

MIDDLESEX UNIVERSITY LONDON

DOCTORAL THESIS

Information Systems Framework for Enterprise Agility

Author:
Joshua C. NWOKEJI

Director of Studies:
Professor Balbir BARN
Supervisor:
Professor Tony CLARK

*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy*

in the

Department of Computer Science

June 7, 2016

Declaration of Authorship

I, Joshua C. NWOKEJI, declare that this thesis titled, “Information Systems Framework for Enterprise Agility” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

“But I say unto you, Love your enemies, bless them that curse you, do good to them that hate you, and pray for them which despitefully use you, and persecute you;”

Abstract

Modern day enterprises operate and transact in an increasingly dynamic business environment. As a result, they are vulnerable to spontaneous changes and uncertainties. These usually reduce effectiveness and optimal performances in enterprises, and can have negative impacts such as loss of competitiveness, and bankruptcy. Enterprise agility, *i.e.*, the ability of enterprises to respond to changes, is a core imperative for effective change management. Yet, it is challenging, difficult to achieve, and a major concern for corporate executives. Enterprises would thus require novel approaches to manage changes and enhance agility.

In order to facilitate or achieve enterprise agility, it would be necessary and vital to develop frameworks or processes that can support effective change management. Such frameworks or processes should include techniques for modelling enterprises changes explicitly, so as to enhance the understanding of how changes relate to or affect enterprises. In addition, there should be techniques for deriving the elements of an enterprise, *e.g.*, business process and data entities, that are required to adapt a given enterprise change. However, concepts, constructs, and techniques for representing changes are often neglected, if available at all, in the existing enterprise modelling approaches such as TOGAF and ZACHMAN. This contributes to the difficulty in applying the existing enterprise modelling approaches to enhance enterprise agility and effective change management.

The work described in this thesis provides a novel approach for supporting enterprise agility and change management. Therefore, this thesis contributes a conceptual process or framework for representing enterprise changes, and deriving enterprise elements such as data entity, business goal, and business process required to adapt a given change. Other contributions made by this approach include a novel conceptual modelling language for representing enterprise changes, an enterprise modelling language, and a set of procedures and rules that can be used to derive the new domain elements required to adapt changes. An industry case study has been used to test the utility of this framework. The results obtained from this case study shows that this framework supports enterprise agility and change management in a number of ways.

Acknowledgements

It was indeed a tougher and more challenging adventure that I ever imagined. "But thanks be to God [my strength], which giveth [me] the victory through our Lord Jesus Christ".

Thanks to Professor Tony Clark, Professor Balbir Barn, and Professor Vinay Kulka-rni. I would not have completed my Ph.D without their supervision, support, fatherly guidance, and encouragement. I owe all I have learnt in this Ph.D to them.

I am very much indebted to my Parents, Sir and Mrs. E.A.N Nwokeji (JP), for the enormous sacrifices they make to provide the financial sponsorship towards my studies. Thanks to my Siblings, especially Adanne m, Dab, and precious for their loyalty, encouragement, support, and prayers. My elder brother, Samuel, was supportive too.

Special thanks my Uncle Engr. Chinyere Anyile, my friends Dr. Gladys Igwe, and Engr. Onyekachi S. Anum for their love, care, prayers, and support. I also thank other people who supported me in various ways, particularly, Pastor and Mrs. Dore Agokem, my inlaws (Innocent Obiora, Joseph Okolie, and Joseph azubuike), Kasonde Mwila, Mr. Yemi Akinwade, Bro. Uche, Timothy, Paul Eke, Sis. Blessing, Elder and Mrs. Felix Eze, Mr. and Mrs. John Umeaku, Mr. David and Dr. Chioma Ude, and Chioma Ogbonna. I also want to thank my research colleagues in TG 23, especially Fred Omondi for their support.

Contents

Declaration of Authorship	iii
Abstract	vii
Acknowledgements	ix
Contents	xi
List of Figures	xvii
List of Tables	xix
List of Abbreviations	xxi
1 Introduction	1
1.1 Problem Definition	1
1.1.1 Deriving Domain Elements	3
1.1.2 Modelling Enterprise Changes	4
1.2 Research Question	5
1.3 Research Aim and Objectives	5
1.4 Brief Overview of the Proposed Framework	6
1.5 Contributions	7
1.5.1 The Change Modelling Language	7
1.5.2 The Concise Enterprise Modelling Language	9
1.5.3 Procedure for Deriving New Domain Requirements (Elements)	10
1.5.4 Conceptual Modelling Techniques and Technologies	11
1.6 Research Methodology	12
1.7 Validation and Evaluation Criteria	13
1.8 Publications and Conference Presentations	14
1.9 Thesis Organization	15
2 Literature Review	19
2.1 Overview	19

2.1.1	Definition and Types of Enterprise Agility	20
2.2	Review Methodology	22
2.3	Establishing Review Protocol	24
2.3.1	Review Objectives	24
2.3.2	Review Questions	24
2.4	Review Strategy	24
2.4.1	Search Keywords	24
2.4.2	Search Expressions	25
2.4.3	Data Sources	26
2.4.4	Selection Criteria	26
2.5	Review Results	27
2.5.1	Data Extraction	27
2.5.2	Results and Answers to Review Questions	28
2.5.2.1	Answer to Review Question 1	28
2.5.2.2	Answer to Review Question 2	29
2.6	The Case Study	31
2.7	Discussion and Analysis	33
2.7.1	The Concepts	33
2.7.2	Discussion of Existing Enterprise Approaches	39
2.8	Conclusion and Chapter Summary	43
3	Meta-Modelling Techniques And Technologies	45
3.1	Overview	45
3.2	Conceptual Model and Modelling	45
3.3	Meta-modelling Techniques	46
3.3.1	Abstract Syntax	46
3.3.2	Concrete Syntax	49
3.3.3	Semantics	51
3.4	Meta-Modelling Technologies	52
3.4.1	Relevant MOF Concepts	53
3.4.2	Eclipse Modelling Framework	55
3.5	Chapter Summary	55
4	The Proposed Framework And Research Methodology	57
4.1	Chapter Overview	57
4.2	The Proposed Framework	57
4.3	Research Methodology	59
4.3.1	Design Science Research	60
4.3.2	Guidelines for Information Systems Research	61
4.4	Motivating Principles	66

4.5	Chapter Summary	68
5	The Change Modelling Language	69
5.1	Language Abstract Syntax	69
5.2	Abstract Syntax Model	70
5.2.1	Concept Identification	70
5.2.2	Concept Definition	72
5.2.2.1	Change	74
5.2.2.2	Goal	75
5.2.2.3	Change Impact	76
5.2.2.4	Business Process	77
5.2.2.5	Data	78
5.2.2.6	Change Driver	79
5.2.2.7	Action	79
5.2.2.8	Actor	80
5.2.2.9	Agility Enabler	81
5.2.2.10	Change Indicator	81
5.2.3	Relationships and Multiplicities Between Concepts	82
5.2.4	Validity Rules	84
5.3	Language Concrete Syntax	85
5.3.1	Model of Diagram	86
5.3.2	Abstract Syntax With Model of Diagram	86
5.4	Language Semantics	88
5.4.1	Semantic Domain	88
5.4.2	Semantic Mapping	89
5.5	Chapter Summary	91
6	Enterprise Modelling Language and The Proposed Rules and Procedures	93
6.1	Overview	93
6.2	Enterprise Modelling	93
6.3	The Enterprise Modelling Language	95
6.4	Model Comparison	99
6.4.1	Conflicts in Conceptual Model Comparison	99
6.4.2	Resolving Conflicts in Model Comparison	100
6.4.3	Resolving Conflicts with Meta-Information	101
6.5	Rules for Relating The Models	104
6.6	Procedures for the Comparison	107
6.7	Chapter Summary	108
7	Application of the Proposed Change Modelling Language	109

7.1	Chapter Overview	109
7.2	Part 1: Modelling These Changes	111
7.3	Application Part 2: Change Management Proforma	118
7.4	Application Part 3: Modelling Enterprise and Deriving Domain Elements	120
7.4.1	Applying Rule 1 to Derive Data Elements	122
7.4.2	Applying Rule 2 to Derive Activities	130
7.4.3	Applying Rule 3 to Derive Goals	131
7.5	Chapter Summary	131
8	Reflection and Conclusion	133
8.1	Chapter Overview	133
8.2	Reflecting on the Research Question	134
8.3	Reflecting on the Research Objectives	135
8.3.1	Identification of Concepts	135
8.3.2	Design and Develop Conceptual Model for Change	135
8.3.3	Concise Enterprise Modelling Language	136
8.3.4	Rules and Procedures	136
8.3.5	Demonstrate Utility	136
8.4	Evaluation and Validation	137
8.5	Discussion of Lessons	138
8.5.1	Lesson From Conceptual Modelling	138
8.6	Limitations	140
8.7	Future Work	142
8.8	Concluding Remarks	142
	Bibliography	147
A	Abstract Syntax Model of the Change Modelling Language	173
B	Abstract Syntax Model of Enterprise	175
C	Enlarged Models of Case Study	177
C.1	Enlarged Change Model From Case Study: Case 1	177
C.2	Enlarged Notation for Change Modelling	179
C.3	Enlarged Change Model From Case Study: Case 2	180
D	Enlarged Case Study Goal Model	183
D.1	Enlarged Case Study Goal Model	183
D.2	Enlarged Case Study Enterprise Model	185

E	Application of the Proposed Change Modelling Language in Database Implementation	187
E.1	Application Part 4: Database Implementation	188
E.1.1	Lessons From Database Implementation	197
F	Formalising the Rules	199
F.1	Rule 1	199
F.2	Rule 2	199
F.3	Rule 3	200

List of Figures

1.1	Example of a Typical Enterprise	1
1.2	Changing Enterprise	3
1.3	Research Approach	13
2.1	Specialisation of Enterprise Agility	21
2.2	Summary Of Processes Involved In Performing SLR	23
2.1	Search Expressions	25
2.3	Systematic Literature Review Screening Process	28
3.1	Features of a Modelling Language	47
3.2	Illustration of UML Concepts	54
4.1	The Proposed Framework For Enterprise Agility	58
4.2	Design Science Research	60
4.3	Guidelines for Information Systems Research	62
4.4	Research Process and DSR Guideline	64
5.1	Abstract Syntax Model of the Proposed Modelling Language	71
5.2	The Structure of Concept Definition	74
5.3	Abstract Model of Diagram	86
5.4	Abstract Syntax with Model of Diagram	87
5.5	Core Concepts and Statechart Concepts	90
6.1	Abstract Syntax Model of the Enterprise Modelling Language	97
6.2	Naming Conflict Resolution Example	102
6.3	Resolving Conflicts with Meta-Information	103
6.4	Procedures For Deriving Domain Elements	107
7.1	Change Model of the Case Study: Case 1	113
7.2	Change Model of the Case Study: Case 2	114
7.3	Change Management Proforma	118
7.4	Procedures For Deriving Domain Elements	120
7.5	Meta-Information for the 2 Change Models in the Case Study	121
7.6	Case Study Enterprise Goal Model	123

7.7	Case Study Enterprise Data Model	124
7.8	Case Study Enterprise Business Process Model	125
7.9	Case Study Enterprise Meta Information	126
7.10	Case Study Enterprise Model	127
7.11	Comparing Enterprise and Change Entities For Case 1 using Meta- Information	128
7.12	Comparing Enterprise and Change Entities For Case 2 using Meta- Information	129
7.13	The Set of Activities for Case 2	130
A.1	Abstract Syntax Model of the Proposed Modelling Language	174
B.1	Abstract Syntax Model of Enterprise	176
C.1	Enlarged Change Model of the Case Study: Case 1	178
C.2	Notations used for Change Modelling	179
C.3	Enlarged Change Model of the Case Study: Case 2	181
D.1	Case Study Enterprise Goal Model	184
D.2	Case Study Enterprise Model	186
E.1	The Relational Database Model	189

List of Tables

1.1	Conference Presentations and Participations	15
1.2	List of Publications from This Research	16
2.1	Difference Between Enterprise Agility and Its Types	22
2.2	Search Keywords	25
2.3	The Systematic Review Strategy	26
2.4	The Quality Criteria Used to Select Primary Studies	27
2.5	Enterprise Agility and Change Concepts	29
2.6	Existing Enterprise Modelling Approaches	30
2.7	Comparison of Agility Modelling Constructs with Enterprise Modelling Constructs	39
4.1	Summary of Motivating Language Design Principles	66
5.1	Concepts of the Proposed Modelling Language	72
5.2	Summary of Concepts in the Change Language	73
6.1	Concepts of the Proposed Enterprise Modelling Language	95

List of Abbreviations

ABS	Abstract State Machine
BMM	Business Motivation Model
BNF	Backus Naur Form
BP	Business Process
BPM	Business Process Modelling Notation
CIMOSA	Computer Integrated Manufacturing Open Systems Architecture
COO	Chief Operating Officer
CML	Conceptual Modelling Language
DC	Design Cycle
DD	Diagram Definition
DSR	Design Science Research
DSL	Domain Specific Language
DTD	Data Type Definition
EA	Enterprise Architecture
EAg	Enterprise Agility
ECMP	Enterprise Change Management Team
EDOC	Enterprise Distributing Object Computing
EEML	Extended Enterprise Modelling Language
EM	Enterprise Modelling
EMF	Eclipse Modelling Framework
ES	Enterprise Systems
ERM	Entity Relationship Modelling
ETL	Extract Transform Load
GL	Guidelines
FSM	Finite State Machine
GML	Graphical Modelling Language
GOML	Goal Oriented Modelling Language
GORE	Goal Oriented Requirements Engineering
GPL	General Purpose Language
HTML	Hyper Text Markup Language
IA	Change Impact Analysis
IS	Information System
ISR	Information Systems Research

IT	I nformation T echnology
ISO	I nternational S tandard O rganization
JDBC	J ava D ata B atabase C onnectivity
MDD	M odel D riven D evelopment
MMA	M eta- M odel A rchitecture
MML	M athematical M odelling L anguage
MOF	M eta O bject F acility
OCL	O bject C onstraint L anguage
OMG	O bject M anagement G roup
PAIS	P rocess A ware I nformation S ystems
PHP	H ypertext P reprocessor
R_B	B usiness process R equirements
R_D	D ata R equirements
RE	R equirements E ngineering
R_G	G oals (R equirements)
SCOR	S upply C hain O perations R eference
SDLC	S ystems/ S oftware D evelopment L ife C ycle
SOA	S ervice O riented A rchitecture
SOC	S ervice O riented C omputing
SLR	S ystematic L iterature R evue
SQL	S tructured Q uery L anguage
SUS	S ystem U nder S tudy
SysML	S ystems M odelling L anguage
TOGAF	T he O pen G roup E nterprise A rchitecture F ramework
UML	U nified M odelling L anguage
UoD	U niverse O f D iscourse
XML	e Xtensible M arkup L anguage
XP	e Xtreme P rogramming
ZAF	Z achman A rchitecture F ramework

*Dedicated to my Parents, Sir and Mrs. E.A.N Nwokeji,
and to Adannem (my first Sister, Chigozirim) for your
untold sacrifices, loyalty, and support.*

Chapter 1

Introduction

1.1 Problem Definition

One major challenge faced by modern day enterprises is the ability to respond to changes, and manage business uncertainties, in an efficient and effective way [49, 70]. The term *enterprise*, as used in this research, is a conceptual representation of the relationships between business objectives, the processes for achieving these objectives, and the set of information required for business operations [54, 55]. As shown in Figure 1.1, business objectives are usually represented as a set of achievable goals, i.e., the rationale for the existence of an enterprise. Similarly the process for achieving business goals can be represented as a set of coherent activities, also called business process, while business information can be represented as a set of data and business entity [54, 55].

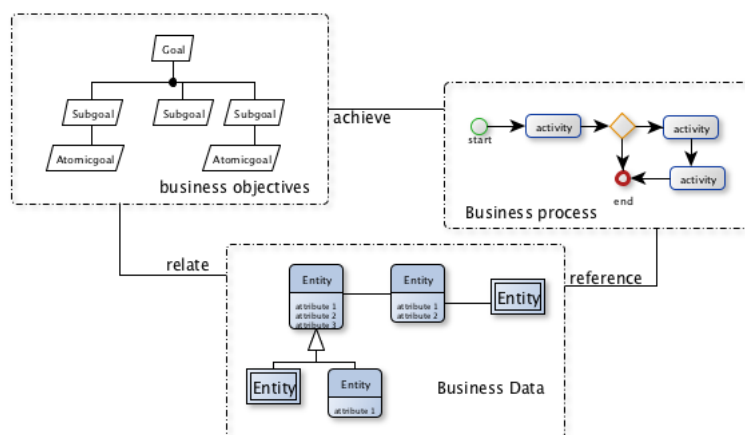


FIGURE 1.1: Example of a Typical Enterprise

Enterprises depend on, interact with, and transact in an increasingly dynamic business environment. Therefore, they can be vulnerable to spontaneous changes and uncertainties. These usually have negative impacts such as financial loss, bankruptcy, changes in business requirements, and loss of competitive advantage [12, 46]. The changes and uncertainties experienced by enterprises are often caused by *change drivers*, broadly defined as circumstances and events that can lead to changes in an enterprise [57, 176]. Examples of popular change drivers include regulatory compliance, technology updates and obsolescence, competition, and changes in business requirements. In order to become competitive and viable in a turbulent business environment, enterprises require agility [12, 49].

Enterprise agility (EAg), broadly defined as the ability of enterprises to adapt to changes timely and effectively, is a core imperative for effective change management [171]. It can improve operational efficiency [178], enhance competitive ability [175], and make enterprises resilient to changes [148]. In addition, it facilitates resource optimization, profitability, and customer retention [119]. Yet, enterprise agility is challenging, prevailing, difficult to achieve, and a major concern for corporate executives [145, 166, 175, 187]. Van Oosterhout et al [145] believe that despite existing scholarly efforts towards actualising agility, till date, there is no definite method for achieving enterprise agility. Therefore, enterprises are still seeking for methods and frameworks for achieving or supporting enterprise agility [171].

Although most existing enterprise modelling approaches are intended, among other reasons, to facilitate or achieve enterprise agility [35, 59]. They are primarily designed to represent the structure, information, and behaviour of an enterprise [55, 194], instead of the changes faced by the enterprise [65, 139]. For instance, enterprise architecture frameworks (EAF) such as the open group architecture framework (TOGAF) [176] and Zachman Architecture Framework (ZAF) [200] provide the means of describing the information, business process, and other aspects of an enterprise. But have no definite means of representing changes faced by enterprises and deriving the elements of an enterprise they are required to adapt changes. Even though, international standard organization framework *ISO/199440*, the de facto international standard for enterprise modelling alluded to change representation as an important requirements for enterprise modelling approaches [98, 122]. Yet, model constructs, concepts, and techniques for representing enterprise changes are lacking in popular enterprise modelling approaches such as, the business motivation model (BMM) [23], i*(i-star language) [196], and universal modelling language (UML) [53].

From the definition of enterprise agility given above, it can be inferred that the capability for adapting to changes is the basic criteria for facilitating or achieving

enterprise agility. Therefore change is a central concept in the enterprise agility domain. But it would be essential that change and its features are specified as well as represented in a systematic, logical, clear, and understandable form, using suitable concepts and constructs, before such capability can be acquired [139]. However, techniques and constructs for change modelling are rarely considered, if available at all, in the existing enterprise modelling approaches. Therefore, they can be considered inappropriate or unfit for facilitating or achieving enterprise agility. This presents two major problems and limitation for enterprises desiring to be agile. First is the the inability to clearly represent enterprise change and its features, which can lead to lack or limited understanding, visualisation, analysis and interpretation of enterprise changes. Secondly lack of process for deriving the enterprise elements, such as data entities and business process activities, that are required to adapt to changes. These two problems are discussed in detail in Sections 1.1.1 and 1.1.2.

1.1.1 Deriving Domain Elements

An enterprise can be captured or modelled at various viewpoints or domains. These can include, but are not limited to, the enterprise business objectives or reasons for its existence (goals), a set of processes for achieving these goals (business process), and the information required to achieve the motivations (data). As shown in Figure 1.2, each of these domains should consist of one or more elements, for instance the data domain can consist of elements such as business entities and attributes.

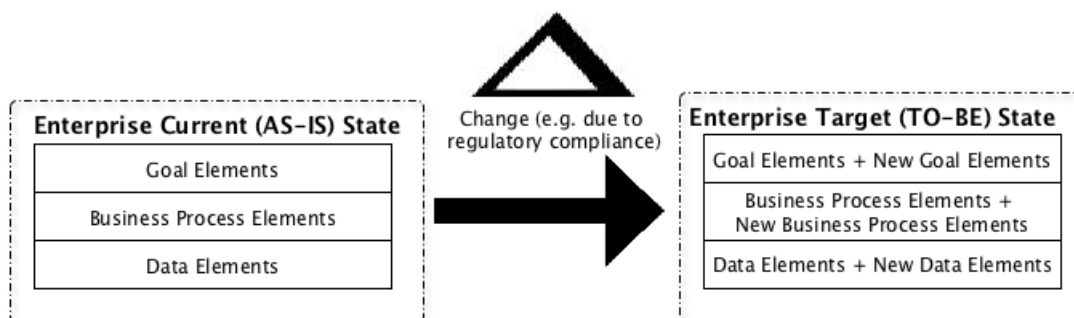


FIGURE 1.2: New Requirements in Changing Enterprise

Consider a change to this enterprise (Δ), due to a regulatory compliance or any other change driver, as shown in Figure 1.2. This change can affect the elements in any, or all, of the domains of the enterprise. Assuming it affects only the business process domain, then the elements of the business process will change from its current state to a target state, while the data domain elements and the goal domain elements will remain constant. The target state of the business process should include a set of new business process elements required to adapt to this change. There should be a consistent and structured process for deriving these new business process elements.

Equally, if this change affects only the data domain, the data elements will change from its current state to a target state, while the business process elements and the goals domain elements will remain constant. Similarly, the target state of the enterprise data should include a new set of data elements required to adapt to this change. The same principle also applies if the change affects only the goal domain. See Figure 1.2 for a diagrammatic explanation.

In any case, a change should cause an enterprise to transit from its current (as-is) state to a target (to-be) state, resulting to modifications *i.e.*, the addition of new domain elements to the affected domain(s). To maintain equilibrium or adapt to a change, an enterprise needs to derive the set of new domain elements, *i.e.*, new goal or data or business process elements, which should be added to the particular domain affected by the change. A structured framework can help an enterprise to facilitate such derivations in a consistent, systematic, and coherent manner. However, such a framework is often neglected, if currently available at all, in the existing enterprise modelling approaches. Therefore it has become increasingly difficult for enterprise change managers to precisely derive and discern the domains elements required to adapt a given change. This can be one of the reasons for the difficulties in achieving enterprise agility and managing enterprise changes.

1.1.2 Modelling Enterprise Changes

In the field of Information Systems, enterprises are usually represented in form of models or architectures, known as enterprise models or architecture, to enhance any form of analysis or manipulations. Hence, in order to relate an enterprise to the changes it can encounter, and derive new enterprise domain elements required to adapt to those changes; there should also be a systematic, logical, and coherent way to represent enterprise changes. A conceptual modelling language can be useful to this end.

Additionally, representing enterprise changes can be useful, to enterprise agility and change management, in some other ways. For instance, it can enhance the ability of an enterprise to clarify any ambiguous and implied concepts, such as change drivers, often associated with changes. Such clarity can be helpful in understanding the origin/causes of changes and deciding suitable actions for mitigating enterprise changes. All these can be possible, and facilitated, if there is a conceptual language for modelling enterprise changes and their features.

Despite their potential benefits and usefulness, to enterprise agility and change management, structured techniques and conceptual modelling language for modelling enterprise changes and their features are currently not available. In this context, *change features or features of change* refer to those concepts or model constructs required to elaborate, specify, and model an enterprise change. Examples of such concepts include *change driver, change impact, agility enabler, change indicator*, etc. Since these are not available in existing enterprise modelling approaches, it has become difficult to use them in understanding the structure of enterprise changes, and performing some analysis such as deriving new enterprise domain elements required to adapt a given change.

1.2 Research Question

This research will answer the following question:

What is the process for deriving the set of new enterprise domain elements required to adapt to a given enterprise change?

1.3 Research Aim and Objectives

The aim of this research is to support enterprise agility and change management initiatives. This is done by developing a structured process¹ (also called framework) for representing an enterprise and its changes, as well as deriving new enterprise domain elements that are required to adapt a given change. This framework can help enterprise stakeholders to precisely determine the data, business process activity, and business goals that should be modified or added to an existing enterprise model, in order to adapt a given change.

In order to achieve this aim, the following objectives will be met:

¹This process is referred to as framework, elsewhere in this Thesis

- (i) To identify the minimal, necessary, and sufficient set of concepts required to represent enterprise changes, through a systematic literature review.
- (ii) To design and develop a conceptual language for modelling enterprise changes, using the concepts identified from the systematic literature review.
- (iii) To design and develop a concise conceptual language for modelling an enterprise, re-using concepts from existing enterprise models and architectures.
- (iv) To provide a set of rules and procedures for deriving new domain requirements when changes occur in an enterprise.
- (v) To demonstrate the utility of the proposed framework using an industry case study.

1.4 Brief Overview of the Proposed Framework

The proposed framework consists of three parts, further description of this framework are provided in Chapter 4. The first part provides a modelling language that can be used to explicitly represent enterprise changes, and hence offers the solution to the problem stated in Section 1.1.2. Given that an enterprise change can be represented explicitly, as a model, using the change modelling language proposed in this research; there should also be a modelling language to represent an enterprise as a model, so that it can be easier to relate a change model to an enterprise model to understand how changes affect enterprise elements.

Hence the second part of the proposed framework provides another modelling language for representing an enterprise. Even though this language is new, it reuses concepts and ideas from existing enterprise modelling frameworks. The third part of this framework proposes some procedures and rules, which can be used to derive the set of new modelling elements required to adapt to a given change. Taken together, the second and the third part of the proposed framework provide the solution to the problem described in Section 1.1.1.

1.5 Contributions

The core contribution of this research is a novel process or framework for modelling enterprise changes and deriving a set of new enterprise domain elements that are required to adapt a given change. This process extends the body of knowledge in Information Systems by making other contributions as follows:

- (i) A novel Change Modelling Language for modelling enterprise changes.
- (ii) Concise Enterprise Modelling Language for modelling an enterprise in terms of its goals, business process activities, and business data.
- (iii) A novel procedure and rules for deriving new domain elements required to adapt to changes.

Each of these contributions are discussed in details in the sections below.

1.5.1 The Change Modelling Language

As mentioned earlier, the proposed change modelling language provides the techniques for the conceptual modelling of enterprise changes. Conceptual modelling is a part of Software Engineering and Information Systems that is concerned with the use of concepts and ideas, called model constructs, to represent a problem domain as well as design the solution domain [127]. By modelling a problem domain using concepts, it is easier to identify, analyse, and understand the causes of the problem together with the requirements for solving it [127].

Enterprise changes, agility, and change management present serious challenges and difficulties to modern day enterprises [49, 70]. Therefore they require immediate strategic solutions. In order to facilitate and speed up these solutions, it would be critical that change, which is the central concept in enterprise agility and change management, and other related concepts are clearly represented or modelled using well defined modelling language and model constructs. Currently, there is no conceptual modelling language and concepts for modelling, or describing the meaning and structure of, enterprise changes to support conceptual understanding. This research contributes to this area by providing unique concepts and model constructs together with a meta-model for modelling enterprise changes, and thus extends the boundary of knowledge and understanding in the field of conceptual modelling.

The proposed change modelling language provides a set of definite concepts that would help to convey clarity and meaning to the nature of an enterprise change. It also includes a set of notation (figures and shapes) which end users can be used to represent or describe the structure of an enterprise change. To ensure that the proposed language is meaningful, its concepts are related to the concepts of, *state chart*, a well known language used to model reactive or changing systems. This language has been fully described in Chapter 5. As discussed in proceeding paragraphs, there are obvious motivations and reasons for using the proposed conceptual language to model enterprise changes.

