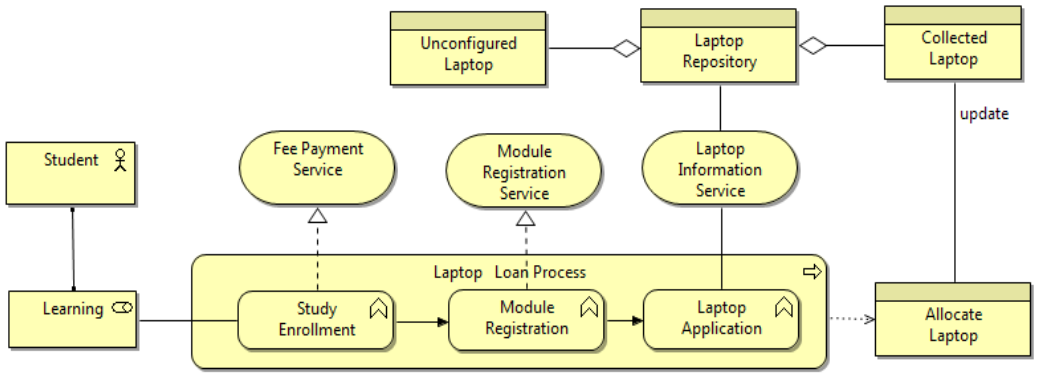

Figure 8-4: Modelling of the Process for Laptop Request for the UME-LLS

This model is constrained by the student registering for a module that require laptop use, payment of fees, enrolment for study and availability of the laptop as specified in the Business Goal and Constraint model of Figure 8-1. Thus three functions Study Enrolment, Module Registration and Laptop Application are identified with associated services for their realization. For the Laptop Information Service, business data objects provide information regarding laptops. Once a laptop is allocated, the Collected Laptop object is updated to maintain the integrity of the Laptop Repository object.

8.1.3 Organisational Model for the UME-LLS

The Organizational model defines the structural composition of the enterprise and includes lines of authority, communications, duties and allocation of resources (TOGAF, 2013). As the taxonomy of this model usually depends on available resources and the value of goals to be achieved, the Organisational model is often considered as a blueprint that depicts the cardinality of active resources needed and their required application within a context in an enterprise. This is inclusive of units, actors, interfaces, locations and collaborations. In some cases, specification of a unit may be recursive or may identify collaborative resources that aid in the completion of a specific work process. In many cases, organisational model can incorporate multiple physical locations, reporting structures, workflow channels and control capabilities. Translating this to discussed frameworks, in the Zachman framework this is defined simply as the "who" and "why" while Figure 8-5 shows the structural composition of an organisation structure in ArchiMate.

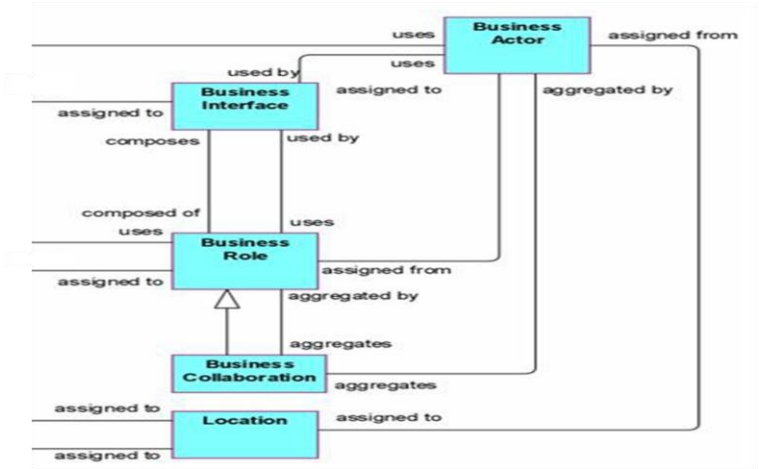

Figure 8-5: Organisational structure of ArchiMate. (Abstract from ArchiMate)

Inconsideration of these detailed specifications, the organizational model elements in the UME-LLS case study would comprise of the following;

i. The UME – the university that implements the laptop loan scheme – Actor/Role.
ii. Student – who loan the laptop – Actor/Role.
iii. IT Unit – that configures laptop with appropriate software and version – Actor/Collaboration/Role.
iv. IT Unit –provide suitable infrastructure in the Learning and Teaching Room by liaising with External vendor and Estate Administration - Collaboration/Role.
v. Procurement Department - procure and store laptop – Actor/Role.
vi. Estates Administration - provide suitable Learning and Teaching Room – Actor/Role.
vii. External vendor - supply laptop and software licenses - Collaboration/Role.
viii. Tutor - teach module that require laptop – Actor/Role.
ix. Finance Department – collect fee, pay contractor and assert costing criteria – Interface/Role.
x. University Portal - register student for studies – Interface/Role.
xi. Departmental Portal –register student for the module and allocate laptop – Interface/Role.

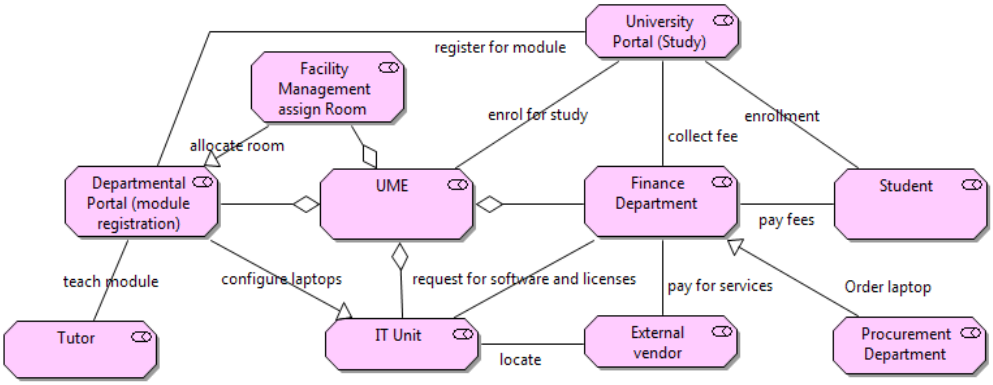

Figure 8-6: Stakeholder Viewpoint of the UME-LLS

These elements are related to each other as presented in the model in Figure 8-6 from Motivation Extension Perspective.

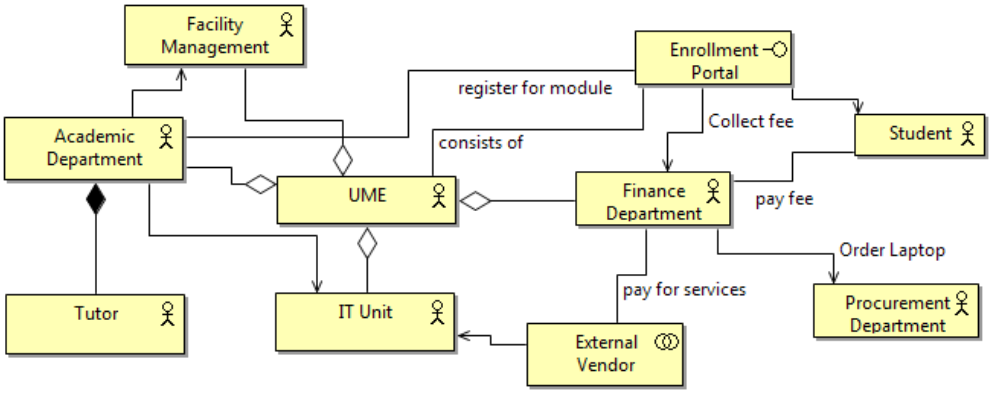

Figure 8-7: Organisational model of the UME-LLS from Business Perspective

From the Business Architecture perspective (Figure 8-7), all units and departments are aggregated to the UME. Tutor is associated directly with Academic Department while the Student accesses Enrolment Portal to enrol as a member of an Academic Department, and to pay fees to Finance Department. The role of the Procurement Department is defined as a specialization of the Finance Department as there needs to be a symbiosis between the two. The Finance Department triggers Procurement Department to perform its functions and to collaborate with External Vendors. The External Vendor collaborates with the IT Unit to provide Software and licensing services. This model thus represents a comprehensive structural composition of the UME-LLS.

8.1.4 Information Model for the UME-LLS

The ArchiMate 2.1 specification defines information viewpoints as comparable to the traditional information models created in the development of most information systems (TOG, 2012). The nature of ISDT is articulated in the building block of the Information model and depicts the structure of the information used in the business processes such as data types and class structures. Exercising this theory in the context of this work, the Information Model shows how the information at the business

level is represented, the data schemas used, and how these are mapped onto the underlying infrastructure such as the RDF ontology. This conforms with the Kernel theories discussed in section 5.4.2 as it enables the redesign of current intra and inter-organizational business process needs prior to implementation. Other entity types that are associated with the Information Model in relation to this work are abstract and formal representation of properties, relationships and functionalities. While Zachman Framework denotes information model with the "What" aspects, ArchiMate 2.1 technical standards identifies element of the information model as in Figure 8-8.

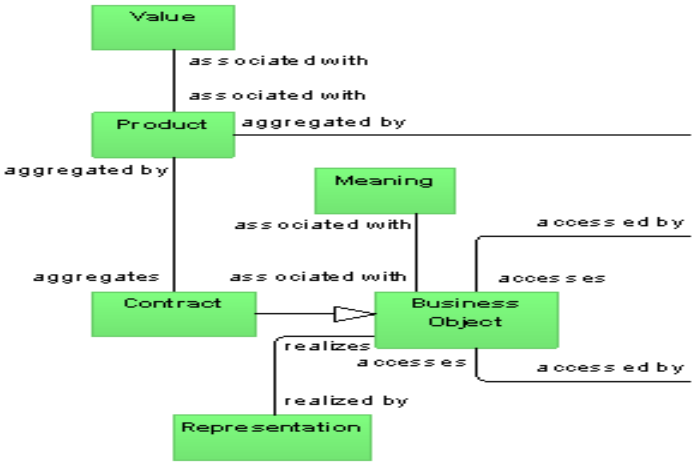

Figure 8-8: Information structure abstraction of ArchiMate. (Abstract from ArchiMate)

The entity types within the model are used to describe the domain and provide constraints by a specific set of properties, relationships and operations. This adheres to the Conceptual Model Validation Theory explained in section 5.1.1 and ensures that inferences about the problem entity are captured to allow the validation of the model to be carried out at a later stage. These theories which provide the grounding for the abstraction of information needed for EA model and validation when synthesized with the VPEC-T concepts, in the case of the UME-LLS the following collection of elements, mental filters and guides are extracted;

Values:

- Alignment of students and the resources used for teaching and learning activities

Policies:

- Principles that guide the process of loaning laptop
- Criteria that ensure that laptop can be loaned at no cost
- A description of the information structures relating to student teaching and learning

Events:

- Allocate a Room to module (constrained by suitability of room, availability and size)
- Student registers for module (constrained by registration for study)
- Student registers for study (constrained by fee greater than cost of laptop)

Contents - represents the business elements:

- Module that require laptops
- Laptop for loan
- Teaching and Learning Room
- Infrastructure at Learning Room
- Suitable Software
- Appropriate Licence

The depiction of these thought analysis is presented in ArchiMate models as follows;

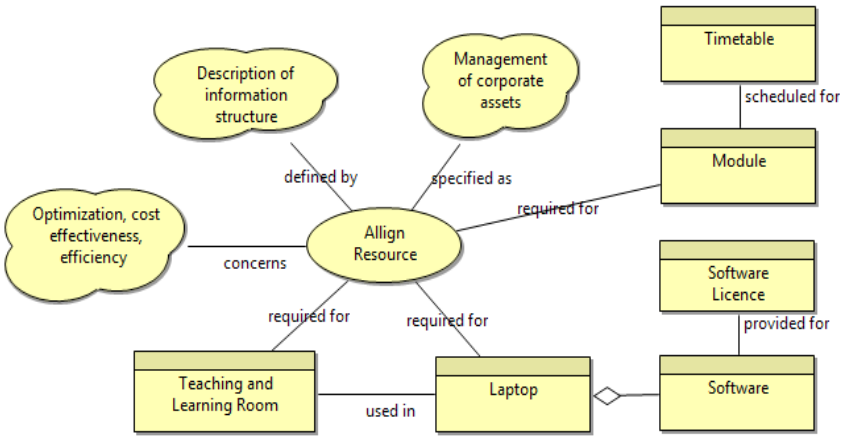

Figure 8-9: Information model of the UME-LLS from Business Perspective

Three elements are considered in this model; business value, meaning and object. The value represents the proposition to align resources within UME-LLS domain. The meaning is expressed by taking into consideration optimization, cost and efficiency concerns. Corporate assets constitute the object and include laptop, learning and teaching room, software, licence, learning module and timetable. The appropriate description of this information forms the information model.

8.1.5 Function and Service Model of the UME-LLS

The function model is a structured representation of activities, actions, processes and operations within the modelled domain and specified viewpoint. The Business Services model provides a formal relationship between business process models and the model elements that realize them. Combining these two viewpoints in the UME-LLS, the rationalisation allows a precise description of the behaviour of the business abstraction; assists with discovery of information requirements; helps identify alternatives; and establishes a basis for determining goals and service impacts. Each service in this respect is a self-contained unit of functionality.

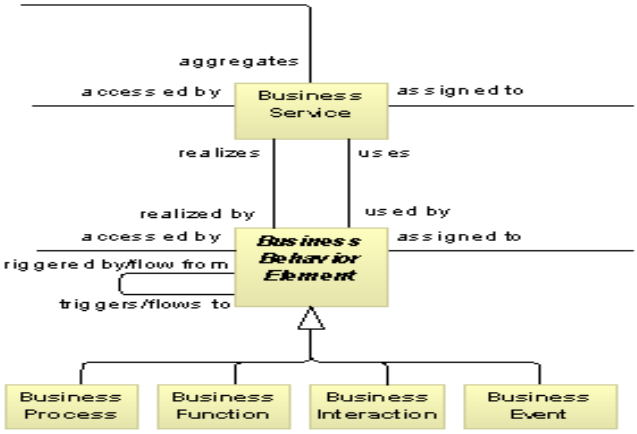

Figure 8-10: Service and Functional abstraction of ArchiMate. (Abstract from ArchiMate)

Therefore in the context of this presentation, the business function and service model combines the strings of business functions, triggers from a business point of view and exposes the functionality of the business roles and collaborations in the project. Figure 8-10 depicts this composition of ArchiMate Business Layer, behaviour which encapsulates service and function and Figure 8-11 the exemplification with the UME-LLS.

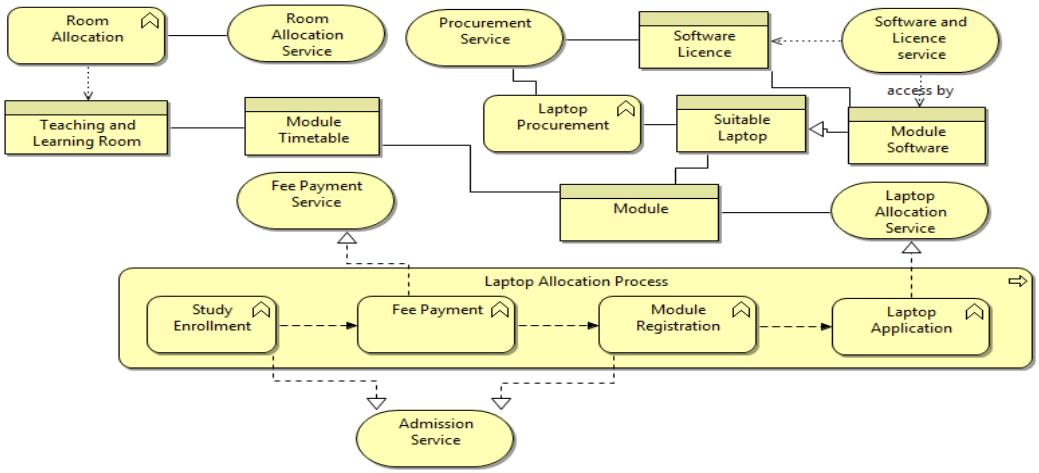

Figure 8-11: Function and Service model of the UME-LLS from Business Perspective

The model depicts functions and associated services required for their realization. This key functionality is accessed through one or more business interfaces. Fee Payment Service and Admission Service are realized by Fee Payment and combination of Study Enrollment and Module Registration respectively through a single Laptop Acquisition process. Several business objects support this operation. The function and service model provide a unit of functionality that is meaningful primarily from the point of view of Laptop Allocation Service and secondarily from Procurement and Room Allocation Services. The internal domain is represented by the business behavior of the student, module, department, procurement and IT unit while the Software and License Services are contained within the external domain.

8.1.6 Mapping of the UME- LLS to Validation Extension Metamodel

Based on the logical and conceptual abstraction of the MDVA presented in Figure 6-1 and the underpinning physical metamodel in Figure 7-9, an instance is developed to visualize a perspective from the student's view of the UME-LLS. This model is presented in Figure 8-12. The model demonstrates the applicability of the proposed contribution of this work as it introduces a new element represented as the BDD Validation into the metamodel instance. The as-is variant of this model is the unavailability of this element and the relationships which it extends to other business functions, services, triggers, data objects and processes. The description of this viewpoint is the business behaviour from the distinctive perspective of the student and shows an implementation of the logical model with the Business Validation artefact and properties. The motivation aspect is related to the ArchiMate core through requirement. For straightforward traceability, component at the business layer are annotated with unique identifiers (UID) also used to distinguish usage in development of the test specifications. The result of the validation supports comparison between obtained model and expected motivation. Transformation to RDFS subsequently ensures that the overall constraints and goals of the enterprise are modelled into the ontology to provide the basis for validation and traceability. To enable mapping, a lithographic profile of the model in Figure 8-12 is presented in Figure 8-13 with annotations of the UIDs and motivation elements. The annotated Figure 8-13 inherits the attributes of its metamodel presented in Figure 7-9 and maintains the structure and composition of its taxonomy.

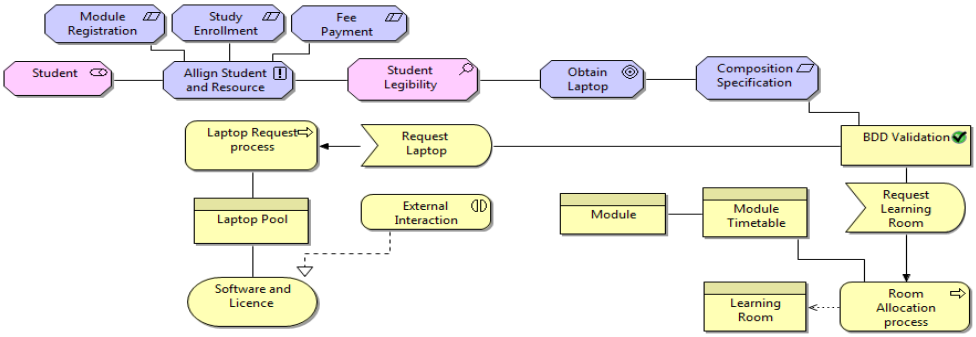

Figure 8-12: Business Layer Model of the UME-LLS with VEM

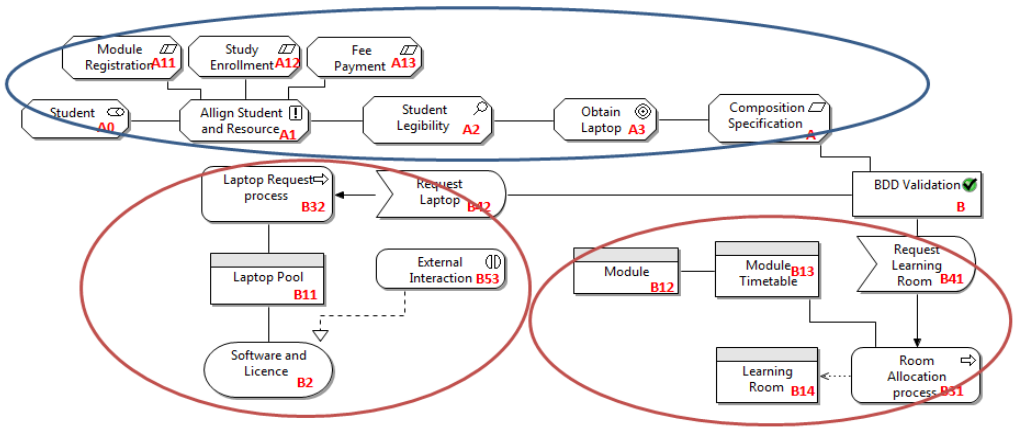

Figure 8-13: Annotating the Business Layer Model of the UME-LLS with VEM

In the profile, a distinction of three main decompositions is identified. These are the motivation concepts and two business behaviour specifications. The two business behaviour aspects are distinct as they relate directly with the BDD Validation element to enable the development of the Behaviour Driven Modelling Constraints Specifications. This is then formalised with the motivation concepts.

8.1.7 Developing Behaviour Driven Modelling Constraint Specifications for the UME-LLS

Having modelled some fundamental perspectives involved in the expression of the UME-LLS, the semantics required to validate the model is developed using the following behaviour driven constraint specifications. This is synonymous with some commonly used ubiquitous language semiotics such as in BDD. The Theoretical Principles for Goal Evaluation discussed in section 5.3 is adopted to substantiate this preposition.

In the case of the UME LLS some examples can be extracted as follows;

Assumptions:

There are 30 Learning and Teaching Rooms;
There are 70 Laptops;
Course of study is computing;
Module that requires laptop is Module A;
The total size of students allowed to study Module A is 50;

Using the process model for the UME-LLS in Figure 8-13 with the more elaborate descriptions of the process models in Figure 8-4, the following scenarios can be developed;

Feature1: Request a laptop.
In order to participate in teachings and learning of module A
As a student
I want to apply for a laptop

Scenario1: Register for study of Computing
Given that I have a student identification number and password
When I log onto the UME web portal
Then grant access to register for studying Computing

Scenario 2: Pay Fees
Given that I have registered for study
When I provide my bank details
Then debit school fee amount from my account
And credit UME with same amount

Scenario 3: Register for module A
Given that I have paid school fee
When I select module A
And indicate that I want to register for module A
Then Allow me to register for module A
And Increase the size of students studying module A by 1
Scenario 4: Apply for a laptop
Given that I have registered for module A
When I apply for laptop
And a laptop is available
Then Allocate a laptop to me
And reduce the number of laptops available by 1

Using the process model for the UME-LLS in Figure 8-13 with the more elaborate descriptions of the process models in Figure 8-2, the following scenarios can be developed;

Feature2: Request free teaching or learning room for use with laptop.
In order to teach module A
As a Tutor
I want to be assigned a teaching room

Scenario 1: I need Room information
Given that there are 30 teaching and Learning Rooms
When I request for a room to teach module A
And provide the time I need the room
And provide the size of the class for Module A
Then check if a free room is available
And the time the room is available

Scenario 2: Obtain a Room
Given that there is a free Teaching or Learning Room available
And the time of Room availability is same as time for module A
And the size of Room is adequate for 50 students
When the Teaching or Learning Room is requested
Then allocate the Room for use in teaching module A
And reduce the number of free Rooms available by 1
But increase the number of Rooms allocated by one

The concept here as in all cases in this research is not to query the RDFS directly as it would be done using SPECFLOW, BEHAT or similar framework for testing business expectations for BDD specifications but to convert this simple and clear specifications to Triples and then use Reasoners, Filters and SPARQL semantics to apply the assertions on the ontology.

8.1.8 Mapping of the UME- LLS to OWL Ontology

From the lithographic profile in Figure 8-13, a table of comprehensive UID is created specifying the Triples, Levels of Hierarchy, Domains and Ranges. Each element in the model is annotated as a Triple consisting of subject, predicate and object.

Table 5: Mapping of UME-LLS Properties to Ontology Hierarchy.

Metamodel TOP- DOWN Decomposition Analysis				RDF TRIPLES			Hierarchies of Properties					D/R	Map
METAMODEL CLASS			Model Instance	Subject UID	Predicate Description	Object UID	L1	L2	L3	L4	L5		
Composite Motivation													ℂ
Motivation Extension	Stakeholder		Student	A0	has interest	A1		L2				A	ℐ≡
	Principles		Align Student and Resource	A1	analyse by	A2		L2				A	ℐ≡
	Constraint		Study Enrolment	A11	restricted by	A1			L3			A	ℐ≡
			Module Registration	A12	restricted by	A1			L3			A	ℐ≡
			Fee Payment	A13	restricted by	A1			L3			A	ℐ≡
	Assessment		Student Legibility	A2	decomposed to	A3		L2				A	ℐ≡
	Goal		Obtain Laptop	A3	specialisation of	A		L2				A	ℐ≡
	Requirement		Composition Specification	A	formalised into	B	L1					A	ℐ≡
BDD Validation Element			BDD Validation	B	factored by	A	L1					A	ℂ
			BDD Validation	B	dependency of	B41	L1					B	ℐ≡
			BDD Validation	B	dependency of	B42	L1					B	ℐ≡
Information Model	Business Object			B1									ℂ
			Laptop Pool	B11	accessed by	B32			L3			B42	ℐ≡
			Laptop Pool	B11	accessed by	B2				L4		B42	ℐ≡
			Module	B12	aggregation of	B13				L4		B41	ℐ≡
			Timetable	B13	accessed by	B31			L3			B41	ℐ≡
			Learning Room	B14	used by	B31			L3			B41	ℐ≡
Business Function / Process Model	**Business Service**		Software and License	B2	accessed by	B11				L4		B42	ℐ≡
			Software and License	B2	realized by	B53					L5	B42	ℐ≡
	Business Behaviour												
		Process	Room Allocation	B31	used by	B14			L3			B41	ℐ≡
			Laptop Request	B32	used by	B11			L3			B42	ℐ≡
	Event		Request Learning Room	B41	triggers	B31		L2				B41	ℂ
			Request Laptop	B42	triggers	B32		L2				B42	ℐ≡
Organisation Model	**Business Role**												
	Interface												
	Collaboration		External Interaction	B53	realised by	B2					L5	B42	ℐ≡
	Actor												

Five levels of hierarchy are also identified with each business elements aggregated onto its motivation, organization, business and organisation aspects. This is then mapped onto ontology using the OWL Protégé adopting a top-down approach. A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the distinctive concepts. Ascribing this approach to the case study, classes for the general concepts of BDD_Validation and Composition_Specification are created first. Then the BDD_Validation is specialised by creating all of its subclasses as depicted in Figure 8-13. These two asserted hierarchy of classes represent the Core ArchiMate artefacts at the business layer and the modelling is specified by the Motivation Extension. Subsequently the entire taxonomy consisting of classes with siblings and in some cases subclasses and super-classes are mapped.

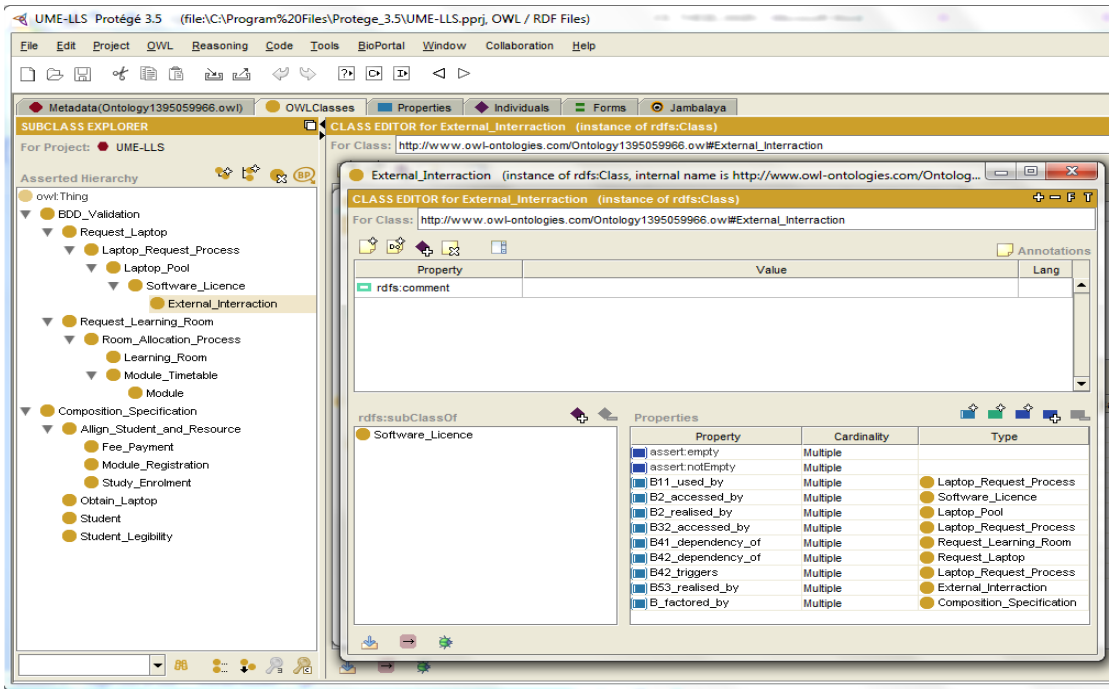

Figure 8-14: OWL implementation of UME-LLS Classes

As implemented in this case study, the top-down development process ensures generally that if a class A is a superclass of class B, then every instance of B is also an instance of A. These mapping is depicted in Table 8 and adheres to the Principles of Mapping Model to Ontology discussed in section 7.2.The ontology in Figure 8-14 represents a formal explicit description of concepts in the UME-LLS. The classes represent the concepts of the model elements. For the BDD_Validation, there are two subclasses for requesting Laptop namely Request_Laptop and Request_Learning_Room. Each of these subclasses has processes that drive the requirement. Similarly, Composite_Specification consist of all the motivation specifications including constraint elements aggregated as subclass for achieving the business principle Allign_Student_and_Resource. Slots are used to describe properties of classes and instances. Properties of each class describe various features, attributes of the model and restricts the role as expressed in Triples. This ontology together with the set of individual instances of classes constitutes the knowledge base of the case study.

Though it has been argued that there is no correct way or methodology for developing ontologies (Uschold & Gruninger, 1996), this research has demonstrated that in order to minimize the complexity of ontology visualization, it is not appropriate to implement the domain and range of the ontology synchronously. This is because the attributes of the domain and ranges are hereditary and by implementing the domain and range from the super-class, the subsequent relationship of the subclasses is overridden.