By modelling enterprise changes using the proposed language, it can be easier to identify, analyse, understand, and communicate the causes of enterprise changes. For instance, the *change driver* model construct included in the proposed meta-model can be used to model the origin and causes of an enterprise change. In this way, enterprise managers can reason about, and understand, the why or rationale behind enterprise changes. Understanding the causes and rationale behind information systems problems is a vital means to finding viable solutions [127]. Hence, understanding the origin and causes of change can motivate enterprise managers to think and proffer viable strategies and actions plans for adapting to changes.

Aside from the change driver, other model constructs such as *impact* can be useful as well. Impact can be used to model the consequences of a change to an enterprise. In so doing, enterprise managers can examine, specify, and communicate the outcome of changes to their enterprise and also identify the potential risks posed by changes. Risk identification is a vital activity in performing change impact analysis (IA), which can facilitate enterprise agility and change management [25]. Furthermore, modelling languages and techniques provide the ability to visualise and clarify confusing concepts in systems [43]. Therefore, the change modelling language proposed in this research can be used to clarify any ambiguous and/or confusing concepts associated with enterprise changes. This can enhance clarity, consistency, and coherency in understanding the underlying principles as well as reinforce the knowledge of enterprise changes.

Furthermore, since conceptual models or modelling languages are usually the basis for developing and implementing database systems [91], the proposed change modelling language can be used to develop and implement a database system. An example of such a database implementation has been shown in Appendix E. A database system can support enterprise agility and change management in many ways. For instance, database systems can enhance information management, integration, and data sharing, these have been found to be useful facilitators of enterprise agility [65, 182].

Database systems provide functionalities for queries, forms, and reports. These can be used to obtain useful business intelligence and information to support decision making for change management. For instance, successful initiatives such as action, agility enablers, etc., used to manage or adapt to previous changes, can be captured with forms and stored in a database system. In the future, when similar or even unprecedented changes occur, the database can be queried to obtain reports of these previous successful change management initiatives. These reports can then be examined and used to support decision of the best initiative to adopt. In this way, the cost and other resources consumed in making decisions towards change management can be reduced. In addition, this can encourage re-use of successful previous change management initiatives, thereby shortening the time and effort required to come up with new initiatives for adapting to changes.

1.5.2 The Concise Enterprise Modelling Language

The enterprise modelling language proposed in this research reuses the concepts and ideas from existing enterprise modelling approaches such as TOGAF and ARCHIMATE. This language describes an enterprise as an embodiment of its business objectives, and the set of (business process) activities together with information (business entities and attributes) required to achieve these objectives. As used in this research, the proposed enterprise modelling language provides a means to model an enterprise, and relate the resulting model with a change model, in order to derive the new enterprise elements required to adapt to the change.

However, the proposed language can also be useful in other areas. For instance it can be used by small scale enterprises, which may not afford to use popular approaches such as TOGAF and ZACHMAN, to describe the structure of their organization in terms of goals, business processes, and data. Existing enterprise modelling approaches such as TOGAF, ZACHMAN, etc., usually require expertise and expensive resources, for effective usage, which some small scale enterprise may not afford.

Modelling of small and medium scale enterprises (SME) is an emerging aspect of enterprise modelling with promising benefits [33, 84, 172]. Yet, there are little or no evidences that existing and popular enterprise modelling approaches can adequately be used for SMEs without rigorous adaptation process. Perhaps, this is because too often, existing approaches can include myriad model constructs and concepts which may not be applicable to SMEs.

The concise enterprise modelling language proposed in this research can be suitable to SMEs and other type of enterprises that may wish to represent their enterprise in three core domains. These include the business goals, the processes required to achieve these goals, and the operating data of the enterprise. In this way, they can exclude other concepts that may not be applicable to them. In addition, this concise representation appears to be easier to use, simpler to understand and provides less complexity relative to most existing approaches.

1.5.3 Procedure for Deriving New Domain Requirements (Elements)

The framework proposed in this research also provides a set of procedures and rules that can be used to derive the requirements (data, business process, and goal) for adapting a given enterprise change. The term *requirements* is derived from requirements engineering (RE), and used in this research to refer to the the essential criteria for the existence of a system [114, 201]. In other words the essential criteria for an enterprise to exist in a target state where it has adapted to a change. Requirements engineering (RE) is a branch of software engineering that deals with the identification, modelling, analysis, and elicitation of the requirements of a system.

In the area of enterprise modelling/architecture (EM/A), RE has been used to gather, analyse, and specify domain requirements for an enterprise. That is, the goals, data, business processes, *etc.*, required for an enterprise to function [55]. But enterprises, as well we as their domain requirements, are vulnerable to steady changes due to change drivers such as government regulations [49, 134]. When an enterprise undergoes changes, it usually transits from its current state to a new (target/to-be) state. As explained in Section 1.1.1, this new state would require a set of new domain elements to complement the current domain of the enterprise. These new domain elements would be vital and are considered to be the requirements for adapting to changes and facilitating enterprise agility.

Hence, it would be useful and necessary to provide a systematic approach for deriving the new domain elements, so that the enterprise can effectively adapt to a given change. But such a systematic approach is often neglected, if considered at all, in the existing enterprise modelling approaches. This research contributes to knowledge in RE by providing a set of rules and procedures that can be used to derive new domain elements or requirements for adapting to changes. These rules provide a structured means to relate and compare the elements (goals, data, business processes) of a change model with those of an enterprise model. By relating

and comparing both models, it can be easier for enterprise managers to precisely discern the business goals, strategies and objectives, as well as information and processes required to adapt a given change.

1.5.4 Conceptual Modelling Techniques and Technologies

The contributions of this research heavily rely on implementation and application of conceptual modelling languages. Hence it is pertinent to discuss some technologies and techniques that are relevant to the design and implementation of the conceptual modelling. Meta-modelling is one of the key techniques in conceptual modelling language development. A meta-model structures and describes the essential features of a modelling language [43], which include the abstract syntax, the concrete syntax, and semantics of the language.

Abstract syntax describes the underlying concepts of a modelling language and how these can be combined into a model. Concrete syntax describes how the model can be presented to the end user using notations. Semantics describes the meaning of the language [43]. Therefore the techniques involved in developing a meta-model and the resulting modelling languages include the description and construction of these essential features. These techniques are applied to develop the proposed modelling languages and are therefore important aspect of this research. Hence they should be described and discussed in detail before applying them to develop the proposed languages. Chapter 3 provides the description and discussion of meta-modelling techniques.

The meta-model of a modelling language should be described using existing technologies such as meta-model architecture (MMA) [43]. The meta-object facility (MOF), an object management group standard, is the meta-model architecture used to describe the modelling languages proposed in this research. The MOF technology is selected because it provides a rich set of concepts that can be used to describe any modelling language. Examples of these concepts include classes and associations. In addition the MOF can be easier to use and understand by a wide range of audience, hence most conceptual modelling languages are described using MOF concepts. The MOF and its concepts play key role in describing the proposed modelling languages. Hence, it would be vital to provide further description and discussion of the concepts and parts of MOF that are applicable to this research. These are discussed in detailed in Chapter 3.

A modelling language should also provide an abstract syntax model, which is a precise representation of the concepts in the modelling language, and the relationships between these concepts. Abstract syntax models are usually constructed using existing modelling framework or editor. The modelling languages proposed in this research also provide abstract syntax models. These are constructed using the eclipse modelling framework (EMF). The EMF is an integrated development platform or software system for model driven development. It provides a set of functionalities, such as the *ecore* modelling editor, that can be used to construct the abstract syntax model of a language. The EMF *ecore* modelling tool is selected because it can be used to model the concepts of the MOF technology, and these concepts are the basis or MMA for describing the proposed languages. The concepts of EMF and *ecore* that are most relevant to this research will be discussed in Chapter 3.

1.6 Research Methodology

This research aims at providing a framework that can be used to support enterprise agility and change management. This framework heavily relies on conceptual modelling, and also provides languages for modelling an enterprise as well as the changes an enterprise can experience. The change modelling language is developed by designing new model constructs that extend existing enterprise modelling approaches with change modelling concepts and capabilities. Model construct is a type of information systems (IS) artefact[90]. The design of IS artefacts is the primary concern of design science research (DSR) proposed in Hevner et al [4, 89, 90]. Therefore, this research will adapt the DSR framework as its research approach.

The DSR ensures high quality IS artefacts by providing a framework for conducting information systems research, and a set of guidelines (GL) [90] to support the realisation of this framework. To ensure that the artefacts and the corresponding modelling languages contributed by this research are of good quality, this research adopts and adapts these framework and guidelines proposed by Hevner et al as its methodology. Although there might be other frameworks for information systems design, the DSR framework supports the creation of artefacts to solve IS problems [67, 90], which is the key aim of this research. In addition, DSR appears to be more detailed and comprehensive, yet easy to adopt. Several successful IS and enterprise modelling research, such as [4, 15, 134, 150], have adopted the DSR approach.

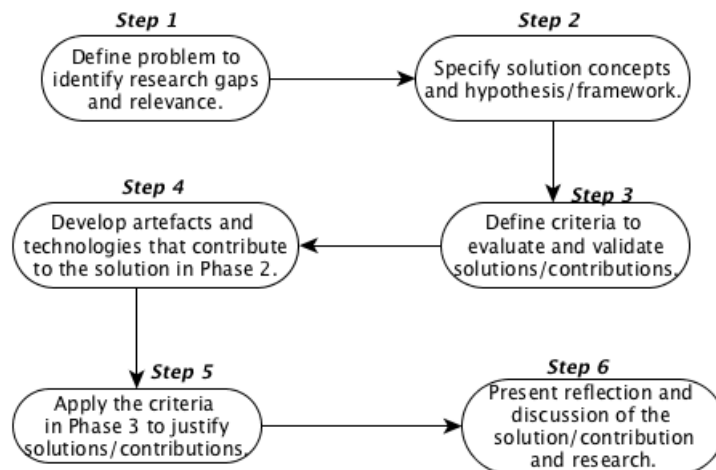


FIGURE 1.3: The Research Approach

To show that the methodology used in this research is consistent with the guidelines for conducting information systems research. The activities followed in this research are summarised into six (6) key steps, as shown in Figure 1.3. These steps are discussed further in Chapter while Figure 1.3 is used to demonstrate how they (these steps) relate to one or more DSR guideline. Chapter 4 also provides the description of DSR framework, and discusses how this research conforms to this framework.

1.7 Validation and Evaluation Criteria

Evaluation and validation of design artefacts is an important guideline in the DSR framework. It involves the use of rigorous method to demonstrate the quality and utility of the design artefacts [15, 134]. Hevner et al [89, 90] propose various methods that can be used to demonstrate the quality and utility of design artefacts. One of such methods is the use of a case study to demonstrate the quality as well as utility of design artefacts. According to Hevner et al [89, 90], the contributions or design artefacts can be evaluated or validated by using them to represent an aspect of a physical world in given domain or universe of discourse. The aspect of a physical world is usually demonstrated or captured in form a case study and/or a detailed scenario [89, 90]. Hence the designer of information systems artefacts or the researcher should obtain a case study that demonstrates the intended aspect of

the physical world. In addition, the designer should specify some criteria which can be used to validate and/or evaluate the design artefacts or the contributions.

In this research, an aspect of the physical world, particularly in the domain of enterprise agility and change management, is captured and demonstrated using a case study. This case study was obtained from my Research Group's industry collaborator. The description of this case study has been provided in Chapter 7. In order to adequately validate and evaluate the contributions or design artefacts of any research, as suggested by Henver et al, some criteria should be specified. Therefore, the criteria for evaluating the contributions/design artefacts of this research are listed below. To be evaluated and valid, the contributions/design artefacts should be able to

- i Represent/model the enterprise changes described in the case study.
- ii Represent/model the enterprise described in the case study.
- iii Derive new domain elements required to adapt the change described in the case study.

If these criteria are demonstrated to be met, then the contributions and design artefacts proposed in this research are valid and evaluated.

1.8 Publications and Conference Presentations

Some of the contributions of this research have been peer reviewed, accepted, presented, and published in reputable conferences and journals. Some others were also rejected but useful, since each of the rejected papers came back with good and useful comments. Even though rejection of papers is not a pleasant experience, the comments that came with them were further fed back to help improve the research contributions.

In addition to these publications, the Author participated, presented, and discussed the research contributions in 6 international conferences including USA, France, Brazil, Spain, UK, and Australia. These conferences are summarised in Table 1.1. Scholarly comments and feedbacks were also received at the end of each presentation and discussions. These were again fed back into the research for further improvements. Apart from international conferences, the Author has also attended and presented twice at the summer conferences in the University where he conducts his research. It is also worthy of mentioning that this research has won

TABLE 1.1: Conference Presentations and Participations

YEAR	CONFERENCE	VENUE
EDOC' 2015:	19th International Conference on Enterprise Distributed Object Computing:	Adelaide, Australia
SAC' 2015:	30th ACM Symposium on Applied Computing:	Salamanca, Spain
ER' 2014:	33rd International Conference on Conceptual Modelling (MReBA workshop):	Atlanta Georgia, USA
MDXSC' 2014:	9th Middlesex University Summer Conference	London, UK
RE' 2013:	21st Int'l Conference on Requirements Engineering (MoDRE Workshop):	Rio de Janeiro, Brazil
ECMFA' 2013:	9th European Conference on Modelling Foundations and Applications. (GMLD Workshop):	Montpellier, France
MDXSC' 2013:	8th Middlesex University Summer Conference	London, UK

an award and attracted research grant, from ACM, during the ACM/SIGAPP Students Research Competition (SRC) sponsored by Microsoft Research.

Table 1.2 presents a summary of the peer reviewed, accepted, and published contributions of this research. Some papers that are currently under review are also included in the first two rows of this table. The papers included in Table 1.2 are those in which the Author of this research is the first/ lead Author.

1.9 Thesis Organization

The remaining part of this thesis is organised as follows. In Chapter 2, the basic concepts required to precisely represent or model enterprise changes are presented. These concepts were identified through a systematic literature review. In addition, this chapter also identified and examined the existing enterprise modelling approaches, so as to understand the extent to which they support enterprise agility and change management. The existing technologies that are used in developing the proposed solution are presented and discussed in Chapter 3. These include technologies such as meta-modelling approaches and conceptual modelling language development. In Chapter 4, the research the proposed solution framework is discussed in detail. Further description of the research methodology, language design principles, and other aspects of this research are also presented in the same Chapter.

²I voluntarily withdrew this paper because there was no funding to register and attend the conference

TABLE 1.2: List of Publications from This Research

PAPER TITLE	RELATIONSHIP TO THIS THESIS	STATUS	PUBLISHER
A Conceptual Model for Enterprise Agility and Change Management	This paper shows how the change modelling language proposed in this research can be used to precisely represent enterprise changes. It relates to Chapter 5 and Chapter 6 of this Thesis	Under Review	SPRINGER: Business and Information Systems Engineering (BISE) Journal
A Conceptual Approach to Enterprise Agility and Change Management	This paper presents a set of concepts that can be used to develop information systems and/or conceptual models to support enterprise agility. This paper relates to Chapter 2 of this Thesis.	Under Review	ELSEVIER: Information and Management Journal
A Modelling Technique for Enterprise Agility	This paper presents a technique for modelling enterprise changes. Its part of Chapter 5 and Chapter 7 of this Thesis.	Accepted ² for publications	IEEE: 49th International Conference on Systems Sciences (HICSS 2016).
A Data Centric Approach to Change Management	This paper demonstrate how the proposed change modelling language can be used to implement a Database Systems to support enterprise agility and change management. This relates to Chapter 5 and Chapter 7.	Accepted and Published, See [140]	IEEE: 19th International Conference on Enterprise Distributed Object Computing (EDOC 2015).
A Conceptual Framework for Enterprise Agility	This paper proposes a conceptual framework for enterprise agility, which was used to develop a proforma to capture, and document change management initiatives and activities. This relates to Chapter 5 and Chapter 7 of this Thesis.	Accepted and Published, See [139].	ACM: 30th ACM Symposium on Applied Computing (SAC 2015).
A Framework for Enterprise Agility	This paper describes the overall idea of applying conceptual modelling to solve enterprise agility and change management problems. It won a research award/grant from the ACM/SIGAP Students Research Competition (SRC) sponsored by Microsoft Research. It is a summary of all the main contributions of this Thesis.	Accepted and published, See [138].	ACM: 30th ACM Symposium on Applied Computing (SAC 2015).
Automated Completeness Check in KAOS	This paper relates to Chapter 5 and Chapter 6 of this Thesis. It implements the set of rules, provided by the KAOS Language, for checking the completeness of a KAOS model.	Accepted and Published, See [141]	SPRINGER: Advances in Conceptual Modelling (BOOK).
Towards A Comprehensive Meta-Model for KAOS	This paper relates to Chapter 2 and Chapter 5 of this Thesis, and provides an integrated meta-model for the KAOS goal oriented modelling approach.	Accepted and Published, See [136]	IEEE: International workshop on Model Driven Requirements Engineering (MoDRE 2013) as part of RE 2013.
A Proposal for Consolidated Intentional Modelling Language (CIML)	This paper extends existing goal oriented modelling approaches with richer, but less cumbersome, model constructs to support enterprise requirements modelling and analysis. It relates to Chapter 5 of this Thesis.	Accepted and Published, See [137]	ACM: International Workshop on Graphical Modelling Language as part of ECMFA 2013.

The proposed change modelling language and its essential features are presented in Chapter 5. These essential features include the abstract syntax, the concrete syntax, and the semantics. Chapter 6 presents the enterprise modelling language together with the rules and procedures for deriving the domain requirements or new domain elements required to adapt a given change. In Chapter 7, a case study is used to demonstrate the utility of the proposed framework, this also validates and evaluates the contributions of this research. This utility is demonstrated in three key areas. These include conceptual modelling of enterprise changes, deriving the domain requirements or new domain elements required to adapt a given change, and implementing a database system to support enterprise agility and change management. Finally, the discussions, reflections, and limitations of this research are presented in Chapter 8.

Chapter 2

Literature Review

2.1 Overview

In order to initiate any meaningful analysis of enterprise changes, such as relating a change to an enterprise and discerning the enterprise elements required to adapt the change, it would be necessary to provide a means of representing enterprise changes explicitly. Such a representation can also enhance understanding, visualisation and interpretation of enterprise changes. One way to provide such means is by designing and developing a conceptual language for modelling enterprise changes.

The first step in developing a conceptual modelling language is usually to identify the necessary and sufficient set of concepts in a given aspect of the physical world or universe of discourse (UoD) [91, 132, 133]. These concepts are then related and integrated into a conceptual schema or modelling language. Therefore, in order to develop a suitable conceptual model to support enterprise change management and agility, it would be vital to, first of all, identify the concepts that can be used for developing such model. The main purpose of this chapter is to identify the necessary and sufficient set of concepts that can be used to develop conceptual models to represent change, as well as support enterprise agility and change management.

To ensure adequacy and wide coverage, these concepts are identified through a systematic review process. The two methods of conducting systematic reviews in Information Systems include systematic mapping study (SMS) and systematic literature review (SLR) [153]. A systematic mapping study aims at identifying, structuring, and classifying a given research area/topic in terms of some metrics such as frequency of publication, database of publication, coverage, etc., with the aim of discovering research trends [14, 152, 153].

But a systematic literature review focuses on gathering, analysing, and synthesising results from a set of selected primary studies/papers to answer more specific questions than SMS, and arrive at vital conclusions [14, 152]. Other differences between systematic literature review and systematic mapping study are available in Petersen et al [152]. Since the purpose of this review is to identify sufficient set of concepts for representing enterprise changes, rather than structuring or classifying the trends or frequency of publications of enterprise changes; a systematic literature review is considered to be more suitable for this review, and hence selected as the literature review method.

The guideline or methodology for conducting systematic literature reviews, especially in Information Systems, have been proposed by Kitchenham in [107]. The systematic review conducted in this chapter is informed by this methodology. But before discussing it, it would be useful to, first and foremost, define enterprise agility, and clarify the differences between its various types. In this way, readers would be able to have the right perspective about enterprise agility and the concepts of enterprise changes proposed in this chapter.

The remainder of this Chapter is organised as follows: Section 2.1.1 provides the definition of enterprise agility and differentiates between its types. Afterwards, a brief discussion of the systematic literature review methodology or guideline is presented in Section 2.2. This is followed by Sections 2.3, and 2.4, which demonstrate how Kitchenham's methodology is applied to conduct this systematic review. Section 2.5 presents the results of this systematic review. These results are then discussed and analysed in Section 2.7. Finally the conclusion to this chapter is given in Section 2.8.

2.1.1 Definition and Types of Enterprise Agility

Enterprise agility has received a growing attention, from Information Systems practitioners and researchers, in recent times. As a result, various scholars and publications, *e.g.*, [12, 27, 41, 145, 146, 148], have defined it in different ways. For instance, Arteta and Giachetti [12] define enterprise agility as the ability to adapt to changes and make use of opportunities that changes bring. Similarly, enterprise agility has been defined as the ability to sense and respond to environmental changes readily [27, 41]. Despite little discrepancies in these definitions, there is a general consensus that enterprise agility involves the ability to respond or adapt to changes in an enterprise. This research generally agrees with existing definitions, but also argues that concepts of enterprise changes can be used to develop conceptual models and information systems that can help enterprise acquire the

capability to adapt or respond to changes. *Therefore this research defines enterprise agility as the ability of an enterprise to respond to changes, timely and effectively, using conceptual modelling techniques.*

Due to its growing importance, various specialisations of enterprise agility have emerged in research and practice. As shown in Figure 2.1, these include manufacturing agility, supply chain agility, agile development, and others. Potentially, researchers and practitioners who are new to enterprise agility and change management may find it difficult to understand the relationships and differences between enterprise agility and its types or specialities. They may also find it difficult to differentiate between the various types of enterprise agility. Based on these, it would be pertinent to provide a brief description of the types of enterprise agility and draw a distinction between them.

Agile manufacturing or manufacturing agility refers to the ability to manufacture goods/services, in a quick and flexible manner, to meet the changing customers need. It is aimed at responding to changes and eliminate delays in the production of goods and services using various techniques. Some examples of these techniques include lean manufacturing, just in time production, and mass customization [73, 74, 181, 182].

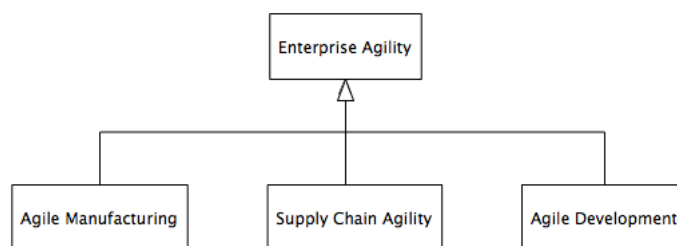


FIGURE 2.1: Specialisation of Enterprise Agility

Similar to agile manufacturing is agile supply chain agility [22, 66, 161]. The focus of a supply chain should be to deliver good and services to customers in faster, seamless, and efficient way. Supply chain agility centres on managing changes and reducing uncertainties in a firm's supply chain. On the other hand, *agile development* [30, 123, 197] is any software or systems development method that is responsive to changes in user/systems requirements, using a set of principles such

as iteration, incremental development, customer representatives, and team collaborations. Aside from these three core specialisations, enterprise agility is still emerging in other domains of enterprise. For instance, the idea of *business process agility* is receiving attention from a considerable number of scholars [86, 103, 164], even though it is yet to mature as a recognised speciality of enterprise agility.

TABLE 2.1: Difference Between Enterprise Agility and Its Types

TYPES OF AGILITY \ DIFFERENCES	Application Area	Examples of Models	Focus
Enterprise Agility	Enterprise Architecture	TOGAF ¹ , ZAF ²	Entire Enterprise
Agile Manufacturing	Manufacturing	Lean Manufacturing	Production of goods and services
Supply Chain Agility	Supply Chain	SCOR ³ Model	Delivery of goods and services
Agile Development	SDLC ⁴	XP ⁵ , SCRUM	Software/Systems user requirements

Table Acronyms:

¹The Open Group Architecture Framework • ²Zachman Architecture Framework • ³Supply Chain Operations Reference • ⁴Systems or Software Development Life Cycle • ⁵Extreme Programming

The central focus in any type of enterprise agility should be to respond and adapt to changes. But the enterprise agility under consideration has some differences from the other types of agility described above. The differences are summarised in Table 2.1 in terms of their applications areas, examples of their models, and focus. For instance, enterprise agility focuses on the entire enterprise and can be applied to an enterprise architecture. In contrast, manufacturing agility focuses on the production of goods and services, and can be applied to the manufacturing function of an enterprise. The idea of this research is that enterprise agility should be holistic. In other words, change management efforts and initiative should be directed towards the entire enterprise, rather focusing on any one speciality of the enterprise. Accordingly, the concepts of enterprise change proposed in this research are intended to focus on the entire enterprise. However scholars can also adopt and adapt these concepts to be fit for use in the other specialised areas of enterprise agility.

2.2 Review Methodology

This literature review is informed by an established guideline, proposed by Kitchenham in [107], for conducting a systematic literature review (SLR). Kitchenham's guideline has been widely used to conduct systematic reviews [8, 29, 52, 63], especially in the field of Information Systems and Software Engineering. This guideline provides the key processes involved in conducting a systematic literature review.

These processes can be summarised into three main activities, which include establishing review protocol, developing review strategy, and presenting review results. As shown in Figure 2.2, each of these activities involves some definite tasks.

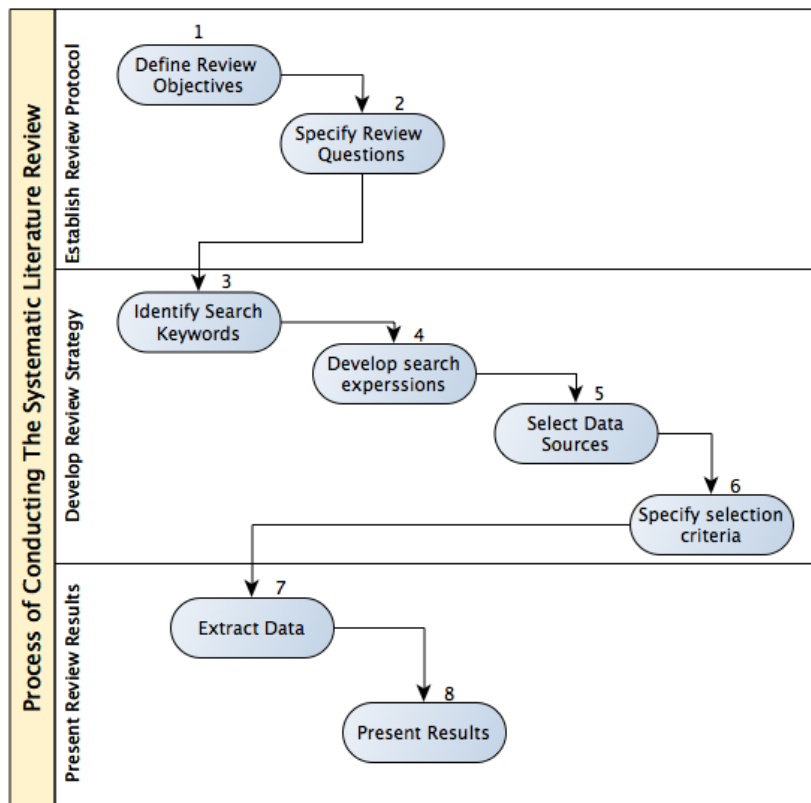


FIGURE 2.2: Summary Of Processes Involved In Performing SLR

Establishing a review protocol involves two main tasks, the first is to define the objectives for conducting the systematic literature review. The second task is to specify the review questions. As shown in Figure 2.2, the tasks involved in developing a review strategy include identifying the search keywords, and developing the search expressions. Others are selecting the data sources, and specifying the selection criteria. The last activity in an SLR process is usually to present the review results or findings. The tasks involved in this are extracting data from the selected data sources and presenting the data. The section that follows shows how each of these tasks are applied to conduct this systematic literature review.

2.3 Establishing Review Protocol

2.3.1 Review Objectives

The objectives of this systematic literature review are as follows:

- To identify the necessary and sufficient set of concepts that can be used to develop conceptual model and information systems to support enterprise agility and change management.
- To use these concepts as the basis for comparing existing enterprise modelling approaches used for enterprise agility, and examine the extent to which they can be used to support enterprise agility and change management.

2.3.2 Review Questions

In order to achieve the objectives of this review, the following questions are answered:

RQ1 What are the sufficient and necessary set of concepts that can be used to develop conceptual model and Information Systems to support enterprise agility and change management?

RQ2 What are the current enterprise conceptual modelling approaches and to what extent do they support enterprise agility and change management?

2.4 Review Strategy

2.4.1 Search Keywords

In order to identify relevant search keywords, the author read through related articles used in his previous research and publications. Few keywords were initially identified from this activity. These keywords were further used to conduct a pilot test search to identify more keywords relating to enterprise agility and change management. After the pilot testing, a set of relevant search keywords were identified, these are presented in Table 2.2.

TABLE 2.2: Search Keywords

A_1 . Enterprise Agility	B_1 . Concepts	C_1 . Frameworks
A_2 . Enterprise Change	B_2 . Terminologies	C_2 . Model ^{*6}
A_3 . Organizational Agility	B_3 . Features	C_3 . Method*
A_4 . Agile Enterprise		

2.4.2 Search Expressions

The next task is to develop the search expressions. To achieve this, the keywords in Table 2.2 are combined into boolean expressions, as shown in Listing 2.1. Some examples of these boolean expressions are as follows:

- (Enterprise Agility **OR** Enterprise Change **OR** Organizational Agility **OR** Agile Enterprise) **AND** (Concept).
- (Enterprise Agility **OR** Enterprise Change **OR** Organizational Agility **OR** Agile Enterprise) **AND** (Features).

LISTING 2.1: Search Expressions

```

1 (A1 OR A2 OR A3 OR A4 ) AND B1
2 (A1 OR A2 OR A3 OR A4 ) AND B2
3 (A1 OR A2 OR A3 OR A4 ) AND B3
4 (A1 OR A2 OR A3 OR A4 ) AND C1
5 (A1 OR A2 OR A3 OR A4 ) AND C2
6 (A1 OR A2 OR A3 OR A4 ) AND C3

```

To obtain a better search result, an asterisk (*) is included in some keywords during the search. This is a wild card operator or a place holder that can allow more search results to be returned. For instance by including asterisks in the 'model' keyword, the search will return results including modelling, models, etc.

This literature search follows a systematic, rigorous, and methodical approach. But in order to ensure that it not overly mechanical, the Author stepped out from the systematic process to conduct manual searches. For instance, manual searches were conducted in Google Scholar. The Author also conducted manual searches in Google, Wikis, and, other relevant websites. The essence of these manual searches is to ensure that the results and identified concepts generally conforms reality and are obtainable concepts in real world enterprises.

⁶This asterisk is a wild card operator that can return results including other keywords such as modeling and methodology

2.4.3 Data Sources

After specifying the search keywords and search expressions, the next step is to identify data sources. In other words to identify databases that store publications relevant to enterprise agility and change management. The selected data sources are shown in the extreme left of Table 2.3.

TABLE 2.3: The Systematic Review Strategy

Data Sources	Inclusion Criteria (Paper must:)	Exclusion Criteria (Exclude:)
<ul style="list-style-type: none"> • Web of Science • Springer Link • Science Direct • IEEE Xplore • ACM Digital Library • EBSCO Host • Google Scholar^{*7} 	<ul style="list-style-type: none"> • be published between 1995 and 2015 • contain the relevant keywords • have full text available • be published in a relevant journal • be a conference or workshop article • be in English Language only • be relevant to enterprise agility and change management 	<ul style="list-style-type: none"> • position papers • keynotes & symposia • duplicated articles • irrelevant titles

These include Web of Science, Springer, Science Direct (Elsevier), IEEE Xplore, ACM Digital Library, and EBSCO Host. But to ensure a better result, a seventh (7th) data source *i.e.*, *Google Scholar* was included. This is marked with asterisk in Table 2.3 to show that it was manually searched. Usually, searching any of these databases would return more search results than necessary. Therefore, to ensure that only relevant and quality publications are selected for primary studies, a set of selection criteria is specified.

2.4.4 Selection Criteria

The selection criteria includes inclusion, exclusion, and quality criteria. Inclusion criteria specifies a set of conditions a publication must satisfy before it can be included for review. For instance, for a paper to be included for review, it must be published between 1995 and 2015. This range is chosen because enterprise modelling and agility started receiving reasonable attention, as per Information Systems, in middle nineties. Although John Zachman introduced the idea of managing enterprise changes using an enterprise model called Information Systems Architecture Framework in 1987, see [198, 199]; enterprise modelling and agility

⁷The asterisk (*) shows that the search in Google Scholar was done manually

became prominent in middle nineties, hence the selection of 1995 to 2015 publication range. On the other hand, exclusion criteria specifies some factors for excluding a publication from review. The centre column of Table 2.3 presents a set of inclusion criteria, while the extreme right column presents the exclusion criteria.

TABLE 2.4: The Quality Criteria Used to Select Primary Studies

Criteria	Description
• Relevance	The selected paper must be relevant to enterprise agility, change management, and enterprise changes.
• Validity	The selected paper must use an appropriate research method or discuss at least one enterprise modelling framework, method, or technique.
• Thorough	The selected papers must be peer reviewed.
• Neutrality	The selected paper must not be published by the author, and any of his supervisory professors.

The essence of defining and using quality criteria is to ensure that adequate rigour is employed in selecting the primary studies. In addition, it ensures that quality publications are selected for primary studies and the SLR produces standard results [107]. The quality criteria used for this review are presented and described in Table 2.4.

2.5 Review Results

2.5.1 Data Extraction

At the end of the search process, a total of 347 papers were extracted and downloaded into BIBTEX, the bibliography management systems used to store and manage search results. The extracted papers were screened, using the screening stages shown in Figure 2.3. The first and second screening stages respectively involve the use of the exclusion and inclusion criteria discussed in Section 2.4.3 and presented in Table 2.3. As shown in Figure 2.3, a total of 119 papers were selected after applying the first and second screening stages.

In the third screening stage, the Author read through the abstract of these 119 papers to identify those that focuses on enterprise agility. A total of 68 papers were selected at the end of this stage. In the final screening stage, the quality criteria discussed in Section 2.4.3 and presented in Table 2.4 are applied to these

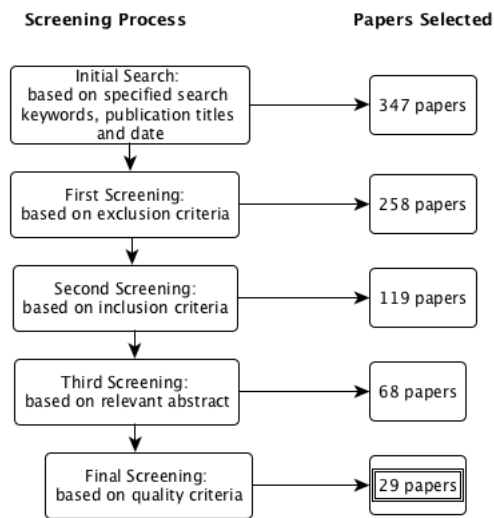


FIGURE 2.3: Systematic Literature Review Screening Process

68 papers. Finally, a total of 29 papers were selected and included in the primary review.

2.5.2 Results and Answers to Review Questions

The Author read through each paper selected for the primary study. Knowledge, ideas, and information gathered from reading the papers in the primary studies are then used to answer the review questions, as presented in the sections below.

2.5.2.1 Answer to Review Question 1

The first review question is related to the first review objective, which intends to identify the set of concepts that can be used to support enterprise agility. This research question is presented as follows:

What are the necessary and sufficient set of concepts that can be used to develop conceptual model and information systems to support enterprise agility and change management?

Table 2.5 presents the necessary and sufficient set of concepts that can be used to develop a conceptual model and/or information systems to support enterprise

agility and change management. These concepts are identified and selected from the primary studies. All possible concepts were initially identified from primary studies. But the concepts that are used by various authors and predominant in the primary studies are considered to be essential to enterprise agility and change management. Therefore, the selected concepts are those that appear in three or more papers of the primary study. Table 2.5 also shows similar terms used to depict each concept in literature and provide references to the primary studies where the concepts come from.

TABLE 2.5: Enterprise Agility and Change Concepts

Concepts	Similar Terms Used in Literature	Primary Study
Change Driver	Agility Drivers, market drivers, drivers of change, change factors	[40, 100, 101, 119, 145, 178, 183]
Change	Environmental Change	[135, 145, 147]
Action	Competitive Actions	[60, 191]
Agility Enablers	Agility Prerequisites, Agile Values, Agility Capabilities, Enablers	[40, 60, 145, 190]
Change Indicator	Change Anticipation, Change perception, change sensing, detecting changes	[49, 95, 147]
Derived Concepts: Data, Actor, Goal, Impact, Business (Process) Activities.	The are derived concepts that can be useful to enterprise agility and change management	These concepts are derived from existing enterprise modelling approaches

In addition to the core concepts identified from the primary studies, there are other concepts which are derived by the Author. These other concepts are considered to be useful to enterprise agility, and can complement the core concepts. For instance, the *goal* concept is not identified from the primary study, as a concept of change. But it is important and has been included because a goal can be used to express an enterprise change in an understandable form. A goal can also provide the means to decompose a change into actionable objectives. This type of concepts are simply called *derived concepts* in Table 2.5, and are further explained in the discussion and analysis Section in 2.7.

2.5.2.2 Answer to Review Question 2

The second review question intends to identify existing enterprise modelling approaches that can be used to support enterprise agility. In addition, it seeks to find out the extent to which these approaches support enterprise agility in their states. This question is as follows:

What are the current enterprise conceptual modelling approaches and to what extent do they support enterprise agility and change management?

Table 2.6 presents a summary of various enterprise modelling approaches that can be used to support enterprise agility and change management. Most of these approaches are derived from the manual search conducted in the Google Scholar database. The enterprise modelling approaches selected and shown in Table 2.6 are those considered popular, deemed to be representatives of other enterprise modelling approaches, and are applicable to the entire enterprise. Other approaches such as the supply chain operations reference model (SCOR) which applies to specific area of an enterprise are not selected.

TABLE 2.6: Existing Enterprise Modelling Approaches

EM Approaches	Reference
TOGAF	[101]
ZACHMAN	[151]
ARCHIMATE	[115]
ARMOR	[55]
BMM	[19]
i*	[56]
UML	[170]

As shown in Table 2.6, the selected enterprise modelling approaches include, the open group architecture framework (TOGAF); the ZACHMAN framework (ZAF), which provides an ontology for describing an enterprise. An enterprise modelling language known as ARCHIMATE, and its extension ARMOR. The business motivation model (BMM) and the distributed intentionality (i*) are techniques for conceptual representation of the motivations and goals of an enterprise. The unified modelling language (UML) is a language that can be used to model enterprise systems and software.

Each of these approaches are discussed in Section 2.7.2. In order to determine the extent to which they can be used to support enterprise agility and change management, the concepts of enterprise changes identified from this review are used as the basis to compare the existing approaches. This comparison is discussed in Section 2.7.2.

2.6 The Case Study

As suggested by Hevner et al [89, 90] in their design science research (DSR), the contributions of Information Systems research can be evaluated and validated using a case study or scenario. This research uses the DSR framework and thus validates its contributions using an industry case study provided by the Research Group's Industry partners. This case study is described in below, and used to describe and cite examples, where necessary, of the concepts and existing enterprise approaches derived from this systematic review and discussed in Section 2.7. Furthermore, references are made to this case study in the rest of this chapter and in subsequent chapters of this Thesis.

Case Study Description: *Enterprise X* is a global leader in Information Technology (IT) Consultancy Services, whose mission is to help achieve Clients' business objectives by providing innovative and world class development, maintenance, and testing of software systems as per stated requirements. At the same time, *Enterprise X* wants to stay competitive, viable as well as maintain its position as a global leader in IT consultancy.

Enterprise X bids for requests for proposals (RFP), staffs the bid won with the right number of suitably skilled resources, follows a set software development process leading to successful delivery, and winds up the project by releasing the human and other resources being utilized for project execution. The enterprise has to function in the face of several delays such as bidding delay. To create value and achieve its corporate objective, the enterprise has to optimize cost and risk associated with its operations.

The key goal of *Enterprise X* is to stay viable and competitive. To achieve this goal, the enterprise has to develop a master plan in two key areas. First, it tends to maintain an outstanding bidding strategy, which ensures that bids are completed timely, and proposals are sent early. Since late submission of bids reduces the chances of winning a bid. In addition, it maintains a strong research department, whose major role is to develop innovative technologies and send unsolicited proposals to clients. Secondly, to stay ahead of its competitors and keep up with world class global IT services, this enterprise maintains a diversified recruitment strategy and workforce. In other words, the enterprise recruits experienced specialists, and graduates with high grades from high ranking academic institutes, from both home and abroad. It also has to keep in pace with advance in technology and functional domains through adequate training or re-skilling of existing workforce.

The enterprise, in essence, is an engine managing the demand, i.e., the number and the nature of RFPs, and the supply, i.e., the number of bid sent out. For a given demand and supply state, the enterprise has to determine costs effective strategies that will help achieve the desired goals. For instance, it can decide to focus solely on having the right number of staff. It can also decide to do more and/or better with less, in which case it hires more experienced and qualitative staff from the job market. Either way, having adequate workforce on its rolls becomes an important goal, but also cost implications are always taken into considerations in all cases and at all times.

In order to enhance operational efficiencies, the organisation defines operational processes for implementing its business transactions. Those processes that are key to achieving enterprise goals are usually given special attention. For instance, the recruitment process should be explicitly defined and followed to ensure that the required and adequate workforce are in place. In addition, it is very important for the enterprise to structure, document, and monitor key information/data that can support effective operational management and governance. For instance, data information about applicants, recruitments, bid, RFPs, etc., should be structured and monitored.

In reality, as expected, this enterprise operates in an environment characterised by spontaneous changes, mainly due to regulatory compliance and competition. Recently, a new government was sworn into power after a just concluded election. Environmental factors such as unemployment, rigging in bidding market, and surge in net migration have necessitated some new government regulations which this and other enterprises have to comply to. Additionally, some hardware manufacturing enterprises have also started providing IT services. These two change drivers are described in details below:

Immigration and Resident Employment Act (IREA): This Act seeks to reduce the immigration of foreign workers, and secure jobs for citizens and residents. In other words, enterprises which recruit workforce from abroad must give preference to those that do not require work/resident permit. In effect, enterprises are required to tighten employment eligibility criteria, for foreign workers, during recruitment. Also enterprises should be ready to prove that they have complied to this Act. In these ways both unemployment and net migration can be reduced.

Competition: Recently, many enterprises are beginning to consider Information Technology (IT) Services as a viable and profitable industry. Hence, four new IT service providers have just entered the industry. To attract clients, they launched radical marketing strategies. These include 18% discounts in all IT

services, together with money back guarantee to clients who are dissatisfied. In addition, they offer longer training and maintenance services to clients. As a result the existing enterprise, under consideration, lost about 24% of its clients base to these competitors.

2.7 Discussion and Analysis

This section defines and analyses the concepts identified from the primary studies. It also provides a brief discussion of the existing enterprise modelling approaches, and examines the extent to which they can be used to support enterprise agility.

2.7.1 The Concepts

Change Driver: A change driver is any event or circumstance that can lead to change(s) in an enterprise. Events, such as organizational politicking and internal strike actions, can originate from within an enterprise boundary. A change driver from this type of events is called an *internal change driver*. Conversely, events such as regulatory compliance can originate from outside the boundaries of an enterprise. Change drivers from such events are called *external change drivers* [100, 101, 183].

In the case study described in Section 2.6, there are two change drivers. The first is government regulation (regulatory compliance) *i.e.*, the immigration and residency employment act, while the second is *competition*. Both change drivers are external since they originate from outside the enterprise.

The change driver concept can be important to enterprise agility and change management in some ways. They can be used as the basis to describe, represent, and communicate the causes as well as the origin of changes in an enterprise. It is widely known that when the root cause or origin of any problem is clearly identified and described, the solutions to that problem are always easier to formulate and articulate. Therefore, understanding the root cause and origin of an enterprise change can help enterprise stakeholders to proffer viable strategies for effective change management.

In addition, the change driver concept can be used to express and communicate the rationale behind an enterprise change. In other words why an enterprise must transit to another state. If enterprise stakeholders can have a clear understanding of why their enterprise must change, then they can view such a change as a shared

concern, and be motivated to make concerted efforts in developing adequate initiatives and actions to manage such a change. Furthermore, understanding the rationale behind a given change can motivate adequate planning and prudent allocation of enterprise resources for adapting to the change. In these ways, enterprise agility can be facilitated or even achieved.

Change: A *change* is a transition, in one or more enterprise domain(s), from a current (*AS-IS*) state to a desirable (*TO-BE*) state, as a result of a change driver. Enterprise domain here refers to the various viewpoints or aspects of an enterprise such as goals, data, and business process.

In the above case study, a change will occur if *Enterprise X* transits from its current state to a target state where it tightens employment eligibility criteria, and provide evidence that it has done so. Any of such transition would lead to some modifications in at least one domain of an enterprise. These modifications usually require the addition of new domain elements, *e.g.*, data entity, to one or more domains of the enterprise. For instance, before it can transit to the state of tightening employment eligibility criteria, *Enterprise X* may require to add a new business goal *e.g.*, 'provide a mandatory psychometric test to all oversea applicants', and new data entities regarding psychometric test *e.g.*, *testScore*, *testType*, *etc.*, to its existing enterprise data. Thus a change can be captured or described in terms of the modifications required by an enterprise to transit to a target state in an enterprise. The target state can be used to refer to the state an enterprise should be in order to effectively adapt to the change drivers, and remain agile.

A change is expected to have some effects, costs, or consequences to the host enterprise. These can include financial losses, bankruptcy, loss of competitive advantage, etc [12, 145]. The term '**change impact**' is a derived concept (see Table 2.5) that can be used to describe and express the consequences or implications of a change to an enterprise. Hence, **Change Impact** is an important attribute of an enterprise change and can be regarded as one of the concepts of enterprise agility. In the case study above, if *Enterprise X* introduces psychometric tests for oversea applicants, then the impact could be the costs, *i.e.*, time, efforts, resources, involved designing and implementing the psychometric testing service.

For enterprise stakeholders to be motivated to make adequate efforts and contributions to change management initiative; the rationale for changes, in other words why an enterprise should transit to another state, should be captured, described, and expressed clearly. More so, in order to formulate effective change management initiatives, and further understand the modifications required for a given transition, changes should be decomposed into actionable objectives [138–140].

Goal oriented modelling (GOM) techniques can be useful to these ends. GOM techniques provide the means to conceptualise the rationale and motivations behind enterprise initiatives such as change management. *Goal*, a term used to capture the rationale, objectives, and motivations of a system or stakeholders thereof, is the central term in GOM [56, 114, 141]. Since goals can be decomposed into subgoals [56, 114, 137], they can be used breakdown a change into actionable objectives, as well as into clear and understandable formats. Hence, **Goal** is another derived concept that can be useful to enterprise agility and change management.

The *change* concept can be important to enterprise agility in some ways. For instance, it can be used to capture, represent, and describe the (target) state an enterprise should transit to, in order to adapt a given change driver and remain agile. This can improve the understanding of enterprise changes, and further clarify any tacit and ambiguous ideas associated with enterprise changes. Additionally, it can provide the means to precisely conceptualise the required modifications an enterprise should make in its domain (s), in other to transit to the needed (target) state; so that it can be easier to understand, interpret, and examine how an enterprise should be structured or configured and how it should operate and function in order to adapt change drivers, facilitate, or even achieve enterprise agility.

Equally the *change impact* concept would be useful to enterprise agility and change management. It can provide the means to conceptualise and reason about the effects, implications, or consequences of a given change in an enterprise. This can make it easier to understand and have knowledge of any negative effects or potential risks associated with a particular change management initiative. Such understanding and knowledge of risks can help in performing risk assessment and change impact analysis. These are known to be advantageous to change management and enterprise agility [25]. Similarly, goals are important, since they can provide the means to express and decompose changes into understandable formats and actionable objectives. In this way, enterprise stakeholders can have a clear understanding of the rationale for change and concerted efforts to contribute suitable ideas for effective change management.

Action: An action can be define as a definite task or piece of work carried out to transit to a desired state of an enterprise. For an enterprise to effectively transit from its current (AS-IS) state to a desired (TO-BE) state, it should define specific actions. For instance, the transition to a target state of tightened employment eligibility criteria, for *Enterprise X* in the case study, would require some definite actions. Examples of such actions could be developing a psychometric testing business process and implementing a testing business service.

To enhance understanding and clarity, each action should be broken down into definite activities. For instance, business process re-engineering (BPR) can be broken down into activities such as identify the process to re-engineer, elicit the requirements for the BPR, model the process, and so on. On the other hand, these activities can be organised in a logical and coherent manner called **business process** [106]. Thus **(Business Process) Activities** is a derived concept of change. Either way, these activities should be expressed clearly and assigned to the relevant enterprise actor(s).

The term *Actor* is a derived concept of enterprise agility that can be used to describe any active entity that can play useful roles in enterprise change management initiatives. An actor can be human such as an enterprise stakeholder, or a system such a computer program [56, 114, 141]. Actors such as business process management team, enterprise data architects, or enterprise business architects can play useful roles if *Enterprise X* decides to develop a psychometric test business process as an action for tightening employment eligibility criteria for oversea applicants. Such actors should be identified, and made to be aware of the tasks they are required to perform or the role they should play to support change management initiatives in enterprises. Furthermore, it can be possible to identify more than one actions for actualising a given enterprise change. In such cases, it would be very useful to identify all available candidate actions. The relative advantages and disadvantages of each candidate actions should be examined to rank the actions in order of preference. The highest ranking action can be selected as the action to be implemented. The unselected actions can be reserved for contingency purposes, and can be used in case the selected action fails.

The action concept can be used to precisely capture and describe what an enterprise should do so as to transit to the target state required for adapting a given change. Such understanding can make it easier to formulate effective strategies and suitable initiatives to support enterprise agility and change management. Once actions are captured and broken into definite tasks, it can be easier to assign them to the concerned enterprise actors. In this way, enterprise stakeholders would know what they should do and further think about how they can contribute their skills and expertise to change management and enterprise agility initiatives. In addition, identifying more than one action can make suitable and credible alternative actions to be ready and handy. So that stakeholders can swiftly select and implement a new actions in case the initially selected actions fails. This can reduce the time, efforts and cost expanded in developing new actions for change management.

Agility Enabler: This can be defined as a function or technology that can support an action. Actions specified to actualise a change should be supported by an agility enabler, so that they can yield optimal results. In the case study above, assuming *Enterprise X* decides to develop a psychometric testing business process as the suitable action for actualising the change, then it should define some functions or technologies to support this action. Technologies such as business process execution language (BPEL) can be used as the enabler to support business process development. Other examples of agility enablers includes information transparency, business and information technology alignment (BIT), and data integration [40, 60, 145, 190].

The agility enabler concept can provide an avenue to clearly capture, describe, and represent technologies or enterprise functions that will be used to support actions for change management initiatives. An enterprise can then check if it currently has such technologies and functions. If not, it can assess the cost of acquiring or outsourcing such technologies, and if it can afford to do so. Assuming the enterprise cannot afford to acquire such technologies, they may consider using an alternative action with a more affordable agility enabler. In these ways, decision about a change management action and its enabler can be based on credible assessment of the available enterprise resources, technologies, and capabilities.

Change Indicator: Change indicator can be defined as any environmental or socio-economic condition that can trigger a change driver. Some examples of change indicator include unemployment and election. Most change drivers are usually triggered by environmental and socio-economic affairs such as surge in immigration and unemployment. For instance increased unemployment or high net migration can trigger change drivers such as government regulations (regulatory compliance) that can hurt enterprises. Similarly, social factors such as recession can trigger fierce competitions among enterprises.

The ability of an enterprise to sense its environment has been recognised as a critical factor for enterprise agility [49, 95, 146]. Therefore, to enhance or realise agility, an enterprise should have concept that can be used to describe or capture the affairs in its environment; so as to monitor and detect environmental affairs that are most likely to trigger change drivers. The change indicator concept can be useful to this end.

The degree of a change indicator, *i.e.*, how predominant or persistent an environmental affair is, should be captured. Social networks, and news media can be used to monitor the degree of a given change indicator. If such a change indicator is very persistent and most likely to trigger change drivers, then its degree is

said to be *severe*. On the other hand, if such change indicator is transient and less likely to trigger change drivers, it can be captured as *mild*. These can provide the means and used as the basis for adequate monitoring of enterprise environment. If enterprises can consistently monitor their enterprise environment, capture and describe change indicators together with their degrees; it can be possible to foresee or anticipate the potential change drivers and changes that can result from those change indicators. In this way adequate actions and initiatives can be developed and implemented, even before the change driver has impacted on the enterprise, to pre-empt such changes from hurting the enterprise.

For instance, assuming that the stakeholders in *Enterprise X* are sensitive to environmental affairs, they could possibly gain awareness, from campaign manifestos and other sources, of what is likely to happen if a new government is sworn in. In this way, they can anticipate and predict any potential regulatory change driver (such as immigration and resident employment act) that would affect their enterprise. Then adequate preparations, planning, strategy development, and actions can be put in place to pre-empt or reduce the effects of the anticipated change drivers and changes.

The idea of change indicator can introduce an important paradigm of '*change pre-emption*' to enterprise agility and change management. '*Change pre-emption*' refers to initiatives that can prevent environment changes from hurting/impacting an enterprise through constant monitoring and detecting environmental social affairs. This can yield a better change management result than current approaches where efforts and initiatives are directed towards managing changes that have already impacted an hurt the enterprise. In addition, the change indicator concept can be used to capture and document current social affairs that have become predominant. So that they can be examined, monitored, and managed at an early stage. This can provide a way for enterprise stakeholders to detect or foresee a change driver long before its occurrence. If a change driver can be detected and analysed at an early stage, *i.e.*, before it impacts on the enterprise; then it can be easier formulate and implement actions, in advance, to forestall the change driver and the resulting change from impacting and hurting the enterprise. This can enhance enterprise change management abilities, and in addition, support change pre-emption paradigm.

Data: Data are abstract terms that can be used to describe business entities and attributes that should be involved in a given change [50, 99]. An *entity* is used to describe objects of interest such as employee, loan, equipment, customer, etc. Each entity would have some properties, such as name and identifier, these are called

TOGAF: The open group architecture framework (TOGAF) [176] provides a guideline and supporting technologies for developing an enterprise architecture. TOGAF includes a capability framework, which is divided into 7 parts [101]. Part 2, *i.e.*, the architecture development method (ADM)⁸, is the core, and probably the most important part in TOGAF capability framework. The ADM describes 8 iterative phases for developing the architecture of an enterprise. Among the ADM phases, the enterprise change management or Phase H defines certain processes for managing enterprise changes. Hence, it is taken to be relevant to enterprise agility. Moreover, Phase H alluded to the concept of change drivers as the major cause of enterprise changes [101, 176].

In addition to Phase H, other phases provide some concepts that can be used to support change management and enterprise agility. For instance, the information systems architecture (ISA) or Phase C includes concepts such as business process activities, data entities. Even though TOGAF does not describe these concepts in terms of change management and enterprise agility, they coincide with the derived concepts shown in Table 2.5, and hence can be used to support enterprise agility. However, other concepts such as change indicator, and agility enabler appear not be described at all in TOGAF.

With respect to the case study in Section 2.6, TOGAF can be used to capture and describe the essential domains of *Enterprise X* such as data, business process, and business goals. More so, since TOGAF identified change drivers, in its Phase H, as the causes of enterprise changes; it can be used to capture and express the change driver experienced by *Enterprise X*. However, as shown in Table 2.7, other concepts such as change, impact, agility enabler, and change indicator that can be useful to describe the change in *Enterprise X* are either implied or not described in TOGAF as concepts of change. Therefore in its current state, TOGAF may only be suitable to provide a partial description and representation of the change in *Enterprise X*.