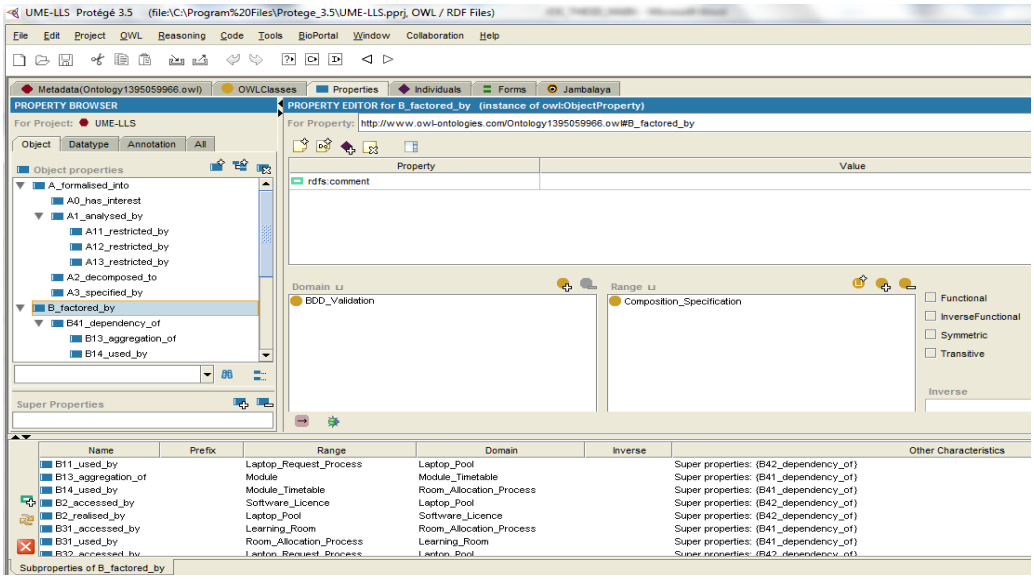

Figure 8-15: Properties and association with Domains and Ranges for UME-LLS.

Thus the approach of bottom-up implementation of the domain and ranges ensures that the appropriate attributes are assigned first between the lower hierarchies of classes before the attribute of the super-class is inherited. This approach maintains every attribute with no override and guarantees that the sequence of inheritance and connectivity integrity is maintained.

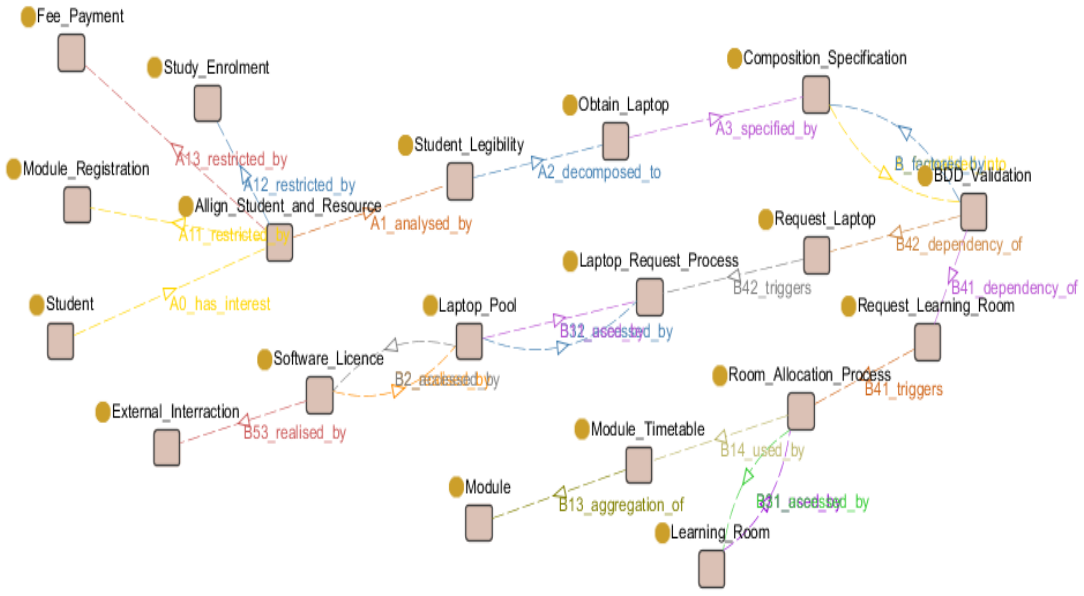

Figure 8-16: RDFS of the ontology showing slot transition of the UME-LLS

The taxonomy of this ontology is represented in Figure 8-16. Starting from the Student Class, there is a clear linearity of dimensions to Composite_Specification class as specified in the definition in Table 5. The constraints are restricted to the Allign_Student_Resourse Class. Between the Composite_Specification and BDD_Validation, there are the Factored_by and Formalised_into properties. This implies that the theme for the validation of the motivation extension is formalised into the BDD Validation elements in order to ensure that desired goals are achieved and ascertain that constraints specified by principles are validated. While the BDD_Validation element factored by the ArchiMate core is canonical to the realization of motivation, Composition_Specification formalized into the motivation extension ensures that its elements can be validated by the Triples.

Slots as implemented in the UME-LLS case study has different facets each describing the value type allowed, the number of the values (referred to in some implementation as cardinality) and other features of the values the slot can take. For example, the value of a B11_used_by slot which is the instances of the class External_Interaction can have multiple values (Figure 8-14). That is, External-Interaction which in the case of the UME-LLS is the vendor relates jointly with the Finance for payment for laptop and services, as well as the IT Unit for provision of software and licenses. In this case the cardinality is multiple but for Learning_Room, slot cardinality can only be single; implying that only one Learning_Room can be allocated to a Module at a time. The OWL protégé distinguish only between single cardinality (allowing at most one value) and multiple cardinalities (allowing any number of values).

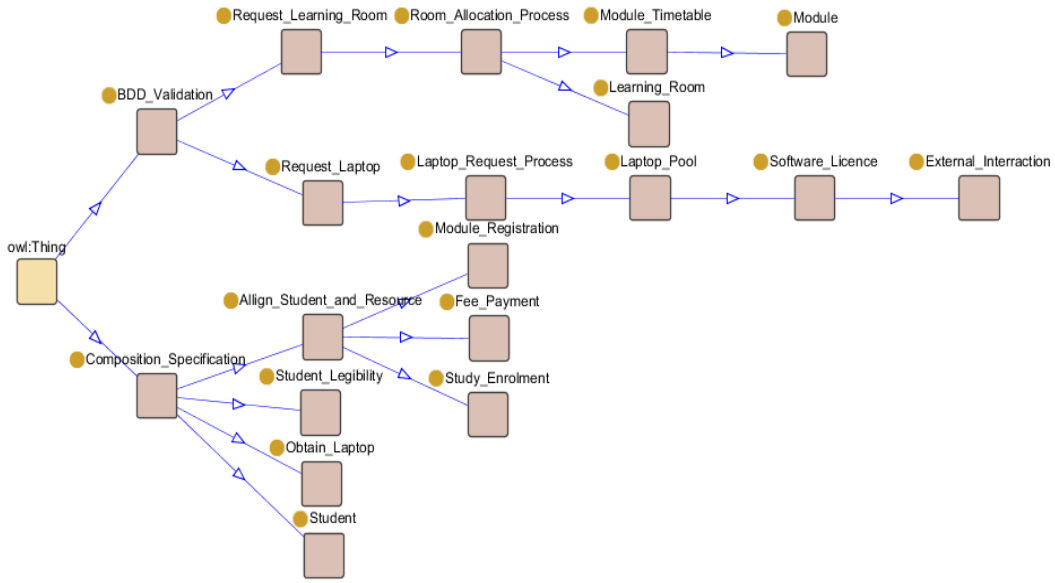

Figure 8-17: Schematic segregation of the UME-LLS showing the class hierarchy

At the intersection of the BDD_Validation element, there are two dimensions which correspond to the two business behaviour (see Figure 8-13). This exact model representation is expressed as two dimensions in the RDF graph in Figure 8-16. Further stratification of the RDFS graph shows the

underlying structure of the taxonomy in Figure 8-17 with the classes aggregated unto their respective super-classes, families and hierarchies. This analogy confirms the efficacy of the overall transformation process of the model to ontology.

8.1.9 Querying the UME- LLS Ontology for Traceability

Querying using Reasoners

The OWL-DL based application provides the Reasoner which ruminates information about inferences and rule processes to potentially provide answers to the questions such as "Why is there an artefact member in this class?" or a domain related question such as "Why is there need to disclose an additional information for an artefact? As OWL axiomatization is used to define the class membership and class derivations based on restrictions of other classes, this approach is applied in the case study to transform the static model to rules for classification of elements in the ontology. The outcomes are queries that are expressive with variables that occur within the validation Class expression bounded by ranges, domains and properties. This structure of query evaluation often referred to as simple entailment equally defines the simple entailment relation between RDF graphs. Thus query answering under the Reasoner ruminate answers that only follow implicitly from the queried entailment RDF graph. While several methods and implementations for SPARQL and OWL-DL under RDFS semantics are available, methods that use OWL semantics have not yet been well studied (Kollia et al., 2011). Therefore the use of Reasoners to exemplify this contribution allows entailment as a Class membership intrinsic to the model for definition of the Student viewpoint in the ontology. This makes the execution of the query for information fairly straightforward as the Module participation is all that is needed to extract the artefacts that constitute the abstract for evaluation. Figure 8-18 demonstrates this and presents a query regarding the elements and properties that exist between Composite_Specification and the BDD Validation element.

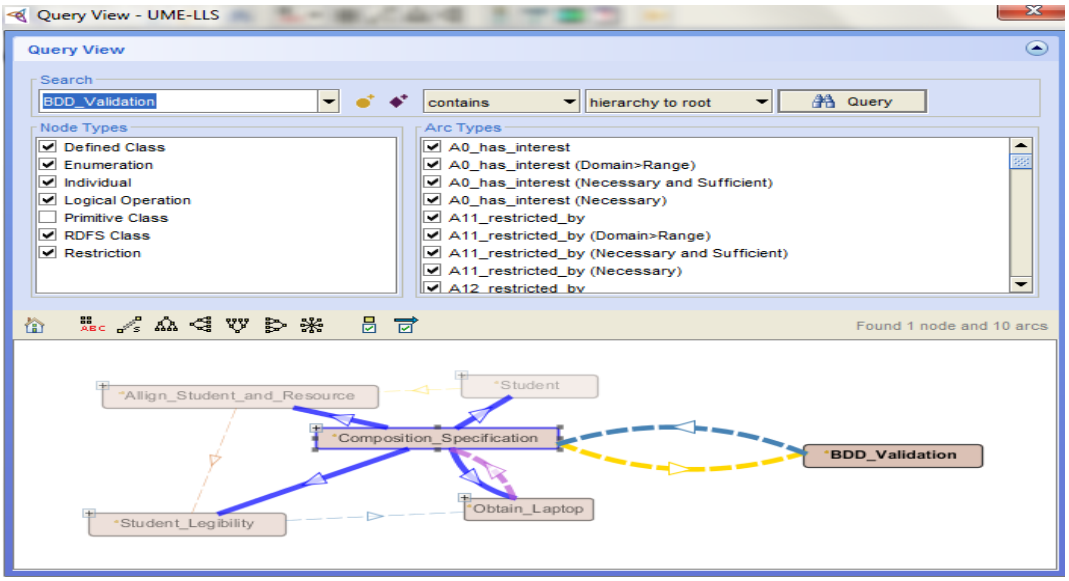

Figure 8-18: Result of Querying the UME-LLS using the OWL Reasoner

Querying using Knowledge Tree

The ontology can also be queried using the knowledge tree method. The knowledge tree offers an epistemological methodology that reveals convergence of traceability or concordance of collaboration within EA artefacts. It is based on the principle that dependencies of distinct artefacts can converge to strong conclusions of multiple coherent associations leading to a strong validation of the business behaviour. Knowledge tree exposes the interlocking of class and property theories in ontology in a coherent, holistic view of perception and offers an alternative perspective on how knowledge is obtained. The necessity for ontology transformation, annotation, validation and as a means for knowledge acquisition is uncontested. In querying using Knowledge Tree, there are criteria for stating whether the query construct can exploit the presence of ontology structure, the ancestor/descendant traversal of class/property hierarchies and filtering conditions that can be imposed on class property hierarchies. The consistent accumulation of knowledge and the emergence of dynamic and practically moderated information repositories in ontology is the basis for knowledge tree method which is a classical method for evolving the hierarchal knowledge structure of EA models (Almeida & Guizzardi 2013). The hierarchical structure of Knowledge Tree is not merely an artificial structure designed to ease the access to EA artefact traceability, it actually represents a more inherent authenticity that enables generalization from a set of details into a general rule. These simplifications allow a more transparent inference to the validation metrics defined for the model. Additionally, because it is impracticable to manually build this sort of knowledge hierarchies in EA, the need for Knowledge Tree is very important from data accumulated in the RDFS Triple stores. Figure 8-19 depicts this Knowledge Tree.

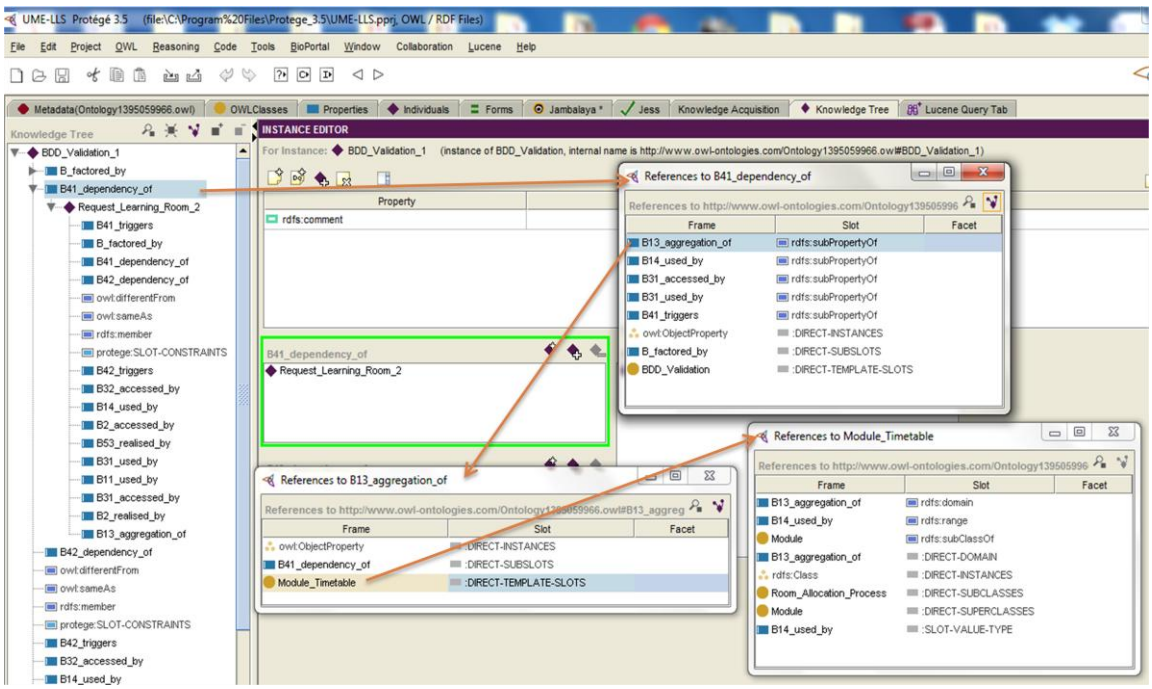

Figure 8-19: Result of Querying the UME-LLS using the Knowledge Tree

Without Knowledge Tree, it has been asserted that uncoordinated deposition of knowledge within the ontology may not be easily detected. Its traceability disposition can lead to the discovery of gaps and overlaps within the taxonomy and self-emerging hierarchies not dictated by any distinct perspective.

Figure 8-19 demonstrates this concept for the UME-LLS case study. The query of the underlying hierarchical structure of references to B41_dependency_of is extracted and linked to B13_aggregation_of within a knowledge repository of Module_Timetable class. This is then validated against the Model Traceability validation metrics to establish its Equivalent Boolean Value and to effectively deploy the result to construct a hierarchy on Goals Realization or conformity with the model's functional behaviour. The Knowledge Tree depiction in this example ultimately uses series of hierarchical position inferences of artefact to validate the model with derivative mapping of UME-LLS Properties to ontology hierarchy table. The comparison of the outcome of this validation may be benchmarked against the Metamodel as in Figure 7-9 or at able of concepts that relate function, service, event and business behaviour with subcategory of state such as in Table 8.

Knowledge tree is recognized as an important basis for reasoning and competitive advantage as many organisations are beginning to establish Knowledge Management Systems (KMS). Although the subject of Knowledge Management (KM) has been explored extensively, the understanding of how the design of the KMS affects ontology, its use and definition in EA is still rather limited. This research has demonstrated a model of the enterprise's knowledge trees related to business behaviour in an ontology transformed from EA model perspective. The key information and understanding here are in their respective capacities to provide the enterprise with greater effectiveness and shared knowledge through traceability. The result of the query benefits not only the design of EAF, but also the business model transformation and validation process.

Querying using Lucene Query principle

Lucene supports modification of queries so as to provide a wide range of searching options. Range associated queries in ontology allow the matching of classes with subclasses and attributes which are associated with the source metamodel. It provides a significance level of matching classes based on the search constraints stipulated in motivation. To boost a constraint, a Class can be nested or negated. More expansive extrapolated results are obtained with more boost factor. Boolean operators can also be used in the Lucene query to allow classes in a search to be combined through logic operations. This is achieved with the use of the *AND* for "*Match All*" and *OR* for "*Match Any*. The *OR* operator is the default conjunction operator and links multiple constraints to find a matching class if either of the constraints apply in the ontology. The *AND* operator on the other hand matches classes where both constraints exist anywhere in the RDFS of the ontology.

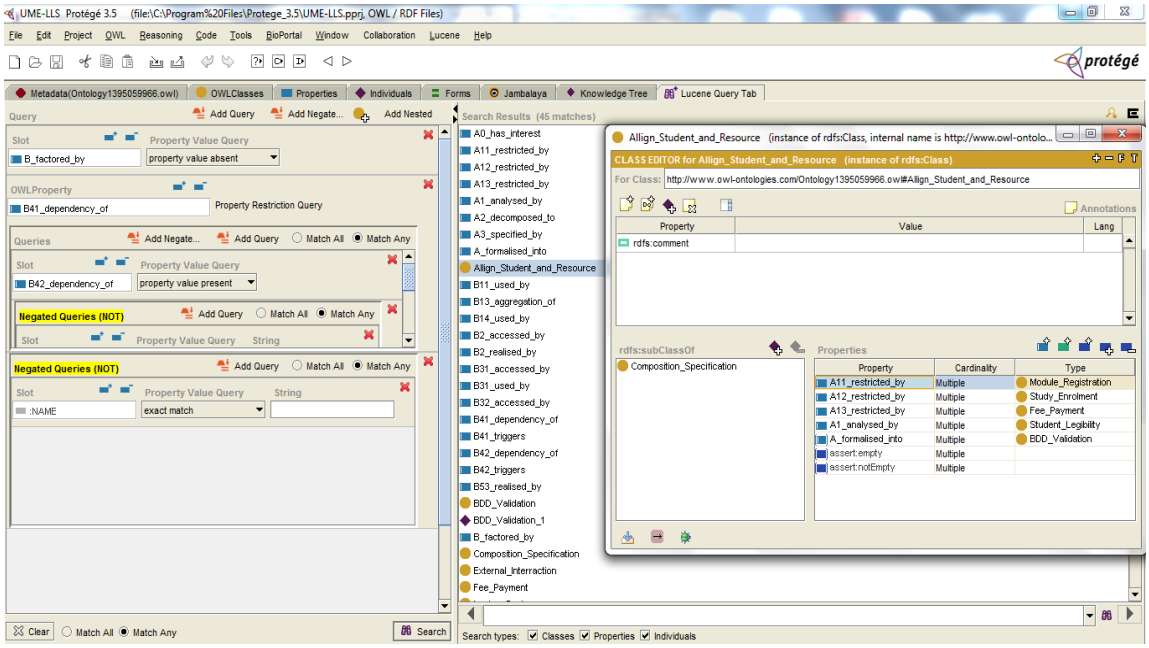

Figure 8-20: Result of Querying the UME-LLS using the Lucene Query for motivation principle

This is demonstrated in Figure 8-20 where this query pattern is used to validate the metamodel instance for Principles in the UME-LLS. In the application, the B_factored_by and B41_dependency_of slots are specified as criteria conjoined with the *OR* Boolean operator. This is negated to abstract the not empty (Availability of artefact in a model) validation constraint (Constraint Assessment). The result of the query is a list of all entities that match the search condition. Drilling deeper on the query that validates Align_Student_and_Resource, an interesting result is displayed predicating the RDFS subclass Composition_Specification where it belongs, properties which restricts it associations with the specific cardinality and ontology classes. The query also extracts the immediate sibling of the Align_Student_and_Resource class as Student_Legibility and extends the coalition as far as the ArchiMate extended core element BDD_Validation.

8.1.10 Querying the UME- LLS Ontology with SPARQL

SPARQL has many similarities with SQL querying over data models. The importance of SPARQL in the execution of queries is well known in ontologies just as SQL in relational databases. However, ordering strategies as implemented in databases is not applicable in triple stores as this would adversely ameliorate the taxonomy of the RDFS. Many practitioners of ontology artifice have focused on the domain semantic model to represent artefacts using the Description Logic (DL). However, in the context of this study, inferences to artefacts are made based on the data-triplets of the RDFS. The rationale for this is that though the SPARQL queries are represented as OWL-DL axioms over the base set, the class membership of an OWL-DL that defines a class is provided by the SPARQL query. Thus the key difference in existing technique which constitutes a contribution of this design science research is that the OWL axiom set which is a property of the state of the model is extended by the

SPARQL query to substantiate the validation result on execution of the query. This ensures optimal use for the SPARQL for straightforward retrieval of artefacts based on a prior known traversal path through the model. This attribute also makes SPARQL very suitable for validation of motivation.

SPARQL Query #1

The following example demonstrates the use of SPARQL query to identify a subset of objects and subjects within the RDFS domain ordered by subject. The result is shown in Figure 8-21.

```
SELECT ?object ?subject
WHERE { ?subject rdfs:domain ?object }
ORDER BY ?subject
```

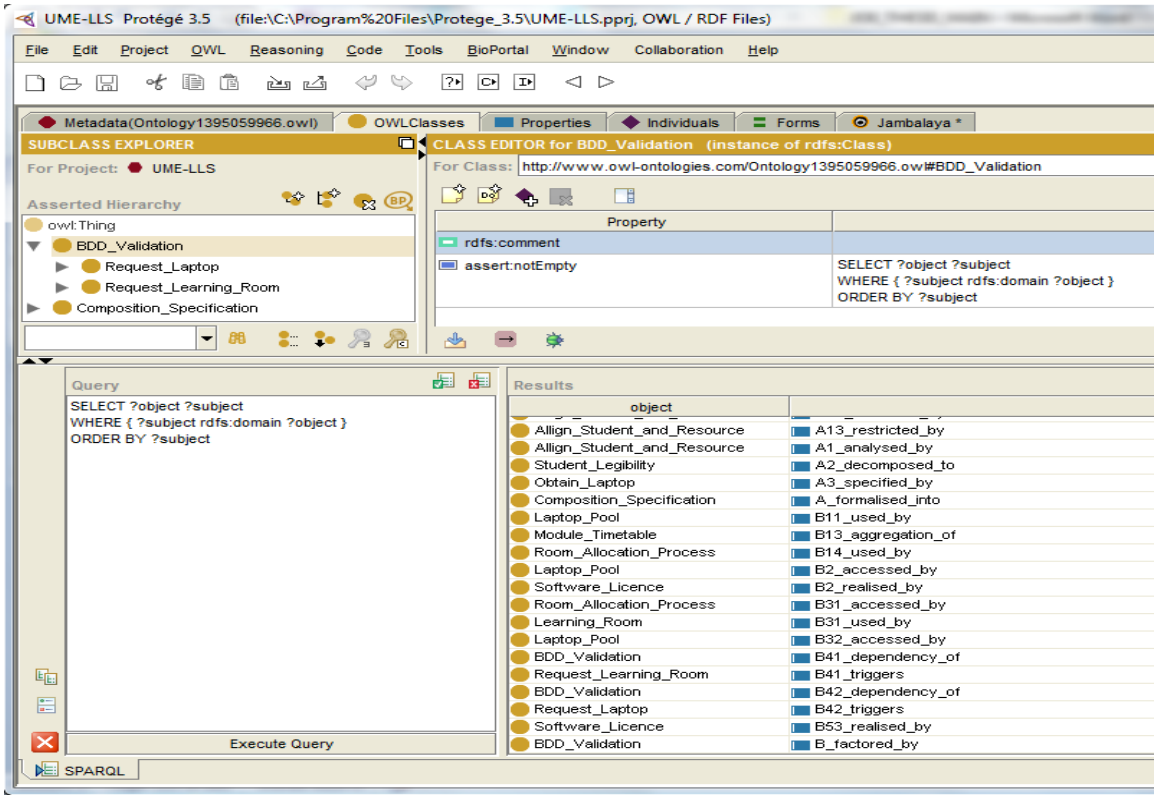

Figure 8-21: Result of Querying the UME-LLS using SPARQL - Class and Slot associations

SPARQL Query #2

The following example demonstrates the use of SPARQL query to identify a subset of subjects and objects within the subPropertyOf predicate. The result is shown in Figure 8-22.

```
SELECT ?subject ?object
WHERE { ?subject rdfs:subPropertyOf ?object }
ORDER BY ?object
```

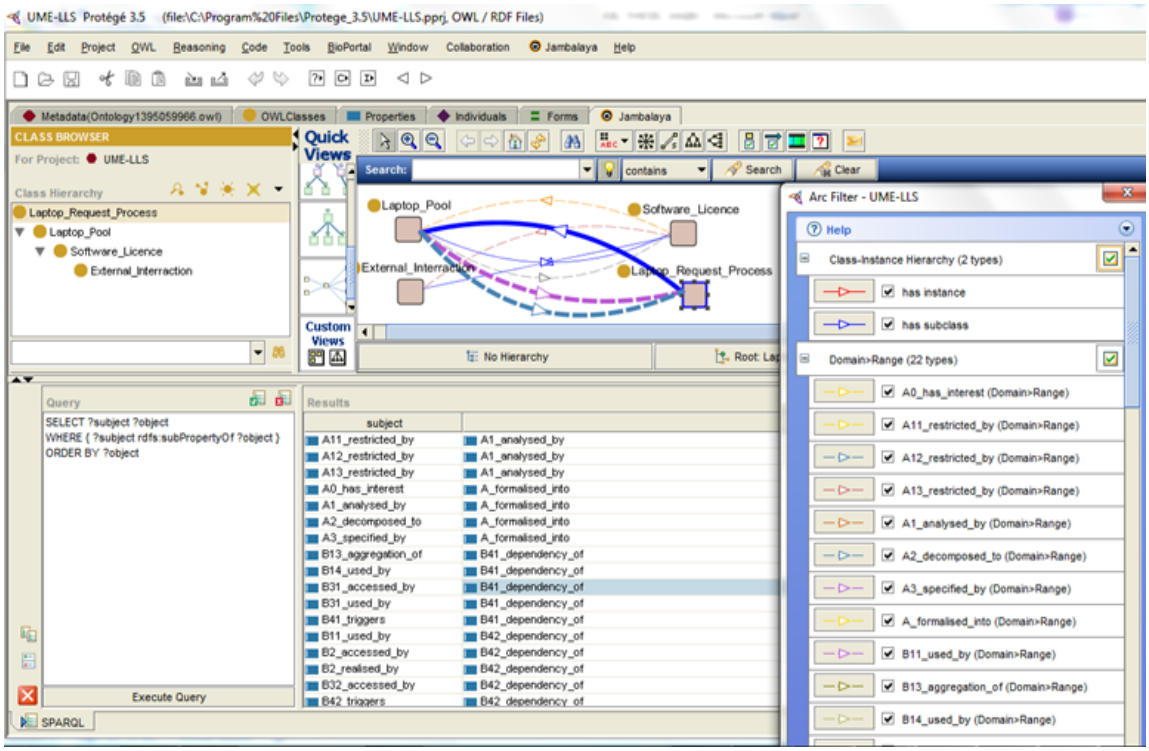

Figure 8-22: Result of Querying the UME-LLS using SPARQL for Class and Superclass association

SPARQL Query #3

The SPARQL approach for validation of model through ontology schematization aims to demonstrate that when axiomatization requires the depth of the inference needed in the solution to be deepened or strengthened as in the case of complex models, SPARQL has been found to be more suitable as a comprehensive tread for probing into the archives of RDF triple data. This contrasts with the OWL-DL axiom filters which are mostly appropriate for preliminary or cursory validation. This is demonstrated in the next examples. The query example in Figure 8-22 is extended to validate the Availability of artefacts using the assert.notEmpty clause. The result is shown in Figure 8-23.