ARCHIMATE/ARMOR: ARCHIMATE [115] is a conceptual modelling language for conceptualising the structure, information and behaviour of an enterprise. Essentially, ARMOR [55] extends ARCHIMATE with goal oriented modelling capabilities. ARCHIMATE partitions the structure of an enterprise into three main layers, namely, the business, application, and infrastructure layers [116]. The business layer describes the business objects and processes of an enterprise. In the application layer, key data, application functions and services required for enterprise operations are captured and described; while the infrastructure layer describes the hardware and software capabilities of an enterprise.

⁸<http://pubs.opengroup.org/architecture/togaf9-doc/arch/>

Although both languages do not describe an explicit approach for enterprise agility and change management; they include some of the concepts presented in Table 2.5, which can be used to support enterprise agility. For instance, the application layer includes the derived concept of data, which can be used to capture and describe the business entities involved in a change management initiative. Also the business layer includes business process, this can be used to express some logical activities for actualising a change outcome. These concepts may be applicable in representing the structure, goals, information, and behavioural aspect of *Enterprise X*. But ARCHIMATE or ARMOR may not be the best fit for representing the change experienced by *Enterprise X*. This is because, as shown in Table 2.7, none of them clearly provide the other concepts of change, such as change driver, agility enabler, and change indicator, that can be useful in representing enterprise changes.

ZAF: The Zachman framework for enterprise architecture (ZAF) [200] provides an approach for classifying the building blocks of an enterprise. The ZAF's approach provides an ontological matrix, which consist of 6 perspectives and 6 dimensions. The dimensions or columns partitions an enterprise into 6 domains or viewpoints. These include data, function, network, people, time, and motivation. Equally, the perspectives or the rows describe 6 levels of abstraction in which each of these viewpoints can be represented [151]. These are contextual, conceptual, logical, physical, out of context, and functioning.

In relation to enterprise agility, ZAF offers some limitations particularly in capturing, representing, and describing enterprise changes and related concepts. As shown in Table 2.7, even though concepts such as data, actor, and business process activities appear in ZAF; they are not necessarily described in relation to enterprise agility and there appears to be no explanation of how they can be used to support change management initiatives. As per the case study, ZAF can be used to represent the other aspects of *Enterprise X* except its change, the causes of this change, what should be done to adapt this change, and some other change concepts presented in Table 2.5.

i* and BMM: The distributed intentionality (i*) modelling framework [56] and business motivation model [23] (BMM) are representatives of goal oriented modelling (GOM) approaches, which are primarily used to model reasons behind the existence of an enterprise. GOM refers to a family of languages which use essential properties such as goals and actors to describe certain properties of an enterprise [141]. *Goal* is a term that describes the rationale behind the actions of an

enterprise [114, 136]. *Actors* are intentional and active entities of a system or an enterprise [56, 137].

The BMM provides a set of modelling techniques for representing and managing enterprise business plans and motivations [23]. It provides a set of modelling constructs and concepts, which can be used, among other things, to, identify and analyse the objectives of an enterprise using ends and means concepts; define business processes for achieving these objectives; express the factors that influences an enterprise to achieve these objectives; and assess the strength, weakness, opportunities, and threats (SWOT) of enterprise objectives.

The *i** framework provides a set of techniques for modelling the *strategic rationale* or goals of an enterprise, and how enterprise actors inter-depend on each other to achieve strategic goals [56]. According to *i**, an enterprise consist of actors who exist to achieve internal intentions, such as a goal, task, and resource, called *strategic rationale*. But actors are not isolated from each other, rather they inter-depend to achieve enterprise rationale. A model of actors' inter-dependencies is called *strategic dependency*. *i** also provides a set of links that defines the relationship between an actor and its intentions, an actor and another actor, etc.

Certain aspect of *Enterprise X* can be clearly represented using BMM and *i**. For instance, the goals or business objectives of this enterprise can be represented and described using the concepts of goals, means, and ends in both *i** and BMM. Additionally, the strategic dependency concepts in the *i** framework can be applied to capture and express how the actors in *Enterprise X* can interact, as well as the role they play, to achieve the enterprise goals. These concepts relates to some derived concepts in Table 2.5, and thus may be applicable to enterprise agility and change management. However, BMM and *i** are considered to be limited in representing the change aspect of *Enterprise X*. Full details of the concepts of change that are available or implied in BMM and *i** are shown in Table 2.7.

UML: The unified modelling language (UML⁹) is a family of languages generally used to capture, describe, or model the structure and behaviour of Information Systems and Software in enterprises [53]. It is generally considered to provide the basis for most enterprise modelling approaches, and can be divided into structure and behaviour languages/models. The structure languages describes modelling

⁹<http://www.uml.org/>

techniques for representing the static elements of an enterprise such as information (using class and object models), etc. On the other hand, the behaviour languages are set of modelling techniques for representing dynamic operations such as business process (using activity model), and use cases.

Some of the concepts presented in Table 2.5 can be found in UML, even though they may not have been described as the essential concepts for change management and enterprise agility. For instance, the concepts of actions, activity, actor, and class are well described and widely used in representing some aspects of an enterprise. Hence they may be applicable in representing some aspect of *Enterprise X*, for example the class concept can be used to represent enterprise data, while the UML activity diagram can be used to represent enterprise business process activities. However, as shown in Table 2.7, most enterprise agility and change management concepts are either lacking or implied in the UML. Therefore, even though UML can be generally useful in enterprise modelling, it may not be the most appropriate language for capturing, representing, and describing the change and related features of change in *Enterprise X*.

2.8 Conclusion and Chapter Summary

Effective change management and enterprise agility are becoming increasingly important in nowadays dynamic business environment. At the same time, they are becoming more and more difficult to achieve. Information systems and conceptual modelling can be applied to support enterprise agility and change management initiatives. Developing such information systems or conceptual models would require a set of concepts. However, as discussed in Section 2.7.2 and summarised in Table 2.7, most concepts that can be used to support enterprise agility and change management are either lacking or implied in existing enterprise modelling approaches. This could be one of the reasons enterprise agility and effective change management are yet to be actualised.

The aim of this chapter/systematic literature review is to identify the necessary and sufficient set of concepts can be used to construct conceptual models or information systems to support enterprise agility and change management. These concepts can also be used as model constructs to extend subsequent version of the existing enterprise modelling approaches such as TOGAF and ARCHIMATE, so that they can be more suitable to support change management and enterprise agility.

Another important aim of this review is to identify the existing enterprise modelling approaches, and examine how these can be used to support enterprise agility and change management. About six popularly used enterprise modelling approaches were identified and discussed. Then the concepts of enterprise agility and change management identified through the SLR are used as the basis to compare these approaches. This comparison can be used as a guide by IS Practitioners and enterprise change managers to select a suitable enterprise modelling approach to support their change management and enterprise agility. It can also help them to identify the gaps, and limitations of the existing approaches in supporting change management and enterprise agility, and the required concepts that can be used to fill these gaps.

The results of the comparison show that the existing enterprise modelling approaches lack most of the enterprise agility concepts. Hence, in their current state, they may not be the perfect fit for enterprise agility and change management. However, since most enterprise modelling languages provide extension mechanisms, they can be extended with these identified concepts to make them more suitable to support enterprise agility and change management. For instance, a new version of the TOGAF or ZACHMAN framework can be extended to include change driver, change indicator, agility enabler, and other related change concepts necessary to support enterprise agility. This can also be applicable to the UML and other goal oriented approaches. In this way, enterprises would have richer concepts for representing, understanding, and analysing enterprise changes; thereby understanding the implications of change, and specifying what should be done to facilitate and/or achieve enterprise agility.

Chapter 3

Meta-Modelling Techniques And Technologies

3.1 Overview

The framework proposed in this research includes conceptual modelling languages that can be used to facilitate enterprise agility and change management. The essential features of these conceptual modelling languages are described in a meta-model. This Chapter describes the techniques and technologies that would be required to develop the proposed conceptual modelling languages. In addition, since the overall contribution of the research relates to conceptual model and modelling; it would be relevant and useful to provide an overview of conceptual model and modelling, so as to enhance readers understanding.

3.2 Conceptual Model and Modelling

Due to its importance and usefulness in Information Systems (IS) development, the term '*conceptual model*' has attracted a plethora of definitions [88, 110, 112, 131, 132, 157, 158, 169, 185]. Some differences in opinion exist in these definitions, and there seems not to be a generally accepted definition of a conceptual model. However, there appears to be a consensus and agreeing point, in these definitions, that a conceptual model involves the abstraction or representation of an aspect of the physical world also called universe of discourse (UoD) or system under study (SUS) [110, 132, 157, 169, 185]. The aspects of the physical that are represented with a conceptual model include problem, knowledge, information, and solutions to problems [10, 110, 169]. The essence of this representation, as captured succinctly by Mylopoulos [132], is to enhance understanding and communication.

Based on these, a conceptual model can be defined as an abstraction or representation of ideas or relevant concepts in an aspect of the physical world to enhance the communication and understanding of its knowledge (information and data), problems, and the solutions to those problems. The act or process of constructing a conceptual model is called *conceptual modelling*. The description of how a conceptual model can be used to represent an instance of a UoD is provided in a language. Hence, a *conceptual modelling language* is a description of the principles and rules for using a conceptual model model [83, 87, 149, 165, 167]. Given the definition and understanding of these important keywords, *i.e.*, *conceptual model*, *modelling*, and *modelling language*, discussion on how to design and develop conceptual models using meta-modelling techniques can now proceed.

3.3 Meta-modelling Techniques

Essentially, meta-model captures and describes the core features of a modelling language. As shown in Figure 3.1, these include abstract syntax, concrete syntax, and semantics [13, 31, 43, 88, 125, 126]. In most cases, conceptual modelling languages are considered incomplete without these essential features. A brief summary of the relationship between a modelling language and its meta-model is shown in Figure 3.1. Semantics describe the meaning of the language, abstract syntax can be used to describe its structure, while the concrete syntax can describe the notations for representing the language.

Since the development of these essential features forms the technique for describing the proposed modelling languages; it would be important to describe them in a more detail way, in order to enhance readability. The aim of this section is to provide the theoretical background that would enhance the understanding of abstract syntax, concrete syntax, and semantics of a modelling language. Hence, Section 3.3.1 describes and discuss the meaning of an abstract syntax. This is followed by the description of concrete syntax in Section 3.3.2, and then semantics in Section 3.3.3.

3.3.1 Abstract Syntax

Abstract syntax is the structural description of the concepts in a modelling language, the relationships between these concepts, and the procedures for integrating them into a model [1, 43]. In order to describe the concepts and rules that

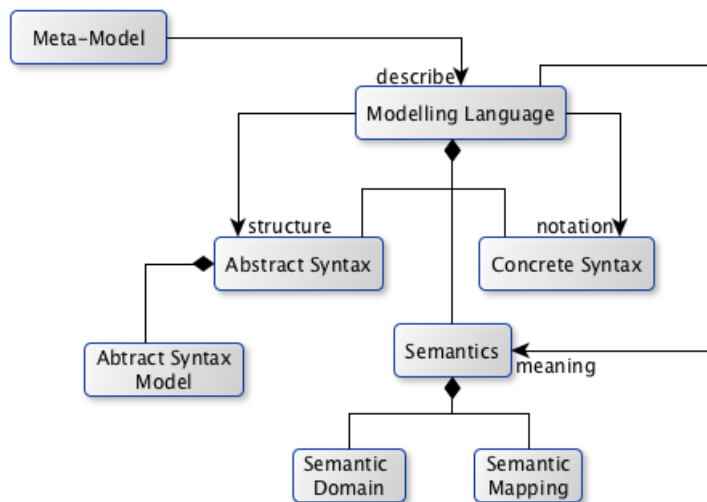


FIGURE 3.1: Essential Features of a Modelling Language

make up the proposed modelling languages abstract syntax techniques are used, see Section 5.1 of Chapter 5.

In an abstract syntax, the concepts of the modelling language are usually referred to as *abstract elements* [118, 130]. Abstract elements are also called *model constructs* or *model elements*. The rules and protocols that can be used to combine model constructs into a model are usually called *validity* or *well-formedness rules* [1, 43]. Therefore an abstract syntax should include validity rules, which help to ensure that all instances or models of a modelling language are consistent, logical, and valid [43].

In addition to validity rules, an abstract syntax should provide an explicit definition of the model constructs alongside the characteristics of each model constructs. Another vital aspect of an abstract syntax is the abstract syntax model, which simply defines the relationships and constraints of model concepts. These aspects, namely, definition of model constructs, abstract syntax model, and validity rules, are usually taken into consideration when designing and developing the abstract syntax of a modelling language. Hence, the processes involved in developing the abstract syntax of a modelling language include identification and definition of model constructs or concepts, building the abstract syntax model, and defining validity rules [1, 43]. Since this research adopts and adapts these processes in developing the abstract syntax of the proposed modelling language; it would be

useful to discuss the activities involved in each process, these discussions are provided below.

Identify and Define Concepts: As mentioned earlier, conceptual modelling languages are usually developed from relevant concepts in a given UoD. Hence, the first activity in conceptual modelling language development is often to identify these concepts and precisely define them. The associated characteristics or attributes of these concepts should also be identified and defined. This research has taken the first step in identifying the useful concepts of enterprise agility and change management through a systematic literature review, see Chapter 2. Further definition of these concepts together with their attributes are defined in Chapter 5 of this thesis. The concepts used as the model constructs of the proposed language can be grouped into two broad categories, these and described below and include:

Primary or Core Concepts: These are the concepts identified from the papers selected for primary studies in the systematic literature review conducted in Chapter 2 of this research. Further definitions of these concepts are provided in Chapter 5.

Auxiliary Concepts: These refer to the concepts that are not originally identified from the the papers selected for primary studies in the SLR. But they are derived to help the primary concepts. For instance, the *goal* concept is not originally identified, from the primary studies of the SLR, as a concept of enterprise agility. However, because goals are expressive and can be decomposed into achievable objectives, they can be used to express the change concept in an explicit and understandable manner. Therefore it has been included as a concept of enterprise agility.

Clark et al [43] provide a general guideline for identifying and defining the concepts included in the abstract syntax of a modelling language. This includes describing other useful information such as the attributes and behaviours of concepts. This research has adapted and adopted this guideline, and will use it as a template for describing the concepts of the proposed modelling languages. Hence, each model construct can include some or all of the following information:

- The Meaning of each concept.
- The Attributes/Property of each concepts, if any.
- The Associations or Relationships between concepts.
- An example of the concepts, if and when necessary.

- Generalisation or Specialisation of concepts, if any.

Develop the Abstract Syntax Model: After identifying and defining model constructs, the next important activity in conceptual modelling language development is to provide a structural representation of these constructs as well as the relationships between them. This is called the abstract syntax model, and should be constructed using existing frameworks and technologies such as UML and/or EMF. For instance, model constructs can be represented as UML *classes/eclasses*, while relationship between these constructs can be described as UML *associations*, and so on. Discussion about these technologies and frameworks are provided in Section 3.4.

Define Rules for Validity: Finally, a set of well defined and precise rules governing the abstract syntax should be provided. It is expected that models or instances constructed using a particular modelling language are consistent, logical, and coherent. To meet this expectation, well-formedness or validity rules should be included in the abstract syntax. These rules can provide the basis for checking that any instance or model constructed using the a conceptual modelling language is valid or well-formed. It can also provide the guidelines and procedures to be followed while creating an instance of the modelling language [43].

Validity rules are often defined informally using natural language. But some executable or behavioural modelling languages define their validity rules formally using formal languages such as first order logic [1, 43, 114, 118, 143]. Since the proposed modelling languages are not intended to be executable or behavioural, in their current state, validity rules are defined informally using natural language, and are provided in Chapter 5 of this thesis.

3.3.2 Concrete Syntax

The *concrete syntax* provides a set of graphical notations, symbols, or shapes for presenting an abstract syntax of a modelling language to the end user [1, 43, 118, 167]. Concrete Syntax technique is important to this Thesis, since defines and provides the notations for describing the proposed modelling languages to users. This is further described in Section 5.3 of Chapter 5.

There are two broad types of concrete syntax, these include textual and diagrammatic concrete syntaxes. A textual concrete syntax provides users with texts, signs,

and symbols for communicating the abstract syntax. A typical example of a textual concrete syntax is the extended Backus-Naur Form (BNF), which has long been used as a basis for describing the concrete syntax of most executable languages. Normally, the textual concrete syntax are used for programming languages, and in some cases executable modelling language. On the other hand, a diagrammatic concrete syntax uses graphical notations such as figures, shapes and images to present the concrete syntax of a modelling language to end users [1, 43]. Diagrammatic syntaxes are considered to be the most popular and have been used in most conceptual modelling languages such as UML and BPMN.

Even though the best type of concrete syntax to adopt for a modelling language is still contentious; the relative advantages and disadvantages, of each type of concrete syntax over the other, have been considered in literature [68, 71, 85, 104]. For instance Gronninger et al [71] argue that textual concrete syntax are platform and tool independent, since text editing do not require specific platform or environment. In addition, they believe that it is easier to develop parsers and code generators for textual concrete syntax than it is for diagrammatic concrete syntax.

Similarly, diagrammatic concrete syntaxes have some advantages over textual concrete syntaxes. Since diagrammatic concrete syntaxes normally use conventional graphical notations such as rectangles, they are easier to learn, use, and understand when compared with textual languages [71, 104]. Therefore they can accommodate a wide range of users, and can be less difficult to use by inexperienced modellers. Furthermore, Heidenreich et al [85] observe that certain information, such as relationships of a modelling language can be visualised and interpreted better in diagrammatic concrete syntax than in textual concrete syntax.

The decision of the type of concrete syntax to adopt for a given modelling language is usually made by the language designer. This is usually based on certain considerations such as the the target audience, and the intended use of the language. Most executable and behavioural modelling languages such as XOCL usually have textual concrete syntax. But non-behavioural modelling languages such as UML, and BPMN have diagrammatic concrete syntax. As mentioned earlier, in their current states, the modelling languages proposed in this research are not intended to be behavioural or executable. Instead, they are intended to provide structural representations of enterprises and the changes enterprises experience. It is also intended that these languages would accommodate a vast majority of users, including inexperienced modellers and users. Based on these, graphical notations, instead of text, will be used for the concrete syntax of the proposed modelling language.

The diagrammatic concrete syntax of a modelling language can be constructed using the two stage process described in [43]. The first stage involves describing a means to parse or interpret the concrete syntax so as to ensure that it is meaningful, useful, and valid. This can be achieved by constructing a *model of diagram*, at a meaningful level of abstraction, which can be used to describe any graphical notation that forms part of the concrete syntax. Sometimes, this model of diagram is called concrete syntax meta-model as in the case of [5]. A model of diagram can be constructed using the graphical primitive and structured data types provided by the OMG's diagram definition (DD) [142].

The DD architecture [142] provides a generic but useful (graphical) abstraction such as figures, nodes, and links, that can be adapted and applied to define the diagrammatic concrete syntax of a modelling language. Further descriptions of DD are beyond the scope of this research, but can be found in OMG specification for diagram definition [142]. In the second stage of constructing diagrammatic concrete syntax, the model of diagram or concrete syntax meta-model is then used to build the abstract syntax [43]. These two stages are followed in Chapter 5 to describe the diagrammatic concrete syntax of the proposed modelling language.

3.3.3 Semantics

The semantics of a modelling language defines its meaning, as well as the meaning of any syntactically correct expression or model constructed with the modelling language [82, 83, 167]. Semantics help to increase the descriptive ability of a modelling language by conveying meaning to the models of that language [31]. For instance, in Section 5.4 of Chapter 5 semantic techniques are used to convey the meaning of the Change Modelling Language. As summarised in Figure 3.1, the semantic definition of a modelling language usually involves two main parts, which include a semantic domain and a semantic mapping [82, 83].

The *semantic domain*, also called semantic unit, refers to an existing modelling language or framework with a well defined semantics or meaning. For instance, languages such as the UML and abstract state machines (ASM) are generally known to have precise semantics and thus can be used as the semantic domain for modelling languages [31, 36, 83]. The semantic mapping relates each concept in the abstract syntax of a modelling language to its equivalent in the selected semantic domain; so that the concepts in the modelling language can be interpreted using a well defined and known concepts in the semantic domain [31, 43, 82, 83].

The semantic domain and semantic mapping of a modelling language can be described in various ways. For instance, they can be defined formally using mathematical formulae, symbolic logic, or any other language. They can also be described informally using natural languages [82, 83]. A known advantage of informal over formal description of semantics is that informal descriptions, particularly in natural language, are always easier to understand, and thus can be easily used by a wide range of audience. Therefore the semantics (domain and mapping) of the proposed modelling languages are described informally in Chapter 5 of this thesis.

3.4 Meta-Modelling Technologies

In addition to describing the key features of a modelling language, a meta-model should also be part of a meta-model architecture [43]. The *meta-model architecture* (MMA) provides a single framework for describing all modelling languages and meta-models in a unified way. By describing all modelling languages in unified way, it can be possible to compare, and relate one modelling language with another [61]. For instance, the change modelling language proposed in this research can be compared or related with an enterprise modelling language. In this way the new domain elements required to adapt changes can be derived. In addition, describing modelling languages in a unified way can support model driven techniques such as model to model transformation, model to model comparison, and model to model integration [43, 47, 61].

Several approaches to meta-model architecture can be found in literature [43, 102, 130]. Among these competing MMA, the meta-object facility (MOF) provide a set of rich but easy to understand and use concepts as well as modelling technologies that can be used to describe the meta-model of the proposed modelling languages. For instance, concepts such as classes, association, and multiplicities can be used to describe the model constructs/concepts of the proposed modelling languages and the relationships between them. Therefore, MOF is adopted as the meta-model architecture for describing the meta-model of the proposed modelling languages. A description of the UML, and its concepts that are relevant and used in developing the proposed languages are described in the section below.

3.4.1 Relevant MOF Concepts

The meta-object facility (MOF¹) is an Object Management Group's (OMG) specification that can be used as meta-model architecture for describing other modelling languages. In addition, it provides a framework for classifying concepts of a modelling language [53, 168]. The MOF concepts that are used to describe the model constructs of the change modelling language proposed in this research include class, attributes, association, composition, generalization, and multiplicity. These become very useful in Section 5.2.1 and Section 5.2.2 of Chapter 5.

This section provides a brief description of MOF concepts, which are relevant to the conceptual modelling languages proposed in this research. However, detailed descriptions and discussions of MOF concepts can be found in [6, 18, 58, 156].

Class: A *class* is a collection of objects with similar characteristics, property, and behaviour. *Objects* are things or entities important in a given context [156]. As illustrated in Figure 3.2, some examples of a class are Person and Account. A class is usually represented using a rectangle with three parts. The first part contains the name of the class. It is usually compulsory for classes to have names. The second part defines the attributes of the class, while the third part describes the common behaviour of the class objects [18]. An *abstract class* is not usually implemented, instead it acts as place holders for other classes [58]. In most information systems modelling, such as database modelling, classes can be called *entities*.

Attribute: This refers to any common property or characteristics shared by all objects in a class [156]. For instance, the attributes of the 'Account' class in Figure 3.2, include *aName*, *aNumber*, and *sortcode*.

Association: An *association* describes the name and the type of relationship that exist between classes [58]. For instance, in Figure 3.2, the 'Person' and 'Account' classes can be associated with *has*. In other words, Person has Account. An association can be used to show that a class, called *sub-class*, is a type of or inherits some property of another class called *super-class*. In this case, the association is called *generalisation*. An example of this can be seen in Figure 3.2, where 'Savings' and 'Current' are subclasses of the 'Account' class.

¹<http://www.omg.org/mof/>

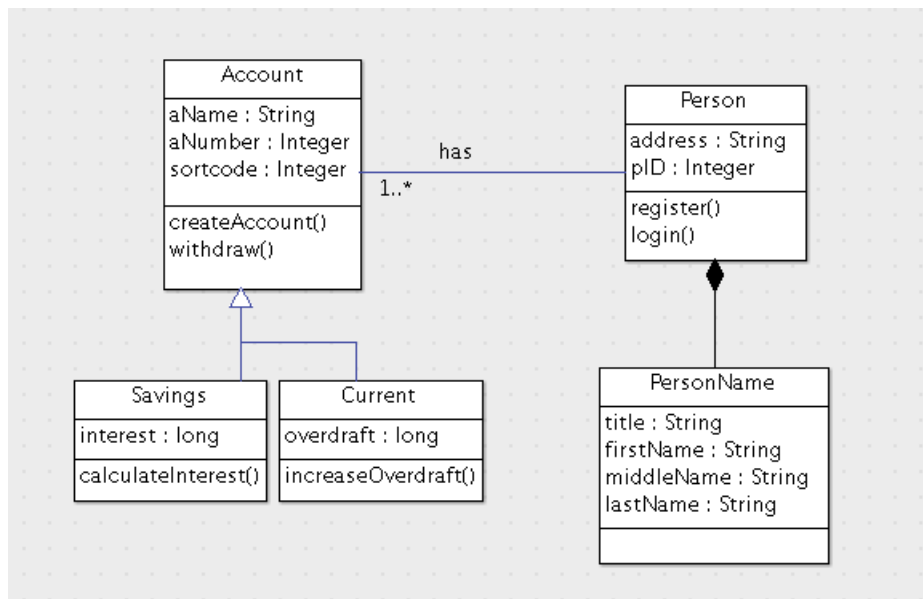


FIGURE 3.2: Illustration of UML Concepts

Equally, an association can be used to show that one or more classes are part of another class called a *whole class*, in which case the association is called *aggregation*. However, if the parts cannot exist independently without the whole class, the association will be called *composition*. As shown in Figure 3.2, the 'PersonName' class cannot exist without the 'Person' class, hence the former is a composite part of the later.

Association can be represented as a plain straight line or a straight line with an arrow head. Generalisation is widely known to be represented with a solid line with hollow arrow head. Aggregation is usually drawn as a straight line with an unshaded diamond end. Composition is similar to aggregation but has a shaded diamond end.

Multiplicity: The term *multiplicity* is used to show the number of objects that can participate in a relationship or association and whether such participation is compulsory or optional [58]. For instance, in the relationship between customer and product, multiplicity can be used to define the number of customers that can buy products, and the number of products that a customer can buy.

Multiplicity is usually written, at the end of the association line, as numbers separated with dots. Sometime an asterisk can be used to show that more than one object can participate in a relationship. For instance the multiplicity, $1..*$, in Figure 3.2 shows that the lower bound is 1 (compulsory association), and upper bound is asterisks (signifying many). This means that a Person must have at least one account.

3.4.2 Eclipse Modelling Framework

Eclipse modelling framework (EMF) has been selected and used as the technology and framework for developing the abstract syntax models of the proposed modelling languages. This is because the EMF², an integrated modelling and model driven development framework, provides a meta-model, model construction technologies, and tools that can be used to develop abstract syntax models of modelling languages [64, 167].

Specifically, the EMF is used as an editor to produce the abstract syntax model of the change modelling language, as shown in Figure 5.1, see Section 5.2 of Chapter 5. Similarly, the abstract syntax model of the concise enterprise modelling language is edited using EMF. This is shown in Figure 6.1 of Section 6.3 in Chapter 6.

The EMF core meta-model or *ecore* is described using the key concepts of MOF defined in Section 3.4.1. Hence these concepts will not be defined again in this sections. Note that in EMF, these concepts are prefixed with a small letter 'e' an acronym for ecore. For instance, MOF Class and Attributes are respectively called eClass and eAttribute in EMF, and so on.

3.5 Chapter Summary

In order to have a better understanding the proposed modelling languages, readers or users need to have knowledge meta-modelling techniques, MOF technologies, and other relevant frameworks. This is because meta-modelling techniques, such as identifying the semantic domain and performing semantic mapping, are the necessary and fundamental techniques for modelling language development. In addition, modelling languages are expected to build from existing meta-model architectures such as MOF and technologies such as EMF. These are relevant to

² <https://eclipse.org/modeling/emf/>

this research and should be described and discussed in, at least, a little more detail. Therefore, this chapter aims at providing the discussion of the techniques and technologies that supports the development of the proposed languages.

Chapter 4

The Proposed Framework And Research Methodology

4.1 Chapter Overview

The limitations or gaps in the existing enterprise modelling approaches, with respect to their ability to support enterprise agility and change management, have been identified in Chapter 2. Their main gap is the inability to represent enterprise changes, and derive the enterprise model elements required to adapt the changes. A framework for filling this gap has been briefly introduced in Section 1.4. Further description of this framework is provided in this Chapter. In addition, this Chapter discusses how the design science research method is applied to design the proposed modelling languages.

4.2 The Proposed Framework

This research proposes a framework for enterprise agility and change management. As shown in Figure 4.1, this framework considers change as an integral part of an enterprise, rather than treating it as an isolated entity happening outside an enterprise. In other words, the enterprise and its problem *i.e.*, the changes it faces, are considered to be parts of a single interrelated system. This idea is borrowed from systems thinking [11, 20, 159, 163, 188]. By considering the enterprise and its changes as a single interacting system, it can be easier to relate a change to the enterprise it affects, and derive a set of new domain elements required to adapt the given change.

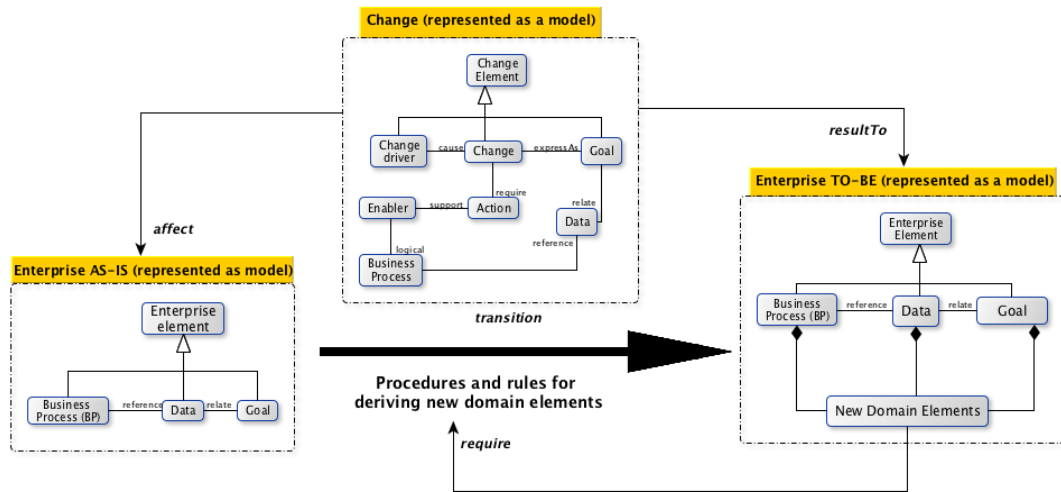


FIGURE 4.1: The Proposed Framework for Enterprise Agility

The proposed framework, as shown in Figure 4.1, can be viewed as a system consisting of four integral and interrelated parts. The first part is the current or *as-is* state of the enterprise, which is usually affected by changes. The second part of this framework is the change that can affect the enterprise. When an enterprise is affected by a change, the enterprise is expected to regain its equilibrium by transitioning to a target or *to-be* state. This state is known as the target or *to-be* state of the enterprise. It is also the third part of the proposed framework.

Notice that the state of the enterprise after a change, *i.e.*, target or *to-be* state, include a set of new domain elements that are not originally in the current state of the enterprise. As discussed in Chapter 1, these domain elements can be goal elements, and/or business process activities, and/or data entities. In any case, they are part of, and should be added to the target state of the enterprise, and are also required to adapt the change. An enterprise should derive these domain elements to be able to adapt a given change. Hence, the fourth part of this framework includes procedures and rules for deriving these new domain elements.

Figure 4.1 suggests that the proposed framework is model driven. This implies that in order to apply this framework to an enterprise, the change, alongside the current and target states of the enterprise should all be represented as models. In this way, it can be easier to understand, visualise, and analyse how changes interact and interrelate with enterprises, as well as derive the set of new enterprise domain elements required to adapt changes. To achieve this, the proposed

framework provides a conceptual modelling language that can be used to define, represent or model an enterprise change. In addition, it provides another conceptual modelling language to define, represent or model both the current and target state of the enterprise.

The change modelling language is made up concepts of enterprise agility and changes identified through the systematic literature review in Chapter 2. Equally the enterprise modelling language is made up of concepts gathered from existing enterprise modelling approaches such as TOGAF and ZACHMAN. This proposed enterprise modelling language limits the representation of an enterprise to its goals, business processes, and data. Every modelling language should have a meta-model, which captures the essential features of that modelling language [13, 31, 43, 88, 125, 126]. The design and development of any meta-model should be based on existing techniques and technologies. These are briefly summarised in Section 1.5.4 of Chapter 1, but full and detailed descriptions of these technologies are provided in Chapter 3.

4.3 Research Methodology

As mentioned earlier, the purpose of this research is to contribute a framework that can be used to facilitate or even achieve enterprise agility and effective change management. As discussed in Section 4.2 and shown in Figure 4.1, this framework relies on conceptual modelling languages. Normally conceptual modelling languages are developed by designing model constructs, from the concepts identified in a given aspect of the physical world. The model constructs of a modelling language are types of information systems (IS) artefacts [90]. But the design of IS artefacts is the primary concern of the design science research (DSR) or information systems research framework proposed by Hevner et al [4, 89, 90]. Based on these, this research adopts and adapts the DSR as its research methodology.

The DSR includes two core parts. The first part is the information systems research (ISR) framework, which describes how to conduct research in the information systems discipline. In order to apply the ISR framework adequately, and realise its benefits, certain guidelines should be followed. These guidelines form the second part of the DSR. The sections that follows provide a brief description of these two core part, and in addition discusses this research applies DSR methodology. Where necessary, specific sections and chapters that relate to each part are mentioned in the discussion. Section 4.3.1 discusses the information systems research framework, while Section 4.3.2 discusses the guidelines for using this framework.

4.3.1 Design Science Research

Design science research (DSR) is a conceptual framework, proposed by Hevner et al [90], for conducting research in the Information Systems discipline. As shown in Figure 4.2, this framework provides three important dimensions through which an information systems research can be realised. The first dimension is the environment, which helps to capture and define the business needs, or the relevance, for conducting an information systems research or study. In other words, the environment examines an enterprise, its people and technology, so as to define and understand a problem domain, together with the opportunities that can be derived from solving such problems. In relation to this research, the 'environment dimension' can be captured in terms of modern day enterprise and the changes they experience. These have already been discussed as problems and motivations for this research in Chapter 1.

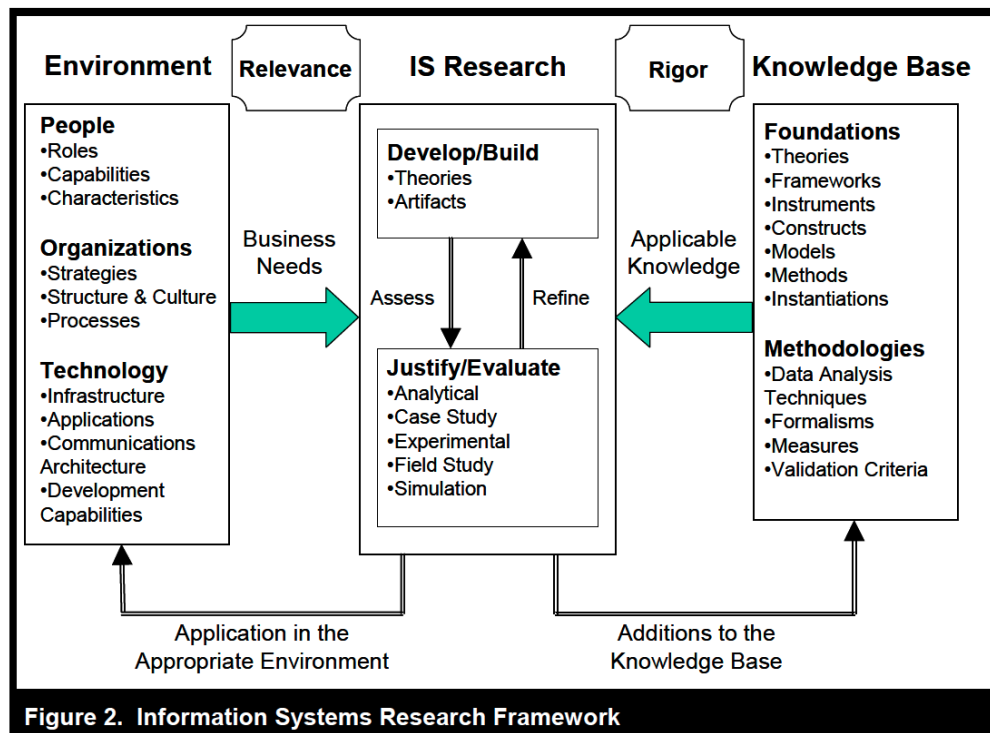


FIGURE 4.2: Design Science Research (This Figure is taken from [90])

In the second dimension, *i.e.*, the information systems research, theories, and artefacts are developed to contribute adequate solution(s) to the problem identified in

the first dimension. These artefacts are then evaluated using any approach such as case study or experiment. The theories and hypothesis/ framework for solving enterprise agility and change management problems have been described in Section 4.2. In addition, Chapter 2 presents a set of concepts, identified through a systematic review, that can be used to construct the artefacts required to solve enterprise agility and change management problems. These concepts are further developed into design artefacts, *i.e.* model constructs and modelling languages, in Chapter 5 and Chapter 6. For the purpose of evaluation, an industry case study has been obtained and are used, in Chapter 7, to demonstrate utility or justify the design artefacts contributed by this research.

The last dimension is the knowledge base which provides the foundations and methodologies for conducting an information systems research. As shown in Figure 4.2, these can include existing technologies, frameworks, models, and techniques. The knowledge of existing technologies and frameworks have also been applied to conduct this research. For instance the UML, an existing modelling technology, is used as a meta-modelling architecture for describing the modelling languages proposed in this research. Similarly, the EMF has been used as the modelling tool for constructing the abstract syntax models of the proposed modelling languages. Summarised discussion of these technologies and frameworks have been discussed in Section 1.5.4, and are further discussed and described in Chapter 3.

4.3.2 Guidelines for Information Systems Research

In the second part of the design science research, Hevner et al [90] provide 7 guidelines for conducting information systems research. A summary of these guidelines is presented in Figure 4.3 below. The aim of this section is to briefly describe each guideline, and explain how it is applied in conducting this research. As mentioned in Chapter 1, the process and activities involved in conducting this research is divided into 6 key steps. Figure 4.4 presents a summary of these steps and shows the specific DSR guideline (GL) each step conforms to.

The DSR guideline 1 or *design as an artefact* states that an information systems research must contribute or produce innovative and viable artefacts. According to Hevner et al [90], an artefact can be a model, or an abstraction that can be used to instantiate a model. The modelling languages and framework contributed by this research are abstractions that can be used to represent enterprise changes and enterprise models. Therefore they are design artefacts. As shown in Figure 4.4, these artefacts are developed in Step 4 of this research. Hence Step 4 of this research

Table 1. Design-Science Research Guidelines	
Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

FIGURE 4.3: Guidelines for Information Systems Research (This Figure is taken from [90])

process conforms to DSR guideline 1. The modelling language for representation enterprise changes is developed in Chapter 5 of this research, while the modelling language that can be used to represent enterprises is developed and presented in Chapter 6.

The *problem relevance* or DSR guideline 2 states that purpose and aim of conducting an information systems research should be to solve critical business problems using technology. Enterprise change drivers and changes are known to have dire consequences, such as financial loss and bankruptcy, in modern day enterprises. At the same time, the ability to respond to these changes, otherwise called enterprise agility, is a core enterprise for enterprise survival and viability. Yet it is challenging, difficult to achieve and a major concern for enterprise executives. Therefore, enterprise agility, and change management are relevant problem areas in enterprises. The first activity in this research, as shown in Figure 4.4 is to identify and define the problems associated with enterprise agility and change management. Hence Step 1 of this research conforms to DSR guideline 2. These problems have been described in Chapter 1 of this thesis.

The third DSR guideline or *design evaluation* emphasizes the importance of demonstrating the utility of design artefacts. According to Hevner et al [90], one good method of demonstrating the utility of Information Systems Research contributions or design artefacts is to use them to represent an aspect of the real which can be captured as a case study or scenario. Therefore to demonstrate utility, the contributions of this research is used to model a case study. This case study, as described in Section 2.6 of Chapter 2, is a real industry case study obtained from an existing industry, and thus an aspect of the real world. In Chapter 7, this case study is applied to test the utility of the design artefacts contributed by this research. As shown in Figure 4.4, steps 3 and 5 of this research process conforms to DSR guideline 3.

The *research contribution* or guideline 4 states that design science research should make clear contributions to the information systems community. It further discusses three key areas where such contributions can be made. One of these areas is in design foundations, which can include constructs, models, or abstractions for representing models. Steps 2, 4, and 5 conform to this guideline. The contributions of this research are identified in Step 2 and discussed in Chapter 1. These contributions are further developed and evaluated in Steps 4 and 5 respectively, which are discussed in Chapter 5, Chapter 6, and Chapter 7 of this research.

The DSR guideline 5 or *research rigour* opines that rigorous methods should be used in both the construction and evaluation of research contributions or design artefacts. One way to demonstrate rigour, according this guideline, is to use the

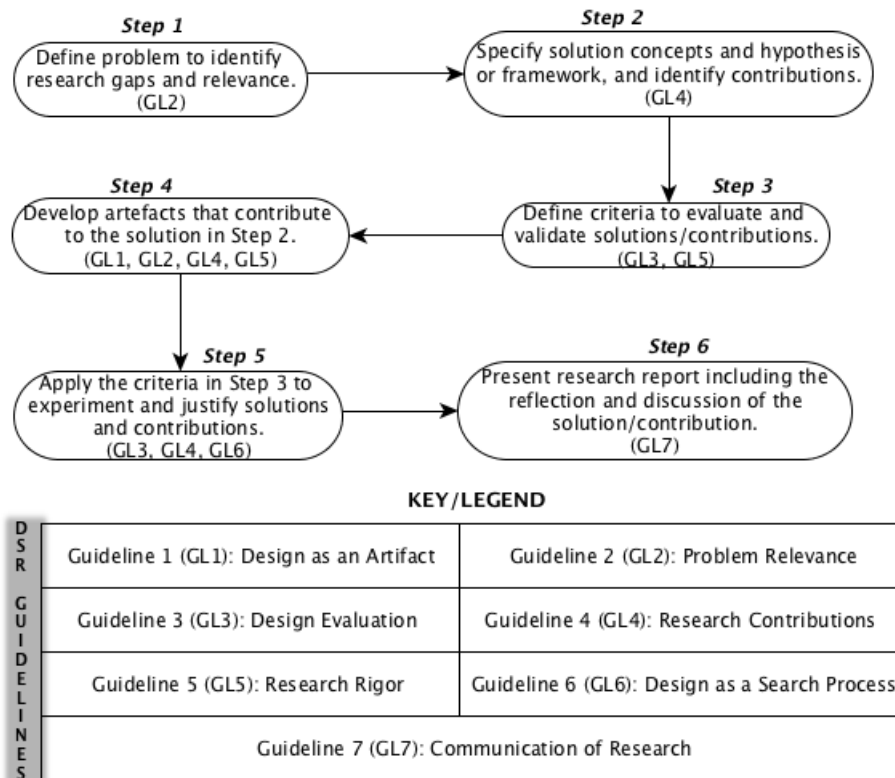


FIGURE 4.4: Research Process and DSR Guidelines

knowledge base effectively. In other words to effectively use existing methodologies, technologies, and framework. This research demonstrate rigour by using existing modelling technologies, and frameworks such as the UML, and EMF, as the basis for constructing its the design artefacts. These technologies, and frameworks have been briefly introduced in Chapter 1 and are further discussed in Chapter 3 of this thesis. In addition, this research used existing design principles to motivate the design of its contributions, *i.e.*, the modelling languages. These principles are discussed in Section 4.4. Another way this research demonstrated rigour is to use existing methods of evaluation, particularly case study, to evaluate its design artefacts. The evaluation criteria discussed in Chapter 1 are applied, in Chapter 7, to evaluate the design artefacts of this research. As shown in Figure 4.4, this guideline also conforms Steps 3 and 4 of this research process.

The DSR guideline 6 or *design as a search process* suggests that the design science research should be an iterative process. In other words, the solutions or artefacts generated from DSR should be experimented on some requirements. Often, these requirements are captured in form of a case study or scenario. The results and feedbacks from such experiments should be used to improve the solutions or design artefacts. This process should be iterative, *i.e.*, repeated until the best possible artefacts or solutions are developed. Earlier versions of the design artefacts of this research, *i.e.*, modelling languages, have been developed previously by the Author and experimented on the requirements captured in case studies, see [138–140]. The results and feedback of these experiments have been presented at various international conferences and published [138–140], as well as used to develop the current version of the solution as reported in this research. Therefore, this research conforms to DSR guideline 6, which relates to Step 5 of the research process shown in Figure 4.4.

The last DSR guideline or *communication of research* relates to the effective presentation of design artefacts to both technology and management oriented audiences. To apply this guideline, this thesis provide both technical and non-technical descriptions. The non-technical descriptions can be applicable to any management audience. For instance, Chapter 1, Chapter 2 and Chapter 4 provides a business perspective to this research. These include problem definition, enterprise agility challenges and prospects, as well as domain analysis through the SLR. Details that can be applicable to technology audience are presented in Chapters 3, 5, 6, and 7. These include the development and description of abstract syntax, concrete syntax, semantics, and syntax models of the proposed modelling languages. As shown in Figure 4.4, Step 6 of this research generally conforms to guideline 7.

4.4 Motivating Principles

As mentioned earlier, Hevner et al [90] suggest that information systems research artefacts or contributions must be developed through the application of rigorous methods. In order to demonstrate the application of such rigour, the development of research artefacts should be motivated by existing principles, theories, and techniques [90]. Therefore, the development of the artefacts or contributions of this research, *i.e.*, the conceptual modelling languages and rules, are motivated by relevant existing principles for designing conceptual modelling languages. Design principles are proposed in [92, 93, 104, 109, 149] for the purpose of providing a set of guidelines that can be used to develop quality and usable conceptual modelling languages. The aim of this section is to discuss the design principles that motivate the development of the proposed conceptual languages. These principles are summarised in Table 4.1, and include completeness, consistency, clarity, correctness, understandability, and flexibility.

TABLE 4.1: Summary of Motivating Language Design Principles

Design Principle	Summary	Reference
Completeness	The modelling languages must contain sufficient constructs to represent an aspect of the physical world in a given universe of discourse.	[111, 124, 128, 173]
Uniqueness or Consistency	Each model construct should represent only one concept of the universe of discourse	[109, 120, 129, 149, 173]
Clarity	The model constructs must be defined explicitly and clearly in such a manner that they can be used easily.	[38, 120, 129, 173]
Correctness	Modelling Languages must conform to existing methods and rules of conceptual modelling	[38, 97, 120, 128]
Simplicity or Understandability	Modelling Languages should be designed in such a simple manner that users can easily understand and use them.	[38, 104, 109, 128, 129, 149]

Completeness Principle: The *completeness* design principles [111, 124, 128, 173] suggests that a modelling language must contain sufficient set of model constructs that can be used to represent the universe of discourse (UoD) or system under study (SUS). Often, but not necessarily, a universe of discourse is captured and described using a case study. In relation to this research, the UoD is an enterprise and the changes it can face, these have been described in a case study obtained from industry collaborators. To facilitate sufficiency of model constructs,

a systematic literature review is conducted, as reported in Chapter 2 of this thesis, to identify a set of concepts sufficient to represent an enterprise and its changes. These concepts are then used as the basis to develop model constructs of the proposed modelling languages. Since these concepts are sufficient to represent the enterprise capture in the case study together with its changes, as reported in Chapter 7, it is therefore considered to satisfy the completeness principle.

Uniqueness and Clarity Principles: The *uniqueness or consistency* principle [109, 120, 129, 149, 173] recommends that model constructs of a modelling language must not have conflicting or ambiguous meaning, and must provide a unique representation of concepts. In other words, each model construct must only be used to represent only one concept of the universe of discourse. Equally, the *clarity* principle [38, 120, 129, 173] ensures that each model construct must be clearly defined and in an understandable format. These can reduce necessary duplication and redundancy of model constructs as well as confusion of concepts. To facilitate and adhere to uniqueness and clarity, explicit definition is assigned to each concept identified in Chapter 2, and each definition is further clarified with the use of a motivating example. These concepts are further defined as model constructs in Chapter 5. In addition, Table 2.5 of Chapter 2 shows various terms or names that can be used to associate each selected concept. These are then grouped into one model construct. For instance, *change drivers*, *agility drivers*, *drivers of change*, and *change factors* are all grouped and expressed as with a single concept called *change driver*.

Correctness and Simplicity Principle: The *correctness* principle [38, 97, 120, 128] recommends that modelling languages must conform to a standard technique for developing conceptual model. Accordingly, the modelling languages proposed in this research are developed using well known and current de-facto standard such as the UML. For instance UML modelling techniques such as associations, and multiplicities, are used to define the proposed modelling languages. Equally, the *simplicity or understandability* principle [38, 104, 109, 128, 129, 149] recommends that a modelling language should be designed in a simple and understandable manner. A simple modelling language should avoid the use of too many unnecessary and cumbersome model constructs or concepts. In this way it can be easy to learn, understand and use. To adhere to this principle, a systematic approach, see Chapter 2, is applied in identifying the concepts used in developing the modelling language. This is to ensure that only necessary and sufficient concepts are used for language development.

4.5 Chapter Summary

In order to contribute and proffer viable solutions to enterprise agility problems, the discussion of the proposed solution framework or hypothesis, the methodology for realising this solution, and motivating principles for developing the solution framework are necessary. The solution framework, discussed in Section 4.2, provides a detailed explanation of the framework required to prove the research hypothesis, and hence contribute to solving enterprise agility problems. This framework contributes design artefacts, *i.e.*, modelling languages and procedures, that can be used to support or even achieve enterprise agility and change management. To ensure that these artefacts are efficacious and of good quality, a relevant and appropriate research methodology should be used. The design science research (DSR), discussed in Section 4.3.1 provides a framework and set of guidelines for ensuring that information systems research products, artefacts, and/or contributions are of good quality. Therefore DSR is considered appropriate and used as the methodology for this research.

Apart from the use of adequate research methodology, there are other ways to ensure efficacy and good quality of the modelling languages proposed in this research. One of such ways is to use existing and well established principles of developing conceptual modelling languages. Hence, the development of the proposed modelling languages is motivated by existing principles of designing conceptual modelling languages, such as completeness and simplicity. These principles are also discussed in this chapter, particularly in Section 4.4. In addition, Section 1.5.4 discussed the technologies that are relevant and can support the development of the proposed solution framework.

Chapter 5

The Change Modelling Language

One of the essential contributions of this research is a *change modelling language*, *i.e.*, a conceptual modelling language for describing enterprise change and its features. Relevant techniques and technologies for developing conceptual modelling languages have been discussed in Chapter 3. These techniques and technologies are applied in this chapter to design and develop the proposed modelling language. The abstract syntax of the proposed modelling language is designed and presented in Section 5.1. Section 5.2 presents the abstract syntax model of the proposed language. This is followed by the Section 5.3, where the concrete syntax is designed and presented. Finally the semantics of the proposed language is presented in Section 3.3.3.

5.1 Language Abstract Syntax

Among other imperatives, a conceptual modelling language should aim to support the representation and communication of knowledge, problems, together with other relevant information in a given universe of discourse. The abstract syntax facilitates this aim by providing the definition of the concepts used in communicating knowledge, problems, and other relevant information. In addition, it defines the attributes, and relationships between concepts, as well as the rules for integrating those concepts into a model. The aim of this Section is to provide the abstract syntax of the proposed modelling language. It starts by identifying and defining the concepts or model constructs of the language, including their attributes. Subsequently, a description of the relationships between the concepts is provided. Then the abstract syntax model is presented. This is followed by a set of validity rules for the proposed language.

5.2 Abstract Syntax Model

The abstract syntax model is shown in Figure 5.1. An enlarged version of this model is provided in Appendix A. The Eclipse Modelling Framework (EMF) discussed in Section 3.4.2 of Chapter 3 is used as a construction tool for the abstract syntax model. The language contains two abstract classes, namely *Change Model Element* and *Data*. The Change Model Element is an abstract superclass, similar to UML named element, for the core model concepts. Abstract classes¹ are more or less 'syntactic sugars' used to make a language better organised and readable. They are not usually represented in a model and thus may not have concrete notations. Data is used as a *namespace*² for enterprise entities.

5.2.1 Concept Identification

The concepts used as the model constructs in the proposed modelling language were identified through a systematic literature review. This is to ensure a wider coverage and inclusions of sufficient as well as necessary set of concepts. About 347 peer reviewed publications were downloaded from computer science and information systems databases. These include publications from experienced academics and practitioners in the industry. The publications were analysed, using systematic techniques, to derive the concepts used for the proposed language. Details about the literature search and review have been reported in Chapter 2.

As shown in Table 5.1, there are two categories of concepts used in the proposed modelling language. These include the core concepts and auxiliary concepts, auxiliary concepts can also called derived concepts. As mentioned in Chapter 2, the core concepts are identified from the papers included in the primary studies of the systematic literature review. They are *change*, *change driver*, *action*, *agility enabler*, *change indicator*, and *impact*. The auxiliary concepts are concepts from existing enterprise modelling approaches that can be used to describe, express, or support the core concepts. These include *goal*, *data*, *business process*, and *agent*. For instance, goal is derived from goal oriented modelling and can be used to decompose a change into achievable objectives. Each of these concepts and their attributes are described in Section 5.2.2.

¹<http://www.uml-diagrams.org/uml-core.html>

²<http://www.uml-diagrams.org/namespace.html>

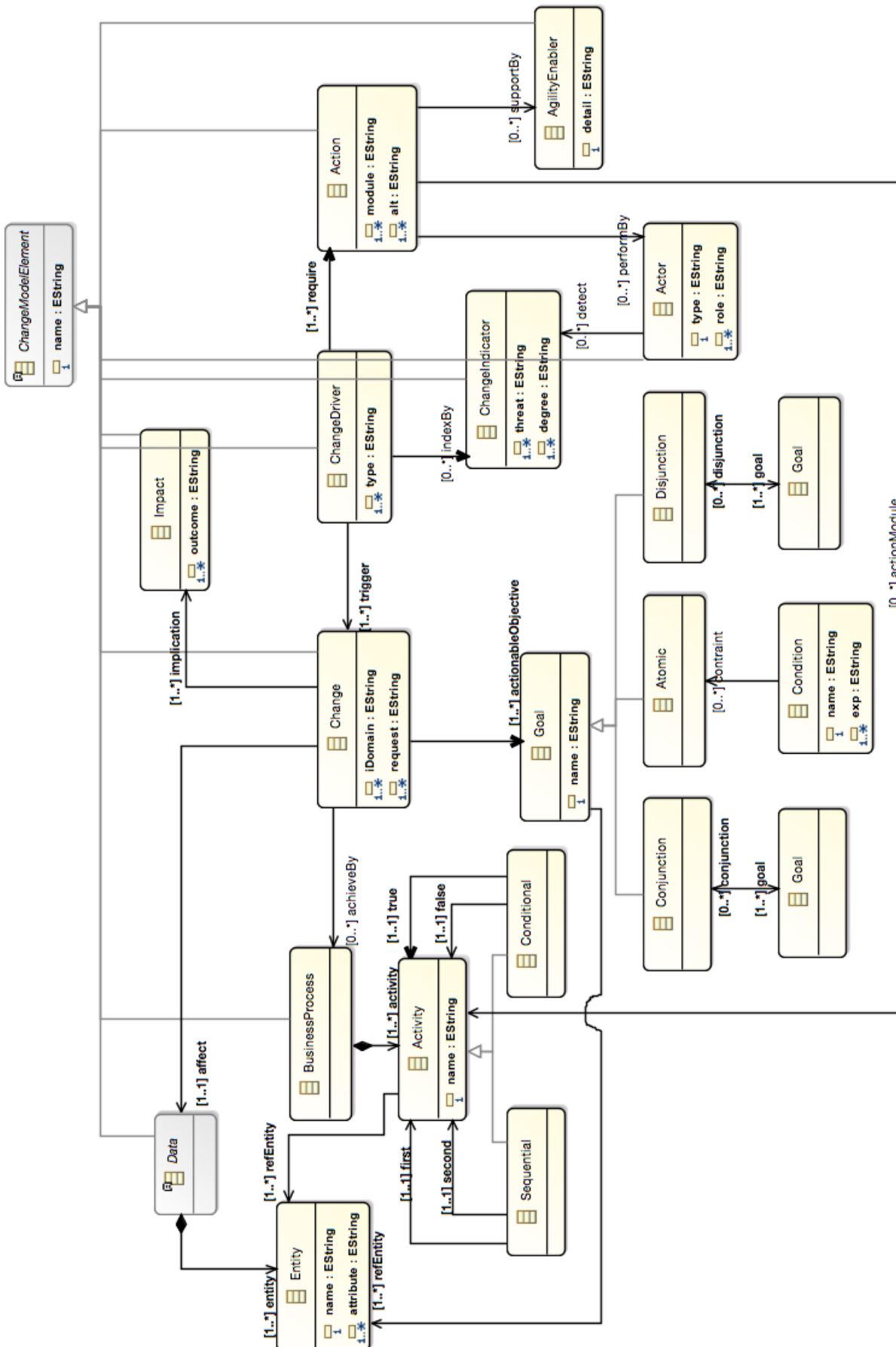


FIGURE 5.1: Abstract Syntax Model of the Proposed Modelling Language

TABLE 5.1: Concepts of the Proposed Modelling Language

Core Concepts	Auxiliary Concepts
Change	Goal
Change Driver	Business Process
Action	Data
Agility Enabler	Actor
Change Indicator	Change Impact

5.2.2 Concept Definition

This section defines the concepts of the change modelling language, the attribute(s) of each concept, and how the concepts relate to each other. The concepts, their attributes and relationships are summarised in Table 5.2.