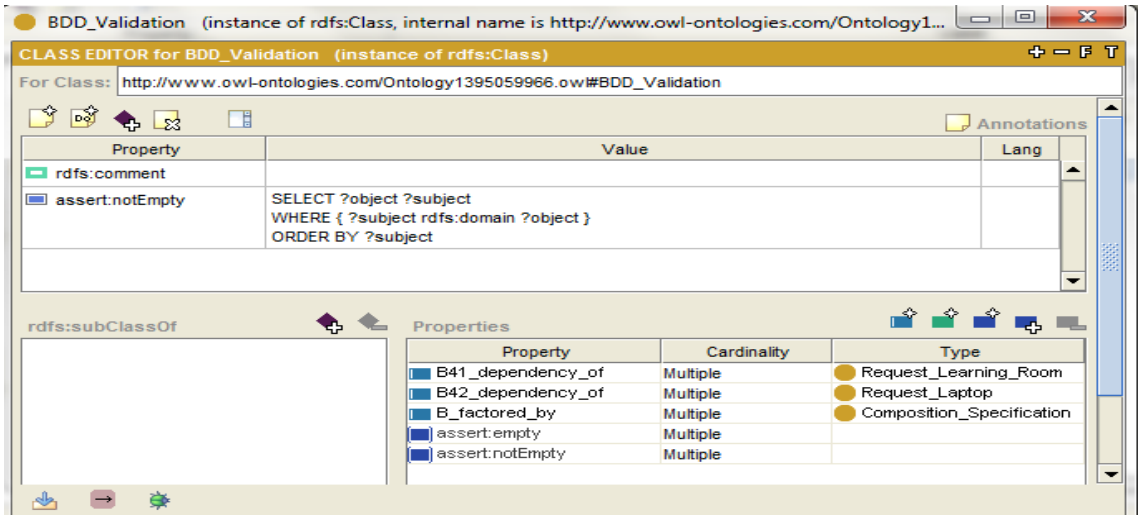

Figure 8-23: Result of Query to assert that Artefact is Available using the assert:notEmpty query

SPARQL Query #4:

This is a specific query which finds all artefacts in the UME-LLS ontology that have association with Laptop_Pool and Software_Licence.

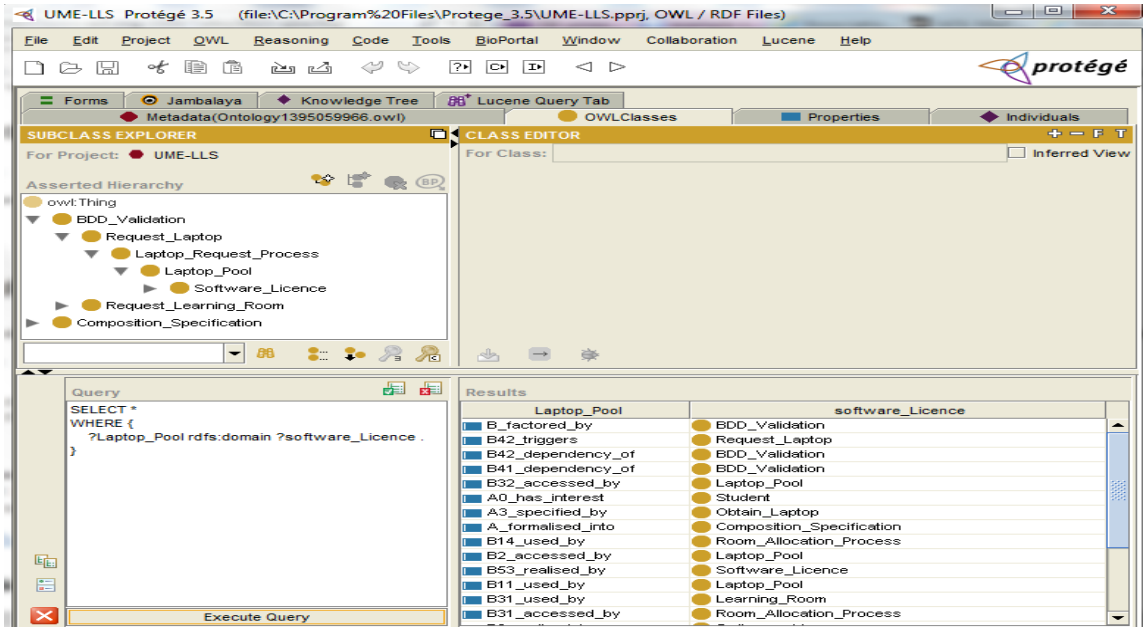

Figure 8-24: Result of Query to validate specific associations with Laptop_Pool.

SPARQL Query #5:

This query demonstrates retrieval of multiple Triple patterns for artefacts validation. The query construct and result is shown in Figure 8-25.

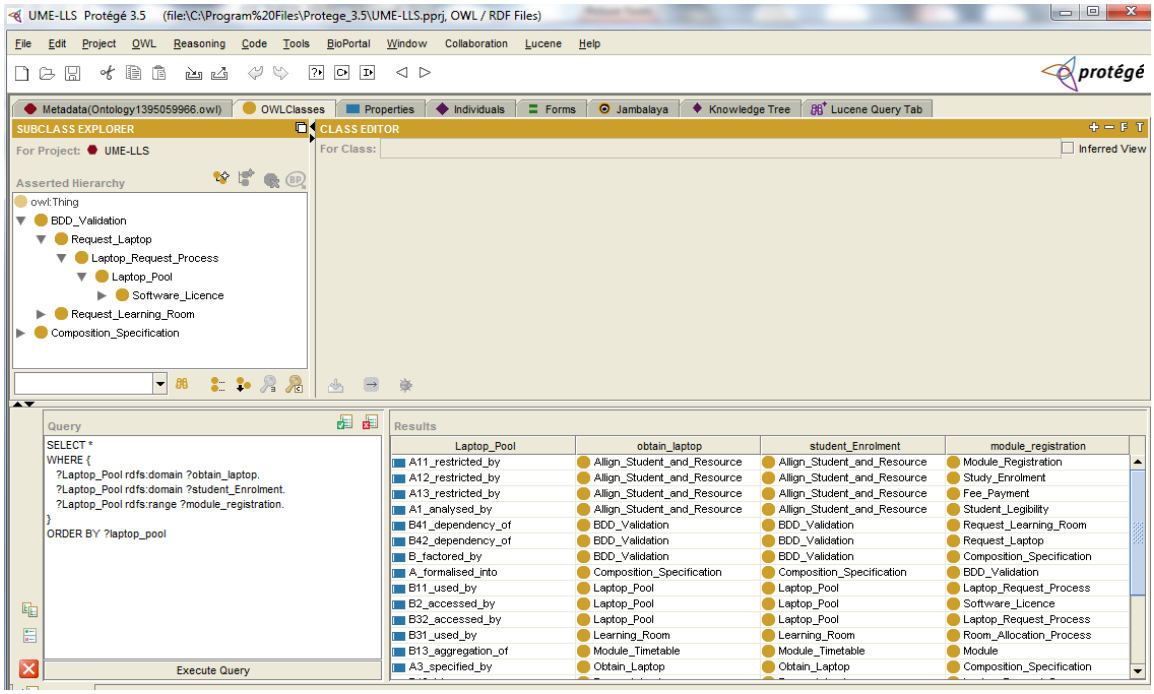

Figure 8-25: Result of Query to specify multiple Triple Patterns for artefacts validation

8.2 Case Study B: University of West London Student Internship Project (UWL-SIP)

Graduate employers, particularly in competitive fields, increasingly view relevant work experience as a must, with employers seeing internship schemes as a filter to identify new employees. Students see an internship as a valuable life experience that can lead to improved academic achievement and a way of making career contacts. Policy development around increased student fees and the sharpening of the student employability agenda requires institutions to respond by demonstrating that they have excellent resources and processes in place to support students in securing graduate employment. The UWL sought to extend this provision to:

- Enhance students' experience,
- Engage students in developing key employment skills and;
- Enhance employer engagement and extend student's career support.

The project aims to enhance student employability and to build better links between the university, businesses and community projects. Some areas of the university already provide an integrated work experience component such as Nursing. However the majority of the schools have no such provision. Building on this background, the UWL planned to implement internship or work experience provision at a school level. In order to implement this, policy internship support processes that were previously used in the graduate internship scheme needs to be adapted and implemented at a school level for the undergraduate cohort. The key stakeholders in this project are students, businesses/community, employment intermediaries and the university (schools, marketing, career/employment service, etc). To develop a prototype scheme, the School of Computing and Technology (SOCAT) is used. Among other benefits, the project is expected to:

- Take an Enterprise Architecture approach and use ArchiMate as the modelling tool.
- Maximise student satisfaction and make a significant contribution to their readiness for employability.
- Increase understanding of how EA can help in the implementation of transformation programmes.
- Once the contact pool has been built with regards to businesses and interns, businesses and UWL should be able to assign interns to in-house projects; not only to complement their studies but also to provide possible employment prospects upon graduation.
- To generate greater business prospects with regards to building better relationships with businesses and UWL students.

The EA modelling is planned to concentrate on the goal, information systems, processes and services modelling. The project will adopt an Enterprise Architecture approach to review the graduate

internship programme, analyse the school/course internship programme stakeholders' requirements and review of the institution technologies and applications. Using EA, the project is aimed to investigate the institution existing systems, with the focus on policy and strategic context including employer engagement, external and internal partner provision and system for student placements. The objective is to build the as-is and to-be architectures with focus on internship provision and also to propose a solution model for the programme using an EA approach. This will include the different viewpoints; i.e. the business processes, the information models, the interaction of the institution with external systems and development of services and portals.

8.2.1 Ascertaining the Values of the UWL-SIP Case Proposition

The preliminary analysis for this project is carried out based on the VPEC-T framework in order to prevent loss in translation from business needs to IT solutions. Due to the broadness of this project, the case study is scoped within the context of this research by identifying and designing the various models which the project intends to implement. This is subsequently transformed to ontology to allow validation of the models against its intrinsic motivation. In essences modelling of the as-is aspect of this project is nominal in this research as validation is not strictly from as-is to to-be but from motivation with the to-be architecture. However, to effectively place in perspective the context of the requirements, investigation of as-is is carried out to ascertain the specifics of current practises and desirables. This is reflected in the values and policies presented in the V-PECT analysis below.

Table 6: Table of VPEC-T Values

Values	
As-is	To-be
No guarantee of placement within entire period of study.	Guarantee of placement within three years of study.
Where placement is necessary, resources not measured.	Minimum resources for the placement management
Employer engagement not maintained.	Maintain employer engagement.
Students need to find their placement.	Automate matching of students with opportunities.
Employability through placement not considered.	Enhance Student Employability through placement.

Table 7: Table of VPEC-T Policies

Policies	
As-is	To-be
No strict requirement for minimum period of placement	Minimum of two weeks internship
Student Feedback at end of placement not mandatory	Student Feedback at end of placement mandatory
Placement not necessarily IT Related	Placement has to be IT Related (SOCAT as exemplar)
No requirement for passing a mandatory IT module	Must pass (Professional IT Level 5) PIT5 course
Businesses are not able to assign interns to in-house projects.	Businesses and UWL should be able to assign interns to in-house projects
No requirement to be met by the employer	Employer to meet minimum requirement set by UWL
No feedback required from Employer	Regular feedback from Employer
No contract with Employer regarding placement	Contract with Employer regarding Internship
No guidelines regarding placement and commitments	Guidelines Regarding Internship and commitments

Events:

Several event filters are identified for the UWL-SIP. These form the business triggers for actions that result in the actualization of one or more business objectives. The VPEC-T framework requires the prioritization and sequencing of these triggers and establishment of relationship between them. The following events (though not exclusive) which can trigger some actions are identified in this project.

- Student is at the end of course year but has not undertaken any internship.
- Employer offers a new opportunity.
- An opportunity has reached a deadline for application (or days to closure of application).
- Employer did not select any student after the selection deadline.

Content:

Content consist of information in any form that is required for the project as well as data held formally in databases and other structured interfaces. The content for the UWL-SIP is classified into four categories namely documents, messages, database repository, Interviews/discussions. Content may be related to pre-requisite information, required for processes and acted upon to result in an event or information generated from an event occurrence.

Documents

The following are identified as documents in the UWL-SIP

- Semester timetable and course outline
- Student enrolment consisting of personal details and profile
- Student skill sets and curriculum vitae
- PIT5 module and examination results
- Student internship monitoring report
- Student's year of study and internship participation statistics

Messages:

The following are identified as messages or notifications in the UWL-SIP

- Student requests for placement
- Available placements for students to apply
- Student applications submission
- Alerts and messages for employer feedback and report
- Alerts and messages for student feedback and report
- UWL_SIP authorisation or approval for placement

- Acceptance of student for placement
- Internship opportunity deadline notification
- Employer acceptance or rejection of student for internship
- Employer advertisement, withdrawal or republishing of internship opportunity

Data Repository:

- Employer profile, internship listing and job description
- Search-find-match: 'keywords search', (postcode, location), skill set, duration, industry
- Student records, Curriculum Vitae and Skill set
- Opportunity being filled, opportunity deleted or available
- New Employer engagement records

Trust:

In the case of the UWL-SIP, Trust is not considered as a deliverable since it is unlikely to be listed as a concern unless it appears as a threat to the success of the project.

Sequel to this analysis, the following EA models has been identified for modelling;

Motivation models

- Motivation Extension for Goal, Constraint and Requirement
- Enterprise Goals and Constraints Model
- Motivation Extension from Student viewpoint.
- Motivation Extension from Career Support viewpoint.
- Motivation Extension from Employer Viewpoint.

Process models

- Process for Internship Application for the UWL-SIP
- Process for Internship Matching for the UWL-SIP
- Process for Placement Monitoring and Feedback for the UWL-SIP

Business Models

- Organizational model
- Information Model
- Functional and Service model

These models depict the development taxonomies for the management of the internship opportunities with an objective to automate the process of matching students with internship opportunities. It also depicts the roles and responsibilities for the career office /school in maintaining complete control over the process.

8.2.2 Motivation Models for the UWL-SIP

The specification stipulates that students will manage their CV, search for internship listings, request and apply for internship and provide feedback once the internship has taken place. Similarly, the internship providers are required to provide information about their internship opportunities to the UWL, track progress on internship listing, partake in the assessment of applications and provide feedback on student internships for the duration and at completion of the internship. The Career Support plays administrative roles in guiding the success of the projects. Figure 8-26 depicts this modelling of the goals as described and associated constraints.

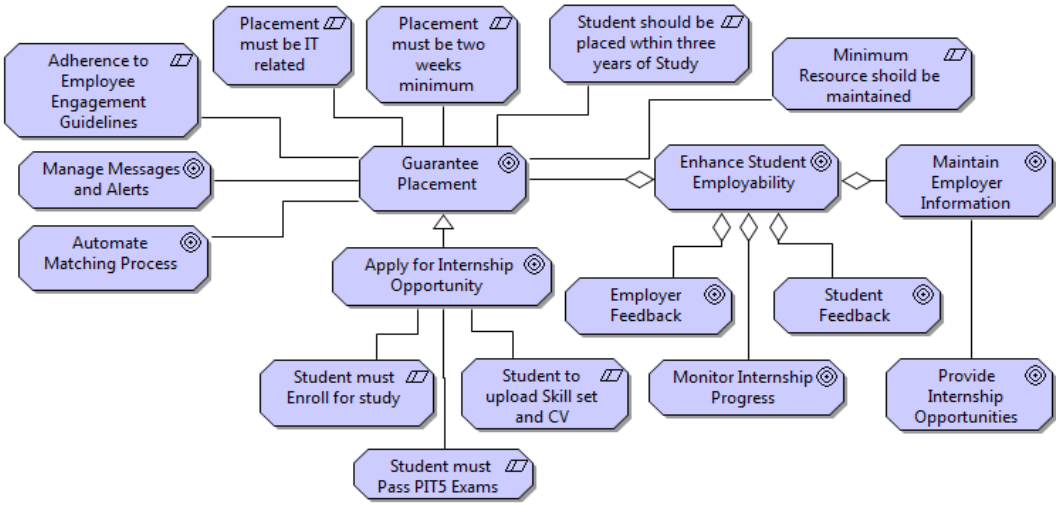

Figure 8-26: Business Goals and Constraints Model

The Business Goals and Constraints model consists of constraints grounded with the principles and requirements that bind their association. This binding establishes congruency and ensures that validation criteria can be developed. The principal goal in the modelling of motivation is representation as "Enhance Student Employability". For this to be achieved five sub-goals are aggregated to it. These five sub-goals have constraints in some cases directly or indirectly through their specific tangential sub-goals; with the constraints assessed and resolved through their associated Requirements.

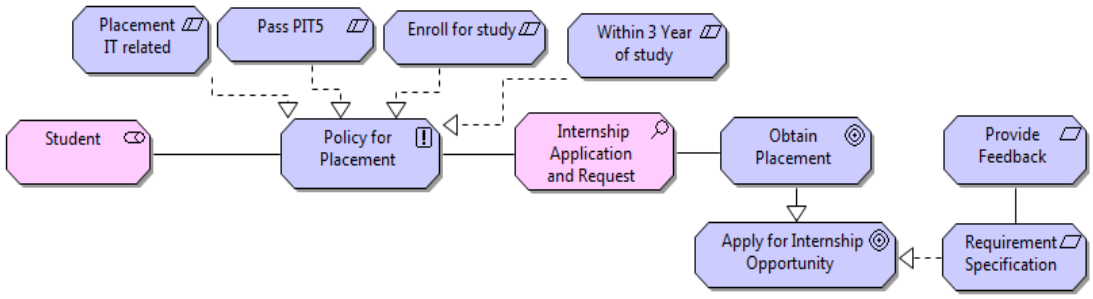

Figure 8-27: Goal and Constraint model for the UWL-SIP from Student viewpoint

The goals that have been modelled express the relationships between assessment of the constraints and the requirements to be validated. This is required in order to enable rationalization of the business processes as well as measurement the model's maturity.

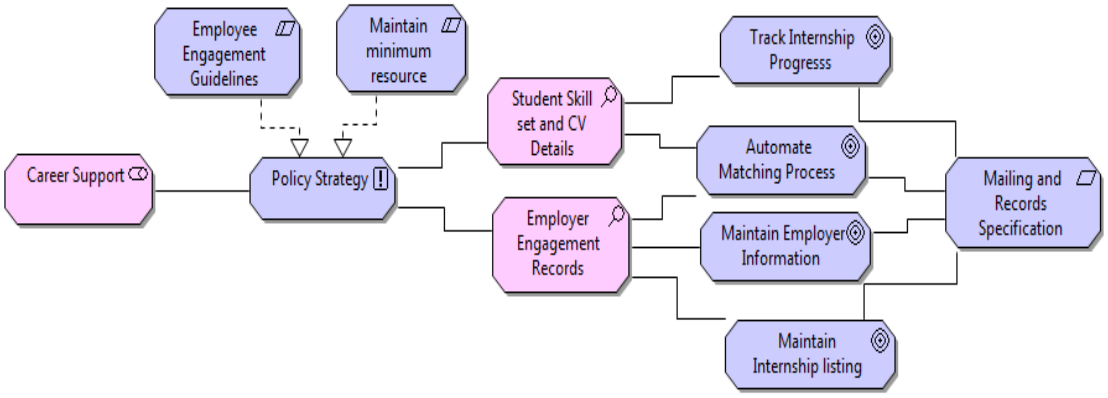

Figure 8-28: Goal and Constraint model for the UWL-SIP from Career Support viewpoint

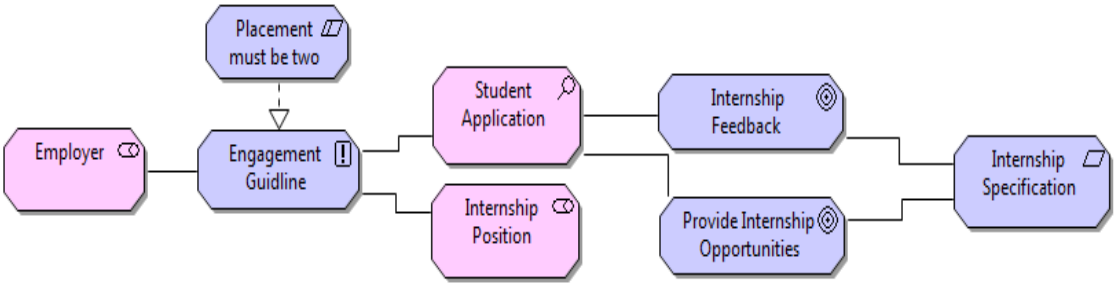

Figure 8-29: Goal and Constraint model for the UWL-SIP from Employer Viewpoint

Thus the classification of motivational components required to achieve the goals specified in the UWE-SIP have been decomposed into three auxiliary models from the perspectives of Student, Career Support and Employer perspectives using the proposition deduced in Table 6 and Table 7. Specifically, the business role here for the student is to apply for internship opportunity, obtain placement and provide feedback (Figure 8-27); the Career Support to track internship progress, automate the matching process, maintain employer information, maintain internship listing as well as manage messages and alerts (Figure 8-28); and the employer to provide and maintain the available internship opportunities as well as send feedback (Figure 8-29). The three models presented in Figure 8-27, Figure 8-28 and Figure 8-29 respectively disintegrate effectively the goals of the proposition into small realizable sub-goals revealing divergent interests between the stakeholders, possible conflicts and overlaps in functionality. The parallelism of the sub-goals with stakeholders is fundamental in the model as it reveals the essential requirements needed to bridge the gaps that may exist in assignment of responsibilities. The decomposition enables the comprehensiveness of requirement to be ascertained. As specific goals are also distinctively connected to their constraints, it ameliorates a more thorough validation exertion.

8.2.3 Process Models for the UWL-SIP

Though many process models can be deduced from this project, three distinctive archetypes are distinguished as they effectively cover the perspectives modelled with the motivational goals in section 8.2.2. The archetypes traverse most of the related events and elements that occur within the UWL-SIP taking into consideration different viewpoints of the three identified stakeholders. The driving principles in the design of these Process models take into cognizance the need to embed predicate logic rules and behaviour patterns that can lead to the actualization of the goals specified in motivation. While the predicate logic rules provide an account of inclusive quantifiers that would be developed to express a wide set of arguments for validation, the behaviour patterns establish an explicit link between artefacts at the business level of abstraction of ArchiMate Business Layer and the requirements that the model needs to fulfil. Thus, the design of these process models conform strictly to the theoretical principles for model validation rules discussed in section 5.2.

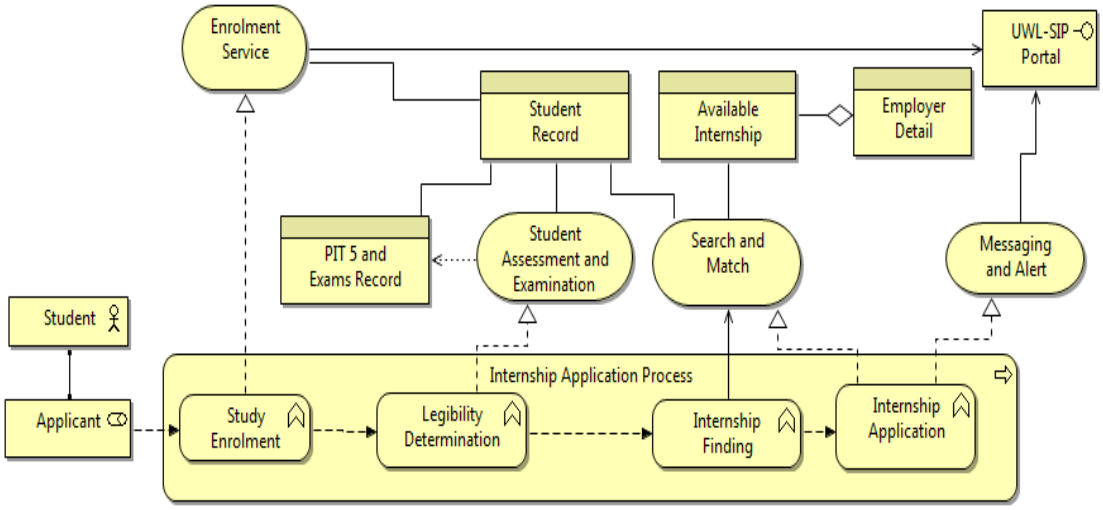

Figure 8-30: Modelling of the Process for Internship Application for the UWL-SIP

The modelling of the process for Internship Application for the UWL-SIP is depicted in Figure 8-30. This process is carried out by the student in the business role of an applicant and involves the execution of the Study Enrolment function. The Study Enrolment function uses the Enrolment Service with access to the Student Records object. Model process flow continues to Legibility Determination function which again uses the Student Assessment and Examination service to determine the suitability of the student with access provided for both the Exam Records and the Student Records objects. The process flow extends to Internship Finding function which in turn triggers the Search and Match Service to match available Internship with student. Subsequently, process flow returns to Internship Application function and uses the results of the Search and Match Service to post the application with a notification through the Messaging and Alert service to the UWL-SIP portal interface that the application has been submitted.

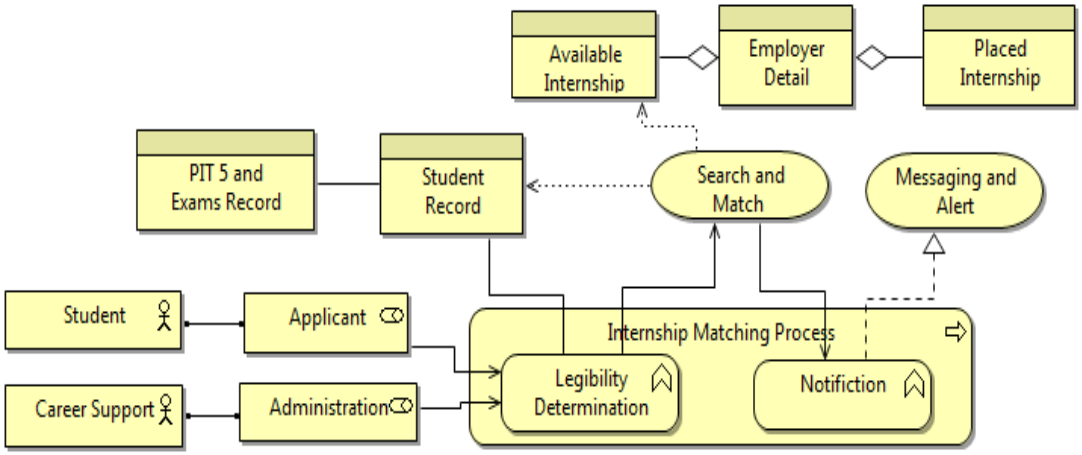

Figure 8-31: Modelling of the Process for Internship Matching for the UWL-SIP

Figure 8-31 depicts a dual scenario where the process for internship matching can be initiated by either the Student or the Career Support. The Legibility Determination function is performed with access to the Student Record and Exams Record business objects to determine whether the student is legible to participate in the internship program. The Internship Marching process on determination of the student's legibility uses the Search and Match service with access to Student Record and Available Internship data objects to execute the match. Afterwards, a notification is sent through the Messaging and Alert Service stating the outcome of the search.

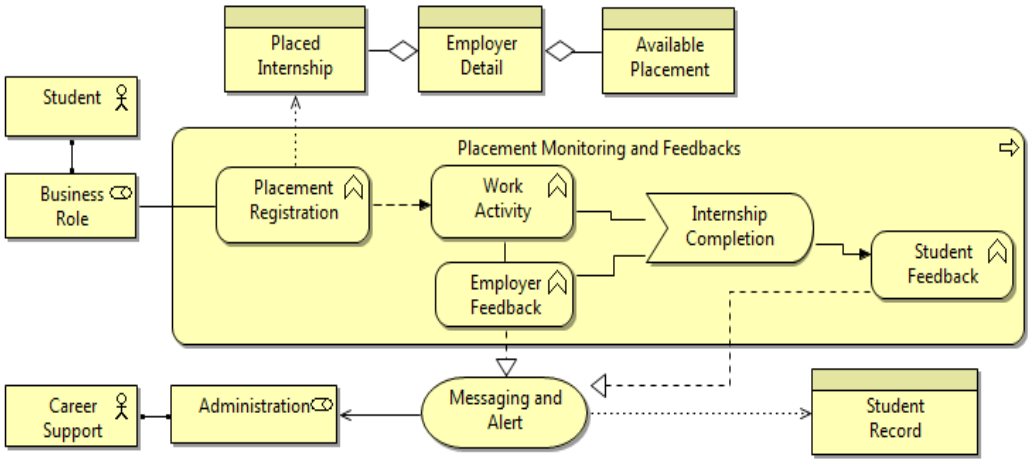

Figure 8-32: Modelling of the Process for Placement Monitoring and Feedbacks

The Process for placement monitoring and feedback can also be initiated by both the Student and Career Support. This process model can only be executed after due registration function and placement of student has been completed. During the Work Activity function, the Career Support through the Messaging and Alert service may prompt the Employer for feedback. However, on completion of the Internship, the Internship Completion Event triggers a final feedback from the student through the Messaging and Alert Service. This event is a notification that occurs in response to the change in state of the student subsequently, the student record is updated.

8.2.4 Business Models for the UWL-SIP

This group of models are strict instantiation from the three aspects (Information, Behaviour and Structure) of the Archimate metamodel. The Information model depicted in Figure 8-33 describes data structures, values defined in the enterprise policies and their meaning in relation to each other and the interoperability provide for the UWL-SIP. The objective of the Information Model is to define a standardized set of structures used to exchange data between artefacts in the EAF. These structures provide the basis for formalization of data bindings, allowing the EA implementation to create a congruent mapping and transitional processes from the model to ontology. Thus the scope of the Information model specification is focused on defining interoperability between elements of the motivation extension and the core ArchiMate Business layer of the EAF.

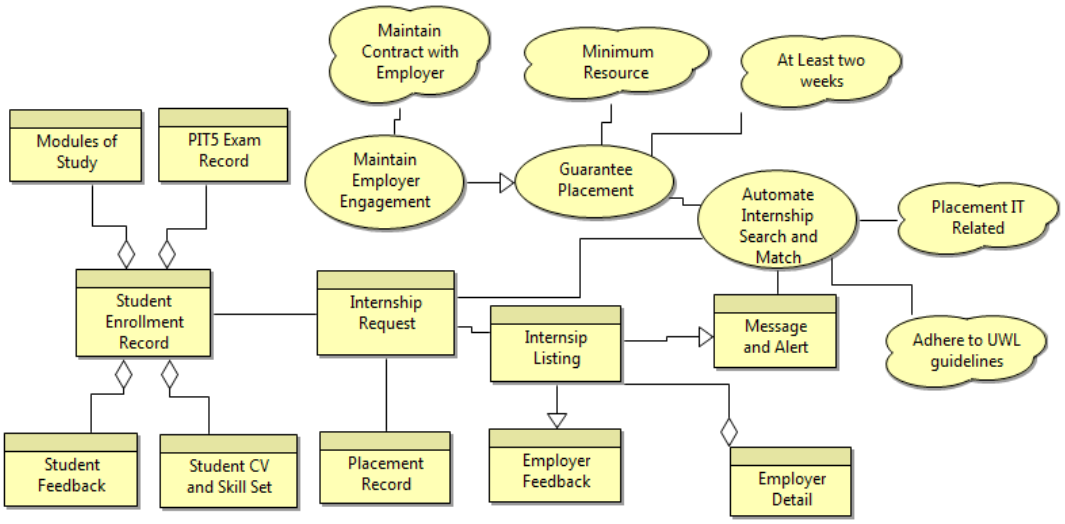

Figure 8-33: Information model for the UWL-SIP to-be from Business Perspective

Central to this information Model is the value Maintain Employer Engagement specialized by Guarantee Placement. To achieve this, there is an association to the value that allows the automation of Internship Search and Match. Various data objects support this paradigm.

The function and service model presented in Figure 8-34 describes the behavioural specifications of the components of the model. The model typically describes what is needed to be accomplished by the stakeholders as well as requested properties of functions and associated services. It deploys the output of the requirement analysis defined by the motivation extension to describe precisely the essential model elements and their relationship to processes. This ensures that the validation to ascertain if specified motivational goals are met can be realized. Annotation of the function and service elements help to avoid overlaps and inconsistencies; allows for accurate extradition of unnecessary redundancies and optimization of resources.

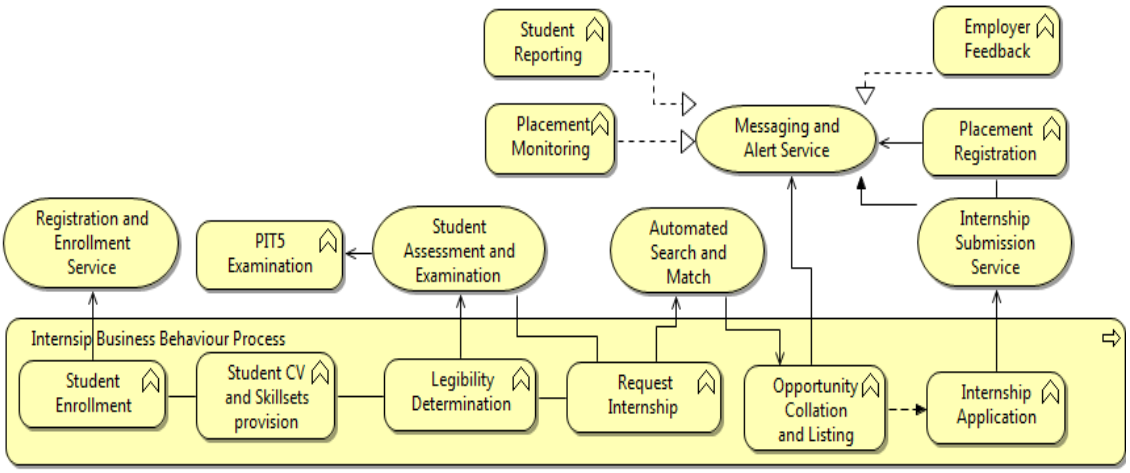

Figure 8-34: Function and Service model for the UWL-SIP from Business Perspective

It also serves as a unique valuable reference for model mapping and transformation, provides documentation of configuration, and allow for consistent communication among the various stakeholders. The function and service model in Figure 8-34 provides a precise idea of the motivation to be achieved such that the model design can substantially be validated.

The organizational model represented in Figure 8-35 defines the structure of the EAF including all the stakeholders identified and associations with each other. It enmeshes a number of elements which identify units, interactions and collaborations that cohere mutually to accomplish the primary goal and sub-goals with specific outcomes.

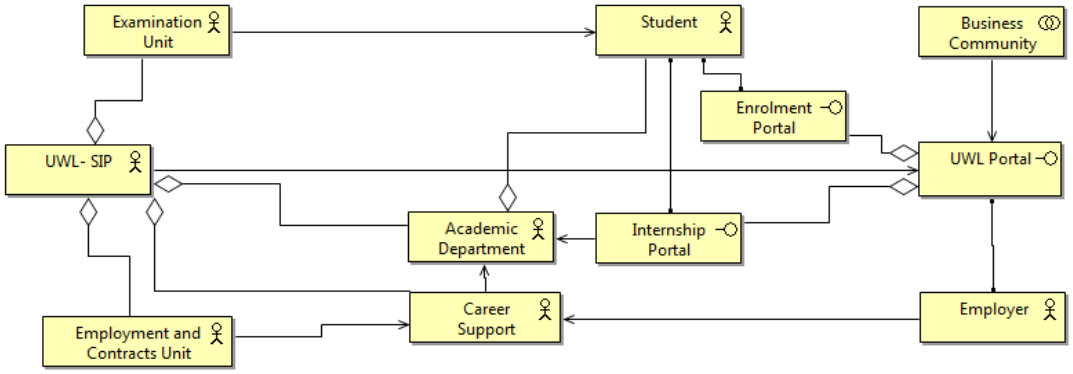

Figure 8-35: Organisational model for the UWL-SIP to-be from Business Perspective

The model also incorporates multiple physical locations internal to the UWL-SIP such as units, department and portals and external locations such as the Internship Provider (Employer). The conjugating artefact is the UWL Portal interface where Student enrols and applies for internship; Internship is listed; Search and Match is carried out by Student and Career Support; Employer manages Internship listing; messages, alerts are generated and feedback routed. Figure 8-36 depicts also the stakeholder viewpoint instantiated from the motivation extension of the Archimate metamodel.

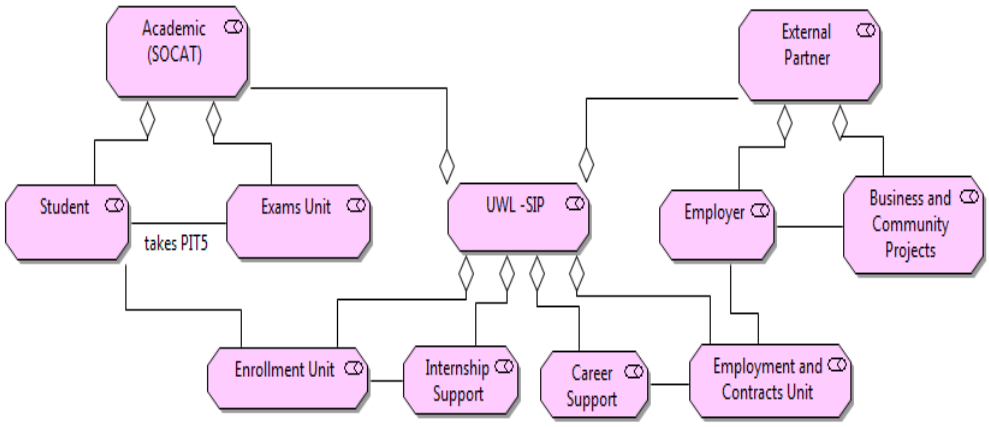

Figure 8-36: Stakeholder Viewpoint of the UWL-SIP

8.2.5 Extending Business Process with Validation Element and Motivation

In order to present a distensible taxonomy that spans motivation, information, business and structural aspects of the UWL-SIP, the model in Figure 8-37 is conceptualized. This model focuses on the intrinsic interest of the Student and is incorporated with the Validation Element to allow substantiation of related goals. The UWL-SIP analysed using the VPEC-T concepts shows an implementation of the logical model with the Business Validation artefact and properties. For coherent traceability, elements at the business layer are annotated with unique identifiers (UID) using the same consistent exegesis of the UML metamodel in Figure 7-9. The affixation is shown in Figure 8-38.

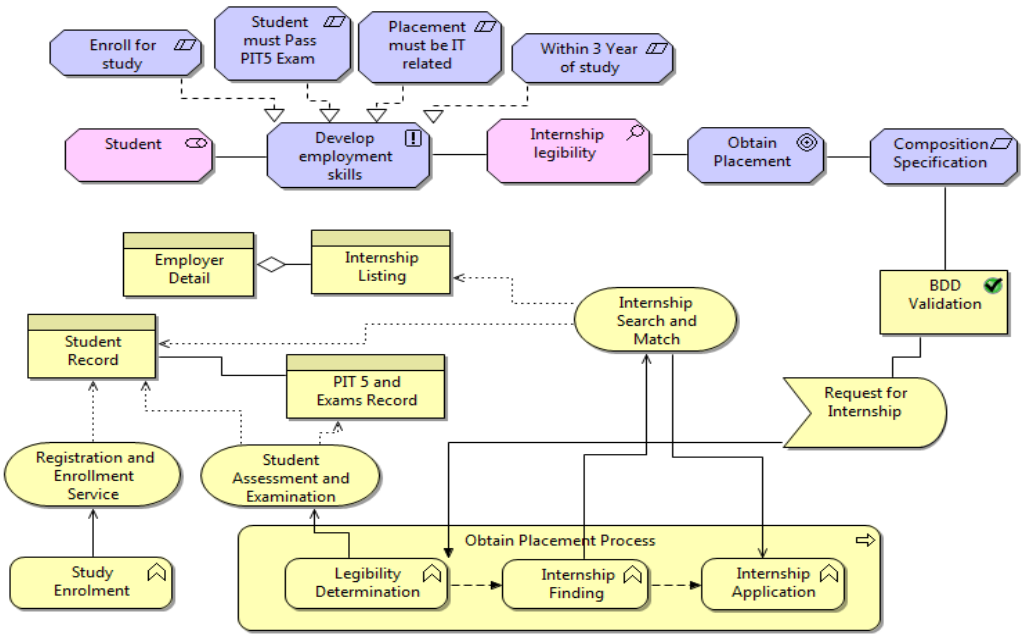

Figure 8-37: Business Layer Model for the UWL-SIP to-be with Extended VEM

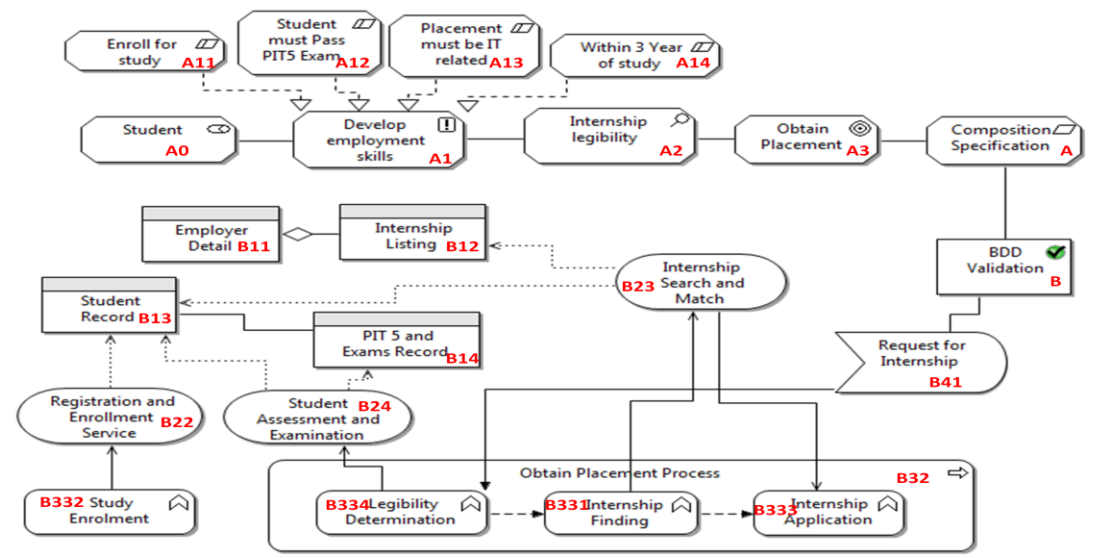

Figure 8-38: Annotating the Business Layer Model of the UWL-SIP with VEM

Figure 8-38 is an amalgamation of related ontogeny for business behaviour from the Student viewpoint of the UWL-SIP models presented so far. This taxonomy incorporates also constraints specified in the motivation and required for the accomplishment of goals peculiar to the student's concerns. Related functions, business objects, processes and services are incorporated in the derivation. The BDD Validation is embedded into this model to indicate the theme for the validation and to interrelate goals with composite constraints. Principles of the methodology proposed in this research are adhered to in order to ground the contributions of this research and evaluate the hypothesis put forward.

8.2.6 Developing Behaviour Driven Modelling Constraints Specification for the UWL-SIP

The model in Figure 8-38 is considered for the development of the validation specification for the UWL-SIP. Some examples are extracted as follow;

Assumptions:

There are 20 Internship providers (Employers).

There are total of 30 Internship Opportunities.

There are 30 Students Legible for Internship.

The Student is enrolled to study an IT or computing related courses.

The following validation scenarios can be developed.

Feature1: Determine Legibility to apply for Internship
In order to qualify to apply for Internship;
As a student
I want to go through prerequisite assessment

Scenario1: Enrol for study in Computing or IT related subject
Given that I have a student identification number and password
When I log onto the UWL web portal
Then grant access to enrol for studying Computing or IT related subject

Scenario 2: I have Passed PIT5 Examination
Given that I have taken PIT5 examination
And I am within my third year of study
When my result indicates a pass
Then assert that I am legible to apply for internship

This feature is based on the Process for Internship Application modelled in Figure 8-30 and Process for Internship Matching modelled in Figure 8-31.

Feature 2: Internship Finding
In order to apply for Internship;
As a student applicant for internship
I want to find an Internship Opportunity

Scenario 1: Request for Internship
Given that I am legible to apply for internship
When I submit my CV and IT Skills Sets
Then search for IT Internship placements that match my skills
And list the available Internship opportunities
And allow me to select an Internship.

Scenario 2: Application for Internship Opportunity
Given that I have chosen an Internship opportunity from a list
And attach my CV to the selected choice of internship
When it is confirmed that my Internship opportunity selection is IT Related
Then submit the application
And Notify me by email that my submission has been successful

8.2.7 Mapping Artefacts of the UWL-SIP to Ontology Elements

The presentation of the UWL-SIP models illustrates salient domains, properties, constraints, forms, instances and cardinalities that can be transformed to ontology. The schema in Figure 8-38 depicts domain knowledge representation on the UWL-SIP case study and identifies the semantic categories that are involved in understanding the discourse from the perspective of Student. This rational schema which can be extrapolated from heterogeneous modelling languages maps objects of the model to annotated classes and their relationship with each other (Table 8). It addresses Information integration at the structural, syntactic and semantic levels thus present content explicitly. This provides the mechanisms for correlation between model objects and their business goals, constraints and requirements.

Table 8: Mapping of UWL-SIP Properties to Ontology Hierarchy.

Metamodel TOP- DOWN Decomposition Analysis				RDF TRIPLES			Hierarchies of Properties					D/R	Map
				Subject	Predicate	Object							
METAMODEL CLASS			Model Instance	UID	Description	UID	L1	L2	L3	L4	L5		
Composite Motivation													©
Motivation Extension	Stakeholder		Student	A0	has interest	A1		L2				A	∃≡
	Principles		Develop employment skills	A1	analyse by	A2		L2				A	∃≡
	Constraint		Enrol for study	A11	restricted by	A1			L3			A	∃≡
			Pass PIT5	A12	restricted by	A1			L3			A	∃≡
			Placement IT related	A13	restricted by	A1			L3			A	∃≡
			Placement within 3 Year of Study	A14	restricted by	A1			L3			A	∃≡
	Assessment		Internship Legibility	A2	decomposed to	A3		L2				A	∃≡
	Goal		Obtain Placement	A3	Specified by	A		L2				A	∃≡
	Requirement		Composition Specification	A	formalised into	B	L1					A	∃≡
BDD Validation Element			BDD Validation	B	factored by	A	L1					B	©
				B	dependency of	B41		L2				B	∃≡
Information Model	Business Object		Employer detail	B11	aggregated by	B12					L5	B	©
			Internship listing	B12	accessed by	B23				L4		B	∃≡
			Student record	B13	accessed by	B22		L2				B	∃≡
				B13	accessed by	B23				L4		B	∃≡
				B13	accessed by	B24					L5		∃≡
				B13	associated with	B14			L3				∃≡
			PIT 5 and Exam record	B14	accessed by	B24					L5	B	∃≡
Business Function / Process Model	Business Service		Registration and Enrolment	B22	used by	B332	L1						©
			Internships search and match	B23	used by	B331				L4		B	∃≡
			Student assessment	B24	used by	B334				L4			∃≡
	Business Behaviour												©
		Function	Internship Application	B333	used by	B23				L4		B	∃≡
				B333	flows from	B331				L5		B	∃≡
			Internship Finding	B331	flows from	B334				L5		B	∃≡
			Legibility Determination	B334	triggered by	B41			L3			B	∃≡
			Obtain Placement process	B32	consist of	B334					L5		∃≡
				B32	consist of	B331					L5		∃≡
				B32	consist of	B333					L5		∃≡
			Study Enrolment	B332	uses	B22	L1						∃≡
	Event		Request for Internship	B41	associated with	B		L2				B	∃≡
Organisation Model	Business Role												
	Interface												
	Collaboration												
	Actor												

The tabulation in Table 8 depicts the constitution and dependent relationships among objects identified in the model of the UWL-SIP. It also exemplifies goals and constraints for validation with hierarchical reasoning required for traceability and query. This sort of mapping and development of ontologies for validation of EAF is easier and more precise when compared to other methodologies (Almeida and Guizzardi, 2013). By decomposing the EAF into business behaviour of interest and retaining the different related artefacts within the perspective of the associated stakeholder, the relationships that exist between the artefacts of the model are explicitly harnessed towards the transformation to ontology. This is depicted in Figure 8-39, Figure 8-40 and Figure 8-41.

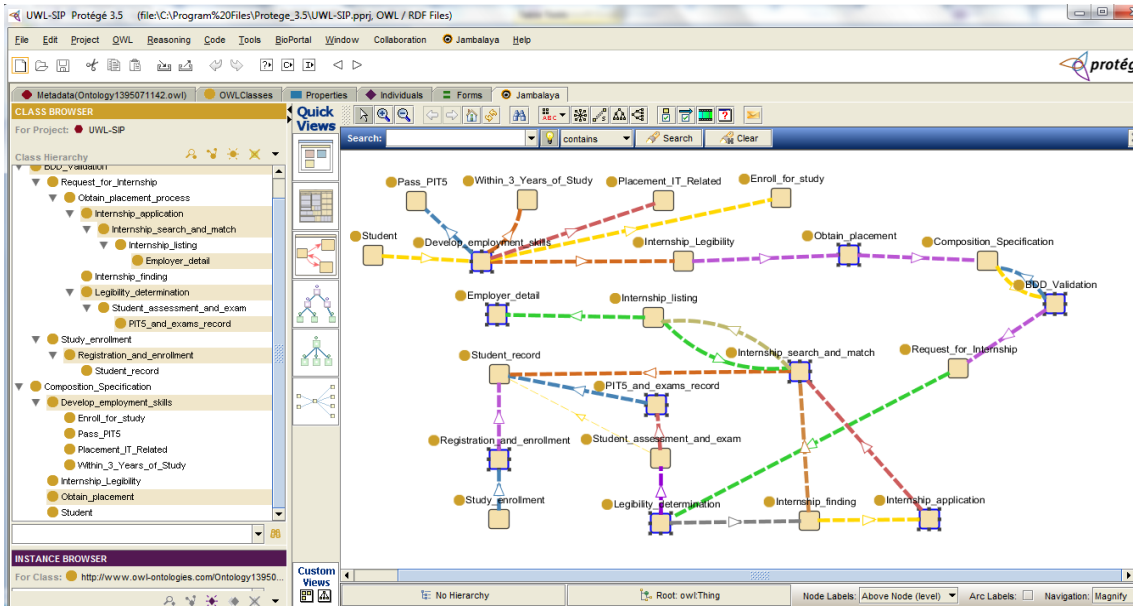

Figure 8-39: OWL implementation of UWL-SIP Classes

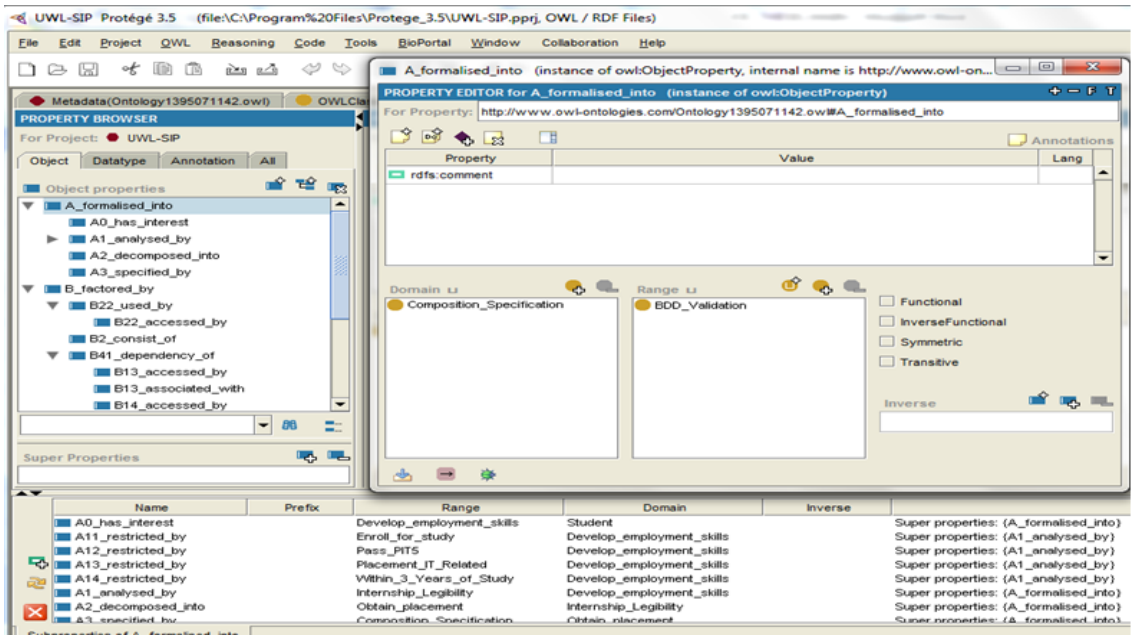

Figure 8-40: Association of properties to domains and ranges for UWL-SIP

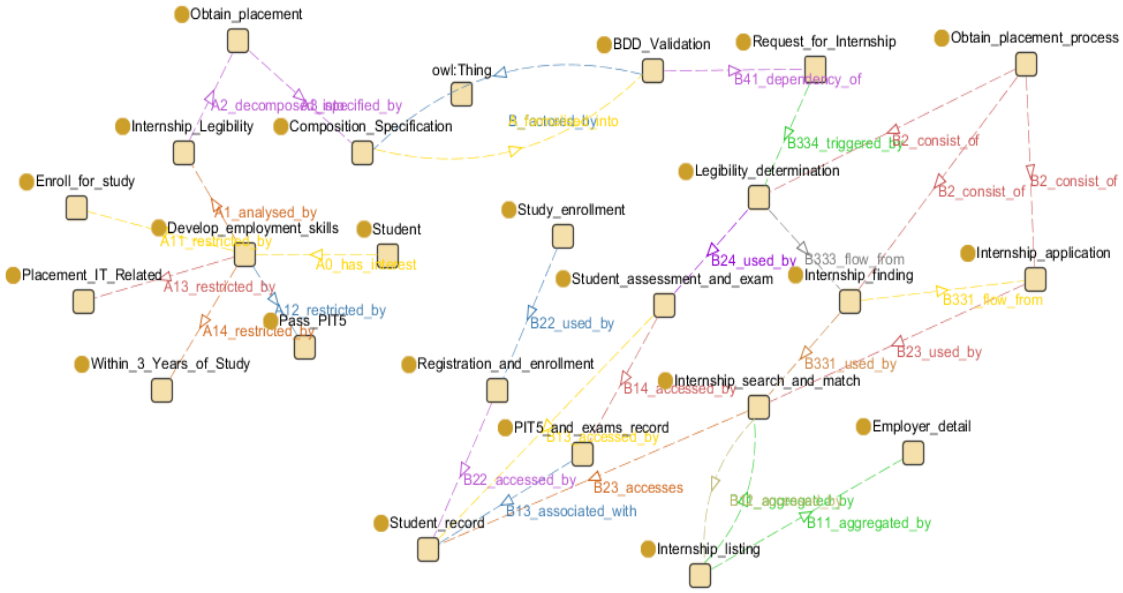

Figure 8-41: Visual Representation of the class, domain and range mappings for UWL-SIP

The UWL-SIP case study demonstrates the use of Enterprise Architecture to implement enterprise interoperability that facilitates the efficient and dynamic management of changing business environment. Focusing on the student viewpoint, the example associated divergent needs (motivation) to multifarious internship design facets. Adopting the model in Figure 8-37, a precise transformation from model to ontology is achieved in Figure 8-41 with generated RDFS (Appendix F), The RDFS in Appendix F presents a congruent pattern of semantics that can easily be validated and averts the communication problems that hinder enterprises from understanding the underlying semantics of a model, implementation of integration and collaborations. The ontology-based Enterprise Architecture model presented by this case study is composed of two levels of ordinances. The first level is the ontology of Motivation and is presented by the left cluster of artefacts in Figure 8-41 and linked through the BDD_Validation to the business layer. The second level is represented by the right cluster of artefacts. The relationships among the components are presented exactly with common notations to support validation, collaborations and ultimately interoperability.

8.2.8 Querying the UWL-SIP Ontology with OWL Reasoners

The semantic Reasoner of OWL is used to infer logical consequences from a set of asserted facts and axioms presented in the UWL-SIP. The rationale for adopting the OWL Reasoner in this cardinal validation is to apply the inherent mechanism of the language to assert whether it is possible to filter artefacts of concerns as existing in the transformed model. The inference rule which is able to progress by either forward chaining or backward chaining is also applied in this case. Forward chaining starts with the available data and uses inference criterion to extract more data as exemplified in the Internship_search_and_match example until a goal is reached while backward chaining starts with a list of goals and works backwards from the consequent to the antecedent to see if there is data available that will support any of the goals. Figure 8-42 demonstrates the use of Reasoner with

forward chaining. The Reasoner presents query view with filters (panel on the right in Figure 8-42) with inclusion of an extensive definition of criteria for properties that can be utilized for delineation of the output RDF graph. Virtually, most querying of OWL ontology carried out using the reasoner deduces implicit knowledge with accurate dependable query results which otherwise may have been ambiguous.

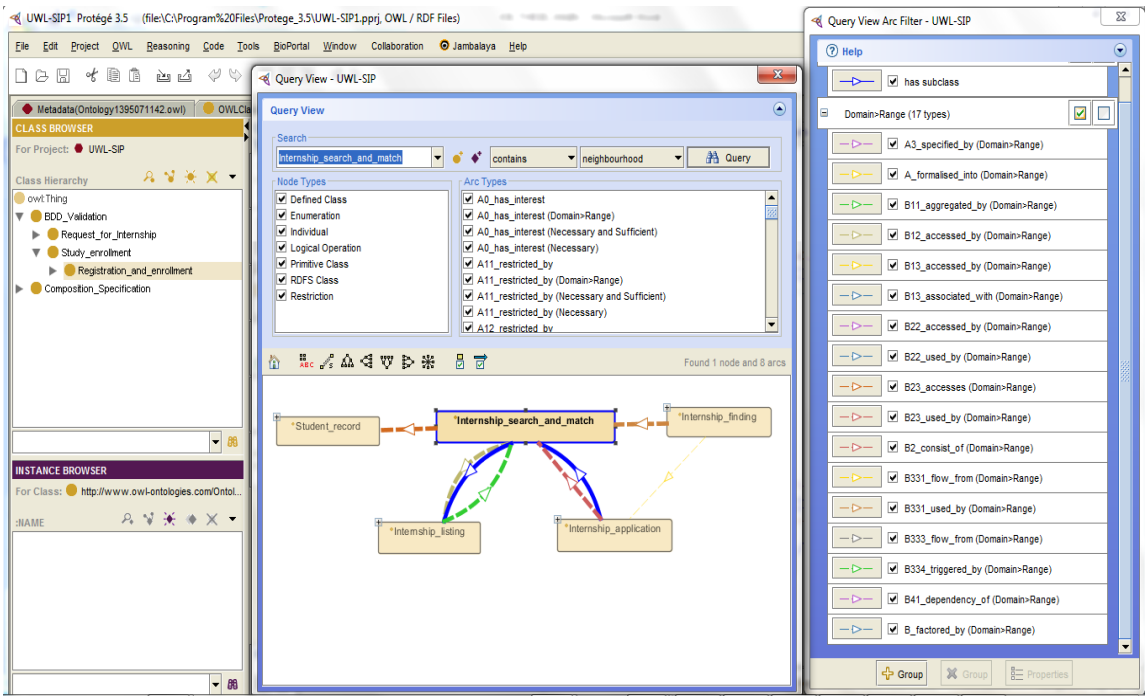

Figure 8-42: Querying the UWL-SIP using the OWL Reasoner

A number of other Reasoning methodologies have been applied for reasoning and querying the ontology. A reminiscence of these methodologies is reflected in the works of Motik et al., (2006) and includes comparisons and performances of ontology reasoning systems. Much effort has also been spent on optimizing standard reasoning tasks such as entailment checking, classification, or realization (Sirin et al., 2006; Glimm et al., 2011). The optimization of query reasoning algorithms have however, mostly been addressed for conjunctive queries in OWL profiles, most notably the OWL QL profile but SPARQL queries are evaluated over RDF graphs which remain the basic data structure even when adopting a more elaborate semantic interpretation.

8.2.9 Querying the UWL-SIP Ontology with SPARQL

RDF and Triples are appropriated in this case study to establish that the right relationship exist in relation to motivation. The queries applied on the RDFS provide a systematic method which interrogates the enterprise architecture model from specific viewpoints, facilitates the validation of constituent artefacts and ascertain interoperability between various existing artefacts. The aim of these

set of queries is to augment earlier exemplifications for justification of the relevance of the model driven validation approach for validating EA models.