One good method of defining the abstract syntax of a conceptual modelling language, as proposed and used by the Object Management Group (OMG) in UML 2.0³ and BPMN 2.0⁴ specifications, is by listing and defining the concepts of the language as well as their attributes.

This research adopts and adapts this method in defining the concepts of the proposed abstract syntax. Hence the definition of concepts in Table 5.2 are structured as shown in Figure 5.2.

As shown in Figure 5.2, each concept must have a *definition*, which states its meaning. Where necessary, *discussion* can be used to provide further explanation to the definition of a concept. Each concept should relate to at least one other concept. The property or characteristics features of a concept such as its name are described as *attributes*. As discussed in Section 3.4.1 of Chapter 1, the three types of relationship that exist between the concepts in the proposed change modelling language include:

Association: Used to describe how a concept relates to another concept.

Generalization: Used to describe the 'types' in a concept.

Composition: Used to describe 'part-whole' relationship between concepts.

³<http://www.omg.org/spec/UML/2.0/Infrastructure/PDF>

⁴<http://www.omg.org/spec/BPMN/2.0.2/>

TABLE 5.2: Summary of Concepts in the Change Language

Concept	Attribute(s)	Relationship(s)
Change	request, and iDomain	The change concept directly relates with the following concepts: <ul style="list-style-type: none"> • BusinessProcess: Change can be <i>achievedBy</i> BusinessProcess. • Goal: Change can be decomposed into <i>actionableObjectives</i> using Goal. • Impact: Change <i>implication</i> to an enterprise is described as Impact. • Data: Changes can affect enterprise information captured as Data.
Goal	name, and condition	The Goal concept relates with: <ul style="list-style-type: none"> • Entity: Goal references entity (<i>refEntity</i>). • Goal has a 'type' association (Generalization), which include: <ul style="list-style-type: none"> – Conjunction – Disjunction – Atomic • Condition is a <i>constraint</i> on Atomic Goals.
Impact	name, and outcome	<ul style="list-style-type: none"> • Impact defines the <i>implication</i> of a Change to an enterprise.
BusinessProcess	name	BusinessProcess has a 'whole-part' association (Composition), which include: <ul style="list-style-type: none"> • Activity.
Activity	name	The Activity concept relates with: <ul style="list-style-type: none"> • Entity: Activity references Entity (<i>refEntity</i>) • Activity has a 'type' association (Generalization): <ul style="list-style-type: none"> – Sequential. – Conditional.
Data	name	Data has a 'whole-part' association (Composition), which include: <ul style="list-style-type: none"> • Entity.
Entity	name, and attribute	<ul style="list-style-type: none"> • Entities describe enterprise information
ChangeDriver	name, and type	The ChangeDriver concept relates with: <ul style="list-style-type: none"> • Change: ChangeDrivers <i>trigger</i> changes. • ChangeIndicator: ChangeDrivers are <i>indexedBy</i> ChangeIndicators. • Action: ChangeDrivers <i>require</i> definite Actions for proper response.
Action	name, and module	the Action concept relates with: <ul style="list-style-type: none"> • Actor: Actions can be <i>performedBy</i> an Actor. • AgilityEnabler: Actions can be <i>supportedBy</i> AgilityEnablers. • Activity: Action can be broken down into Activity (actionModule).
Actor	name, type, and role	The Actor concept relates to: <ul style="list-style-type: none"> • ChangeIndicator: Actors can monitor enterprise environment to <i>detect</i> ChangeIndicators.
AgilityEnabler	name, and detail	AgilityEnablers can be used to <i>support</i> an Action.
ChangeIndicator	name, threat, and degree	ChangeIndicators <i>index</i> ChangeDrivers.

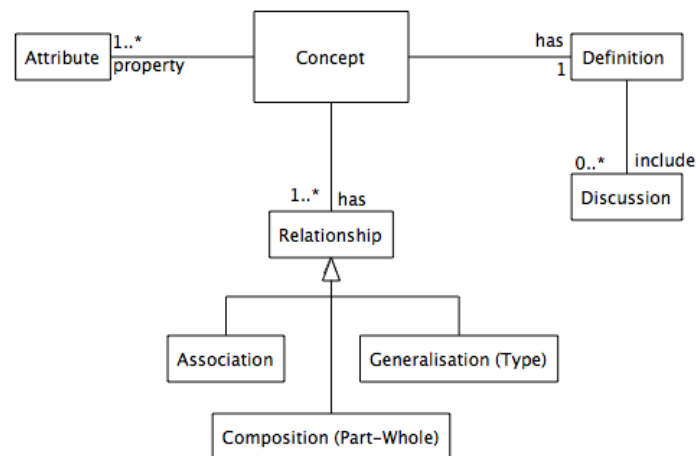


FIGURE 5.2: Structure of Concept Definition

Note: Concepts, attributes, auxiliary concepts, and generalisation in 'bold face' fonts means that they are represented as standalone classes in the abstract syntax model of the proposed modelling language. The core concepts are defined according the order in which they are listed in Table 5.1. But if a core concept has auxiliary concepts, they (auxiliary concepts) will be defined first, before defining the next concept.

5.2.2.1 Change

Definition: A **Change** is a transition, in one or more domains of an enterprise, from a current state to a target state, as a result of a change driver.

Discussion: Usually, such a transition would require or lead to modification(s) in one or more domains (goal, business process, and data) of an enterprise. For instance, if an enterprise desires to transit to a target state of manufacturing new products or offering new services, as result of competition; then it would require some modifications in some of its domains, such as addition of new business entities and attributes in the data domain, to record information about the new products or services.

These modification can be captured and described using the goal model construct. In order words, the change concept can be used to describe, as well as convey

information about the modification(s) which should be made in the domain(s) of an enterprise in order to achieve a transition to a desired state. In addition, it can help to capture the exact domain(s) where the required modification(s) should take place.

The auxiliary concepts of change include **Goal**, **Change Impact**, **Business Process** and *Data*. Details of these auxiliary concepts are discussed in the sections below.

Attributes: The attributes of change concept include *request* and *iDomain*. Since enterprise changes normally require or lead to modifications in at least one enterprise domain. An enterprise change can be described using the *modification request*, simply called the *request* attribute. In other words, the request attribute gives information about the aspect of an enterprise that should be modified in order to achieve the desired transition to a target state. It can also describe the modification(s) a given enterprise change can lead to, and the domain where such modification should take place. An example of the change request from the from the above example could be "*modify manufacturing data to add the new product information*".

The *iDomain* attribute, which stands for intended domain, is used to convey information about the domain of the enterprise where the modification should take place. This can be across the entire domains (goal, business process, data) of an enterprise, a combination of two or more domains, or a single domain.

5.2.2.2 Goal

Definition: A **Goal** is an intention of a system under consideration or a stakeholder thereof [114].

Discussion: Goals can also be used to describe the rationale or motivation behind enterprise actions. Usually, goals are expressed using goal oriented modelling (GOM) techniques. GOM, a product of requirements engineering, has been extensively described in extant publications such as [54–56, 114, 137, 141, 179, 180]. Therefore they will not be described in further details here. Since this research does not seek to propose or re-invent GOM language, only those aspect of GOM that are useful for this research are discussed.

Goal is used as an auxiliary concept, for a change, that can be used to express a change request into actionable objectives. As mentioned earlier, it is required that

a change is clearly represented, decomposed to an understandable set of objectives, and properly articulated, before adequate change management plans can be formulated [138, 139]. As explained in Chapter 2, a goal can be decomposed into actionable objectives [114, 137] and thus can be useful to these ends. Therefore, goals can be used as means of representing changes explicitly, and decomposing them into achievable objectives.

Generalisation: A goal can be of any of the following types: parent, sub-goal, disjunction, conjunction, and atomic goal [114, 136, 141]. **Parent goals** are goals that can be decomposed into one or more sub-goals. A **sub-goal** provides some alternatives for achieving a parent goal. Two or more sub-goals can be obligatory alternatives for achieving a parent goal, in this case they are called **conjunction goals** (con). Equally, sub-goals can be optional alternatives for achieving a parent goal, in which case they are called **disjunction goals** (dis). An **atomic goal** is any goal that has no alternative and cannot be further decomposed [137, 141, 179, 180]. Atomic goals usually make reference to an enterprise entity such as cost, product, staff, etc. For instance, the atomic goal, "*employ first class graduates only*", make reference to the employee entity of this enterprise.

Attributes: The main attribute of all goals concept is name, but an atomic goal should have **Condition**. A condition is any constraint placed on the property of a given enterprise entity referenced by an atomic goal. For instance, the atomic goal, "*employ first class graduates only*", places a condition over the 'grade' property of the 'employee' entity it references. This condition can be communicated using an expression. For example *employee.grade = 'first class'*. Expression is represented in the model using an acronym **exp**. The name attribute can be used to describe the given name of a goal.

5.2.2.3 Change Impact

Definition: A change impact, simply called **Impact**, is the potential or anticipated implication(s) of a change to an enterprise.

Discussion: Expectedly, a transition or change of state in an enterprise would have some effects, implications, or consequences. These can be described using the impact model construct. For instance, the impact of the change in the example

given in 5.2.2.1 could be the cost, time, resources, efforts, *etc.*, expended in manufacturing the new products or offering new services. Even though, the effects of changes or impacts are generally considered to be negative to enterprises. Some changes can also result to a positive effect. For instance, addition of new products or services can lead to customer satisfaction, attract new customers, and increase profit. Hence, its possible to have a positive change impact.

Attributes: In addition, to its name, an impact should have an **outcome** attribute, which can be used to precisely describe the impact. For instance, the impact of adding new products can be described as follows: "*the cost involved in developing the new products and services*". This cost could be in terms of money, time, human resources, etc.

5.2.2.4 Business Process

Definition: A **Business Process** is the set of logically organised activities for achieving definite outcomes [2, 113].

Discussion: A coherent set of activities may be involved, or required, for an enterprise to transit to a desired state. These are usually identified by enterprise managers or stakeholders concerned with a change initiative. Thus, business process, a derived concept, can then be applied to expressed the set activities required by a change. Research on business process modelling (BPM) technique has progressed significantly. Equally business process models and notation (BPMN) are well known and covered in existing literature such as [7, 9, 32, 48, 144]. However some BPM construct relevant to this research are briefly discussed below.

Composition: A business process is composed of a series of **Activities**. An **Activity** is a piece of work to be completed [144]. Association link, usually a directed arrow or a simple straight line, can be used to show the coherent flow of activities in a business process model. Two types of business process activities are available in the proposed modelling language. These are **Conditional** and **Sequential** activities. A *conditional* activity is one that gives a boolean result (true or false) and directly links to two other activities, while a *sequential* activity does not give a boolean result and directly links to one activity. Each activity is identified by its name.

Attributes: A business process has a `name` attribute, which describes the given name of the process.

5.2.2.5 Data

Definition: **Data** are enterprise entities and the attribute of these entities that should be involved in a given change.

Discussion: Data can also be used to describe the business entities required by the enterprise to transit to the desired state. These entities together with their attributes are represented as data model. The idea of data modelling is not new to information systems and software engineering. Instead, it has been widely applied in database design using entity relationship (ER) diagrams. This research applies some aspect of data modelling relevant to the proposed modelling language. But further information about data models are available in literature, see [3, 24, 26, 37, 189]

Data is a derived concept which can be used to capture the entities affected by change. In general, some or all enterprise entities can be affected during a transition or change. This implies that some entities and their attributes may require some modifications such as updating, deleting, adding, etc., in order to achieve the desired transition. For instance, adding a new product or service, require new business entities and attributes to capture information about such products or services.

Composition: Data are composed of one or more entities. An **Entity** is known to be a thing or an object of interest, such as `Employee` and `Equipment`, to an enterprise. Entities have `attributes` and `names`. An attribute is the property of a given entity. For instance, the `Employee` entity would have attributes such as *identify, address, salary, etc.*

Attribute: Data is modelled as an abstract class in the proposed language, and thus does not have an attribute of its own. Instead its attributes are those assigned to an entity, *i.e.* `name` and `attribute`.

5.2.2.6 Change Driver

Definition: A **Change Driver** is an event, circumstance or condition that can lead to a change in an enterprise.

Discussion: Examples of such events include merger and acquisition, competition, and regulatory compliance. Essentially, the change driver model construct can be used to express and share information about the origin or the root cause of an enterprise change. This can facilitate an understanding and knowledge about the causes or rationale behind a change. So that enterprise stakeholders and managers can, first of all, know and understand why their enterprise should change.

Attributes: A change driver should have a **name** and **type** attributes. The name attribute captures and describes the event(s), circumstance(s), and/or condition(s) that triggers the change. The type attribute says whether the change driver is external or internal. A change driver is external if it originates from outside the boundary of an enterprise. Some examples of external change driver include regulatory compliance and competition. On the other hand, a change driver is said to be internal if it originates from within the enterprise boundaries. Examples include organizational culture and politics, and overhaul of business strategy to improve enterprise effectiveness.

5.2.2.7 Action

Definition: An **Action** is a set of definite activities or tasks carried out to respond to a change driver or achieve a change.

Discussion: In order to achieve the transition to a desired state, and/or respond to a given change driver, an enterprise is expected to carry out some activities. These activities can be described and articulated using the action model construct. Actions are usually carried out enterprise *actors*, which can be machine or human. Thus, actor is considered to be an auxiliary concept of action and described in the following section.

Attributes: An action may have a name, module and/or an alternative attribute. The name attribute simply expresses the name of the action. For instance, an enterprise may decide to implement a new technology, or re-engineer its business process in order to respond to a given change driver. In this case, the name of the action would be: *implement new technology* or *re-engineer business process*.

The action module attribute is used to breakdown each action into a set of manageable activities that can be used to actualise the action. These can be organised into a set of logical activities, *i.e.*, business process or they can be ordinary actions realised by humans. For instance, there are various activities that can be involved if an enterprise chooses to implement a new technology as the suitable action. These can include requirements analysis, design, modelling, etc. The action module attributes are used to capture and describe these activities.

As described in Chapter 2, it's possible to have more than one actions for responding to a given change driver or actualising a given change. Usually, one of these actions would be implemented at any given time. The Alternative attribute is used to express any candidate action that has not yet been implemented. This can be useful in cases where the selected and implemented action fails or did not yield the desired result. In such cases, stakeholders can select and implement a substitute action from the specified alternatives actions. Alternative is represented in the proposed language with the acronym *alt*.

5.2.2.8 Actor

Definition: An **Actor** is an entity that has special interest and/or plays specific role in an enterprise.

Discussion: Change management initiatives are often accompanied with some roles such as identifying actions required to adapt to change drivers, and monitoring an enterprise environment to detect and mitigate potential change drivers. These roles are played by certain enterprise actors, (stakeholder or system) and can be captured using the agent concept.

Actor is not a new concept, particularly, in the field of information systems and enterprise modelling frameworks. It is taken to be a product of agent oriented modelling and agent oriented software engineering. Detailed description of agent oriented modelling is beyond the scope of this research. Further information about agent and agent oriented modelling can be found in existing publications such as [44, 192, 193, 195].

Attributes: An actor can have a name, type, and role attribute. The role attribute describes the specific duty of an actor in a change management initiative. The type attribute can be used to specify whether the agent is a human stakeholder or a machine (system). While the name attribute specifies the given name of a particular agent.

5.2.2.9 Agility Enabler

Definition: An **Agility Enabler** is a function or technology that supports an action towards a change driver or the entire change management initiative.

Discussion: Actions specified against a change driver should be supported by an agility enabler, so that they can yield optimal results. For instance, if an enterprise decides to re-engineer its business process as a suitable action for responding to a competition change driver, then it should identify and specify some enterprise functions and/or technologies that can support this action. These can include technologies such as business process execution language (BPEL). Agility enablers can also include some enterprise functions such as information transparency, business and information technology (BIT) alignment, etc.

Attributes: Agility enabler has a name and detail attributes. The name simply identifies what the agility enabler is. The detail attribute gives more information about the agility enabler. For instance, assuming an agility enabler is identified as *BPEL technology*. Then the detail attribute should be used to convey more information such as the specification, versions, etc., of the technology.

5.2.2.10 Change Indicator

Definition: A **Change Indicator** is any environmental or social affair that can trigger a change driver.

Discussion: Usually, change drivers are indicated by certain social or environmental affairs. For instance, a new entrant into an industry can be an indicator of a potential competition change driver. Similarly, a election, surge in immigration, and increase in unemployment can all indicate possible new government regulations that might cause changes in enterprises. These types of social affairs can be

captured and conveyed to the enterprise stakeholders using the change indicator concept. This can help an enterprise and its stakeholders to be sensitive to their environment as well as detect potential threats to business stability. More detailed information about the change indicator model construct can be found in Chapter 2.

Attribute: A change indicator has three attributes, namely threat, degree and name. The threat attribute describes the type of change driver the change indicator can index. For instance, the new entrant change indicator can index a competition change driver, thus its threat is competition. The degree attribute is used to express the severity of the change indicator. If a change indicator is prevalent, serious, and most likely to cause a change, then its degree can be described as high. Otherwise it can be low or mild. The name attribute simply specifies the name of the change indicator.

5.2.3 Relationships and Multiplicities Between Concepts

This section highlights and describe the names of the association/relationships between the concepts described above and shown in the abstract syntax model, see Figure 5.1.

Trigger: Recall that change drivers are the primary causes of change in enterprise. Hence, a relationship should exist between a change and the change driver that leads to the change. The *trigger* association can be used to capture such a relationship. In other words a change driver can trigger at least one change.

ActionableObjective: As mentioned earlier, a change should be decomposed into achievable or actionable objectives, so as to facilitate the formulation of strategies, processes, and action plans for its management. Since goals are decomposable into objectives, they can be useful to this end. Therefore, a change should be associated with the goal that expresses its objectives. This can be done using *ActionableObjective* association.

Both conjunction and disjunction goals have a bidirectional association. This simply implies that a conjunction or disjunction are types of a *goal* and a goal can be a *conjunction* or *disjunction*. Each atomic goal may be *constraint* by more than one conditions. Thus atomic goal can be related with condition using the *Constraint* association.

Implication: Since each change is expected to have at least one impact, which defines its consequences or potential consequences, in the host enterprise. Each change should thus be related to its impact(s). This can be done with the *implication* association.

Achieve By: A transition (change) to a target state can be achieved through a set of logical activities called business process. Therefore a relationship should exist between a change and a business process. This can be expressed using the *achievedBy* association, *i.e.*, each change be achieved by at least one business process. Each business process contains at least one *activity*. Sequential activities to a sequence of other activities (*first/second*). While conditional activities can give a boolean result (*i.e.*, *true or false*).

Affect: Part or whole of an enterprise data are usually modified (updated, deleted, added, etc.) in order to achieve the desired state in a change initiative. Thus, a relationship can be defined, between a change and enterprise data, using the *affect* association. Data is an abstract class and a named space for entities. Hence a data contain one or more entities.

Require: Commensurate and appropriate actions are required to respond to change drivers. Hence a relationship should be defined between a change driver and its required action. The *require* association can used for this purpose. That is a change driver *require* one or more actions.

PerformBy: The actions required to respond to a given change are normally performed by agents, which can be a human stakeholder or a machine. This implies that there is should be a relationship between an action and one or more agents that carry out those actions. This relationship can be defined using the *performBy* association.

SupportBy: Actions designed for responding to a change can be enabled using certain functions called agility enabler. A relationship can thus be established between an action and its agility enabler using the *support* association. A particular agility enabler can be used to support one or more actions.

Detect: Change indicator refers to social affair, within the surroundings of an enterprise, that can precipitate to change drivers. Stakeholders, a type of agent, are expected to monitor change indicators. Hence there should be a relationship between a change driver and agent. This can be represented using the *detect* association.

IndexBy: Change indicators are usually taken to signal change drivers. This suggests that a relationship can be defined between a change driver and the change indicator that leads to it. This relationship can be expressed using the *index* association.

5.2.4 Validity Rules

This section outlines a set of rules that determine the legitimacy of expressions or models constructed with the proposed modelling Language. These rules are called validity or well-formedness rules. One important benefit of validity rules is to ensure that models are well formed, consistent, and coherent. As mentioned earlier in Chapter 3, validity rules can be defined either formally or informally. Usually formal methods can be used to define validity rules when the language is intended to be executable. But formal methods can be difficult to comprehend, learn and use, particularly for end users without programming background. This generally makes formal methods disadvantageous and restrictive.

Defining validity rules in natural language can be a better and good option. Informal methods such as using a natural language are generally known to be easier to understand. Hence, by using a natural language to define validity rules, it can be easier to understand, learn, and use by a wide range of audience and end users, when compared with formal methods. In addition, an clear, precise, and explicit definition of validity rules using a natural language can make easier for users with programming background to even formalise such rules. In this way a wider audience can use the language. Based on this, the validity rules of the proposed language are defined informally using natural language, as expressed in the paragraphs below.

UniqueName: For a change model to be valid, all '*ChangeModelElement*' must have a unique name.

SpecifyAction: A change model shall be invalid if the action(s) required to respond to the change driver is not specified.

ChangeDomain: For a change model to be well formed, a change must take place in at least one enterprise domain which can be the data domain, business process domain, and/or the goal domain. If this domain is not expressed, then the change model shall be considered invalid.

CoreElement: To be considered valid, the change in any change model must have actionable objective(s), affect at least a data element, and achieved by a definite process.

5.3 Language Concrete Syntax

As described in the previous chapter, the concrete syntax of a modelling language provides a set of graphical notations, figures, and symbols that describes how the language can be presented to users. Depending on the intended use as well as the target audience, a language's concrete syntax can be textual or diagrammatic. The relative advantages and disadvantages of diagrammatic and textual concrete syntaxes have been discussed in the previous chapter, but are also available in literature [68, 71, 85, 104]. Diagrammatic concrete syntax is adopted for the proposed modelling language. This is because diagrammatic syntax are easier to understand, learn, and use by a wide range of audience including inexperienced users.

A generic framework, known as diagram definition (DD) [142], for describing diagrammatic concrete syntax has been proposed by the object management group. A brief overview of this framework has been provided in Section 3.3.2 of Chapter 3, but a detailed description is available in [142]. This framework has been summarised into two key stage process by Clark et al [43]. This research adopt and adapt this two stage process of describing the diagrammatic concrete syntax of a modelling language. The first stage include constructing a model of diagram which can then be used as the basis for interpreting the concrete syntax. The second stage involves using the model of diagram to build the abstract syntax of the proposed language. These two stages are discussed detail in the sections below.

5.3.1 Model of Diagram

A model of diagram is an abstract model that can be used to interpret the diagrammatic concrete syntax of a modelling language [43]. Figure 5.3 shows the model

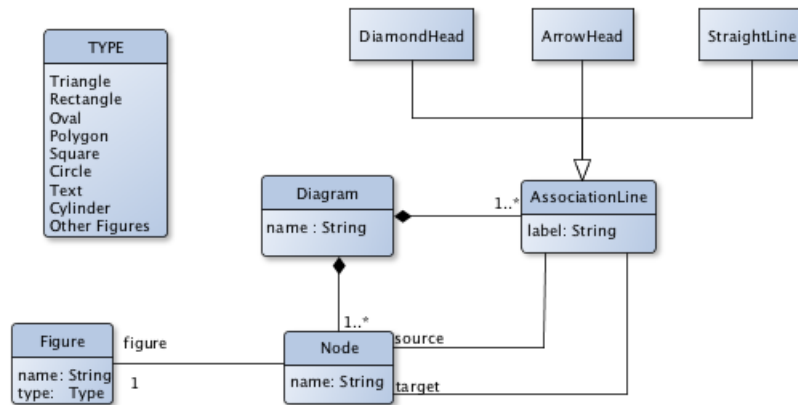


FIGURE 5.3: Abstract Model of Diagram

of diagram for the diagrammatic concrete syntax of the proposed modelling language. The root element of this model is **diagram**, which has a name. The diagram consist of at least one **node** and **AssociationLine**. A node must have a **figure** with a name and type attributes. Common types of a figure, such as *triangle*, *rectangle*, *oval*, and *polygon* have been listed as **type**. Association line can be used to show the association between between two or more nodes. It can be of any of the three types as shown in the figure. *DiamonHead* is usually used to show composition association. The **arrowhead** shows generalisation association while the bf straightline shows association between two nodes.

5.3.2 Abstract Syntax With Model of Diagram

In the second and last stage of determining the diagrammatic concrete syntax of a modelling language, the model of diagram, shown in Figure 5.3, is used to construct the abstract syntax.

The resulting model, as presented in Figure 5.4, shows relationship between the elements of the concrete syntax model (model of diagram) and the concepts of

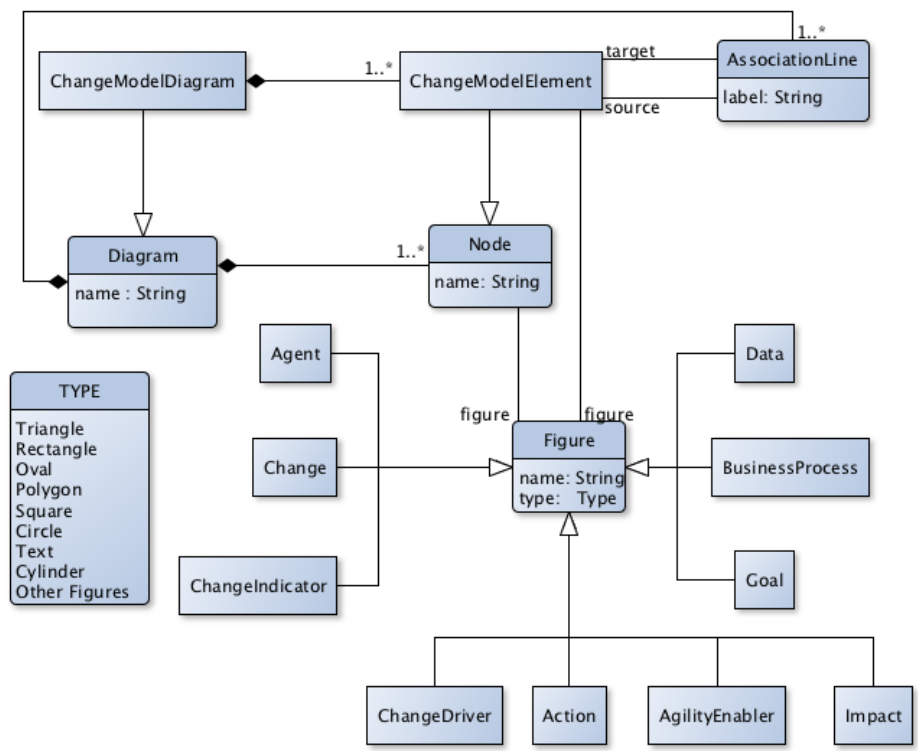


FIGURE 5.4: Abstract Syntax with Model of Diagram

the abstract syntax. For instance, a *ChangeModelDiagram* is subclass of *Diagram*, *i.e.*, (the root element in model diagram, see Figure 5.3). Each *ChangeModelElement* is a subclass of *Node* and a type of *Figure*. All the concepts, or model elements, in the abstract syntax can be represented with any type of figure listed in the *Type* class. For example, a *Change* can be represented with a *Figure* such as an *Ellipse* and so on. The *AssociationLine* shows the relationship between two or more change model element. For instance, a change can be related to its change driver using the association line.