SPARQL Query #6

Developing further the concept of multiple Triple patterns for artefacts validation demonstrated in query #5, a SPARQL query is issued to validate the conformity of the subject Obtain_placement, whether there are appropriate predicate for the rdfs:domain and rdfs:range hierarchy of association with the objects Internship_listing, Intership_legibility and Student_enrollment. The result is generated in Figure 8-43.

```
SELECT ?Obtain_placement
        ?Study_enrollment ?Internship_Legibility
WHERE {
        ?Obtain_placement rdfs:domain ?Internship_listing .
        ?Obtain_placement rdfs:range ?Internship_Legibility .
        ?Obtain_placement rdfs:domain ?Study_enrollment .
}
ORDER BY ?Obtain_placement
```

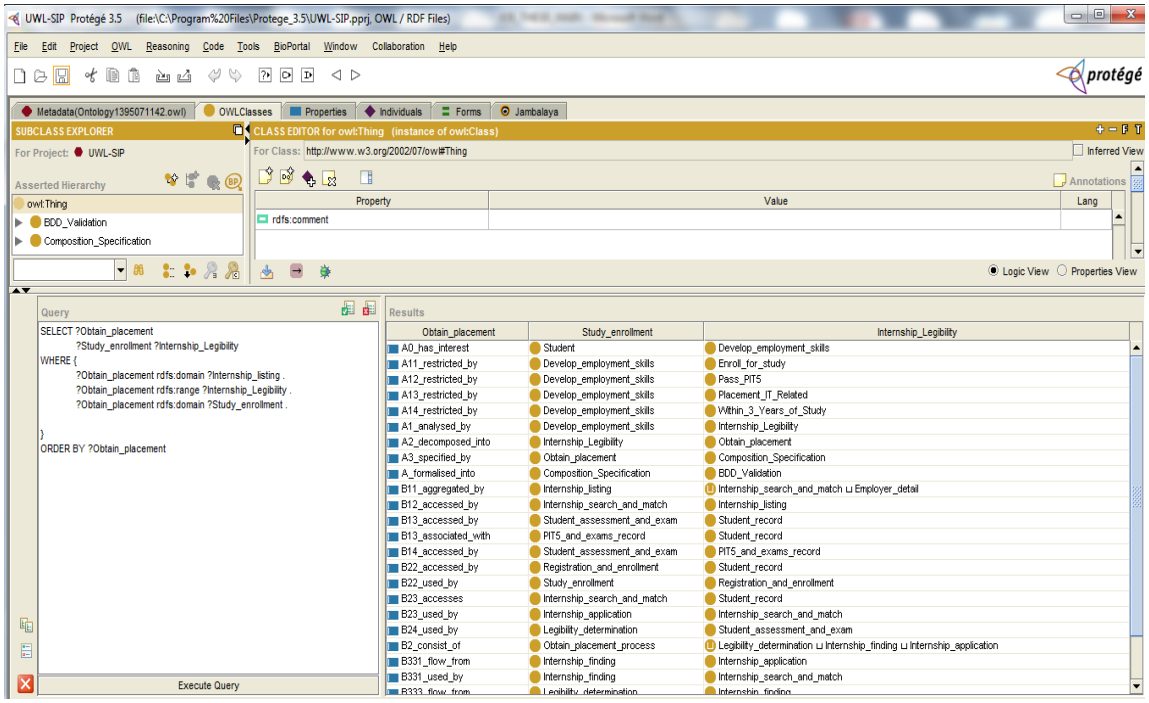

Figure 8-43: Multiple Triple patterns validation for the artefacts conformity and Behaviour Analogy

Evaluation of the result of the query shows three columns each corresponding to the SELECT request subject. The constraints specified in the WHERE clause refines each output as it is delimited by the object. Thus the Obtain_placement subject is associated with domain resources for Study_enrolment and Internship_Legibility. Of particular significance is the description of the element Internship_listing (second

column for the object Study_enrollment) which correlates the element as a disjunction aggregated by B11_aggregated_by with Employer_detail for the Internship_Search_and_match element. Similarly, conformity is asserted by the Obtain_placement_process which annotates appropriately the disjunction description for three elements as consisting of and specified as B2_consist_of of Legibility_determination, Intenship_finding and Internship_application. Also worthy of note on this multiple Triple SPARQL query is the Develop_employement_skills which accurately associates the four constraints Enroll_for_study, Pass_PIT5, Placement_IT_Related and Within_3_Years_of Study as constraints restricted by A11_restricted_by, A12_restricted_by, A13_restricted_by and A14_restricted_by respectively. Without these affirmative coherences exhibited by the result of this query, the integrity of the model would have been uncertain.

SPARQL Query #7

This query example demonstrates how the RDFS can be explored to locate a motivational element Obtain_placement in the ontology's RDFS dataset with associated core Business Architecture elements.

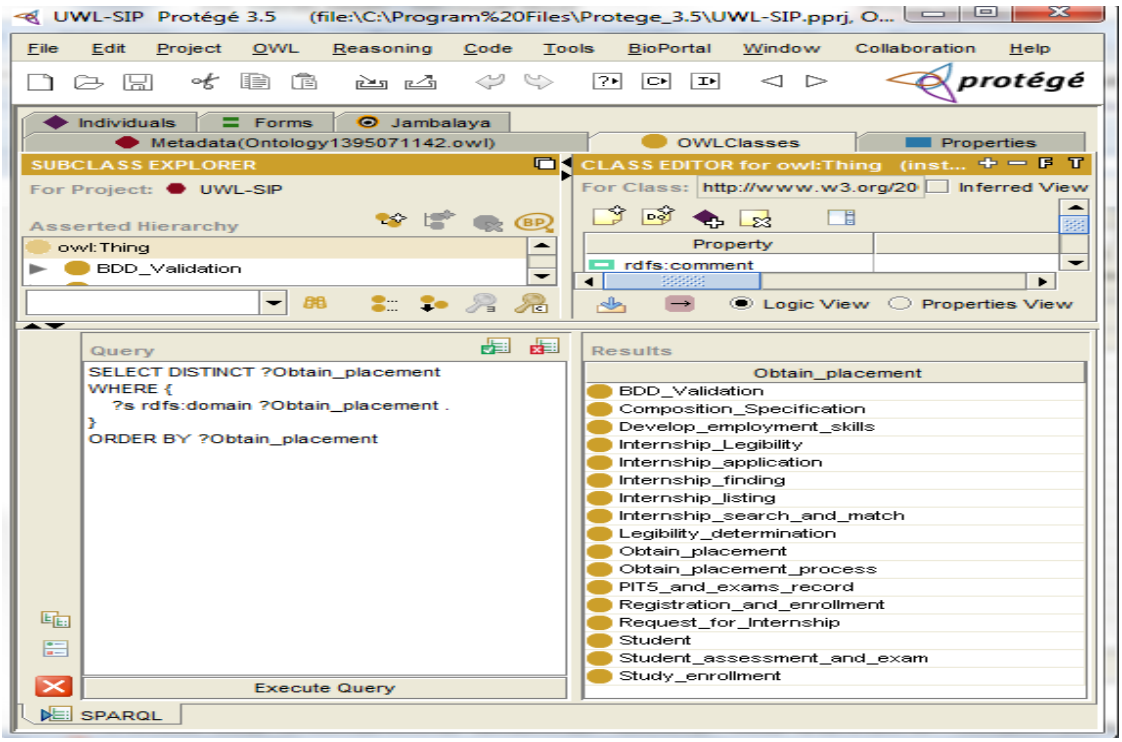

Figure 8-44: Ascertaining Availability Validation of Motivational element

The DISTINCT solution modifier used in this query eliminates duplicate solutions. Only one solution that binds the same elements to the same RDFS is returned from the validation.

SPARQL Query #8

Filtering of SPARQL query results can be achieved by using the optional FILTER expression NOT EXISTS and EXISTS. The filter expression EXISTS is utilized to test whether the pattern is authenticated in the data while the NOT EXISTS filter expression validates the conformity of the model abstract pattern to authenticate whether it is in disparity or does not match the dataset, given the elements in the group

graph pattern in which the filter occurs. Both options do not generate any additional bindings. The NOT EXISTS presents a way of thinking about negation in the ontology. This is demonstrated in Figure 8-45.

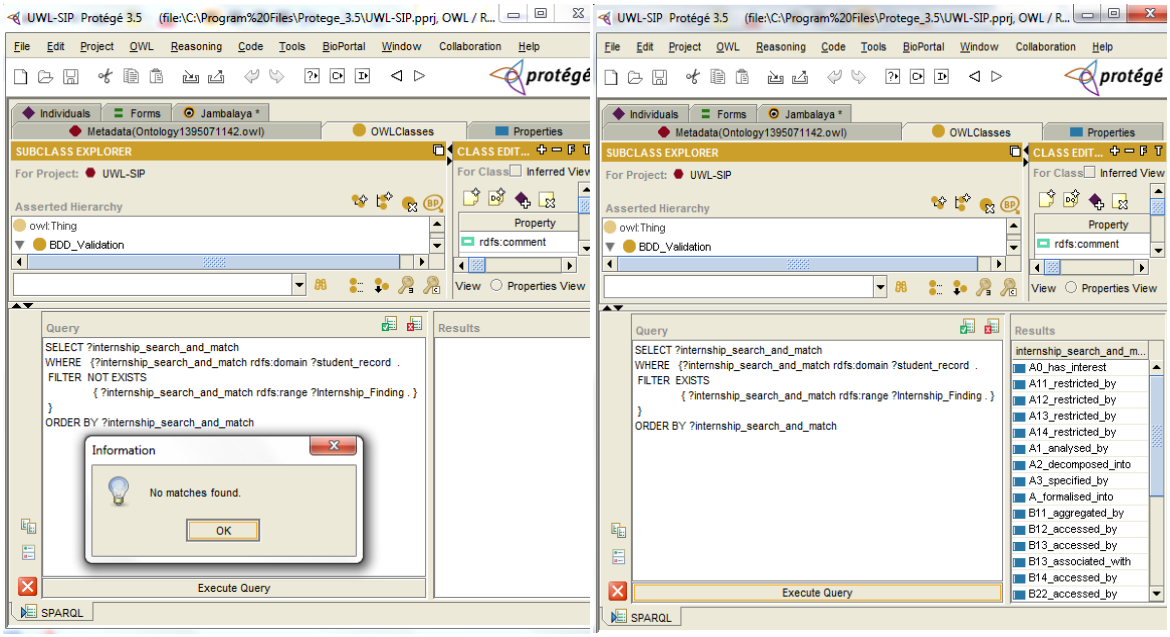

Figure 8-45: Application of Query with FILTERS

The exact validation query is issued with the NOT EXISTS and the EXISTS options. The results are displayed accordingly with a "No matches found" and a solution sequence, corresponding to the way in which the queried graph pattern matches the data.

SPARQL Query #9

Using the Namespace Prefixes defined for the UWL-SIP shown in Figure 8-46, the DESCRIBE form can also be applied on the ontology to validate for a single result RDF graph containing RDF data about SPARQL information resources. In this example, the query validates that the Student_record has accessed_by relationship with Internship_Search_and_Match artefact.

```
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1395071142.owl#"
   xml:base="http://www.owl-ontologies.com/Ontology1395071142.owl"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:assert="http://www.owl-ontologies.com/assert.owl#">
```

Figure 8-46: Namespace Prefixes defined for the UWL-SIP

The query pattern produces result set by taking each of the resources identified in a solution, together with any resources directly named by the IRI, and assembles a single RDF graph through a "DESCRIPTION" which comes from available information, including the target RDF dataset.

Thus the resources taken from the bindings are identified with query variable in the result set. This enables description of resources whether they are identified by IRI or by blank node in the dataset: The example statement below demonstrates this.

```
DESCRIBE ?x
WHERE {?x rdfs:student_record < http://www.w3.org/2000/01/rdf-schema#>
```

The Triple pattern object rdfs:student_record is defined as being a class individual in the RDFS vocabulary and the query returns information about at most one object that relates to rdfs:student_record.

8.3 Evaluation of Results of Case Studies

The two case studies presented delved into EA validation by use of objective evidence to confirm that a model meets the intrinsic goals defined by their requirements. To achieve this, the case studies modelled the business layer of EAF from normative perspectives namely motivation, organisation, business, structure and stakeholders. The research also evaluated the intrinsic nature of the model artifacts with the aim to answer some critical questions such as "how can an EAML be extended in order to incorporate validation attributes in its metamodel taxonomy?", "what needs to be in a model to allow its validation?" and "how can traceability be attained with EA models?" The approach conceptualized a hypothesis that adopted theories of model validation, principles for model validation rules, theoretical principles for goal evaluation, theoretical foundations for computing, information systems design theories in conjunction with ideas from domain-driven design and object-oriented analysis to compose a paradigm that transforms models to RDF Triples, articulate the semantic structures in form of object, predicate and subject to facilitate EA model validation. The domain-driven design concept is based on the business behaviour of the model and describes the test scenarios needed to facilitate the validation of the resultant RDF Triples. From the case studies, it can be inferred that it may be possible with some variation to apply the same methodology across all layers of the EA framework to demonstrate a pragmatic disposition of the concept within and across the different EAF compositions. The case studies have also excluded ambiguities about the methodology presented in this research. The metamodel, models and queries have exemplified consistency, ascribing articulate representation of structural components in relation to their behavioural attributes, impact on other elements, dependencies and their correlatives within given ontologies.

With respect to addressing the research sub questions, the two distinct case studies affirmed that the extended metamodel could be used to create formalized models that encapsulate the multifarious perspectives and validation requirements within the enterprise. The first study delved into the development of frameworks and models that addressed the need to allocate laptop resource to students and assign teaching rooms for lectures with cost and availability as constraints. The second study involved a more complex IT environment as it delved into the requirements of internal and external

collaborations in the provision of internship opportunities to students, communications and maintenance of employer engagement afterwards. The results from both case studies demonstrated and affirmed that simple and complex models could be transformed to ontologies in a formalised way, maintaining consistency and contextual integrity. Two of the stated expected benefits amongst others of the UWL-SIP were to adopt an Enterprise Architecture approach using ArchiMate as the modelling tool and to increase understanding of how EA can help in the implementation of transformation programmes. These two expectations were demonstrated conclusively by this research.

The results from both studies provided a number of insights. Firstly it was observed that though the triple format could create difficulties when it comes to the formulation of more intricate assertions, such assertions could be split into sets of simpler assertions so as to fit the triple format. However, a drawback here is that many concurring concepts within a model of a given viewpoint may need to be forked to compose their information semantics thus splitting may cause interoperable. Irrespective of this silo effect, a single uniform formalization approach for representation of the model is emphasized where possible.

However, a limitation was also noted with the ontology representation scheme. For instance, it is not obvious in some cases how assertions must to be interpreted. For example the assertion "Student applies for Laptop" as in the UME-LLS case study, could be interpreted for *each module* as "Student applies for Laptop", or for *each request* for a Learning Room, "Student applies for Laptop" or even at *registration for study*, "Student applies for laptop". Without additional knowledge about how to interpret the relation it cannot be decided which alternative is meant in any given case to aid appropriate modelling and validation. Certainly, in normal daily events, humans communicate well when using ambiguous statements. But this is so because humans are able to associate statements spontaneously with a relevant context of implicit background assumptions. In the case of model transformation to ontology , such implicit knowledge is lacking, and it is for this reason that logical definitions and axioms expressed in an appropriate formal language are required to preclude, or at least constrain, competing interpretations.

Despite this limitation, both case studies provide solid evidence that the research question can be collaborated. In conclusion of this summary therefore, it can be said that it is possible to incorporate validation into Enterprise Architecture models so as to ascertain the realization of motivational goals and ensure traceability. Thus the presentations affirm the hypothesis propounded affirmatively.

9 RESEARCH EVALUATION AND CONCLUSIONS

Evaluation provides evidence that a new technology developed in a Design Science Research (DSR) process achieves the purpose for which it is designed. Without evaluation, outcomes of DSR are unsubstantiated assertions that the designed artefacts if deployed in practice will achieve its purpose. Scientific researches require rigorous and sufficient evaluation (Venable, 2010). The importance of evaluation of DSR artefacts is emphasized as crucial and supported in many literatures (Venable, 2012). According to Gregor & Hevner (2013), these artefacts should be evaluated with criteria based on the requirements of the context in which the artefact is implemented. Examples of criteria have been identified as functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organisation and other relevant quality attributes.

Evaluation in Design Science Research (DSR) is concerned with assessment of the design science outputs, including the underlying theory and artefacts. The outputs framed as DSR artefacts usually consist of constructs, models, methods, and instantiations. In certain evaluations of DSR, it has been proposed that outputs should conform to the IS Design Theories (Walls et al., 2004). Hevner et al (2007) identified evaluation as an important substantiative phase in the DSR which demonstrates the utility, quality, and efficacy of a design artefact and advocates that using rigorous evaluation methods, the designed artefacts must be analyzed to authenticate their relevance and performance in driving changes and effectuating improvements in the behaviour of systems, people, and organizations.

9.1 Choice of Design Science Research Evaluation Method

The concept of discrete testable hypotheses for unambiguously evaluating two components of IS Design Theories (ISDT), the design process and the design artefacts has been proffered by many practitioners (Walls et al., 2004). Amongst these axioms, Hevner *et al* (2004) summarized five kinds of evaluation strategies namely observational, analytical, experimental, testing, and descriptive methods. Although guidance for making a choice amongst these evaluation strategies is not specified in their submissions, evaluation in Computer Science research has always emphasized the adoption of a positivist approach with a proposition for exemplification with experiments. However, since little work in the DSR domain has been acknowledged as sufficiently addressing choices and strategies for DSR evaluation, other authors have contested the adequacy of these strategies and classified DSR evaluation approaches into two primary forms: artificial and naturalistic (Venable, 2010). Venable explains that artificial evaluation evaluates a solution technology in a contrived and non-realistic way as obtainable in laboratory experiments, simulations, criteria-based analysis, theoretical arguments, and mathematical proofs while naturalistic evaluation explores the performance of a solution technology in its authentic environment such as the enterprise. Of these two, Venable contrary to Hevner *et al.*, summates that naturalistic evaluation is preferable as DSR evaluation strategy as a naturalistic setting is more dependable than a hypothetical artificial setting. While other researchers

argue that the interpretivist or even critical techniques may still be used with the DSR, Venable contest that these methods generally supplement the main goal of proving or disproving the design theory and not the utilization of the DSR artefact (Venable, 2010).

As artificial evaluation of DSR is deemed unreal in many ways as it deploys unreal users, unreal systems and unreal settings, the probability that the outcome of the artificial evaluation may be unreal for the valuation of DSR is legitimate and makes it inapplicable in this research evaluation. By performing evaluation in a real environment (real people, real systems, artefacts), and real settings naturalistic evaluation embraces all of the complexities of human practice in real organisations (Sun & Kantor, 2006) and is adopted in this work. Therefore in this scheme, guidance on evaluation presumes a positivist approach and conforms to the position of many practitioners that evaluation of DSR should be spurred by development of criteria and the assessment of the artefact's performance in comparison with the criteria. This it is argued would ensure that beyond simply establishing workability, the evaluation would achieve the responsibility of determining the rationale for that workability.

9.2 Formulating Strategic Framework for Research Evaluation

The purpose of this section is to formulate a strategic framework for evaluation in this Design Science Research. A strategic framework serves three purposes. It is used to help build strategies for evaluation of research outcomes, to achieve improved rigor in DSR and to descriptively improve understanding of unstated evaluation implications in the case under consideration. This strategic framework for DSR evaluation is based on several valuable principles of the ISDT explained in section5.4.1. One of such principle is the distinction between ex ante and ex post evaluations. Ex ante evaluation take place before a system is constructed while ex post evaluation take place after the system is constructed. A substantial body of literature and principles upon which this design research draws from assumes that "ex ante" and "ex post" are unproblematic concepts.

Applying these concepts in this study, DSR evaluation anchor on artefact which consists of the meta model extension, metamodel instances, proposed validation methodology and relationships which enhance traceability of business behaviour to goals. Thus the artefact as an anchor in this work is clearly defined. Drawing upon the above principles, a strategic framework can be formulated by choosing the prominent alternatives described above featuring *when* evaluation takes place, *what* is actually evaluated, and *how* it is evaluated. "When" to evaluate may be selected from ex ante, ex post, or both. It incorporates aspects such as the evaluation context (real stakeholder, organizations, views and viewpoints). "What" is evaluated involves choosing between the design processes or the design artefact. This also includes the granularity of evaluation. "How" to evaluate may be selected from naturalistic or artificial forms of evaluation (Venable, 2010). The strategic framework is designed to be used both normatively to guide the design of DSR evaluation and descriptively to ascertain conformity with extant DSR evaluation principles. The application of the framework for these two purposes is discussed in the following section.

9.3 Applying the Strategic Framework for Research Evaluation

The content measures for evaluation of information systems designs and artefacts are closely linked with quality criteria. Quality can be described in terms of measurable variables. The availability of these quality measures clarifies the selection of content within the strategic framework. For characterization of DSR evaluation where the design artefact is a model, quality standards such as ISO 9126, COBIT, and IEEE are used as inspiration. This is discussed in section 5.8 and illustrated in Figure 5.3 of this work. The standards applied in the context of EA suggest a number of potential measures that leads to an evaluation of the design artefacts against the characteristics and motivation of the model. These are Traceability, Perspective Visualization, Business Behaviour Analogy, Constraint Assessment, and Goal Realization. The main assumption of understandable Requirement-Oriented process is that these standards when adhered to leads to the realization of quality goals and artefacts (Engelsman & Wieringa, 2014). Following a well described and logical process also yields a better chance of producing quality. Evaluating whether a process is sound is not easy or obvious, but can be done. Firstly, the components of the process identified can be evaluated individually against some criteria or opinion of the method or process. Secondly, the development of quality models that can be validated as well as satisfy its motivation can also be evaluated by applying the framework for interpreting and describing the results and outputs.

In using the framework descriptively, three main questions are asked; (1) what is actually being evaluated? In the case of this study, EA artefact and its methodology; "what" may also include the evaluation granularity? In the case of this work, evaluation granularity levels are (a) whether the individual artefact was retrieved, (a) whether the business function which involved the artefact was completed, and (c) whether the completed task had a valuable impact on the associated goal or motivation. These granularity levels represent levels of means to achieving the goals and may be graded from level (a) to (c) ranging from low, intermediate and high. (2) How is it evaluated? As applicable to this study, using the naturalistic evaluation approach and (3) when is the evaluation carried out. This can be ex ante evaluation, ex post evaluation or both. To illustrate this evaluation approach, the following assertions are considered based on the theoretical principles for model validation discussed in section 5.2.

9.3.1 Evaluation Based on Principles for Active Validation Level

***What* is actually evaluated?**

The ability of extended metamodel to produce models with validation attributes that can be transformed to ontology. The RDFS of the ontology should be query-able using SPARQL semantics. The outcomes are design artefacts that are derived from a consistent and systematic methodology following existing theories for modelling and computing.

In terms of granularity, all related artefacts were retrieved, the business functions involving the artefact were completed, and the completed task had a valuable impact on the associated goal.

***How* was the evaluation carried out?**

The evaluation was a naturalistic evaluation in which the model was enacted against test scenarios on the basis of previously defined constraints and goals defined in motivation. As outcomes, it was affirmed that the deeper semantic expressiveness of the queries on the ontological provided a means to ascertain goal realization, perspective visualization and artefact traceability.

***When* was it evaluated?**

The evaluations were carried out both ex ante (before all the design artefacts were developed) and ex post (after the design artefacts were developed). For ex ante, evaluation was carried out to ascertain that the models were exact instantiation of the metamodel and that the transformed ontology was exact replication of the model. For ex post, evaluation was carried out to compare test execution output with the requirements specified as in motivation.

9.3.2 Evaluation Based on Principles for Passive Validation Level

***What* is actually evaluated?**

Interpretative analysis of the interfaces and relationship types of the RDF graphs to determine that there is an all inclusive traceability from constraints to goals. The level of granularity required in this case depends on the complexity of the RDF graph evaluated. The Interpretative analysis examines the collaboration between classes of the ontology and properties that establish interactions with different parts of the taxonomy. The outcome is a coherent impression of activity matrix that correlates a distinctive source to designated target.

In terms of granularity, all properties with associated range and domain are retrieved, the traceability tree is uninterrupted from source to target, and there is no ambiguity regarding the capability to iterate the traceability path for a similar initiative to achieve the same goal.

***How* was the evaluation carried out?**

The evaluation was a naturalistic evaluation in which the RDF graph was created based on forward and backward chaining using inference criterion and filters to extract elements as specified in the validation theme. As outcome, it was observed that the level of complexity of the graph could be controlled by reducing the filtration of Node Types and selectively increasing the Arc type filters. This minimizes redundancies and enhances the usability of the graph.

***When* was it evaluated?**

The evaluations were carried out both ex ante (before all the design artefacts were developed) and ex post (after the design artefacts were developed). For ex ante, evaluation was carried out to ascertain that the originating source model for the RDF graph was the exact replication of the source model. For ex post, evaluation was carried out to compare the RDF graph produced from the filtered abstraction with the RDF graph produced from the transformed source model. The expected outcome is that the abstracted RDF graph must be a subset of the source RDF graph.

9.3.3 Evaluation of the MDVA workflow

The application of scientific theoretical principles presented in this work towards the conceptualization of a Model Driven Validation Approach (MDVA) is a contribution that provides a solution to issues concerning EA model validation. The methodology describes the guidelines for model transformation to ontology, with capability for validation using Reasoners and a unified query language. This approach adds dexterity to the process of EA modelling and limitless potentials and threshold for validation of the model. Though the development of an ontology that precisely define concepts and properties of enterprise architecture is a challenging task, this approach is justified as it provides a clear understanding of several challenges in validation of EA models.

***What* is actually evaluated?**

The MDVA workflow allows the definition of both the behavioural and the structural attributes of the EA components. Validation themes are defined by a set of motivational goal and specify the components to be tested in the model. The precept of the MDVA workflow is presented in Appendix C.

In terms of granularity, every entity in the workflow is accessible, there are no dead ends or isolated branches; the workflow results in completed process; and completed activities have one outcome which allows comparison of results.

***How* was the evaluation carried out?**

The evaluation was a naturalistic evaluation in which two case studies were adopted. The process of generating RDFS was tested for the two case studies. As outcomes, the result of validations performed on the RDFS and RDF graphs were compared to motivation to establish goal realization and traceability.

***When* was it evaluated?**

The evaluations were carried out ex post (after the design artefacts were developed). This was necessary in order to allow the entire workflow to be tested as a whole.

9.4 Reflective Assessment of the Research Contributions

RC1: Extension of metamodel of Business Layer of EAF with Validation Element:

For the Research Contribution RC1, the ArchiMate EAML was successfully extended with validation capabilities. This provides a capability for expressing metamodels and models in a form that allow motivation aspects to be associated with business architecture artefacts. This is demonstrated with the business layer models that represented the Student's view for both the UME-LLS and UWL-SIP case studies

RC2: Development of EAF model to ontology Transformation Approach:

When it comes to ontology transformation, modelling of EA was carried out from varied perspectives and with the extended validation element. For each of the case studies, a model was transformed to ontology. As a contribution, it was affirmed that the approach enabled clarity in the presentation of EA model in terms of goals that are required and business artefacts that constitute the processes needed to achieve those goals.

RC3: Application of domain-driven design and object-oriented analysis concepts in the Validation of EAF models:

The analysis of the use of domain-driven design and object-oriented concepts as a tenet adaptable to the formalisation and development of semantics for EAML, description of queries for EA validation demonstrate consistent dependencies as the queries indicated significant compliance with set principles and theories for model validation rules. The outcomes are queries that are expressive with variables that occur within the validation Class expression bounded by ranges, domains and properties. The challenges associated with the implementation this approach and combination with other levels of validation for other factors was negligible when supported with Reasoners. Reasoners specifically fit with this method of validation as it colludes information about inferences and rule processes to potentially provide answers to enquiries. Therefore the use of Reasoners to support and exemplify this contribution allows entailment which is intrinsic to the model for definition of the viewpoint in the ontology. This makes the execution of the query for information fairly straightforward.

RC4: Validation of Enterprise Architecture Models using Ontology querying methodology:

Contemporary approaches that have been preferred as a means of validating EAF and models have been maturity matrices, balanced scorecards and reference models. These approaches which are based on qualitative evaluation are very subjective as they are often susceptible to many inhibitions such as user bias, levels of respondent's discernment and sometime organizational intricacies. This contribution which is logical is demonstrated using SPARQL. It

is also objective and targeted based on input and output artefacts that must adhere to set constraints and business rules.

RC5: RDFS Triple store for EAF Model:

An EA model transformed to ontology provides the capability to create a unified store house for triple patterns, conjunctions, disjunctions, and RDFS. Triple stores are incrementally developed with each transformation to ontology and can enhance deep querying and traceability within the EAF. This also enables the development of regression testing of EAF models thus improve the overall quality of the framework. The contribution is achieved and presented in appendices D, E and F. This contribution is also the subject of relative importance and of recent is providing the basis and significant framework for scientific investigations and area for further research.

RC6: Model-Driven Validation Approach:

The Model-Driven Validation Approach (MDVA) is contributed. MDVA validates a model iteratively by testing primarily elements and attributes of the model against goals and constraints in its motivation extension. The outcome is an improved workflow process for incorporating quality into the design of the model through goals to component association. The traceability process is also simplified through his methodology. The validation scenarios for MDVA describe the behaviour and attributes of the component to be validated in order to realize set motivation goal. Granted that the methodology is adhered to, it ensures better conformance to user Goals and motivation. This contribution in exemplified and evaluated in the UME-LLS and UWL-SIP case studies.

9.5 Novelty and Originality of Findings

The construct of the ArchiMate complemented with concepts of validation produced an extended EAML with added capabilities. This enabled the placement of the artefacts within the context of validation while establishing veritable relationship between the core EA artefacts and motivational elements. Modelling with this extended metaphor exemplified by two case studies, the UME-LLS and the UWL-SIP provided the modularity for transformation to ontology. Adopting ideas from domain-driven design, testing and validation were addressed from four perspectives;

- Use of the ontology language Reasoner.
- Use of the class and property filters to abstract portions of related RDF graphs thus demonstrate traceability
- Demonstration of forward chaining or backward chaining and
- Use of the SPARQL to issue direct queries for specific artefacts and business behaviour based on constraints specified in motivation.

The affirmation and conformity of this approach with extant theories and principles provide the evidence that the methodology developed in this thesis has competency for validating models for enterprise architecture frameworks.

More techniques for validating models which are predominantly based on experiences gained with construction of ontologies are still evolving. Formalization and annotation of concepts within the techniques are considered critical for effective incorporation of the structural layers, artefact types and dependencies of the models. This practice has been found to expose gaps and overlapping functionalities that may exist between classes and properties of the models. The case studies presented in this research has shown that the development of modelling techniques that can validate the framework would minimise inconsistencies within EAF as well. This implies also that it may be very feasible to amalgamate divergent models by decomposing its business behaviour and consolidating logically related archetypes into perspectives and validation theme for effective interoperability and validation. It appears however that what is needed to achieve this is the development of a common vocabulary specifically for use with EA models for application over its transformed RDFS. This induced semantics would also support effective validation of the model and establishment of traceability.

It is also significant to note that though the focus of the approach presented in this work did not dwell on evaluation through maturity indices, the systematisation and abstraction of EAF perspectives through modelling and transformation to ontology promoted the validation of motivation and ascertainment of traceability. It also enabled the visualization of gaps and overlaps that exists in the heterogonous model expositions that are found in many businesses EAF. Given that the existence of complexities and lack of semantic integrity has continued to pose difficulties in contemplating the use of ontologies as a means of consolidating and validating EA models, a major benefit of the validation approach presented in this work is that it has advanced a technique with a precise standard which supports traceability and validation of EAF; providing a foundation for alignment of EA abstractions to technological infrastructures.

9.6 Sustainability of Contributions and Findings

Ontologies have existed for a very long time and has been conceptualised as the basic structure or armature around which knowledge base can be built (Uschold and Gruninger, 1996). Ontologies are characterized mostly in relation to their means and content. For instance, while a simple ontology may include a hierarchy to concepts bound by assumption for its relationship (Wache et al., 2001), more complex ontologies may include axioms that handle increased complexities of relationships, concepts and constraints desired to bind the intended interpretation (Noy, 2004). Thus as ontology development expands within more inter-organisational projects and integration of disparate views, their integration is expected to produce subsequent opportunities for refinements. Concern for appropriate

representation of content may later be augmented using specification languages or other formal logic. In relation to the contributions presented by this work, considerable facets of opportunities provided by ontologies have been exploited to further its benefits in the following ways;

i. The exposition of traceability for the Enterprise Architecture artefacts through the use of ontology filters and logical reasoners has allowed dependencies and effect of change to be more apparent. This innovation apart from facilitating clarity in the presentation of EA model in terms of goals that are required and business artefacts that constitute the processes needed to achieve those goals; it has also aided alignment of business strategy with goals as well as determines gaps and overlaps with the EA taxonomies.

ii. The contributed Model Driven Validation Approach (MDVA) precipitates a methodology for improvement of the quality and design of an EA model through goals to traceable component association, as the composition of the MDVA notion subsumes both the behavioural and the structural attributes of the EA components. The validation metrics associated with the MDVA specifies what types of test are to be carried out on the components and the context of the expected results as defined by motivation.

iii. The concept of storage of EA models for the purpose of validation as an RDFS within ontology triple stores is a contemporary alteration emerging from this work. Given the complexities associated with modelling of EAF, varieties of viewpoints, and the use of discordant artefacts resulting from the adoption of heterogeneous modelling languages, an obvious and pragmatic way for formalization is presented with use of ontologies. EA models transformed to ontology provide this capability to create a unified store house for triple patterns, conjunctions and disjunctions. Triple stores can be incrementally developed with each transformation of model to ontology thus enhance deep querying and traceability within the EAF. This contribution paves the way for the development of regression testing of EAF models for the first time.

iv. Validation of Enterprise Architecture models using ontology querying methodology provides an opportunity for development of new validation semantics that are particularly specific to EA artefacts. This research has adapted the domain-driven design and object-oriented analysis, a contemporary approach understandable by many stakeholders of the enterprise to achieve this aim. However there is need for more work to be done in this area in terms of further development of the lexicon of the language to address the multifarious validation needs of the ontology. This is likely to be an ongoing heuristic enhancement process for the identification of the most useful generic queries applicable to EA validation domain. The adoption of the approach proposed in this work on more EA modelling projects across divergent establishments would provide further information

that can authenticate the generality for the ambience of the vocabulary. The contribution of this research towards this aspiration are queries adapted using SPARQL semantics to interrogate RDFS and to obtain results that can be compared against the associated goals, establish traceability and ensure alignment. The language semantics is built with preconditions and post conditions. Domain-driven design and object-oriented analysis concepts are applied in a formal way to prognosticate the query for the constraints specified by motivation. Evaluation of the result yields three outcomes; (a) values that allow comparison to ascertain if the tested goal is realized, (b) component traceability and (c) reusable test basis for validation iterations. This validation contribution is logical, objective and targeted based on input and output artefacts that adhere to set constraints and business rules.

Past and recent developments have shown that there are actually two main research communities active in the ontogenesis of enterprise architectures. They represent two different perspectives classified as information technology and enterprise modelling. The main problems cited have continued to be concerns with the different semantics and languages used. Therefore the capability to extend modelling languages by these communities is of crucial importance. As contributed by this work, the extension of the ArchiMate EAML with validation capabilities has provided a methodology for expressing metamodels and models in a form that allow motivation aspects to be associated with business architecture artefacts. Furthermore, the extended models encapsulated with motivation are transformed to ontology description schema with that efficacy, allowing validation to be performed.

It is necessary to continue this effort of harmonization of EAF by establishing collaboration between enterprise modelling communities, developing business-oriented architectures and technology engineering people while working on the IT-oriented ones. Simulation of regression testing could be added to the extensions in order to leverage the potential of the stored RDF triples for the purpose of automated validation. The triples could be expanded to support efficient storage, organisation and retrieval of model related documents and schemas.

From another point of view, enterprise architectures need to accommodate change, evolve with the application of new technologies, transformations and with the developments in the business environment. This research has contributed and demonstrated a transformation workflow from EA model to ontology in support to this principle. The continuous alignment of business architecture to IT architecture is one of the challenges to the implementation of EA in industry and management of change. More joint research efforts are needed to develop unified enterprise architecture in particular with unique semantics that allow adept mapping between business and IT architectures in order to facilitate this postulation.

9.7 Recommendations and Further Research

As validation of EAF is an area that currently draws very little diligence amongst practitioners due to complexities, this thesis presents a novelty methodology through which much research can be initiated. This include amongst many others a case for integration of divergent EAFs through a common vocabulary using ontology so as to allow better congruency, traceability, validation and alignment of business objectives to Information Technology.

Overall, research to date has indicated that architecture concepts are not sufficiently exploited. One of the reasons is the lack of appropriate architecture representation formalism supporting the characterization of features and properties of enterprise systems at a high abstraction level. Existing enterprise architecture proposals are represented in different ways with neither a rigorous syntax nor semantics. Existing architecture principles are seldom developed to a satisfactory level which allows amalgamation of significant improvement to enterprise architecting. Developing architecting principles can be bottom-up based on best practices, or top-down by studying some theoretical paradigms. Furthermore, available enterprise architectures are deficient of justifications in many cases. It is difficult to know why architecture is arranged in one way rather than another. Principles and patterns for designing architectures for various purposes (interoperability, flexibility, modularity, etc) would allow grounding future architecture development on a more scientific basis.

As found in this research and presented in the thesis, previous, past and existing enterprise architecture research and development suffer from lack of methods for evaluating architecture proposals. Evaluation criteria such as for example maturity, security, interoperability, modularity, robustness, openness, etc that are used to characterize an architecture proposal need to be more precisely defined. One main challenge here is to define the concepts and to elaborate metrics allowing measuring different degrees of maturity, security, interoperability, etc. Architecture evaluation criteria are also related to architecture design principles. These criteria actually reflect possible architecture properties and can be an area of further research.

Other areas of further research may include the exploitation of the ability to merge several ontologies developed within the same enterprise, possibly using divergent or cascaded EA framework for effective interoperability and validation. From this work, it seems feasible that divergent EAFs can be amalgamated through a common vocabulary using ontology. Further research needs to be carried out to define the parameters through which this can be achieved. There are also other concerns such as validation of integrated scattered information critical for the support of enterprise architecture models from sources such as Linked Data.

Future research and development on enterprise architecture should be based on a rigorous and precise ontology definition of the set of concepts, relations and properties of enterprise architecture. There is also an important need to develop an agreed architecture representation language and evaluation

method/metrics so that architecture proposals can be properly described, assessed and compared. Also, architecture design principles and patterns for promoting proven and justified architectural solutions in the industry need to be reviewed regularly. This research has contributed towards that inquisition of knowledge and has used extant theories and open source platform to allow recycling of knowledge; conceptualized and executed the research reusing knowledge-based applications with extensions to develop an approach for validating EA frameworks.

10 BIBLIOGRAPHY

Abdullah, I., Umair, T., Rashid, Y., & Naeem, B. (2013). Developments on Balanced Scorecard: A Historical Review. World Applied Sciences Journal, 21(1), 134-141.

ACM (1992). Codes of ethics and professional conduct [Online], Available at: URL: http://www.acm.org/about/code-of-ethics. Accessed: December, 2014.

Al-Mudimigh, A. S. (2007). The role and impact of business process management in enterprise systems implementation. Business Process Management Journal, 13(6), 866-874.

Alhumaidan, F., & Zafar, N. A. (2014). Possible Improvements in UML Behaviour Diagrams. In Computational Science and Computational Intelligence (CSCI), 2014 International Conference on (Vol. 2, pp. 173-178). IEEE.

Almeida, J. P. A., & Guizzardi, G. (2013). An ontological analysis of the notion of community in the RM-ODP enterprise language. Computer Standards & Interfaces, 35(3), 257-268.

Alturki, A., Gable, G., Bandara, W., & Gregor, S. (2012). Validating the Design Science Research Roadmap: Through the Lens of "The Idealised Model For Theory Development".

Alvarez, M. M. R., Fernández, M. M. M., Conroy, B. V., & Martínez, A. C. (2008). Criteria of the peer review process for publication of experimental and quasi-experimental research in Psychology: A guide for creating research papers. International Journal of Clinical and Health Psychology, 8(3), 751-764.

Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D., & Sycara, K. (2002). DAML-S: Web service description for the semantic web. In The Semantic Web-ISWC 2002 (pp. 348-363). Springer Berlin Heidelberg.

Apt, K. (2003). Principles of constraint programming. Cambridge University Press.

ArchiMate (Version 2.4). Available at: http://archi.cetis.ac.uk/download.html, Accessed: April 2013.

Ashmore, P., Henson, J., Chancellor, J., & Nelson, M. (2004). Is Your Enterprise Architecture All It Can Be? Lessons From The Front-Line. Business Process Trends (June 2004).

Astels, D. (2003). Test driven development: A practical guide. Prentice Hall Professional Technical Reference.

Bahill, A. T., Botta, R., & Daniels, J. (2006). The Zachman framework populated with baseball models. Journal of EA, 2(4), 50-68.

Baker, P., Dai, Z. R., Grabowski, J., Haugen, O., Lucio, S., Samuelsson, E., & Williams, C. E. (2004). The UML 2.0 testing profile. In Proceedings of the 8th Conference on Quality Engineering in Software Technology, Nuremberg (Germany) (pp. 181-189).

Baker, D.C., & Janiszewski, M. (2005). 7 Essential Elements of EA, Enterprise Architect, Fawcette Technical Publications (FTP).

Bakhshadeh, M., Morais, A., Caetano, A., & Borbinha, J. (2014). Ontology Transformation of Enterprise Architecture Models. In Technological Innovation for Collective Awareness Systems (pp. 55-62). Springer Berlin Heidelberg.

Barbau, R., Lubell, J., Rachuri, S., & Foufou, S. (2014). Towards a reference architecture for archival systems: use case with product data. Journal of Computing and Information Science in Engineering, 14(3), 031005.

Barra, E., Génova, G., & Llorens, J. (2004). An approach to Aspect Modelling with UML 2.0. In Aspect-Oriented Modeling Workshop, AOM.

Bay-Borelli, M., Rozunick C., Way W., & Weisman E. (2010). Considerations For Developing Test Specifications- For Common Core Assessments, Adopting Curriculum Standards—Only the First Step, A white paper from Pearson, Available at: http://www.pearsonassessments.com/research.

BCS, 2006. Code of conduct for BCS members. Available at: URL: http://www.bcs.org/server.php?shownav.6030. Accessed: December, 2013.

Beck, K. (2003). Test-driven development: by example. Addison-Wesley Professional.

Bernus, P., & Nemes, S. (2003). Handbook on Enterprise Architecture, Springer-Verlag.

Berrisford, G., & Lankhorst, M. (2009). Using ArchiMate with TOGAF {Part 1}: Answers to nine general questions about methods. Via Nova Architectura.

Beznosov, K. (2000). Information enterprise architectures: problems and perspectives. In Written for the Advanced Topics in Software Engineering seminar given by Dr. Michael Evangelist at the School of Computer Science, Florida International University.

Bicchierai, I., Bucci, G., Nocentini, C., & Vicario, E. (2013). Using Ontologies in the Integration of Structural, Functional, and Process Perspectives in the Development of Safety Critical Systems. In Reliable Software Technologies–Ada-Europe 2013 (pp. 95-108). Springer Berlin Heidelberg.

Bittler, R. S., & Kreizmann, G. (2005). Gartner Enterprise Architecture Process. Evolution, 21.

Blessing, L. T., & Chakrabarti, A. (2009). DRM, a design research methodology. London: Springer.

Bloch, M., Blumberg, S., & Laartz, J. (2011). Delivering large-scale IT projects on time, on budget, and on value. Harvard Business Review.
Boury-Brisset, A. C. (2003). Ontology-based approach for information fusion. In Proceedings of the Sixth International Conference on Information Fusion (pp. 522-529).
Brame, A., & Barlow, G. (2010). KPMG New Zealand Project Management Survey 2010, Available at: http://www.kpmg.com/NZ/en/IssuesAndInsights/ArticlesPublications/Documents/Project-Management-Survey-report.pdf. Accessed: January, 2014.
Braun, C., & Winter, R. (2005). A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Meta-modelling Platform, GI Edition, LNI, Enterprise Modelling and Information systems, Architectures, Proc. of the Workshop in Klagenfurt, Klagenfurt, 24.10.2005, P75, pp. 6479.
Bredemeyer Consulting, (2013). Software Architecting Success Factors and Pitfalls, Capgemini, Available at: http://www.capgemini.com/ Accessed: 2014.
Calì, A., Gottlob, G., & Lukasiewicz, T. (2012). A general datalog-based framework for tractable query answering over ontologies. Web Semantics: Science, Services and Agents on the World Wide Web, 14, 57-83.
California Technology Agency EA 1.1, (CTA), (2011). Enterprise Architecture Glossary by Set, Available at: http://www.cio.ca.gov/Government/IT_Policy/pdf/SIMM_58C_Enterprise_Architecture_Glossary_04132011.pdf. Accessed: August, 2014.
Calvanese, D., De Giacomo, G., & Lenzerini, M. (2002). A framework for ontology integration. In The Emerging Semantic Web—Selected Papers from the First Semantic Web Working Symposium (pp. 201-214).
Carla M. P., C. M., & Sousa, P. (2005). Enterprise architecture: business and IT alignment. ACM SAC '05, Proceedings of the 2005 ACM symposium on Applied computing.
Carlsson, S. A. (2010). Design science research in information systems: A critical realist approach. In Design Research in Information Systems (pp. 209-233). Springer US.
Ceh, I., Crepinsek, M., Kosar, T., & Mernik, M. (2011). Ontology driven development of domain-specific languages. Computing Science Information Systems, 8(2):317{342}.
Chapurlat, V., & Braesch, C. (2008). Verification, validation, qualification and certification of enterprise models: Statements and opportunities. Computers in Industry, 59(7), 711-721.
Chelimsky, D., Astels, D., Helmkamp, B., North, D., Dennis, Z., & Hellesoy, A. (2010). The RSpec book: Behaviour driven development with Rspec, Cucumber, and friends. Pragmatic Bookshelf.
Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. Computer and Industrial Engineering, 59:647659.
Choi, N., Song, I. Y., & Han, H. (2006). A survey on ontology mapping. ACM Sigmod Record, 35(3), 34-41.
Chrissis, M.B., Konrad, M. & Shrum, S. (2003). CMMI: Guidelines for process integration and product improvement. Addison-Wesley Professional.
CIO-Council, Federal Enterprise Architecture Framework version 1.1, Available at: http://www.cio.gov/archive/fedarch1.pdf. Accessed: April, 2012.
Clark, T., Barn, B. S., & Oussena, S. (2011). Leap: a precise lightweight framework for enterprise architecture. In Proceedings of the 4th India Software Engineering Conference (pp. 85-94).
Clark, T., Evans, A., Kent, S., & Sammut, P. (2001). The MMF approach to engineering object-oriented design languages. In Ws. on Language Descriptions, Tools and Applications (LDTA), Genova, Italy.
Clinger-Cohen Act of 1996, 1996. Available at: www.tricare.osd.mil/imtr/ppm/documents/clingercohen.pdf. Accessed: April 2013.
COBIT, Available at: http://www.isaca.org/cobit/pages/default.aspx; Accessed: February 2014.
Codd, E. F. (1982). Relational database: a practical foundation for productivity. Communications of the ACM, 25(2), 109-117.
Coleman, P., & Papp, R. (2006). Strategic Alignment: Analysis of Perspectives. Proceedings of the 2006 Southern Association for Information Systems Conference.
Coryn, C. L., Noakes, L. A., Westine, C. D., & Schröter, D. C. (2011). A systematic review of theory-driven evaluation practice from 1990 to 2009. American Journal of Evaluation, 32(2), 199-226.
Cumming, G. (2012). Understanding The New Statistics: Effect Sizes, Confidence Intervals, and Meta-Analysis. New York, USA: Routledge. pp. 27–28.
Cummings, T., & Worley, C. (2014). Organization development and change. Cengage learning.
Curran, C. (2005). Link IT Investments to Business Metrics. Enterprise Architect (3:1), pp. 16-18.
Da Xu, L. (2011). Enterprise systems: state-of-the-art and future trends. Industrial Informatics, IEEE Transactions on, 7(4), 630-640.
Dahalin, Z. M., Razak, R. A., Ibrahim, H., Yusop, N. I., Kasiran, M. K., & Malaysia, U. (2010). An enterprise architecture methodology for business-it alignment: adopter and developer perspectives.

Damanpour, F., & Schneider, M. (2006). Phases of the adoption of innovation in organizations: Effects of environment, organization and top Managers1. British Journal of Management, 17(3), 215-236.
Danesh, M. H., & Yu, E. (2014). Modeling Enterprise Capabilities with i*: Reasoning on Alternatives. In Advanced Information Systems Engineering Workshops (pp. 112-123). Springer International Publishing.
Dashofy, E. M., Van der Hoek, A., & Taylor, R. N. (2002, May). An infrastructure for the rapid development of XML-based architecture description languages. In Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on (pp. 266-276). IEEE.
Delgado, J. C. (2014). An Enterprise Interoperability Framework based on Compliance and Conformance. Revolutionizing Enterprise Interoperability Through Scientific Foundations, 280.
Department of Defence (DoD), Department of Defence Architecture Framework Version 1.0 - Vol. 1 Definition & Guideline & Vol. 2 Product Descriptions. Available at: www.aitcnet.org/dodfw, Accessed: April, 2013.
Di Maio, P. (2011). 'Just enough' ontology engineering. In Proceedings of the International Conference on Web Intelligence, Mining and Semantics (p. 8). ACM.
Dietz, J.L.G. (2006). Enterprise Ontology: Theory and Methodology, Springer.
DoD (2013). DoDAF. Systems & Software, Consortium. Available at: www.software.org/pub/architecture/dodaf.asp. Accessed: March 2013.
Dragan G., Dragan D., & Vladan D., (2007)., MDA-Based Automatic OWL Ontology Development. International Journal for Software Tools Technology Transformation, 9(2):103–117.
Dryer, D. A., Bock, T., Broschi, M., & Beach, T. D. (2007). DoDAF limitations and enhancements for the Capability Test Methodology. In Proceedings of the 2007 spring simulation multiconference-Volume 3 (pp. 170-176). Society for Computer Simulation International.
Dustin, B., & Elfriede, S. (2002). Effective Software Testing. Addison Wesley. p. 3.
Dutoit, A., McCall, R., Mistrk, I., & Paech, B. (2006). Rationale management in software engineering: Concepts and techniques. In A. Dutoit, R. McCall, I. Mistrk, and B. Paech, editors, Rationale Management in Software Engineering, pages 1-48. Springer.
Engelsman, W., & Wieringa, R. (2014). Understandability of Goal-Oriented Requirements Engineering Concepts for Enterprise Architects. In Advanced Information Systems Engineering (pp. 105-119). Springer International Publishing.
Engelsman, W., Quartel, D., Jonkers, H., & van Sinderen, M. (2011). Extending enterprise architecture modelling with business goals and requirements. Enterprise Information Systems, 5(1), 9-36.
Erder, M. & Pureur, P. (2003). QFD in the Architecture Development Process. IT Professional (5:6), pp. 44-52.
Ernst, A. M. (2010). A pattern-based approach to enterprise architecture management (Doctoral dissertation, München Technical University, Dissertation.
Essien, J., & Oussena, S. (2013). Enterprise Architecture Models for Description of Integrated Components for Validation - A Case Study of Student Internship Programme. In the 15th International Conference on Enterprise Information System.
FEA Consolidated Reference Model Document, May 2005 and FEAF Practice Guidance, December 2006. Published by the Federal Enterprise Architecture Program Management Office, Office of Management of Budget. Available at: www.whitehouse.gov. Accessed: November, 2013.
Fernández-López, M., & Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review, 17(02), 129-156.
Fischer, C., Winter, R, & Aier, S. (2010). What Is an Enterprise Architecture Principle? Towards a Consolidated Definition, Computer and Information Science, SCI 317, pp. 193–205.
Fox, M.S. (1992). The TOVE Project: A Common-sense Model of the Enterprise, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Belli, F. and Radermacher, F.J. (Eds.), Lecture Notes in Artificial Intelligence # 604, Berlin: Springer-Verlag, pp. 25-34.
Frank, L., & Krogstie, J. (2008). Active Knowledge Modelling of Enterprises. Springer,
Frank, U. (1998). The MEMO Object Modelling Language (MEMO-OML). Arbeitsberichte des Instituts für irtschaftsinformatk, Nr. 10, Koblenz.
Frank, U., & Jung, J. (2001). The MEMO Organisation Modelling Language (MEMO-OrgML). Arbeitsberichte des Institut für Wirtschaftsinformatik der Universität Koblenz-Landau.
Frank, U. (2002). Multi-Perspective Enterprise Modelling (MEMO): Conceptual Framework and Modelling Languages". In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-35). Los Alamitos, CA. Ralph H. Sprague, Jr. (eds.). IEEE Computer Society Press.
Frankel, D. S. (2003). Model Driven Architecture Applying MDA. John Wiley & Sons.
Frankel, D. S., Harmon, P., Mukerji, J., Odell, J., Owen, M., Rivitt, P., Rosen, M., & Soley, R. M., (2003).The Zachman framework and the OMG's model driven architecture." Business Process Trends ". pp. 1-14.
Franz, B. Diego, C., Deborah L. McGuinness, Daniele, N, & Peter, F. (2003). Patel-Schneider, editors. The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA.

Fraser, P., Moultrie, J. & Gregory, M. (2002). The use of maturity models/grids as a tool in assessing product development capability. In the Proceedings of the IEEE International Engineering Management Conference, Cambridge, pp. 8-20.

Frederiksen, N., Glaser, R., Lesgold, A., & Shafto, M. G. (Eds.). (2013).Diagnostic monitoring of skill and knowledge acquisition. Routledge.

Funnell, S. C., & Rogers, P. J. (2011). Purposeful program theory: effective use of theories of change and logic models (Vol. 31). John Wiley & Sons.

Gangemi, A., Catenacci, C., Ciaramita, M., & Lehmann, J. (2006). Modelling ontology evaluation and validation (pp. 140-154). Springer Berlin Heidelberg.

GARTNER (2013). Available at: http://www.gartner.com. Accessed: April, 2013.

GENECA RESEARCH REPORT (2011). "Doomed From the Start? Why a Majority of Business and IT Teams Anticipate Their Software Development Projects Will Fail" Available at: http://www.geneca.com. Winter 2010/2011 Industry Survey. Accessed 2013.

George, B., & Williams, L. (2003). An initial investigation of test driven development in industry. In Proceedings of the 2003 ACM symposium on Applied computing (pp. 1135-1139). ACM.

Gerrit A. & Blaauw (1972). Computer Architecture", Elektronische Rechenanlagen, Vol 4, p. 154-159.

GITHUB (2011). Available at: https://github.com/cucumber/gherkin/wiki/tool-Support. Accessed: April 2013.

Glimm, B. (2011). Using SPARQL with RDFS and OWL entailment. In Reasoning Web. Semantic Technologies for the Web of Data (pp. 137-201). Springer Berlin Heidelberg.

Goethals, F. (2003). An Overview of Enterprise Architecture Framework Deliverables. Available at: www.econ.kuleuven.ac.be/leerstoel/sap/FramesPage.htm. Accessed: March 2013.

Goldkuhl, G. (2004). Design theories in information systems-a need for multi-grounding. Journal of Information Technology Theory and Application (JITTA), 6(2), 7.

Golnam, A., Viswanathan, V., Moser, C. I., Ritala, P., & Wegmann, A. (2014). Designing Value-Oriented Service Systems by Value Map. In Business Modeling and Software Design (pp. 150-173). Springer International Publishing.

Golnam, A. (2013). Problem Structuring with the Systemic Enterprise Architecture Method.

Green, N., & Bate, C. E. T. (2007). Lost in Translation: A Handbook for Information Systems in the 21st Century.

Greefhorst, D., Greefhorst, E., & Proper E. (2011). Architecture Principles: The Cornerstones of Enterprise Architecture, Springer.

Gregor, S. (2006). The nature of theory in information systems. Mis Quarterly, 611-642.

Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. MIS Quarterly, 37(2), 337-356.

Grüninger, M., Atefi, K., & Fox, M. S. (2000). Ontologies to support process integration in enterprise engineering. Computational & Mathematical Organization Theory, 6(4), 381-394.

Gudas, S. & Lopata, A. (2007). Meta-model based development of use case model for business function. Information Technology and Control, 36(3), 302-309.

Gustas, R. (2010). A look behind conceptual modelling constructs in information system analysis and design. International Journal of Information System Modelling and Design (IJISMD), 1(1), 79-108.

Gyongyi, Z., Garcia-Molina, H., & Pedersen, J. (2006). Web content categorization using link information

Hailpern, B. & Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. IBM systems journal, 45(3), 451-461.

Halttunen, V., Lehtinen, A. & Nykänen. R. (2005). Building a Conceptual Skeleton for Enterprise Architecture Specifications. The Proceedings of the 15th European - Japanese Conference on Information Modelling and Knowledge Bases, Tallinn, Estonia, May 15-19.

Hao, W., Jun. G., & Xinning, S. (2013). Research on the Model and Its Application of Ontology-driven Knowledge Management System. Journal of Library Science in China.

Haring, L., & Ronald, D. (2011). Behaviour Driven development: Better than Test Driven Development. Java Magazine (Veen Magazines) (1): pp: 14–17.

Heflin, J., Volz, R., & Dale, J. (2013) Requirements for a Web Ontology Language, Working draft of the W3C Ontology Working Group. Available at: http://www.w3c.org/TR/webont-req. Accessed: September 2013

Heinz-Dieter K. (2003). Optimising Business Performance with Standard Software Systems. pp: 95.

Hevner A. R. (2007). The three cycle view of design science research. Scandinavian Journal of Information Systems. 19(2): 87

Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley

Höfferer, P. (2007). Achieving Business Process Model Interoperability Using Metamodels and Ontologies. In ECIS (pp. 1620-1631).

Hoogervorst, J. (2004). Enterprise Architecture: Enabling Integration, Agility and Change. International Journal of Cooperative Information Systems (13:3), 2004, pp. 213-233.
Horridge, M. (2009). A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition1. 2. University Of Manchester.
Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21, 79.
Huang, J., Abadi, D. J., & Ren, K. (2011). Scalable SPARQL querying of large RDF graphs. Proceedings of the VLDB Endowment, 4(11).
IEEE (1992). Codes of ethics. Available at: http://www.computer.org/cms/Computer.org/Publications/code-of-ethics.pdf. Accessed: December, 2013.
IEEE (1998). IEEE standard for software test documentation. Version 7, 1998. New York: IEEE - 1998.
IEEE (1998). Test Case Specification Template, Test Case Specification Identifier, (IEEE 829-1998).
IEEE (2000). IEEE Recommended Practice for Architectural Description of Software Intensive Systems. IEEE Standard 1471-2000.
Imran, S., Foping, F., Feehan, J., & Dokas, I. M. (2010). Domain specific modelling language for early warning system: using IDEF0 for domain analysis. IJCSI International Journal of Computer Science Issues, 7(5), 1694-0814.
Industry Advisory Council (2005). Advancing Enterprise Architecture Maturity, version 2.0. Developed for The Federal CIO Council (CIOC) by Industry Advisory Council (IAC).
ISO 9000, 9001, 9004, Available at: http://www.praxiom.com/iso-definition.htm. Accessed: April 2013.
ISO/IEC 42010:2007, Systems and Software Engineering – Recommended Practice for Architectural Description of Software-Intensive Systems, Edition 1.
ITU Recommendation X.901 | ISO/IEC 10746-1:1998, Information Technology – Open Distributed Processing – Reference Model – Part 1: Overview, International Telecommunication Union.
Iyer, B., & Gottlieb, R. (2004). The Four-Domain Architecture: An approach to support enterprise architecture design. IBM Systems Journal, 43(3), 587-97.
Jan L., & Dietz, G. (1999). DEMO: towards a discipline of Organisation Engineering. In: European Journal of Operations Research.
Jan L., & Dietz G. (2006). Enterprise Ontology - Theory and Methodology, Springer-Verlag Berlin Heidelberg. p.118.
Jan L., & Dietz, G. (2008). Architecture: Building strategy into design. Academic Service. ISBN 9789012580861 p.32
Johnson, P., Ullberg, J., Buschle, M., Franke, U., & Shahzad, K. (2014). An architecture modeling framework for probabilistic prediction. Information Systems and e-Business Management, 1-28.
Jones, D., & Gregor, S. (2007). The anatomy of a design theory. Journal of the Association for Information Systems, 8(5), 1.
Jonkers, H., van Burren, R., Arbab, F., De Boer, F., Bonsangue, M., Bosma, H., & van Zanten, G. V. (2003).. Towards a language for coherent enterprise architecture descriptions. In Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International (pp. 28-37). IEEE.
Jorgensen, H., Owen, L., & Neus, A. (2008). Making Change Work. Available at: http://www-935.ibm.com/services/us/gbs/bus/pdf/gbe03100-usen-03-making-change-work.pdf. Accessed: January, 2014.
Kaisler, S.H., Armour, F., & Valivullah, M. (2005). Enterprise Architecting: Critical Problems. Proceedings of the 38th Hawaii International Conference on System Sciences, HICSS'05. Hawaii, IEEE Computer Society.
Kang, D., Lee, J., Choi, S., & Kim, K. (2010). An Ontology-based enterprise architecture. Expert Systems with Applications, 37(2), 1456-1464.
Kaplan, R. S. & Norton, D. P. (2001). Transforming the balanced scorecard from performance measurement to strategic management: Part I. Accounting horizons, 15(1), 87-104.
Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., & Wimmer, M. (2006). Lifting metamodels to ontologies: A step to the semantic integration of modeling languages (pp. 528-542). Springer Berlin Heidelberg.
Kecheng L. (2002). Information, Organisation, and Technology: Studies in Organisational Semiotics. pp.198.
Keith, M., & Schincariol, M. (2013). Query Language. In Pro JPA 2 (pp. 193-226). Apress.
Kerzner, H. R. (2013). Project management: a systems approach to planning, scheduling, and controlling. John Wiley & Sons.
Khoury, G. R. (2007). A unified approach to enterprise architecture modelling. Doctoral dissertation, University of Technology, Sydney.
Kilov, H. (2004). Using RM-ODP to bridge communication gaps between stakeholders. In Workshop on ODP for Enterprise Computing (WODPEC04), California USA September.