5.4 Language Semantics

This section describes the semantics of the proposed modelling language. As mentioned earlier in Section 3.3.3 of Chapter 3, there are two main steps in defining the semantics of a modelling language. These include identification and definition of a semantic domain. In the second stage, the elements of the semantic domain are used to describe the meaning of the concepts or model constructs in the proposed languages. This is called semantic mapping. The semantic domain is described in Section 5.4.1, while the semantic mapping is described in Section 5.4.2.

5.4.1 Semantic Domain

As discussed in Chapter 3, a semantic domain refers to an existing modelling language or framework with a well defined meaning. To enhance the understanding of a conceptual modelling language, it should be related to a well known and meaningful domain [82, 83]. The proposed change modelling language uses a *Statechart* as its semantic domain. Statechart is a conceptual modelling language for describing how a reactive system can change its state [79]. Invented by Professor David Harel, the statechart has long been adopted as part of the UML standard diagrams.

The statechart is selected as the semantic domain since it can be used to describe change of state in reactive systems, and any enterprise can be described as a system that constantly reactive to the changes in its environment. In addition, the statechart is an existing conceptual modelling language known to have precise meaning, *i.e.*, it has a formal semantics. The concepts in a statechart have been semantically defined in existing publications such as [78, 80, 81, 96]. Hence it can be used as the basis to describe the meaning of the proposed change modelling language.

A typical startchart consist of the following basic concepts or model elements: 1. *State*, which describes a system, or the components thereof, at any given time. A system can be in any of the two states, namely *initial state* and *final state*, at any given time. 2. *Event*, which describes activities that can cause a system or any component of the system to change state. 3. *Action*, which captures the tasks carried out whenever there is a change of state. 4. *Transition*, which represents any change of state in a system. The detailed description of the statechart modelling language, including its abstract syntax, concrete syntax, and semantics can be found in publications such as [43, 76–81, 96].

5.4.2 Semantic Mapping

Semantic mapping establishes a relationship between the elements of a semantic domain and the concepts of abstract syntax of a modelling language. The semantic domain, *i.e.*, *statechart*, of the proposed modelling language has been described in Section 5.4.1. This section uses this main concepts of the semantic domain or statechart to describe the meaning of concepts in the abstract syntax of the proposed modelling language.

Recall that some concepts, such as *actor*, *data*, *goal*, and *business process*, in the abstract syntax of the proposed modelling language are derived from existing modelling languages. For instance, goal concepts are derived from goal oriented models and requirements engineering research [54–56, 114, 137, 141, 179, 180], agents are derived from agent oriented modelling [44, 192, 193, 195], while business process is derived from business process modelling and notation [7, 9, 32, 48, 144], and so on. Therefore, these are not part of the semantic mapping, as their semantics have been defined in their respective language definition.

Figure 5.5 shows an illustration of the core concepts, in the proposed modelling language, with the concepts of a typical statechart. Each statechart concept, as shown in a circle with double lines, is linked to the concepts of the proposed language using a bended arrow.

An enterprise can be described as a system consisting of goals (business objectives), business processes (processes for achieving these goals), and data (information required to actualise the goals and operate the enterprise). Hence an enterprise as used in the proposed change modelling language can be described as the system in a statechart. The state of an enterprise at any given time is the current or as-is state. In relation to a statechart, this means the initial state of a system as

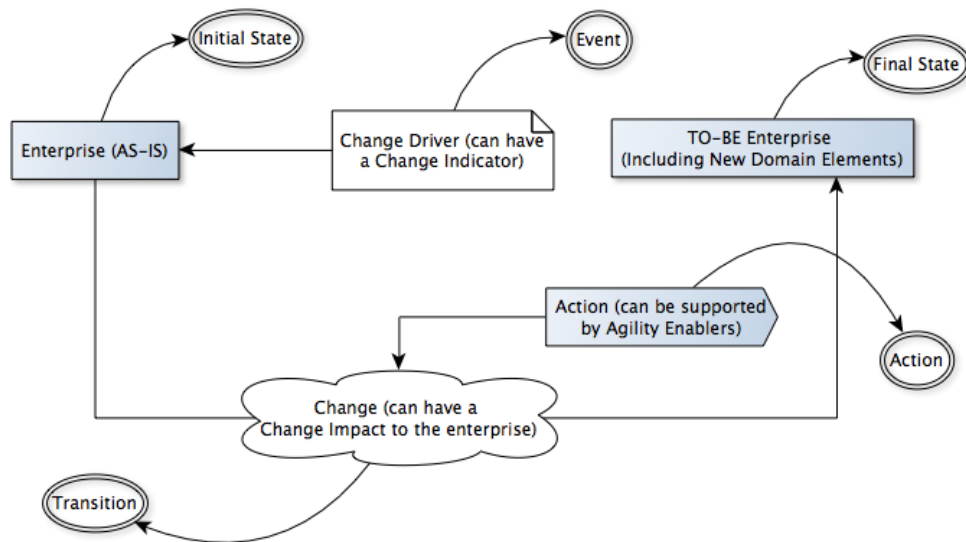


FIGURE 5.5: Core Concepts of the Proposed Language and Statechart Concepts

shown in Figure 5.5. A change driver, such as competition or regulatory compliance can cause the enterprise to change or transit to a target or to-be state.

In statechart events cause a system to change to a final state. Therefore, change driver in the proposed change modelling language means events in a statechart. Similarly a change in the proposed change modelling language can be described as the transition in a statechart, while the target or to-be state in the change modelling language means the final state of a statechart. In the proposed change modelling language, a change driver can have an indicator, known as change indicator, which can be used to index any environmental affair that can trigger that change driver.

In a statechart, an action(s) is usually invoked whenever there is a change of state to help realise the transition. As shown in Figure 5.5, this can be used to describe the action, in the proposed change modelling language, carried out to actualise a change. In the proposed change modelling language, this action can be supported using some technologies or functions called *agility enablers*. Furthermore, even though this is not explicitly described in the statechart, usually when a system changes state there are some results or implications. For instance, if an alarm system changes from off state to on state, the result or implication would be the noise or sound of the alarm. Hence, the change impact in the proposed change

modelling language can be described as the result or implication when a system changes its state.

5.5 Chapter Summary

This chapter applies the techniques and technologies, discussed in Chapter 3, for designing and developing a conceptual modelling language to develop the proposed change modelling language. The essential features of the proposed change modelling language, *i.e.*, abstract and concrete syntaxes as well as the semantics are also described in this chapter. In the next chapter, *i.e.*, Chapter 6, the enterprise modelling language is presented using the same techniques and technologies. The next chapter also provides a set of rules and describes the procedures for relating a change model with an enterprise model to derive the set of new domain elements required to adapt a given change.

Chapter 6

Enterprise Modelling Language and The Proposed Rules and Procedures

6.1 Overview

This chapter describes the second key contribution of this research, that is, a systematic approach for deriving the set of enterprise domain (data, business process, and goals) that are affected by a given change. A key step in this approach is to construct a change model as well as an enterprise model and compare the former with the later. Already, a language for modelling enterprise changes has been proposed and described in the previous chapter. The first part of this chapter proposes and describes a language for modelling an enterprise. Afterwards a set of rules and procedures for comparing a change model with an enterprise model are outlined. It is important to note that the proposed enterprise modelling language builds on and re-uses concepts from existing enterprise modelling approaches.

6.2 Enterprise Modelling

Before describing essential features of the proposed enterprise modelling language, it will be useful to provide a brief overview of an enterprise model. Enterprise modelling is a useful practice both in research and industry as it helps to express, understand, and communicate knowledge about the operations of an enterprise [184]. Various approaches and definitions of enterprise modelling are available in literature, for instance see [34, 51, 62, 72, 121, 184, 186]. Even though there are some variations in these definitions, there is a generally consensus that an enterprise modelling should represent the vital aspects (also called domains or viewpoints), such as information and processes, of an enterprise. The essence of

this representation is to convey knowledge about the operations of the enterprise. Usually, enterprise models are used for different purposes, hence the aspects of the enterprise to be represented in an enterprise model would depend on the purpose of the enterprise mode [69].

This research generally agrees with existing definitions of enterprise modelling found in literature [34, 51, 62, 72, 121, 184, 186]. Hence it defines enterprise modelling as the act of representing the goals, business processes, and data of an enterprise to enhance the understanding and communication of knowledge about enterprise operations. This definition suggests that the enterprise model proposed in this research will represent three core aspects or domains, namely goals, business processes, and information of an enterprise. It can be possible to extend this definition to include the representation of other enterprise aspects such as technology and resources. However, these other aspects are beyond the purpose and scope of the proposed enterprise modelling language and this research.

Various enterprise models have been proposed and used for specific purposes in both academia and industry. Some of these such as TOGAF, and ZACHMAN, are used to represent and provide a holistic view of an entire enterprise. Others are used to represent an aspect or domain of an enterprise. For instance BPMN is mainly used to represent an enterprise business processes, while ER and UML Class Diagrams can be used to represent enterprise data. Ordinarily, any of these existing enterprise models can be used in this research. However, the concepts or modelling constructs contained in them are either too generic or not sufficient for the purpose, scope, and use of this research. The intention and scope of this research is consider potential changes in three main domains, *data, goals, business processes* of an enterprise. ER or UML class diagram could be used to represent the data domain, but either of them would be inappropriate for business process and goal domains. Similarly, BPMN is mainly used to represent business process but would be inappropriate for the other domains.

Equally, other approaches such as TOGAF and ZACHMAN can be generic, and contain too many aspects together with constructs that are beyond the scope of this research. For instance, the technology aspect of the TOGAF framework is beyond the scope of this research. In the same way, the network and function viewpoints of an enterprise as described in the ZACHMAN framework are not required in this research. This is because the focus of this research is to describe an enterprise as embodiment of three key domains, which include goals, business process, and information, similar to the enterprise models proposed in [162]. Detailed description of these key domains have been described in existing enterprise modelling approaches. For instance, BPMN describes how an enterprise business process

can be represented. Hence these are reused in the proposed enterprise modelling language.

6.3 The Enterprise Modelling Language

As mentioned in the previous Chapter, the essential features of a modelling language are defined in its meta-model. These features include the abstract syntax, concrete syntax, and semantics of the modelling language. Abstract syntax defines each concept of a language and describes how these concepts can be combined to form a logically correct model. Concrete syntax defines the concrete notation and how the language can be presented to the end users. The semantics provides a meaning to any logically correct expression or model of a language using a semantic domain and a semantic mapping.

TABLE 6.1: Concepts of the Proposed Enterprise Modelling Language

Concepts	Source Ent. Modelling Language.	References
<ul style="list-style-type: none"> • Data • Entities • Attributes 	Entity Relationship Modelling (ERM)	[3, 24, 26, 37, 189]
<ul style="list-style-type: none"> • Goal • Goal Types • Condition 	Goal Oriented Modelling (GOM)	[54-56, 114, 137, 141, 179, 180]
<ul style="list-style-type: none"> • Business Process • Activities 	Business Process Model and Notation (BPMN)	[7, 9, 32, 48, 144].

The proposed enterprise modelling language does not include new concepts or model constructs. Instead, it reuses the concepts of existing enterprise modelling approaches identified in the systematic literature review in Chapter 2. These concepts are shown in the extreme left of Table 6.1. The center column of this Table shows the existing enterprise modelling languages where these concepts are derived from. Since the essential features, *i.e.*, *abstract syntax*, *concrete syntax*, and *semantics* of these languages are already described in available publications, their description will not be repeated in this thesis. Publications that describe these have been included in the reference column at the extreme right of Table 6.1.

Figure 6.1 shows the abstract syntax model of the proposed enterprise modelling language. The abstract syntax model is constructed using eclipse modelling framework (EMF) and meta-object facility (MOF) concepts, see Section 3.4.2 and Section 3.4.1 of Chapter 3.

The diagram in Figure 6.1 provides the conceptual and logical description of the proposed enterprise model. Aside this description, an enterprise can also be structured or represented in layers, for instance see [116, 162]. In layered representation, each key domain of the enterprise forms a layer. The first layer is the business architecture/model, which normally includes business objectives (goals), this is followed by the business process layer that captures the main activities required to actualise business objective. The information layer describe the essential business entities and attributes (Data) that support business objectives. Depending on the scope of the enterprise model, other layers can also be added. Layered approach to enterprise modelling is beyond the scope of this research, but further details about layered approach can be found in [116, 162].

As mention earlier, the enterprise model proposed in this research consist of three main domains of an enterprise. These include data, goal, and business process. Enterprise data is an abstract term used to capture and structure the essential information in an enterprise. As shown in Figure 6.1, data is an abstract class which can be represented as entities using the entity-relationship model (ERM) [3, 24, 26, 37, 189]. A typical ER model describes the business entities of an enterprise and relationship between them. Entities are known to be objects of interest, such as products, services, and customers, to a given enterprise. Each entity has a set of characteristics or properties called attributes. For instance a customer should have a name, and address.

An enterprise should have a set of definite goals that describe its rationale for existence as well as what it wants to achieve. These can be represented using a goal oriented modelling (GOM) language. A typical goal oriented model is made up of a goal (root goal), subgoals, and atomic goals. A goal is an abstraction that defines why an enterprise exists, and what it must achieve. A parent goal can be achieved by decomposing it into objectives, called subgoals. Some subgoals can be an optional to achieving a parent goal, these are called disjunction. Conversely if a set of subgoals are mandatory to achieving a subgoal, they are called conjunction. An atomic goal cannot be decomposed further and it usually attached to a condition. Further details about goal oriented model and its the essential features can be found in [54–56, 114, 137, 141, 179, 180].

A set of coherent and logically defined activities, called business processes, are

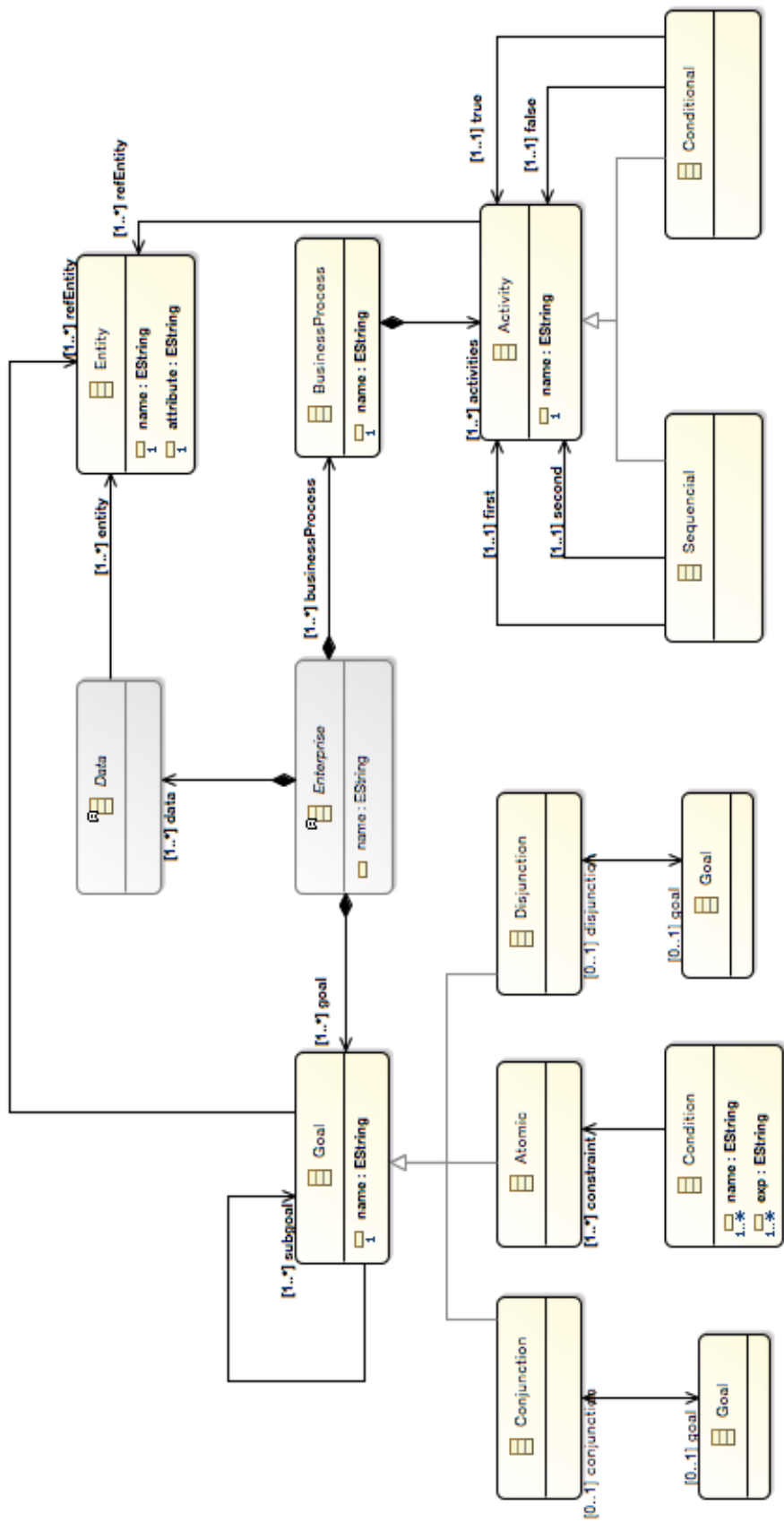


FIGURE 6.1: Abstract Syntax Model of the Enterprise Modelling Language

usually required to actualise business outcomes and goals. An enterprise business process can be represented using the business process model and notation (BPMN). Essentially the BPMN is made up of activities, generally referred to as pieces of work to be completed in order to fulfil a business outcome. An activity can be sequential if it can lead to other activities. Other activities can lead to boolean results, these are called conditional activities. Detailed description of the syntax and semantics of business process model and notation can be found in [7, 9, 32, 48, 144].

Usually, business process activities reference one or more enterprise information, which are represented as entities or attributes in the data model. For instance, the business process activity such as *'download customer details'* references the customer entity in the data model. Equally, enterprise goals can also reference one or more enterprise entities. This is because the rationale for the existence of an enterprise, specified using goals, usually relates to one or more business entities. For instance, an enterprise can exist to satisfy customers, create excellent products, and deliver valuable services. These rationale (goals) reference customer, product, service information, which can be represented as entities of a data model. Based on this, each goal and business process activity must reference one or more entity as shown in Figure 6.1. This constraint is very important because it can be used to resolve the naming conflict that can occur when comparing an enterprise model with a change model. See Sections 6.4.1 and 6.4.2 for conflicts in comparing conceptual models.

Now that the modelling language for an enterprise has been described. The next steps are to describe the rules for relating a change model to an enterprise model. As mentioned earlier, the essence of this relationship is to derive the elements (entities and attributes) of an enterprise data domain, elements (activities) of the business process domain, and the elements (goals) of the goals domain that can be affected by a given change. These rules are based model comparison *i.e.*, comparing a change model with an enterprise model. Therefore, before presenting these rules, it would necessary to provide a brief overview of model comparison, and the type of conflicts that can occur when comparing two models. The section that follows will give a brief discussion of model comparison. Afterwards rules for comparing enterprise and change models will be presented.

6.4 Model Comparison

Conceptual model comparison is a useful practice in the Information Systems and Software Engineering disciplines. It helps to identify matching or non-matching elements or concepts between two conceptual models [108]. There can be various motivations for comparing two or more conceptual models. For instance, conceptual models can be compared in order to obtain an optimal model and enable interoperability during information systems development, data schema integration, information systems merger and integration, and so on [17, 75, 154, 155].

However, the motivation for model comparison in this research is to support enterprise agility and change management process by deriving a set of enterprise domain elements that can be affected by change; so that enterprise stakeholders can understand and estimate precisely the aspects of their enterprise that are affected by a given change, and how these can be modified to effectively adapt to the change. In this way, adequate change management initiatives can be developed.

A model comparison can involve two or more conceptual models of similar or same type. In other words models of similar syntax, notations, and semantics. This is known as homogeneous model comparison. On the other hand, a heterogeneous model comparison involves two or more conceptual models of different types, *i.e.*, syntax, notation and semantics [17, 105, 160]. The change and enterprise models compared in this research can be considered to be homogeneous. Since they have the key elements of the enterprise model, *i.e.*, *entities*, *attributes*, *goals*, and *activities* have the same syntax, notation, and semantics with those in the change model. Various conflicts can occur during the comparison of conceptual models, especially homogeneous conceptual models [17, 117, 155]. Successful comparison of conceptual models would thus depend on identifying and resolving these conflicts. The next section presents an overview of these conflicts and how they can be resolved.

6.4.1 Conflicts in Conceptual Model Comparison

Batani et al [17] identify and describe two broad categories of conflicts that can occur while comparing conceptual models or schemas. These include structural and naming conflicts. Other researchers such as Lawrence and Barker [117], Kashyap and Sheth [105], as well as Pfeiffer and Gehlert [155] also agree with these two broad categories. According to Batani et al [17] a structural conflict can occur when the same concept is represented with different model constructs in two or more

models. In relation to this research, a structural conflict can occur if the model construct or notation used to represent a concept, say entity, is different in both change model and enterprise model. But this is not likely to occur in this research, since the concrete notations of the core concepts (entity, attributes, goal, and activities) of the enterprise model are the same with their counterpart in the change model. That is, the same concrete notation is used to represent an entity, attribute, activity, and a goal in both change and enterprise models.

On the other hand, a naming conflict involves inconsistencies in the nomenclature of concepts in both models. There are two broad types of naming conflicts, these are synonyms and homonyms [17, 117, 155]. A synonymous naming conflict can occur if two different names are used to describe the same concept in the corresponding models. For instance, the concept of workforce can be represented as staff in an enterprise model but represented as employee in the corresponding change model. Conversely, homonym can occur when different concepts are expressed with the same name. For example the name customer may be used to represent a single individual in one model, but used to represent a company or firm in another model.

Potentially, a synonymous naming conflict can exist in the comparison proposed in this research, since there is no definite constraints or rules for choosing the names of concepts. Usually modellers have the flexibility of choosing suitable names to represent concepts they intend to model. However, naming conflict can be avoided if the same modeller constructs both the enterprise and change models at the same time. But this is not always feasible since change models are usually constructed during a change management process (*i.e.*, when a change occur), and hence can be constructed at different times and by different modeller than the enterprise model. Based on this it can be necessary to describe how naming conflict may be resolved when comparing both enterprise and change models.

6.4.2 Resolving Conflicts in Model Comparison

Various approaches and techniques for identifying and resolving naming conflicts, in model comparison, have been proposed in literature [17, 75, 105, 155]. Resolving naming conflict is usually a manual process, which involves users discretion and judgement; since it can be difficult for a program, machine, or computer to assign semantics to concepts and thus detect whether the same name is used for different concepts or vice versa [117, 155].

In the approach proposed by Lawrence and Barker, see [117], naming conflicts can be resolved by constructing a standard dictionary of names, terms and concepts. The proposed dictionary is similar to a typical English dictionary and should provide the meaning of concepts. Similarly, the use of manual domain term extractor and domain model builder has been proposed by Pfeiffer and Gehlert in [155]. The domain term extractor identifies and builds a glossary terms or names used for the concepts in each conceptual model. These terms can then be used to resolve naming conflicts during while comparing the models. Another possible means of resolving naming conflict is using the Extensible Markup Language (XML) [28]. This process involves annotating the conflicting names with XML namespaces, and carrying out transformations using Extensible Stylesheet Language Transformation (XSLT) [42].

The three approaches described above can be useful for comparing conceptual models particularly for the purpose of database schema integration, for instance XML has been used to integrate heterogeneous data warehouses [177]. But the purpose of comparing enterprise and change models in this research is neither to merge nor integrate the two languages or schemas. Instead the intention is to identify the domain elements that are possibly affected by a given change. More so, apart from enterprise data, other domains of the enterprise such as goals and business process models are also compared in this research. It may be inappropriate to represent business process and goal models using XML. Hence, a consistent but slightly different approach is used for resolving the naming conflict that can arise when comparing enterprise and change models in this research. Rather than constructing a standard dictionary of terms, or using domain extractor, this research proposes the use of *meta-information*. Details of the proposed meta-information is discussed the Section 6.4.3 below.

6.4.3 Resolving Conflicts with Meta-Information

This approach involves listing the core domain elements of both models, in this case enterprise and change models, to be compared in a table called *meta-information* table. An example of a meta-information for an enterprise model is shown in Figure 6.2. This consists of a table with three columns, *i.e.*, *RefEntity*, *Activity*, *Goal*. Each cell in the first or 'RefEntity' column contains the entities to be compared in the enterprise model. Similarly, each cell in the second or 'Activity' column contains the activities to be compared in the enterprise model; while the cells in the third or goal column contain the goals of enterprise model to be compared. Usually goals contain longer strings than activities and entities. Hence, instead

of writing long strings of goals, which can make the goal column unnecessarily long, users can annotate each goal in the model and use this annotation in the meta-information table. For example, in Figure 6.2, the goal: *ensure higher grade for overseas applicants* is annotated with S1 in the model. S1 is then used in the meta-information table instead of writing the long strings in this goal.

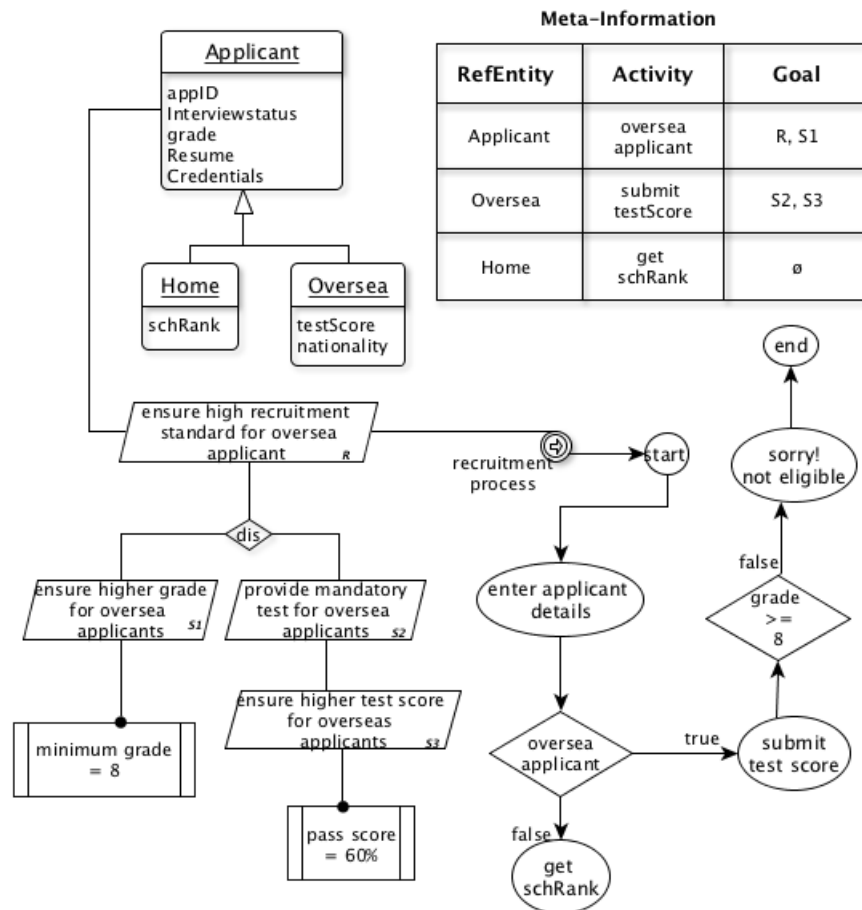


FIGURE 6.2: Example of Naming Conflict Resolution Table

Recall that in the proposed enterprise modelling language shown in Figure 6.1, and described in Section 6.2, activities and goals are usually referenced by some entities. The same also applies to the proposed change modelling language discussed in Chapter 5. Hence when constructing the meta-information table, each activity and goal should be placed next to the entity they reference. For instance, as shown in Figure 6.2, the *submit testScore* activity and the two goals *i.e.*, S2 and S3 reference the *oversea (applicant)* entity. This can help to accurately identify and

resolve synonymous naming conflicts, since it can be inaccurate to say whether or not two concepts are synonymous based on their names alone. In cases where one entity is referenced by two or more activities, the user may not need to repeat all the activities referenced by that one entity. As this can make the data in the meta-information unnecessarily long, clumsy, and redundant. Instead, one of the activities can be selected, especially the one that can be more relevant.