Kim, C. H., Weston, R. H., Hodgson, A., & Lee, K. H. (2003). The complementary use of IDEF and UML modelling approaches. Computers in Industry, 50(1), 35-56.
Klein, J., & Gagliardi, M. (2010). A Workshop on Analysis and Evaluation of Enterprise Architectures. Carnegie Mellon Software Engineering Institute (SEI) in April of 2010. Available at: http://www.sei.cmu.edu.
Kobryn, C. (1999). UML 2001: a standardization odyssey. Communications of the ACM, 42(10), 29-37.
Kollia, I., Glimm, B., & Horrocks, I. (2011). SPARQL query answering over OWL ontologies. In The Semantic Web: Research and Applications (pp. 382-396). Springer Berlin Heidelberg.
Koning H. (2008). Communication of IT-Architecture. Thesis Dutch Research School for Information and Knowledge Systems. ISBN 978-90-5335-163-5. pp.94.
Kosanke, K. & Zelm, M. (2014). CIMOSA European Enterprise Integration Concept Last updated December 29, 2013 Available at: http://www.cimosa.de/. Accessed: January 2014.
Kosar, T., Oliveira, N., Mernik, M., Pereira, V.J.M., Crepinsek, C.D. Da, & Henriques, R.P. (2010). Comparing general-purpose and domain-specific languages: An empirical study. Computer Science and Information Systems, 7(2): pp. 247- 264.
Kourouthanassis, P. E., & Giaglis, G. M. (2006). A Design Theory for Pervasive Information Systems. In IWUC (pp. 62-70).
Krogstie J. (2008). Using EEML for Combined Goal and Process Oriented Modelling: A Case Study. IDI, NTNU, Trondheim, Norway. Proceedings of EMMSAD.
Kruppke, H., Jost, W., & Kindermann, H. (2006). ARIS—Software, Method and Instrument. In AGILITY by ARIS Business Process Management. pp. 3-10. Springer Berlin Heidelberg.
Kuechler B, & Vaishnavi V. (2008). On theory development in design science research: Anatomy of a research project. European Journal of Information Systems. 17(5): 489–504.
Kulkarni, V., Roychoudhury, S., Sunkle, S., Clark, T., & Barn, B. (2013). Modeling and enterprises-the past, the present, and the future. MODELSWARD'13.
Kumar, S. K., & Harding, J. A. (2013). Ontology mapping using description logic and bridging axioms. Computers in Industry, 64(1), pp. 19-28.
Lacher, M. S., & Groh, G. (2001). Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. In FLAIRS conference. pp. 305-309.
Lakhrouit, J., Baïna, K., & Benali, K. (2014). Model and Application Architecture Indicators of Evaluation the Enterprise Architecture. In New Perspectives in Information Systems and Technologies, Volume 2 (pp. 63-71). Springer International Publishing.
Lam, W. (2005). Investigating success factors in enterprise application integration: a case-driven analysis. European Journal of Information Systems (2005:14). pp. 175-187.
Lankhorst, M. M. (2013). Enterprise Architecture at Work: Modelling, Communication and Analysis, Springer.
Lapouchnian, A. (2005). Goal-oriented requirements engineering: An overview of the current research. University of Toronto.
Larman, C. (2012). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e. Pearson Education India.
Lecklin, O. (2002). Laatu yrityksen menestystekijänä (Quality as a company's success factor), Gummerus, 2002.
Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G. (2001). Composing domain-specific design environments. Computer, 34(11), pp. 44-51.
Lederer, M., Schott, P., Huber, S., & Kurz, M. (2013). Strategic Business Process Analysis: A Procedure Model to Align Business Strategy with Business Process Analysis Methods. In S-BPM ONE-Running Processes (pp. 247-263). Springer Berlin Heidelberg, pp. 41.
Lochmann H. & Hessellund, A. (2009). An integrated view on modelling with multiple domain-specific languages. In Software Engineering. ACTA Press.
Logica Management Consulting (2014). Failing business process change projects substantially impact financial performance of UK business, Available at: http://www.cgi.com/en. Accessed: January 2014.
Lopez, M. (2000). An Evaluation Theory Perspective of the Architecture Trade-off Analysis Method (ATAM)", Technical Report CMU/SEI-2000-TR-012. The Software Engineering Institute, Carnegie Mellon University.
Maarten W., & Herman H. (2010). The Integrated Architecture Framework Explained: Why, What, How. pp. 157.
Mabey, B. (2008). Imperative vs. Declarative Scenarios in user stories. Available at: http://benmabey.com/2008/05/19/imperative-vs-declarative-scenarios-in-user-stories.html. Accessed: May 2013.
Machado, M. (2013). Balanced Scorecard: an empirical study of small and medium size enterprises. RBGN Review of Business Management, 15(46), 129-148.
March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. Decision support systems, 15(4), 251-266.

Markus, M. L., Majchrzak, A., & Gasser, L. (2002). A design theory for systems that support emergent knowledge processes. Mis Quarterly, 179-212.
Martin, R. A., Robertson, E. L., & Springer, J. A. (2004). Architectural Principles for Enterprise Frameworks. In CAiSE Workshops (1). pp. 151-162.
Martin, R., & Robertson, E. (2000). A Formal Enterprise Architecture Framework to Support Multi-model Analysis. In Proceedings of the 5th CAiSE/IFIP8 (Vol. 1).
Maxwell, S. E., & Delaney, H. D. (2004). Designing experiments and analyzing data: A model comparison perspective (2nd edition.). Mahwah, NJ. Lawrence Erlbaum Associates.
McGuinness, D. L. (2002). Ontologies Come of Age, In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, Spinning the Semantic Web: Bringing the WWW to Its Full Potential. MIT Press.
McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, 10(2004-03), pp. 10.
McShane, M., & Nirenburg, S. (2013). Use of Ontology, Lexicon and Fact Repository for Reference Resolution in Ontological Semantics. In New Trends of Research in Ontologies and Lexical Resources (pp. 157-185). Springer Berlin Heidelberg.
Menzel, C., & Mayer, R. J. (2006). The IDEF family of languages. In Handbook on architectures of information systems (pp. 215-249). Springer Berlin Heidelberg.
Mertins, K., & Jaekel, F. W. (2006). MO²GO: User Oriented Enterprise Models for Organizational and IT Solutions. In: Bernus, P.; Mertins, K.; Schmidt, G.: Handbook on Architectures of Information Systems. Second Edition. Springer-Verlag Berlin. ISBN 3-540-25472-2.
Moody, D. L. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. Software Engineering, IEEE Transactions on, 35(6), 756-779.
Morganwalp, J.M., & Sage, A.P. (2004). Enterprise Architecture Measures of Effectiveness." International Journal of Technology, Policy and Management (4:1), pp. 81-94.
Moss, H. K. (2007). Improving Service Quality with the Theory of Constraints. Journal of the Academy of Business & Economics, 7(3).
Motik, B., & Sattler, U. (2006). A Comparison of Reasoning Techniques for Querying Large Description Logic Aboxes. In Proceedings of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning(Phnom Penh, Cambodia.
Motwani, J., Prasad, S. & Tata, J. (2005). The Evolution of TQM: An Empirical Analysis Using the Business Process Change Framework. The TQM Magazine (17:1), pp. 54-66.
Muller, T., Black R., Eldh, S., Graham D., Olsen K., Pyhajarvi, M., Thompson G., & Veendendal E. (2005). Certified Tester Foundation Level Syllabus, International Software Testing Qualifications Board.
Myers, A., Glenford J. (1979). The Art of Software Testing. John Wiley and Sons. pp. 145–146.
National Association of State Chief Information Officers (NASCIO), "NASCIO Enterprise Architecture Maturity Model, v. 1.3". (2003). Available at: https://www.nascio.org/publications/index.cfm. Accessed January 2014.
Neaga, E. I., & Harding, J. A. (2005). An enterprise modelling and integration framework based on knowledge discovery and data mining. International Journal of Production Research, 43(6), 1089-1108.
Nell, J.G., NIST. (1997). An Overview of GERAM. ICEIMT'97 International Conference on Enterprise Integration Modelling Technology.
Noran, O. (2003). An Analysis of the Zachman Framework for Enterprise Architecture from the GERAM perspective. Annual Reviews in Control, 27, pp.163-183.
North, D. (2006). Introducing BDD. Available at: http://dannorth.net/introducing-bdd/. Accessed: August 2012.
Noy, N. (2012). Ontology mapping and alignment. In Fifth International Workshop on Ontology Matching collocated with the 9th International Semantic Web Conference ISWC-2010, Shangai, China.
Noy, N. F. (2004). Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record, 33(4), pp. pp. 65-70.
Office of Management and Budget. (2005). OMB Enterprise Architecture Assessment Framework Version 1.5. OMB FEA Program Management Office; The Executive Office of the President, USA.
OMG. Available at: http://www.omg.org/. Assessed: December 2013.
OMG (2008): Model Driven Architecture. Technical Report, Available at: http://www.omg.org/cgi-bin/doc?omg/. Accessed: February, 2012.
Pacheco, D. (2014). Theory Of Constraints And Six Sigma: Convergences, Divergences And Research Agenda For Continuous Improvement. Independent Journal of Management & Production, 5(2), 331-343.
Parreiras, F. S., Staab, S., & Winter, A. (2007). On marrying ontological and metamodeling technical spaces. In ESEC-FSE '07: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pages 439–448, New York, NY, USA. ACM.

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation. In Design Science Research in Information Systems. Advances in Theory and Practice (pp. 398-410). Springer Berlin Heidelberg.
Perkins, A. (2003). Critical Success Factors for Enterprise Architecture Engineering. Visible Solutions.
Popescu, G., & Wegmann, A. (2014). Using the Physics of Notations Theory to Evaluate the Visual Notation of SEAM. In IEEE 16th Conference on Business Informatics (CBI 2014) (No. EPFL-CONF-198951).
Prasse, M. (1998). Evaluation of Object-Oriented Modelling Languages: A Comparison between OML and UML. In: M. Schader, A. Korthaus (Eds.): The Unified Modelling Language -Technical Aspects and Applications, Heidelberg: Physica, pp. 58-78.
Pulkkinen, M. & Hirvonen, A. (2005). EA Planning, Development and Management Process for Agile Enterprise Development. In: Sprague, R.H. Jr: Proceedings of the Thirty-Eighth Annual Hawaii International Conference on System Sciences. Big Island, Hawaii, 2005, IEEE Computer Society.
Quartel, D., Engelsman, W., Jonkers, H., & Van Sinderen, M. (2009). A goal-oriented requirements modelling language for enterprise architecture. In Enterprise Distributed Object Computing Conference. EDOC'09. IEEE International, pp. 3-13.
Quilitz, B., & Leser, U. (2008). Querying distributed RDF data sources with SPARQL. In The Semantic Web: Research and Applications, pp. 524-538. Springer Berlin Heidelberg.
Raghupathi, W. (2007). Corporate governance of IT: A framework for development. Communications of the ACM, 50(8), 94-99.
Rahm, E. (2011). Towards large-scale schema and ontology matching. In Schema matching and mapping (pp. 3-27). Springer Berlin Heidelberg.
Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., & Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In Engineering Knowledge in the Age of the Semantic Web (pp. 63-81). Springer Berlin Heidelberg.
Regev, G., Bajic-Bizumic, B., Golnam, A., Popescu, G., Tapandjieva, G., Saxena, A. B., & Wegmann, A. (2013). A Philosophical Foundation for Business and IT Alignment in Enterprise Architecture with the Example of SEAM. In Proceedings of the Third International Symposium on Business Modeling and Software Design (No. EPFL-CONF-198962, pp. 131-139). SCITEPRESS-Science and Technology Publications.
Rehkopf, T.W. & Wybolt, N. (2003). Top 10 Architecture Land Mines. IT Professional (5:6), pp. 36-43.
Riege, C., & Aier, S. (2009). A contingency approach to enterprise architecture method engineering. In Service-Oriented Computing–ICSOC 2008 Workshops (pp. 388-399). Springer Berlin Heidelberg.
Rosati, R., & Almatelli, A. (2010). Improving Query Answering over DL-Lite Ontologies. KR, 10, 51-53.
Roth, S., Hauder, M., Farwick, M., Breu, R., & Matthes, F. (2013). Enterprise architecture documentation: Current practices and future directions.
Rudawitz, D. (2003). Why Enterprise Architecture Efforts Often Fall Short. EA Community Whitepaper.
Salmans, B., & Kappelman, L. A. (2010). The State of EA: Progress, Not Perfection. The SIM guide to enterprise architecture, pp.165-187.
Sargent, R. G. (2005). Verification and validation of simulation models. In Proceedings of the 37th conference on Winter simulation (pp. 130-143). Winter Simulation Conference.
Sarker, S., & Lee, A. S. (2002). Using a positivist case research methodology to test three competing theories-in-use of business process redesign. Journal of the Association for Information Systems, 2(1), 7.
Scheer, A. W., & Nüttgens, M. (2000). ARIS architecture and reference models for business process management, pp. 376-389. Springer Berlin Heidelberg.
Schekkerman, J. (2003). How to survive in the jungle of enterprise architecture frameworks: Creating or choosing an enterprise architecture framework. Trafford Publishing.
Schekkerman, J. (2004). Enterprise Architecture Validation - Achieving Business-Aligned and Validated Enterprise Architectures. Institute for Enterprise Architecture Developments. Available at: http://www.enterprise-architecture.info/. Accesses: January 2014.
Schneider, A. W., Schulz, C., & Matthes, F. (2013). Goals in Enterprise Architecture Management--Findings from Literature and Future Research Directions. In Business Informatics (CBI), 2013 IEEE 15th Conference on (pp. 284-291). IEEE.
Sessions, R. (2007). A Comparison of the Top Four Enterprise-Architecture Methodologies, Object Watch, Inc.
Simon, D., Fischbach, K., & Schoder, D. (2014). Enterprise architecture management and its role in corporate strategic management. Information Systems and e-Business Management, 12(1), 5-42.
Sintek, M., & Decker, S. (2002). TRIPLE - A query, inference, and transformation language for the semantic web. In The Semantic Web—ISWC 2002, pp. 364-378. Springer Berlin Heidelberg.
Siponen, M. (2006). Six design theories for IS security policies and guidelines. Journal of the Association for Information systems, 7(1), 19.

Sirin, E., & Parsia, B. (2007). SPARQL-DL: SPARQL query for OWL-DL. In Golbreich, C., Kalyanpur, A., & Parsia, B. (Eds.), Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Vol. 258 of CEUR Workshop Proceedings. CEURWS. org.

Smart J. F. (2013). BDD in Action, Behaviour-driven development for the whole software life cycle, MEAP Edition, Manning Early Access Program, BDD in Action, Manning Publications.

Smartlogic (2013). Available at: http://www.smartlogic.com/home/products/products-overview. Accessed: December 2013.

Sogetti (2011). Fields covered by DYA. Available at: www.Dya.info. Accessed: July, 2013.

Soley, R. (2000). Model driven architecture. OMG white paper, 308.

Stanford University (2013). Protégé Version 3.5. Available at: http://protege.stanford.edu/. Accessed: September, 2013.

Stanley, R., & Uden, L. (2013). Why Projects Fail, from the Perspective of Service Science. In 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing, pp. 421-429. Springer Berlin Heidelberg.

Steyn, H. (2002). Project management applications of the theory of constraints beyond critical chain scheduling. International Journal of Project Management,20(1), 75-80.

Stumme, G., & Maedche, A. (2001, August). FCA-Merge: Bottom-up merging of ontologies. In IJCAI (Vol. 1, pp. 225-230).

Stvilia, B. (2007). A model for ontology quality evaluation. First Monday, 12(12).

Sun, Y., & Kantor, P. B. (2006). Cross-Evaluation: A new model for information system evaluation. Journal of the American Society for Information Science and Technology, 57(5), 614-628.

Tang, A., Han, J., & Chen, P. (2006). A Comparative Analysis of Architecture Frameworks, School of Information Technology, Centre for Component Software Volume VII, No. 2, 23 Issues in Information Systems.

Taplin, D., Clark, H., Collins E., & Colby, D. (2013). Technical Papers: A Series of Papers to support Development of Theories of Change Based on Practice in the Field. New York: Actknowledge and The Rockefeller Foundation.

Tarí, J.J. (2005). Components of successful total quality management. The TQM Magazine (17:2), pp. 182-194.

TEAF (2005). The Enterprise Architecture Framework. Systems & Software Consortium. Available at: www.software.org/pub/architecture/teaf.asp, Accessed: April 2013.

Terry W. (2013). Available at: http://hci.stanford.edu/winograd/papers/language-action.html, Human-Computer Interaction 3:1 (1987-88), pp.3-30. Accessed: June 2013.

The Atlantic Zoo. Available at: http://www.emn.fr/z-info/atlanmod/index.php/Zoos. Assessed: February 2015.

The Business Rules Group. (2010). The Business Motivation Model, Business Governance in a Volatile World. Available at: http://www.businessrulesgroup.org/second_paper/BRG-BMM.pdf. Accessed: June, 2014.

Theuerkorn, F. (2004). Lightweight enterprise architectures. CRC Press.

TOG (2013). Version 9.1. The Open Group. Available at: http://www.opengroup.org/togaf/. Accessed: June 2013.

TOG (2012). ArchiMate 2.1 Specification . The Open Group. Available at: http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html. Accessed: July 2014.

Turner, J.R., & Müller, R. (2005). The Project Manager's Leadership Style As a Success Factor on Projects: A Literature Review. Project Management Journal (36:1), pp. 49-61.

TURTLE (2013). Available at: http://www.w3.org/TeamSubmission/turtle/#sec-tutorial. Accessed: September 2013.

Ulbrich, F. (2006). Improving shared service implementation: adopting lessons from the BPR movement. Business Process Management Journal, 12(2), 191-205.

Unified Modelling Language: Infrastructure, Version 2.1.2, formal/2007-11-04, Object Management Group, March 2006. Available at: http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF. Accessed: December 2013.

Unified Modelling Language: Superstructure, Version 2.2, formal/2009-02-02. Object Management Group, August 2005. Available at: http://www.omg.org/spec/UML/2.2/Superstructure/PDF. Accessed: December 2013.

United States Department of Commerce (2003). IT Architecture Capability Maturity Model. United States Government Accountability Office, "A Framework for Assessing and Improving Enterprise Architecture Management, V. 1.1", Government Accountability.

Urbaczewski, L., & Mrdalj, S. (2006). A comparison of enterprise architecture frameworks. Issues in Information Systems, 7(2), 18-23.

Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11(2).

Van Aken J. E. (2005). Management research as a design science: Articulating the research products of mode 2 knowledge production in management". Br J Manage. 16(1): 19–36

van der Raadt, B., Soetendal, J., Perdeck, M., & van Vliet, H. (2004). Polyphony in Architecture. Proceedings of the 26th International Conference on Software Engineering, IEEE Computer Society.

Van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: an annotated bibliography. SIGPLAN Not. 35(6):26, pp. 36.

Van Grembergen, W., & Saull, R. (2001). Information technology governance through the balanced scorecard. In Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS), CD-ROM. Maui.

Vazifedoost, A. R., Oroumchian, F., & Rahgozar, M. (2007). Finding similarity relations in presence of taxonomic relations in ontology learning systems. In Computational Intelligence and Data Mining, CIDM 2007. IEEE Symposium on (pp. 215-220). IEEE.

Venable, J. R. (2010). Design science research post hevner et al.: criteria, standards, guidelines, and expectations. In Global Perspectives on Design Science Research (pp. 109-123). Springer Berlin Heidelberg.

Venable, J., Pries-Heje, J., & Baskerville, R. (2012). A comprehensive framework for evaluation in design science research. In Design Science Research in Information Systems. Advances in Theory and Practice (pp. 423-438). Springer Berlin Heidelberg.

Venkatraman, N., & Henderson, J. (2010). Strategic Alignment: Leveraging IT for Transforming Organisations, IBM Systems Journal, Vol 32 No 1.

Vernadat, F. B. (2014). Enterprise Modeling in the context of Enterprise Engineering: State of the art and outlook. International Journal of Production Management and Engineering, 57.

Veryard R., & Ian G. MacDonald (1994). EMM/ODP: A methodology for federated and distributed systems. In: Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, Maastricht, The Netherlands. Pp. 26–28.

Vicente, M., Gama, N., & da Silva, M. M. (2013). Using ArchiMate and TOGAF to Understand the Enterprise Architecture and ITIL Relationship. In Advanced Information Systems Engineering Workshops (pp. 134-145). Springer Berlin Heidelberg.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). Ontology-based integration of information-a survey of existing approaches. In IJCAI-01 workshop: ontologies and information sharing (Vol. 2001, pp. 108-117).

Wagenhals, L. W., Levis, A. H. (2009). Service oriented architectures, the DoD architecture framework 1.5, and executable architectures. Systems Engineering, 12(4), 312-343.

Walls, J. G., Widermeyer, G. R., & El Sawy, O. A. (2004). Assessing information system design theory in perspective: how useful was our 1992 initial rendition?. Journal of Information Technology Theory and Application (JITTA), 6(2), 6.

Walter, T., Parreiras, F. S., Staab, S. (2009). OntoDSL: An ontology-based framework for domain-specific languages. In Andy Schürr and Bran Selic, editors, MODELS, volume 5795 of Lecture Notes in Computer Science, pages 408–422. Springer.

Wan, H., Luo, X., Johansson, B., & Chen, H. (2013). Enterprise architecture benefits: the divergence between its desirability and realizability. In 14th International Conference on Informatics and Semiotics in Organizations (ICISO2013, IFIP WG 8, 1 Working Conference). SciTePress.

Wang S., Jin. L. Jin. C., (2006). Ontology definition Metamodel based consistency checking of uml models. pages 1 –5.

Wegmann, A. (2002). The Systemic Enterprise Architecture Methodology (SEAM). Business and IT Alignment for Competitiveness (No. LAMS-REPORT-2002-009).

Wegner, P., & Eberbach, E. (2004). New models of computation. The Computer Journal, 47(1), 4-9.

Weiss, S., Aier, S., & Winter, R. (2013). Institutionalization and the Effectiveness of Enterprise Architecture Management.

Weiss, S., Aier, S., & Winter, R.(2012). Towards a Reconstruction of Theoretical Foundations of Enterprise Architecture Management, De Marco, M., Te'eni, D., Albano, V., Za, S. (Eds.): Information Systems: Crossroads for Organization, Management, Accounting and Engineering, Rome, Physica-Verlag, Heidelberg, pp. 461-468.

Weston, J., & Defee, J. (2004). Performance Based Enterprise Architecture Planning – A white Paper, 2004, Available at: http://www.caci.com/, Accessed: April 2013.

Winter, R., & Fischer, R. (2007). Essential layers, artifacts, and dependencies of enterprise architecture. Journal of Enterprise Architecture, 3(2), pp.7-18.

Wongrassamee, S., Simmons, J. E. L., & Gardiner, P. D. (2003). Performance measurement tools: the Balanced Scorecard and the EFQM Excellence Model. Measuring Business Excellence, 7(1), 14-29.

Wout, J., Waage, M., Hartman, H., Stahlecker, M., & Hofman, A. (2010). The Integrated Architecture Framework Explained.

Wynne, M., Hellesoy, A. (2012). The cucumber book: behaviour-driven development for testers and developers. Pragmatic Bookshelf.

Xu, J., & Feng, C. (2014). Multimode Resource-Constrained Multiple Project Scheduling Problem under Fuzzy Random Environment and Its Application to a Large Scale Hydropower Construction Project. The Scientific World Journal.

Ylimäki, T. (2008). Potential critical success factors for enterprise Architecture Evaluation of enterprise and software architectures: critical issues, metrics and practices:[AISA Project 2005-2008]/ Eetu Niemi, Tanja Ylimäki & Niina Hämäläinen (eds.). Jyväskylä: University of Jyväskylä, Information Technology Research Institute, 2008.-(Tietotekniikan tutkimusinstituutin julkaisuja, ISSN 1236-1615; 18). ISBN 978-951-39-3108-7 (CD-ROM).

Ylimäki, T., Halttunen, V., Pulkkinen, M., & Lindström, T. (2005). Methods and Tools for Enterprise Architecture. Larkki Project October 2001 - April 2005. Publications of the Information Technology. Research Institute 16, University of Jyväskylä. Available at: http://www.titu.jyu.fi/larkkipublication.

Zachman, J. (1987). A Framework for Information Systems Architecture. IBM Systems Journal, 26(3), IBM Publication G321-5298, pp. 276-292.

Zachman, J.A., & Sowa, J.F. (1992). Extending and Formalizing the Framework for Information Systems Architecture, IBM Systems Journal, Volume 31, No. 3, pp. 590-616.

Zachman, J. (2002). The Zachman framework for enterprise architecture. Zachman International.

Zachman, J. (2008). John Zachman's Concise Definition of The Zachman Framework. Zachman International. 2008. Available at: http://www.zachman.com/about-the-zachman-framework, Accessed: April, 2013.

Zimmerman, J., Stolterman, E., & Forlizzi, J. (2010). An analysis and critique of Research through Design: towards a formalization of a research approach. In Proceedings of the 8th ACM Conference on Designing Interactive Systems (pp. 310-319). ACM.

Appendix A: Ethical Considerations

Ethical concerns have been taken into consideration in carrying out this research to ensure that the study does not consciously or unconsciously contravene any legislation or cause physical, emotional or social harm to anyone. Cognizance of the fact that there is no direct participation of users in this research, no requirement for sensitive information, no need for deployment of safety precautions or critical systems, requirement for obtaining ethical approval is inapplicable. Participants who are part of any collaborations as in the case study used or for the purpose of sharing opinions within the confines of this research directly or indirectly have been notified about the purpose of this research, processes involved, outcomes and/or benefits arising from the research. Any data used or applications deployed has been in accordance with the Data Protection Act (1998). Confidentiality is maintained throughout the entire course of this research and participants are treated with respect, fairness and equality. Participants had the option to withdraw from the study at any time without any consequences and the option to retain their data or destroy all their data given. Finally, as the researcher, I have conducted myself under the code of conduct for the British Computer Society (BCS) (BCS, 2006), the code of ethics for the Association for Computing Machinery (ACM, 1992) and Institute of Electrical and Electronics Engineers (IEEE) Computer Society (IEEE, 1992).

Appendix B: Papers, Presentations and Seminars

Over the course of this research, the following work has been presented in the form of publications.

Kim, H., Oussena, S., Essien, J., & Komisarczuk, P. 2013. Towards Event-Driven Enterprise Architecture. Uncovering Essential Software Artifacts through Business Process Archaeology, P.285.

Samia Oussena, Joe Essien. , 2013. Validating Enterprise Architecture Using Ontology-Based Approach. Publication in the 3rd International Symposium ISKO-Maghreb, 2013.

Samia Oussena, Joe Essien, Peter Komisarczuk, 2014. Formalization of Validation Extension Metamodel for Enterprise Architecture Frameworks, Publication in the 16th International Conference on Enterprise Information System 2014.

Essien, J., Oussena, S., 2013. Enterprise Architecture Models: Description of Integrated Components for Validation - A Case Study of Student Internship Programme. Submitted and accepted as a Doctoral Consortium on Enterprise Information Systems at the 17th International Conference on Enterprise Information Systems - ICEIS 2013

Papers completed and undergoing review with co-authors before publishing.

Joe Essien, Samia Oussena, Peter Komisarczuk, Stephen Roberts: Model Driven Validation Approach for Enterprise Architecture and Motivation Extensions.

Joe Essien, Samia Oussena: Ontology Based Approach for Validating Heterogeneous Enterprise Architecture Models.

Appendix C: Model-Driven Validation Approach (MDVA) workflow process

The following design methodology defines the steps that need to be taken in order to adopt the MDVA:

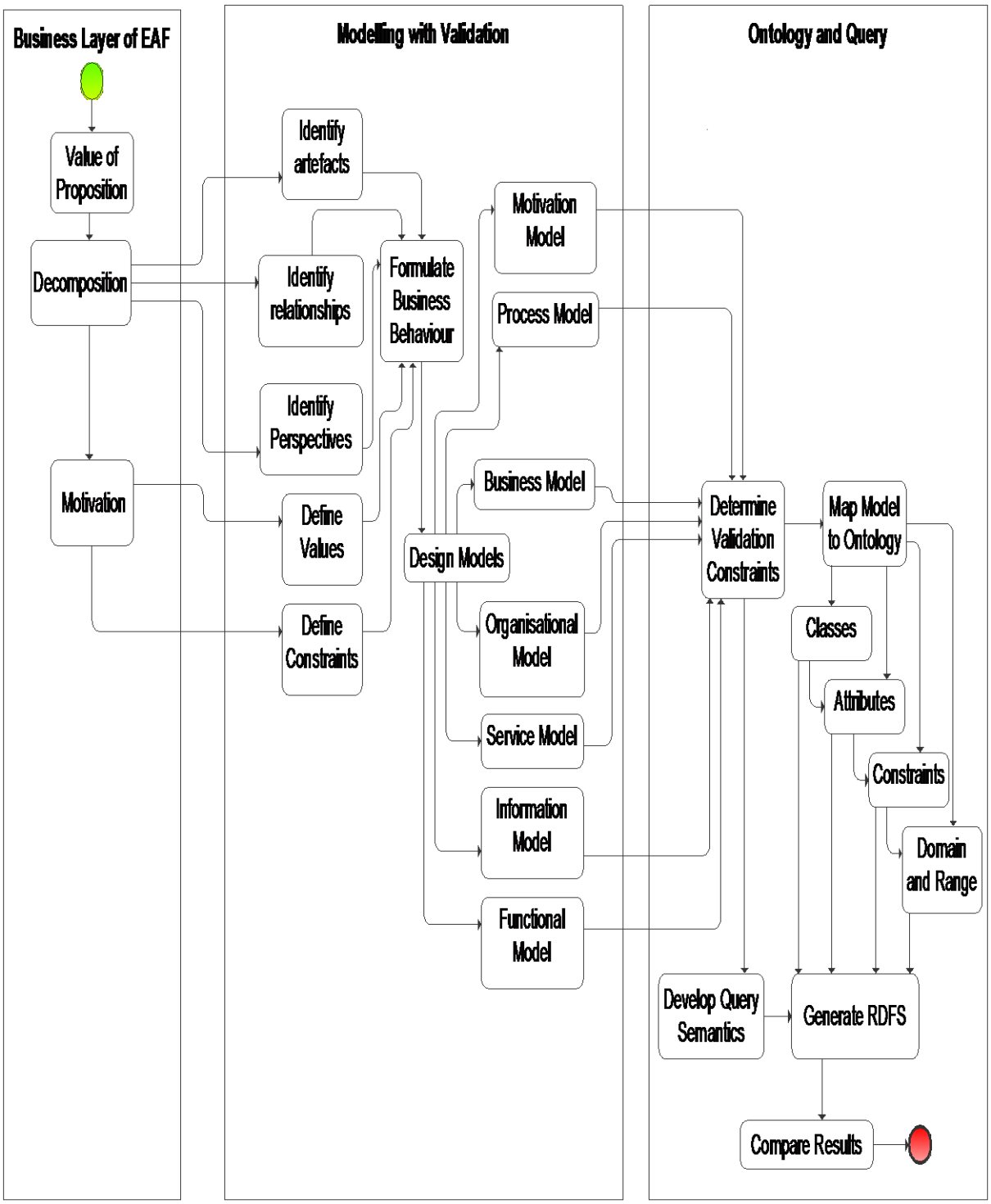

Figure C1: Workflow Diagram for the MDVA

STEP 1: Determine the aspects of the Business Layer that needs to be validated

- **a)** Identify artefacts
- **b)** Identify relationships between artefacts
- **c)** Identify the perspectives to be modelled (Stakeholders and Roles)

STEP 2: Define Motivation for the Proposition

- **a)** Define the values, policies, events, contents that are involved.
- **b)** Define the constraints, principles, assessment criteria, goals and requirements
- **c)** Relate the 2a to 2b to conceptualise Business Behaviour.

STEP 3: Design models with Validation Extension based on perspectives

a) Model required may be based on aspects, stakeholder or any of following;

i. Motivation models
ii. Process models
iii. Business Models
iv. Organizational model
v. Information Model
vi. Functional and Service model

STEP 4: Developing Behaviour Driven Modelling Constraints Specification

- **a)** Develop Assumptions
- **b)** Develop Features
- **c)** Develop Scenarios based on constraints and criteria
- **d)** Develop Test Data
- **e)** Develop Triples (Subject, Predicate, Object)

STEP 5: Mapping Artefacts of the Model to Ontology Elements

- **a)** For each artefact, identify corresponding ontology element
- **b)** Create all ontology classes and subclasses
- **c)** Create all ontology properties
- **d)** Associate properties to classes and subclasses
- **e)** Establish Domains and Ranges

STEP 6: Determine how the ontology will be queried and traceability achieved.

- **a)** Discover key stages of business behaviour.
- **b)** Transform BDD test specifications for the business behaviour to SPARQL queries
- **c)** Query the RDFS created by the ontology
- **d)** Query the RDF Graph using Reasoners to establish Traceability

STEP 7: Compare the Results of the query and Traceability Graph

- **a)** Affirm that Constraints are implemented
- **b)** Affirm that artefacts associated with Goals are realized in the traceability graph
- **c)** Affirm that Goals are aligned with associated requirements

Appendix D: RDFS for Validation Extension Metamodel

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
   <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY assert "http://www.owl-ontologies.com/assert.owl#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
   <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
   <!ENTITY swrla "http://swrl.stanford.edu/ontologies/3.3/swrla.owl#" >
   <!ENTITY sqwrl "http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/URIVEM#"
    xml:base="http://www.owl-ontologies.com/URIVEM"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#">
   <owl:Ontology rdf:about="">
      <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
      <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
      <owl:imports rdf:resource="http://www.owl-ontologies.com/assert.owl"/>
   </owl:Ontology>
   <owl:ObjectProperty rdf:ID="A11_restricted_by">
      <rdfs:domain rdf:resource="#Constraint"/>
      <rdfs:range rdf:resource="#Principles"/>
      <rdfs:subPropertyOf rdf:resource="#A1_has_interest"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="A1_has_interest">
      <rdfs:domain rdf:resource="#Stakeholder"/>
```

```
    <rdfs:range rdf:resource="#Principles"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A2_analysed_by">
    <rdfs:domain rdf:resource="#Principles"/>
    <rdfs:range rdf:resource="#Assessment"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A3_decomposed_to">
    <rdfs:domain rdf:resource="#Assessment"/>
    <rdfs:range rdf:resource="#Goal"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A4_specified_by">
    <rdfs:domain rdf:resource="#Goal"/>
    <rdfs:range rdf:resource="#Requirement"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A_formalised_into">
    <rdfs:domain rdf:resource="#Requirement"/>
    <rdfs:range rdf:resource="#Composite_Motivation"/>
</owl:ObjectProperty>
<rdfs:Class rdf:ID="Actor">
    <rdfs:subClassOf rdf:resource="#Business_Role"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Assessment">
    <rdfs:subClassOf rdf:resource="#Composite_Motivation"/>
</rdfs:Class>
<owl:ObjectProperty rdf:ID="B11_associated_with">
    <rdfs:domain rdf:resource="#Business_Object"/>
    <rdfs:range rdf:resource="#Meaning"/>
    <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B12_aggregated_with">
    <rdfs:domain rdf:resource="#Product"/>
    <rdfs:range rdf:resource="#Contract"/>
    <rdfs:subPropertyOf rdf:resource="#B21_aggregated_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B12_specified_by">
    <rdfs:domain rdf:resource="#Business_Object"/>
    <rdfs:range rdf:resource="#Contract"/>
    <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="B13_assigned_to">
   <rdfs:domain rdf:resource="#Location"/>
   <rdfs:range rdf:resource="#Representation"/>
   <rdfs:subPropertyOf rdf:resource="#B511_assigned_to"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B13_realised_by">
   <rdfs:domain rdf:resource="#Business_Object"/>
   <rdfs:range rdf:resource="#Representation"/>
   <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B1_available_in">
   <rdfs:domain rdf:resource="#BDD_Validation_Element"/>
   <rdfs:range rdf:resource="#Business_Object"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B211_associated_with">
   <rdfs:domain rdf:resource="#Product"/>
   <rdfs:range rdf:resource="#Value"/>
   <rdfs:subPropertyOf rdf:resource="#B21_aggregated_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B21_aggregated_by">
   <rdfs:domain rdf:resource="#Busines_Service"/>
   <rdfs:range rdf:resource="#Product"/>
   <rdfs:subPropertyOf rdf:resource="#B2_accessed_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B2_accessed_by">
   <rdfs:domain rdf:resource="#Business_Object"/>
   <rdfs:range rdf:resource="#Busines_Service"/>
   <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B2_realized_by">
   <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
   <rdfs:range rdf:resource="#Busines_Service"/>
   <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B31_specialized_by">
   <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
   <rdfs:range rdf:resource="#Interaction"/>
   <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B32_specialized_by">
   <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
   <rdfs:range rdf:resource="#Process"/>
```

```
    <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B33_specialized_by">
    <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:range rdf:resource="#Function"/>
    <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_accessed_by">
    <rdfs:domain rdf:resource="#Business_Object"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_conforms_with">
    <rdfs:domain rdf:resource="#BDD_Validation_Element"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_effectuality">
    <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_integrity">
    <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_triggered_by">
    <rdfs:domain rdf:resource="#Business_Event"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B4_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B3_used_by">
    <rdfs:domain rdf:resource="#Busines_Service"/>
    <rdfs:range rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:subPropertyOf rdf:resource="#B2_accessed_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B4_accessed_by">
    <rdfs:domain rdf:resource="#Business_Object"/>
    <rdfs:range rdf:resource="#Business_Event"/>
    <rdfs:subPropertyOf rdf:resource="#B1_available_in"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B4_dependency_of">
```

```
    <rdfs:domain rdf:resource="#BDD_Validation_Element"/>
    <rdfs:range rdf:resource="#Business_Event"/>
    <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B4_realized_by">
    <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
    <rdfs:range rdf:resource="#Business_Event"/>
    <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B511_assigned_to">
    <rdfs:domain rdf:resource="#Actor"/>
    <rdfs:range rdf:resource="#Location"/>
    <rdfs:subPropertyOf rdf:resource="#B51_assigned_to"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B51_aggregated_by">
    <rdfs:domain rdf:resource="#Collaboration"/>
    <rdfs:range rdf:resource="#Actor"/>
    <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B51_assigned_to">
    <rdfs:domain rdf:resource="#Business_Role"/>
    <rdfs:range rdf:resource="#Actor"/>
    <rdfs:subPropertyOf rdf:resource="#B5_authenticated_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B52_associated_with">
    <rdfs:domain rdf:resource="#Busines_Service"/>
    <rdfs:range rdf:resource="#Interface"/>
    <rdfs:subPropertyOf rdf:resource="#B2_accessed_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B52_used_by">
    <rdfs:domain rdf:resource="#Business_Role"/>
    <rdfs:range rdf:resource="#Interface"/>
    <rdfs:subPropertyOf rdf:resource="#B5_authenticated_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B53_specialized_by">
    <rdfs:domain rdf:resource="#Business_Role"/>
    <rdfs:range rdf:resource="#Collaboration"/>
    <rdfs:subPropertyOf rdf:resource="#B5_authenticated_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B5_aggregated_by">
    <rdfs:domain rdf:resource="#Collaboration"/>
    <rdfs:range rdf:resource="#Business_Role"/>
    <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B5_assigned_to">
   <rdfs:domain rdf:resource="#Business_Behaviour_Element"/>
   <rdfs:range rdf:resource="#Business_Role"/>
   <rdfs:subPropertyOf rdf:resource="#B3_conforms_with"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B5_authenticated_by">
   <rdfs:domain rdf:resource="#BDD_Validation_Element"/>
   <rdfs:range rdf:resource="#Business_Role"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B5_composed_of">
   <rdfs:domain rdf:resource="#Interface"/>
   <rdfs:range rdf:resource="#Business_Role"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B_factored_by">
   <rdfs:domain rdf:resource="#Composite_Motivation"/>
   <rdfs:range rdf:resource="#BDD_Validation_Element"/>
</owl:ObjectProperty>
<rdfs:Class rdf:ID="BDD_Validation_Element"/>
<rdfs:Class rdf:ID="Busines_Service">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Business_Behaviour_Element">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Business_Event">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Business_Object">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Business_Role">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Collaboration">
   <rdfs:subClassOf rdf:resource="#Business_Role"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Composite_Motivation"/>
<rdfs:Class rdf:ID="Constraint">
   <rdfs:subClassOf rdf:resource="#Principles"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="Contract">
   <rdfs:subClassOf rdf:resource="#Business_Object"/>
</rdfs:Class>
<owl:DatatypeProperty rdf:ID="False">
   <rdfs:domain>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <rdfs:Class rdf:about="#BDD_Validation_Element"/>
            <rdfs:Class rdf:about="#Composite_Motivation"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
   <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
<rdfs:Class rdf:ID="Function">
   <rdfs:subClassOf rdf:resource="#Business_Behaviour_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Goal">
   <rdfs:subClassOf rdf:resource="#Composite_Motivation"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Interaction">
   <rdfs:subClassOf rdf:resource="#Business_Behaviour_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Interface">
   <rdfs:subClassOf rdf:resource="#Business_Role"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Location">
   <rdfs:subClassOf rdf:resource="#Business_Role"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Meaning">
   <rdfs:subClassOf rdf:resource="#BDD_Validation_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Principles">
   <rdfs:subClassOf rdf:resource="#Composite_Motivation"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Process">
   <rdfs:subClassOf rdf:resource="#Business_Behaviour_Element"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Product">
   <rdfs:subClassOf rdf:resource="#Contract"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Representation">
   <rdfs:subClassOf rdf:resource="#Business_Object"/>
```

```
    </rdfs:Class>
    <rdfs:Class rdf:ID="Requirement">
        <rdfs:subClassOf rdf:resource="#Composite_Motivation"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Stakeholder">
        <rdfs:subClassOf rdf:resource="#Composite_Motivation"/>
    </rdfs:Class>
    <owl:DatatypeProperty rdf:ID="True">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdfs:Class rdf:about="#BDD_Validation_Element"/>
                    <rdfs:Class rdf:about="#Composite_Motivation"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <rdfs:Class rdf:ID="Value">
        <rdfs:subClassOf rdf:resource="#Contract"/>
    </rdfs:Class>
</rdf:RDF>
```

Appendix E: RDFS for UME-LLS Case Study

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
   <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
   <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1395059966.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1395059966.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>
    <owl:ObjectProperty rdf:ID="A0_has_interest">
        <rdfs:domain rdf:resource="#Student"/>
        <rdfs:range rdf:resource="#Allign_Student_and_Resource"/>
        <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A11_restricted_by">
        <rdfs:domain rdf:resource="#Allign_Student_and_Resource"/>
        <rdfs:range rdf:resource="#Module_Registration"/>
        <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A12_restricted_by">
        <rdfs:domain rdf:resource="#Allign_Student_and_Resource"/>
        <rdfs:range rdf:resource="#Study_Enrolment"/>
        <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A13_restricted_by">
        <rdfs:domain rdf:resource="#Allign_Student_and_Resource"/>
        <rdfs:range rdf:resource="#Fee_Payment"/>
```

```
    <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A1_analysed_by">
    <rdfs:domain rdf:resource="#Allign_Student_and_Resource"/>
    <rdfs:range rdf:resource="#Student_Legibility"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A2_decomposed_to">
    <rdfs:domain rdf:resource="#Student_Legibility"/>
    <rdfs:range rdf:resource="#Obtain_Laptop"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A3_specified_by">
    <rdfs:domain rdf:resource="#Obtain_Laptop"/>
    <rdfs:range rdf:resource="#Composition_Specification"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A_formalised_into">
    <rdfs:domain rdf:resource="#Composition_Specification"/>
    <rdfs:range rdf:resource="#BDD_Validation"/>
</owl:ObjectProperty>
<rdfs:Class rdf:ID="Allign_Student_and_Resource">
    <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
</rdfs:Class>
<owl:ObjectProperty rdf:ID="B11_used_by">
    <rdfs:domain rdf:resource="#Laptop_Pool"/>
    <rdfs:range rdf:resource="#Laptop_Request_Process"/>
    <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B13_aggregation_of">
    <rdfs:domain rdf:resource="#Module_Timetable"/>
    <rdfs:range rdf:resource="#Module"/>
    <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B14_used_by">
    <rdfs:domain rdf:resource="#Room_Allocation_Process"/>
    <rdfs:range rdf:resource="#Module_Timetable"/>
    <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B2_accessed_by">
    <rdfs:domain rdf:resource="#Laptop_Pool"/>
    <rdfs:range rdf:resource="#Software_Licence"/>
    <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B2_realised_by">
   <rdfs:domain rdf:resource="#Software_Licence"/>
   <rdfs:range rdf:resource="#Laptop_Pool"/>
   <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B31_accessed_by">
   <rdfs:domain rdf:resource="#Room_Allocation_Process"/>
   <rdfs:range rdf:resource="#Learning_Room"/>
   <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B31_used_by">
   <rdfs:domain rdf:resource="#Learning_Room"/>
   <rdfs:range rdf:resource="#Room_Allocation_Process"/>
   <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B32_accessed_by">
   <rdfs:domain rdf:resource="#Laptop_Pool"/>
   <rdfs:range rdf:resource="#Laptop_Request_Process"/>
   <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B41_dependency_of">
   <rdfs:domain rdf:resource="#BDD_Validation"/>
   <rdfs:range rdf:resource="#Request_Learning_Room"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B41_triggers">
   <rdfs:domain rdf:resource="#Request_Learning_Room"/>
   <rdfs:range rdf:resource="#Room_Allocation_Process"/>
   <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B42_dependency_of">
   <rdfs:domain rdf:resource="#BDD_Validation"/>
   <rdfs:range rdf:resource="#Request_Laptop"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B42_triggers">
   <rdfs:domain rdf:resource="#Request_Laptop"/>
   <rdfs:range rdf:resource="#Laptop_Request_Process"/>
   <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B53_realised_by">
   <rdfs:domain rdf:resource="#Software_Licence"/>
```

```
    <rdfs:range rdf:resource="#External_Interraction"/>
    <rdfs:subPropertyOf rdf:resource="#B42_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B_factored_by">
    <rdfs:domain rdf:resource="#BDD_Validation"/>
    <rdfs:range rdf:resource="#Composition_Specification"/>
</owl:ObjectProperty>
<rdfs:Class rdf:ID="BDD_Validation"/>
<rdfs:Class rdf:ID="Composition_Specification"/>
<rdfs:Class rdf:ID="External_Interraction">
    <rdfs:subClassOf rdf:resource="#Software_Licence"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Fee_Payment">
    <rdfs:subClassOf rdf:resource="#Allign_Student_and_Resource"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Laptop_Pool">
    <rdfs:subClassOf rdf:resource="#Laptop_Request_Process"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Laptop_Request_Process">
    <rdfs:subClassOf rdf:resource="#Request_Laptop"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Learning_Room">
    <rdfs:subClassOf rdf:resource="#Room_Allocation_Process"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Module">
    <rdfs:subClassOf rdf:resource="#Module_Timetable"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Module_Registration">
    <rdfs:subClassOf rdf:resource="#Allign_Student_and_Resource"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Module_Timetable">
    <rdfs:subClassOf rdf:resource="#Room_Allocation_Process"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Obtain_Laptop">
    <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Request_Laptop">
    <rdfs:subClassOf rdf:resource="#BDD_Validation"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Request_Learning_Room">
    <rdfs:subClassOf rdf:resource="#BDD_Validation"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Room_Allocation_Process">
```

```
        <rdfs:subClassOf rdf:resource="#Request_Learning_Room"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Software_Licence">
        <rdfs:subClassOf rdf:resource="#Laptop_Pool"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Student">
        <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Student_Legibility">
        <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Study_Enrolment">
        <rdfs:subClassOf rdf:resource="#Allign_Student_and_Resource"/>
    </rdfs:Class>
</rdf:RDF>
```

Appendix F: RDFS for UWL-SIP Case Study

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
   <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY assert "http://www.owl-ontologies.com/assert.owl#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
   <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1395071142.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1395071142.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:assert="http://www.owl-ontologies.com/assert.owl#">
    <owl:Ontology rdf:about="">
       <owl:imports rdf:resource="http://www.owl-ontologies.com/assert.owl"/>
    </owl:Ontology>
    <owl:ObjectProperty rdf:ID="A0_has_interest">
       <rdfs:domain rdf:resource="#Student"/>
       <rdfs:range rdf:resource="#Develop_employment_skills"/>
       <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A11_restricted_by">
       <rdfs:domain rdf:resource="#Develop_employment_skills"/>
       <rdfs:range rdf:resource="#Enroll_for_study"/>
       <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A12_restricted_by">
       <rdfs:domain rdf:resource="#Develop_employment_skills"/>
       <rdfs:range rdf:resource="#Pass_PIT5"/>
       <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A13_restricted_by">
       <rdfs:domain rdf:resource="#Develop_employment_skills"/>
       <rdfs:range rdf:resource="#Placement_IT_Related"/>
       <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="A14_restricted_by">
```

```
    <rdfs:domain rdf:resource="#Develop_employment_skills"/>
    <rdfs:range rdf:resource="#Within_3_Years_of_Study"/>
    <rdfs:subPropertyOf rdf:resource="#A1_analysed_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A1_analysed_by">
    <rdfs:domain rdf:resource="#Develop_employment_skills"/>
    <rdfs:range rdf:resource="#Internship_Legibility"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A2_decomposed_into">
    <rdfs:domain rdf:resource="#Internship_Legibility"/>
    <rdfs:range rdf:resource="#Obtain_placement"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A3_specified_by">
    <rdfs:domain rdf:resource="#Obtain_placement"/>
    <rdfs:range rdf:resource="#Composition_Specification"/>
    <rdfs:subPropertyOf rdf:resource="#A_formalised_into"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="A_formalised_into">
    <rdfs:domain rdf:resource="#Composition_Specification"/>
    <rdfs:range rdf:resource="#BDD_Validation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B11_aggregated_by">
    <rdfs:domain rdf:resource="#Internship_listing"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Employer_detail"/>
                <owl:Class rdf:about="#Internship_search_and_match"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
    <rdfs:subPropertyOf rdf:resource="#B23_used_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B12_accessed_by">
    <rdfs:domain rdf:resource="#Internship_search_and_match"/>
    <rdfs:range rdf:resource="#Internship_listing"/>
    <rdfs:subPropertyOf rdf:resource="#B23_used_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B13_accessed_by">
    <rdfs:domain rdf:resource="#Student_assessment_and_exam"/>
    <rdfs:range rdf:resource="#Student_record"/>
    <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B13_associated_with">
    <rdfs:domain rdf:resource="#PIT5_and_exams_record"/>
    <rdfs:range rdf:resource="#Student_record"/>
    <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="B14_accessed_by">
   <rdfs:domain rdf:resource="#Student_assessment_and_exam"/>
   <rdfs:range rdf:resource="#PIT5_and_exams_record"/>
   <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B22_accessed_by">
   <rdfs:domain rdf:resource="#Registration_and_enrollment"/>
   <rdfs:range rdf:resource="#Student_record"/>
   <rdfs:subPropertyOf rdf:resource="#B22_used_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B22_used_by">
   <rdfs:domain rdf:resource="#Study_enrollment"/>
   <rdfs:range rdf:resource="#Registration_and_enrollment"/>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B23_accesses">
   <rdfs:domain rdf:resource="#Internship_search_and_match"/>
   <rdfs:range rdf:resource="#Student_record"/>
   <rdfs:subPropertyOf rdf:resource="#B23_used_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B23_used_by">
   <rdfs:domain rdf:resource="#Internship_application"/>
   <rdfs:range rdf:resource="#Internship_search_and_match"/>
   <rdfs:subPropertyOf rdf:resource="#B334_triggered_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B24_used_by">
   <rdfs:domain rdf:resource="#Legibility_determination"/>
   <rdfs:range rdf:resource="#Student_assessment_and_exam"/>
   <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B2_consist_of">
   <rdfs:domain rdf:resource="#Obtain_placement_process"/>
   <rdfs:range>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Internship_application"/>
            <owl:Class rdf:about="#Internship_finding"/>
            <owl:Class rdf:about="#Legibility_determination"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:range>
   <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B331_flow_from">
   <rdfs:domain rdf:resource="#Internship_finding"/>
   <rdfs:range rdf:resource="#Internship_application"/>
   <rdfs:subPropertyOf rdf:resource="#B334_triggered_by"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="B331_used_by">
   <rdfs:domain rdf:resource="#Internship_finding"/>
```

```
      <rdfs:range rdf:resource="#Internship_search_and_match"/>
      <rdfs:subPropertyOf rdf:resource="#B334_triggered_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="B333_flow_from">
      <rdfs:domain rdf:resource="#Legibility_determination"/>
      <rdfs:range rdf:resource="#Internship_finding"/>
      <rdfs:subPropertyOf rdf:resource="#B334_triggered_by"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="B334_triggered_by">
      <rdfs:domain rdf:resource="#Request_for_Internship"/>
      <rdfs:range rdf:resource="#Legibility_determination"/>
      <rdfs:subPropertyOf rdf:resource="#B41_dependency_of"/>
    </owl:ObjectProperty>
    <owl:TransitiveProperty rdf:ID="B41_dependency_of">
      <rdf:type rdf:resource="&owl;ObjectProperty"/>
      <rdfs:domain rdf:resource="#BDD_Validation"/>
      <rdfs:range rdf:resource="#Request_for_Internship"/>
      <rdfs:subPropertyOf rdf:resource="#B_factored_by"/>
    </owl:TransitiveProperty>
    <owl:ObjectProperty rdf:ID="B_factored_by">
      <rdfs:domain rdf:resource="#BDD_Validation"/>
      <rdfs:range rdf:resource="#Composition_Specification"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="BDD_Validation">
      <assert:notEmpty rdf:datatype="&xsd;string"
        >SELECT ?object ?subject
WHERE { ?subject rdfs:range ?object }
ORDER BY ?subject</assert:notEmpty>
      <owl:disjointWith rdf:resource="#Composition_Specification"/>
    </owl:Class>
    <owl:Class rdf:ID="Composition_Specification">
      <owl:disjointWith rdf:resource="#BDD_Validation"/>
    </owl:Class>
    <owl:Class rdf:ID="Develop_employment_skills">
      <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
      <owl:disjointWith rdf:resource="#Internship_Legibility"/>
      <owl:disjointWith rdf:resource="#Obtain_placement"/>
      <owl:disjointWith rdf:resource="#Student"/>
    </owl:Class>
    <owl:Class rdf:ID="Employer_detail">
      <rdfs:subClassOf rdf:resource="#Internship_listing"/>
    </owl:Class>
    <owl:Class rdf:ID="Enroll_for_study">
      <rdfs:subClassOf rdf:resource="#Develop_employment_skills"/>
    </owl:Class>
    <owl:Class rdf:ID="Internship_application">
      <rdfs:subClassOf rdf:resource="#Obtain_placement_process"/>
    </owl:Class>
    <owl:Class rdf:ID="Internship_finding">
      <rdfs:subClassOf rdf:resource="#Obtain_placement_process"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Internship_Legibility">
   <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
   <owl:disjointWith rdf:resource="#Develop_employment_skills"/>
   <owl:disjointWith rdf:resource="#Obtain_placement"/>
   <owl:disjointWith rdf:resource="#Student"/>
</owl:Class>
<owl:Class rdf:ID="Internship_listing">
   <rdfs:subClassOf rdf:resource="#Internship_search_and_match"/>
</owl:Class>
<owl:Class rdf:ID="Internship_search_and_match">
   <rdfs:subClassOf rdf:resource="#Internship_application"/>
</owl:Class>
<owl:Class rdf:ID="Legibility_determination">
   <rdfs:subClassOf rdf:resource="#Obtain_placement_process"/>
</owl:Class>
<owl:Class rdf:ID="Obtain_placement">
   <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
   <owl:disjointWith rdf:resource="#Develop_employment_skills"/>
   <owl:disjointWith rdf:resource="#Internship_Legibility"/>
</owl:Class>
<owl:Class rdf:ID="Obtain_placement_process">
   <rdfs:subClassOf rdf:resource="#Request_for_Internship"/>
</owl:Class>
<owl:Class rdf:ID="Pass_PIT5">
   <rdfs:subClassOf rdf:resource="#Develop_employment_skills"/>
</owl:Class>
<owl:Class rdf:ID="PIT5_and_exams_record">
   <rdfs:subClassOf rdf:resource="#Student_assessment_and_exam"/>
</owl:Class>
<owl:Class rdf:ID="Placement_IT_Related">
   <rdfs:subClassOf rdf:resource="#Develop_employment_skills"/>
</owl:Class>
<owl:Class rdf:ID="Registration_and_enrollment">
   <rdfs:subClassOf rdf:resource="#Study_enrollment"/>
   <owl:disjointWith rdf:resource="#Request_for_Internship"/>
   <owl:disjointWith rdf:resource="#Study_enrollment"/>
</owl:Class>
<owl:Class rdf:ID="Request_for_Internship">
   <rdfs:subClassOf rdf:resource="#BDD_Validation"/>
   <owl:disjointWith rdf:resource="#Registration_and_enrollment"/>
   <owl:disjointWith rdf:resource="#Study_enrollment"/>
</owl:Class>
<owl:Class rdf:ID="Student">
   <rdfs:subClassOf rdf:resource="#Composition_Specification"/>
   <owl:disjointWith rdf:resource="#Develop_employment_skills"/>
   <owl:disjointWith rdf:resource="#Internship_Legibility"/>
</owl:Class>
<owl:Class rdf:ID="Student_assessment_and_exam">
   <rdfs:subClassOf rdf:resource="#Legibility_determination"/>
```

```
    </owl:Class>
    <owl:Class rdf:ID="Student_record">
      <rdfs:subClassOf rdf:resource="#Registration_and_enrollment"/>
    </owl:Class>
    <owl:Class rdf:ID="Study_enrollment">
      <rdfs:subClassOf rdf:resource="#BDD_Validation"/>
      <owl:disjointWith rdf:resource="#Registration_and_enrollment"/>
      <owl:disjointWith rdf:resource="#Request_for_Internship"/>
    </owl:Class>
    <owl:Class rdf:ID="Within_3_Years_of_Study">
      <rdfs:subClassOf rdf:resource="#Develop_employment_skills"/>
    </owl:Class>
</rdf:RDF>
```