Meta-Information For Enterprise Model			Meta-Information for Change Model		
RefEntity	Activity	Goal	RefEntity	Activity	Goal
Staff	submit timesheet	R, S1	Employee	submit timesheet	R, S1
Cusomer	submit PersonNum	S2	Buyer	submit CompanyID	∅
Applicant	sign in	S2, S3	Applicant	login	S2, S3

FIGURE 6.3: Resolving Conflict with Meta-Information

As an example, consider the meta-information for both enterprise and change models as shown in Figure 6.3. Without considering the corresponding activities and/or goals, a user can judge that the two entities, *i.e.*, *staff* and *employee*, have similar meanings and thus amount to a synonymous naming conflict. This may be accurate in this case since the staff and employee entities relate to the same activity *i.e.*, *submit timesheet* and/or the same goal *R,S1*. But the same may not be applicable to other entities such as *customer* and *buyer*. Considering their names alone, the customer and buyer entities may appear to be a synonymous naming conflict, which can possibly be resolved by merging the two entities into a single entity. However, by further considering the activities and/or goals that reference these entities, it can be seen that there is no synonymous naming conflict in them. From the activity, *i.e.*, *submit PersonNum*, associated with the customer entity, it can be seen that a customer refers to an individual with a different set of characteristics. Whereas from the activity, *i.e.*, *submit CompanyID*, the buyer entity refers to a company with a different set of characteristics than an individual. Therefore there

is no naming conflict between the customer and the buyer entities, even though they appear to be synonymous.

This same principle is also applicable to other domain elements or columns in the meta-information table. The idea here is that candidate synonymous naming conflicts should be confirmed. As described in the paragraph above, the confirmation process involves considering other model element(s) related to the model element under consideration. In other words, if an activity of a change model is synonymous to an activity of an enterprise model, then the entities these activities reference, and/or goals they relate to should be used to confirm whether or not this synonym amounts to a naming conflict. Applying the meta-information to identify and resolve synonymous conflicts is not an automated process, rather it is based on human judgement. Usually user discretion is employed to identify synonyms in model elements of both enterprise and change models. Once a synonym between two model elements is confirmed to be a naming conflict, then the two model elements can be merged into a single entity. For example, staff and employee can be resolved by merging them into a single entity called staff.

6.5 Rules for Relating The Models

As described earlier, and shown in Figure 6.1, an enterprise can be represented in terms of business objectives *i.e.*, goals, a set of processes for achieving these goals *i.e.*, business process, and information required to achieve those goals *i.e.*, data. Each of these viewpoints or domains *i.e.*, goals, business process, and data can be represented using some concepts or model constructs, usually called *domain elements*. For example, the elements of the data domain include entities and attributes, while the goal domain consists of goals and goal types.

Consider a change, due to a government regulation, that causes this enterprise to transit from its current state to a target (TO-BE) state. Accordingly, this change will affect at least one domain of the enterprise, and thus lead to modifications in the elements of that domain. For instance the change may require the addition of a business entity, or attribute in the data domain, or the addition of an activity in the business process domain. In order to adapt to this change effectively and maintain equilibrium, there should be a way to accurately derive the set of domain elements that can be affected (and thus need to be modified) by a given change. The derivation of these elements can be supported by a consistent and coherent set of rules and procedures. But such rules and procedures cannot be found in existing approaches. This section contributes to filling this gap by providing a set

of rules that can be used to derive the new domain elements required to adapt changes.

These rules are generally based on comparing each domain element in a change model with its counterpart in the enterprise model. In other words, activities of change model are compared with corresponding activities in enterprise model, and so on. The reference point for this comparison is normally the change model, since a change model can be used to represent the domain elements involved in a given change. Hence, any domain element represented in a change model is considered necessary to adapt a given change. If this same domain element is not available in the enterprise model, then it should be identified as one of the domain elements that should be added to the enterprise model in order to adapt that change.

Therefore, the general principle of this comparison is to identify the domain elements that are part of the change model but are not part of the enterprise model. If such a domain element exists, then it will be needed to adapt to the change, which has been represented using the change model. However, the domain elements that are available in the enterprise model but not available in the change model are not affected by the change and are not needed to adapt the change. Hence, such domain elements are not considered.

In order to make it easier to learn, understand, and use by a wide range of audience including inexperienced modellers, these rules are defined in natural language. The first rule, called *Rule 1*, provides a way to compare entities of a change model with entities of an enterprise model. The second rule, *i.e.*, *Rule 2*, provides a way to compare goals of a change model with those in an enterprise model. *Rule 3* provides a way to compare business process activities in a change model with the business process activities in an enterprise model.

Note: These rules are independent of any formal approach, and are designed to be used without formalising them. Hence it is not a requirement or mandatory to formalise them. However it can be possible to formalise them using any formal language such as formal logic. Please refer to Appendix F for examples of how each rule can be formalised using formal logic.

Rule 1: compareEntities: This rule compares *similar* entities in both change and enterprise model and determine if they are exactly the same. Users can check whether or not an entity of a change model is similar to an entity of an enterprise model by looking at their attributes and the information that both entities represent. Candidate similar entity can be determined by checking if that entity

references the same activity and/or goal in both change and enterprise models. If this is the case, then the attributes of such entities should be checked in both models. For the purpose of this research, if the attributes of an entity in a change model is exactly the same with that in an enterprise model, then the two entities can be taken to be the same. The meta-information table described in Section 6.4.3 can help to identify entities and the activities and goals that are referenced by them. This rule can be defined as follows:

If an entity exists in the change model, but similar entity do not exist in the corresponding enterprise model, then that entity is required to adapt to the change (represented by the change model).

Rule 2: compareActivities: This rule compares *similar* activities in both change and enterprise models. Users can identify candidate similar activity by examining the entity and/or goal it relates to. If an activity of a change model relates to the same entity and/or goal to that of an enterprise model, then that activity is a candidate similar activity. If this is the case, then users can further examine both entities to decide if they are the same. The meta-information table described in Section 6.4.3 can help to identify activities together with the entities they reference and the goals that are related to them. For instance, in the meta-information shown in Figure 6.3, the *sign in* activity in the enterprise model, and the *login* activity in the change model relate to the same entity *applicant*. Thus they can be considered similar. This rule can be defined as follows:

If an activity exists in the change model, but similar activity does not exist in the corresponding enterprise model, then that activity is required to adapt to the change (represented by the change model).

Rule 3: compareGoals: This rule compares *similar* goals in both change and enterprise model. A candidate similar goal would relate to the same entity and/or activity in both change and enterprise models. If this is the case, then both goals can be examined further to determine if they are exactly the same. The meta-information table described in Section 6.4.3 can help to identify goals as well as the entities they reference, and the activities they are related to. This rule can be defined as follows:

If a goal exists in the change model, but similar goal does not exist in the corresponding enterprise model, then that goal is required to adapt to the change (represented by the change model).

6.6 Procedures for the Comparison

The rules for comparing and deriving model elements that are affected by change have been described in the sections above. The next step is to describe the procedures that the user can follow to actualise this. As mentioned earlier, one key motivating principle of the proposed framework is simplicity. In other words, research framework should be designed in a simple and easy to understand manner; so as to make it usable by a wide range of audience, including inexperienced modellers. In addition, the simplicity principle can make the research contributions easy to learn and use. Based on these, the proposed procedure is designed as a stepwise approach that can be followed and learnt easily.

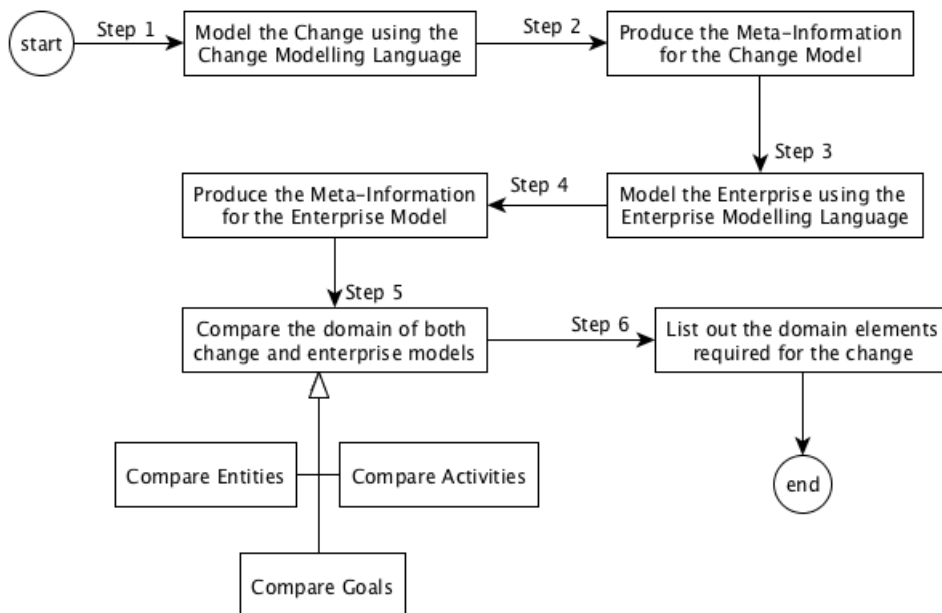


FIGURE 6.4: Procedures For Deriving Domain Elements

Figure 6.4 presents the proposed procedure that can be used to derive a set of domain elements that are affected by a given enterprise change. This procedure consists of six steps. The first step is to model the given enterprise change, this can be realised using the change modelling language proposed in this research. Section 7.2 provides an examples of how an enterprise change can be modelled using the proposed modelling language. This example can be adopted and adapted to model any enterprise change. The second step, after modelling the enterprise

change, is to build the meta-information of the change model. Details about how to build the meta-information can be found in Section 6.4.3. The essence of building this meta-information is to resolve any possible naming conflict and make the comparison process easier.

As shown in Figure 6.4, step 3 involves producing a model of the enterprise using the proposed enterprise modelling language. An illustration of how this step can be achieved has been presented in Section 7.4. In step 4, the meta-information for the enterprise model is produced. Afterwards, a comparison of the three key domain elements (entities, activities, and goals) are done in step 5. The rules for these comparisons have been provided in Section 6.5. Further examples, of how these rules can be applied, is provided in Section 7.4. The final step of this procedure involves listing of the derived domain elements. These can be listed in any form suitable to the modeller.

6.7 Chapter Summary

This Chapter presents the proposed enterprise modelling language, discusses a set of rules and procedures for deriving the set of domain elements that are affected by a given enterprise change. These rules are based on comparing the three key domain elements, namely entities, activities, and goals, in both enterprise and change models. Before presenting these rules, some conflicts that can arise when comparing conceptual models were considered. A resolution approach for these conflicts have also been proposed in this chapter. Finally, the chapter ended with a set of stepwise procedures that can be applied to derive domain elements affected by a given change.

As mentioned earlier, this research proposes an information systems framework for enterprise agility. This framework includes a change modelling language, an enterprise modelling language, a set of rules for relating enterprise and change models, and the procedures for deriving domain elements that can be affected by change. Given that all these parts of the proposed framework have been described; the next chapters will now focus on the application of this framework in three key areas as proposed in Chapter 1. These include application in representing the structure of an enterprise change (conceptual modelling), application in deriving the domain elements that can be affected by a given change, and the application in database design and implementation.

Chapter 7

Application of the Proposed Change Modelling Language

7.1 Chapter Overview

The purpose of this research is to provide a unique approach for representing enterprise changes and deriving the enterprise domain elements required to adapt those changes. In Chapter 5, the design and implementation of the proposed change modelling language is discussed in detail. More so, the rules and procedures for relating changes to an enterprise and deriving the enterprise domain elements required to adapt changes, have been presented and discussed in the previous chapter. To demonstrate utility, it would be necessary to show how the proposed framework, *i.e.*, the modelling language, rules, and procedure can be applied in a real world setting.

Hence, the focus of this Chapter is to use an industry case study to show how the proposed framework can be used to represent enterprise changes and derive the domain elements required to adapt changes. The case study used in this chapter is obtained from the Research Group's Industry partners. This case study is described in Section 2.6 of Chapter 2, and will be repeated below.

In order to effectively apply the proposed modelling language to this case study, some steps were followed. These steps have been described in Section 7.2. Even though these steps are intended not to be suggestive rather than prescriptive, they can make it easier to use the proposed language in representing any enterprise change case. Besides these, Appendix E shows an example of how the proposed modelling language can be used to implement a database to support enterprise change management initiatives.

Case Study Description: *Enterprise X* is a global leader in Information Technology (IT) Consultancy Services, whose mission is to help achieve Clients' business objectives by providing innovative and world class development, maintenance, and testing of software systems as per stated requirements. At the same time, *Enterprise X* wants to stay competitive, viable as well as maintain its position as a global leader in IT consultancy.

Enterprise X bids for requests for proposals (RFP), staffs the bid won with the right number of suitably skilled resources, follows a set software development process leading to successful delivery, and winds up the project by releasing the human and other resources being utilized for project execution. The enterprise has to function in the face of several delays such as bidding delay. To create value and achieve its corporate objective, the enterprise has to optimize cost and risk associated with its operations.

The key goal of *Enterprise X* is to stay viable and competitive. To achieve this goal, the enterprise has to develop a master plan in two key areas. First, it tends to maintain an outstanding bidding strategy, which ensures that bids are completed timely, and proposals are sent early, since late submission of bids reduces the chances of winning a bid. In addition, it maintains a strong research department, whose major role is to develop innovative technologies and send unsolicited proposals to clients. Secondly, to stay ahead of its competitors and keep up with world class global IT services, this enterprise maintains a diversified recruitment strategy and workforce. In other words, the enterprise recruits experienced specialists, and graduates with high grades from high ranking academic institutes, from both home and abroad. It also has to keep in pace with advance in technology and functional domains through adequate training or re-skilling of existing workforce.

The enterprise, in essence, is an engine managing the demand, i.e., the number and the nature of RFPs, and the supply, i.e., the number of bid sent out. For a given demand and supply state, the enterprise has to determine costs effective strategies that will help achieve the desired goals. For instance, it can decide to focus solely on having the right number of staff. It can also decide to do more and/or better with less, in which case it hires more experienced and qualitative staff from the job market. Either way, having adequate workforce on its rolls becomes an important goal, but also cost implications are always taken into considerations in all cases and at all times.

In order to enhance operational efficiencies, the organisation defines operational processes for implementing its business transactions. Those processes that are key to achieving enterprise goals are usually given special attention. For instance, the

recruitment process should be explicitly defined and followed to ensure that the required and adequate workforce are in place. In addition, it is very important for the enterprise to structure, document, and monitor key information/data that can support effective operational management and governance. For instance, data information about applicants, recruitments, bid, RFPs, etc., should be structured and monitored.

In reality, as expected, this enterprise operates in an environment characterised by spontaneous changes, mainly due to regulatory compliance and competition. Recently, a new government was sworn into power after a just concluded election. Environmental factors such as unemployment, rigging in bidding market, and surge in net migration have necessitated some new government regulations which this and other enterprises have to comply to. Additionally, some hardware manufacturing enterprises have also started providing IT services. These two change drivers are described in details below:

Immigration and Resident Employment Act (IREA): This Act seeks to reduce the immigration of foreign workers, and secure jobs for citizens and residents. In other words, enterprises which recruit workforce from abroad must give preference to those that do not require work/resident permit. In effect, enterprises are required to tighten employment eligibility criteria, for foreign workers, during recruitment. Also enterprises should be ready to prove that they have complied to this Act. In these ways both unemployment and net migration can be reduced.

Competition: Recently, many enterprises are beginning to consider Information Technology (IT) Services as a viable and profitable industry. Hence, four new IT service providers have just entered the industry. To attract clients, they launched radical marketing strategies. These include 18% discounts in all IT services, together with money back guarantee to clients who are dissatisfied. In addition, they offer longer training and maintenance services to clients. As a result the existing enterprise, under consideration, lost about 24% of its clients base to these competitors.

7.2 Part 1: Modelling These Changes

This section explains how the proposed change modelling language can be applied to model the two change drivers described in Section ?? of the case study above. Henceforth, the immigration and resident employment act (IREA) will be called **Case 1**, while competition will be called **Case 2**. The change model for *Case*

1 is shown in Figure 7.1, while the change model for *Case 2* is shown in Figure 7.2. Modelling these changes can enhance understanding and help to convey knowledge about: (i) The courses and origin of the given change. (ii) How the change can be expressed. (iii) The enterprise data that can be affected by the change. (iv) The actions required to adapt to a change. (v) The indicators of the change, and so on. The steps described below can be followed to achieve this, however as mentioned earlier, these steps are not prescriptive. Instead, they can be adapted to suit the enterprise's specific situation and change case.

Step 1–Identify and model the change driver : This can be done by studying and examining the enterprise change case or situation. For instance, by reading through the case study above, one can deduce that the change driver for *Case 1* is government regulation or *regulatory compliance*. This requires enterprises to reduce the number of workers hired from abroad by raising employment/recruitment criteria for foreign applicants. Similarly, the change driver in *Case 2* is *competition* as a result of the activities of new IT service enterprises. In this case some new IT service providers have entered the industry leading to loss of competitive advantage. The identified change driver should be represented using a suitable notation, as shown in Figures 7.1 and 7.2. The *type* attribute of the change driver can be written on top of the change driver notation as shown in the Figures. In both cases, the change drivers are external to the enterprise, since their origin is not from within the enterprise. Notice that the KEY shown in Figure 7.1 does not include notation for some elements such as data, business process activities, and goals. This is because these notations are popularly known, widely used, and are readily available in related publications. Hence this research does not consider it necessary to repeat them.

Step 2–Specify and model the change(s) : Usually change drivers must lead to some changes. Hence, the change driver identified in STEP 1 above would lead to at least one change in this enterprise. As discussed earlier in Chapter 5, changes are transitions to a new state of an enterprise, and can be written in form of requests. As in STEP 1, the desired change can be identified by carefully examining the change case under consideration. In *Case 1*, the enterprise should transit to a state where recruitment criteria for oversea applicants are tightened. As shown in Figure 7.1, this change is represented as the request: *modify the recruitment process to tighten the eligibility criteria for foreign applicants*. Equally, in *Case 2*, the enterprise is required to transit to a new state where bidding strategies are modified to clients. This can be represented as the request: *modify the bidding strategy to attract and retain new clients*, as shown in Figure 7.2. In both cases, the *iDomain* attribute, *i.e.*, the

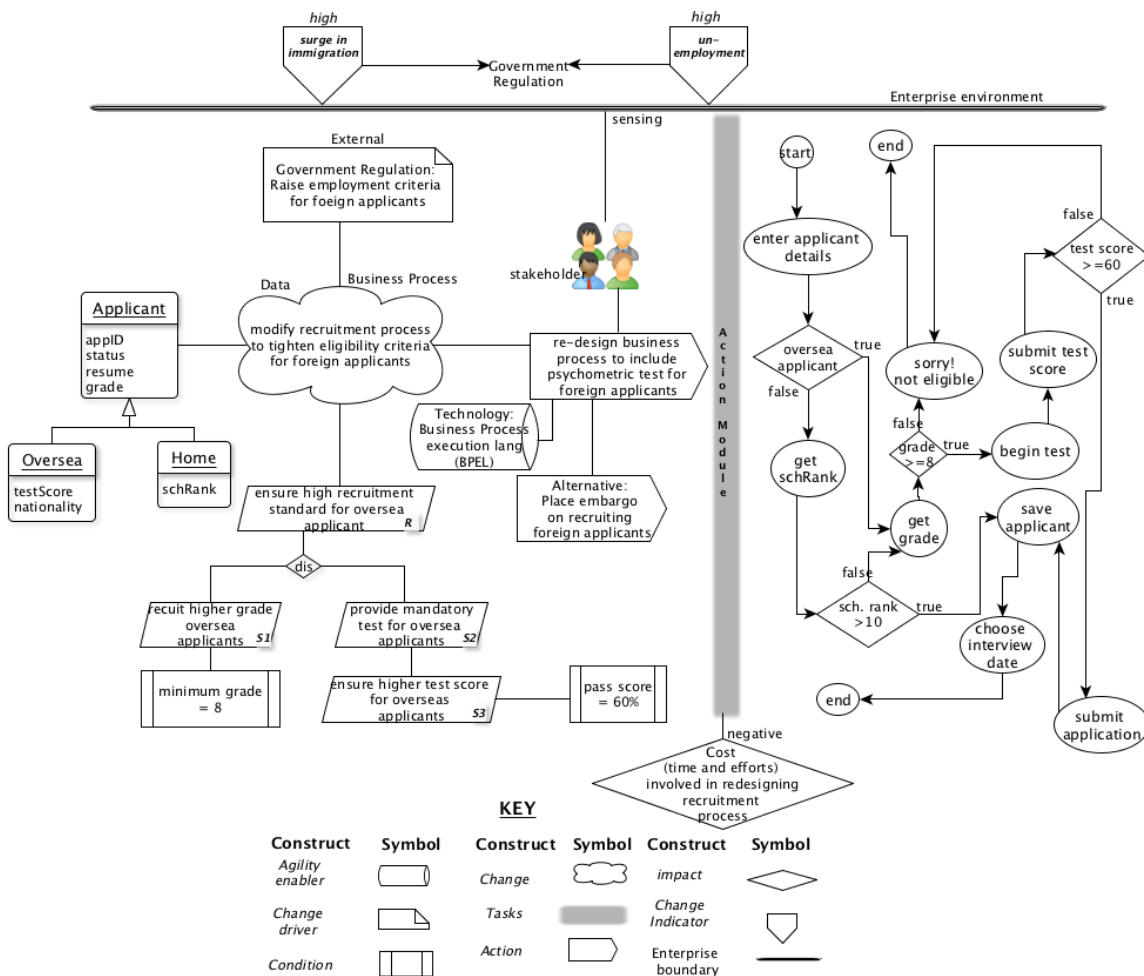


FIGURE 7.1: Change Model of the case Study: Case 1(Enlarged in Appendix C)

intended domain where the modification would take place should be represented. For instance, the change in *Case 1* would require a modification in business process and data. To make this obvious, these domains can be written on top of the change notation as shown in Figure 7.1, and 7.2.

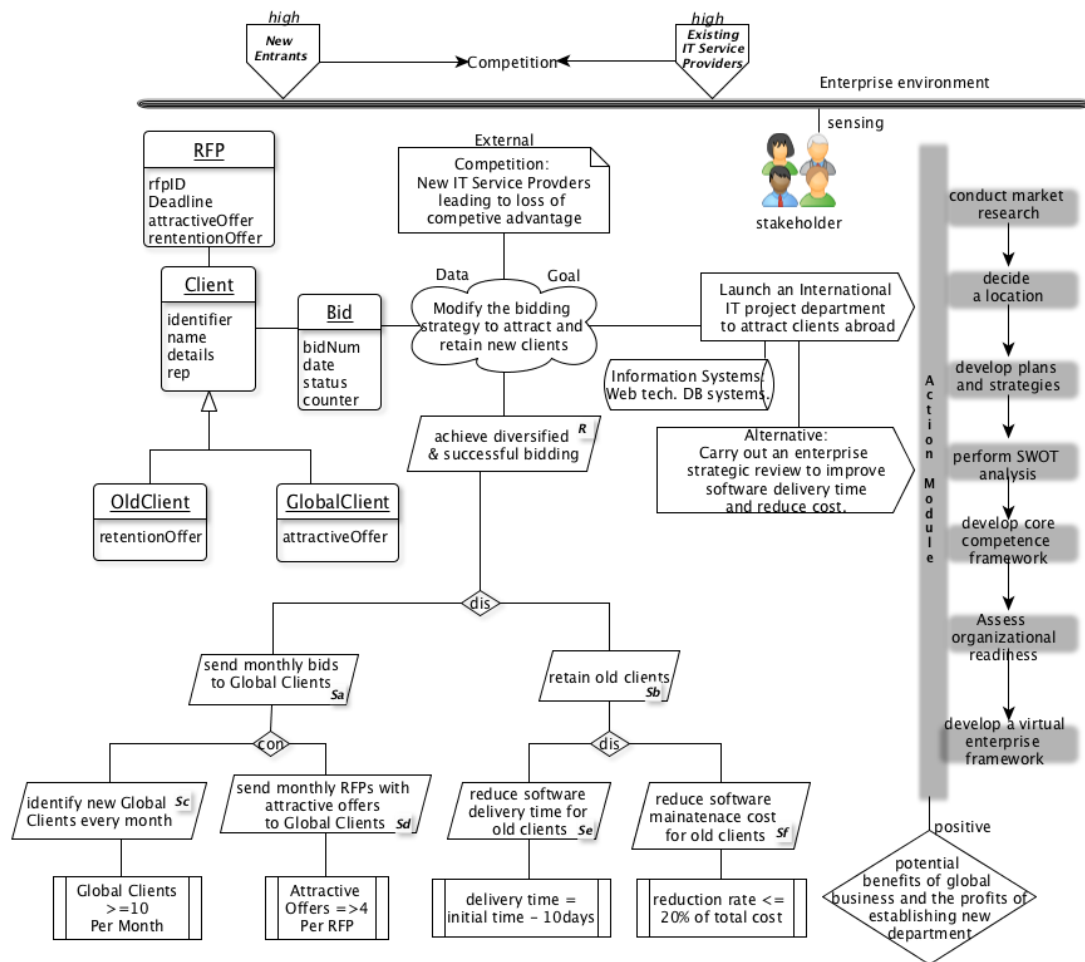


FIGURE 7.2: Change Model of the case Study: Case 2(Enlarged in Appendix C)

Step 3–Decompose the Change into Objectives : The change represented as a request in STEP 2 above should be expressed in a precise, and clear manner as well as decomposed into achievable objectives. This implies that the change request should have concrete meaning and be explicit, so that stakeholders using

the model can understand, for instance, what is meant by tightening recruitment criteria for oversea applicants, diversifying bidding process, and so on. Since goal modelling provides a means of decomposing a goal into achievable objectives. STEP 3 can be achieved by expressing the change request as goals, and using goal decomposition to break down each change into achievable objectives. So that the desired changes can be understood and shared as a common objectives, and concerted effort can be made to develop winning strategies for adapting to the change. As shown in Figure 7.1 and 7.2, the change request in both cases are linked to a root goal, and further decomposed into achievable objectives using goal modelling techniques. The annotation, such as *S1*, and *Sa*, attached to each goal will be used when producing the meta-information for each change model. The essence of annotating each goal has been discussed in Chapter 6.

Step 4—Identify and Model Actions : After specifying and expressing the change, the next step is to decide on the action(s) required to adapt the change. Actions are tasks carried out in response to a given change driver/change. This implies that action(s) are usually specific or peculiar to a given change driver or change rather than being generic. To support the decision about adequate actions, both the change driver and the change can be examined. For example, the overall aim of the government regulation change driver in *Case 1* is to reduce the immigration of foreign workers. This requires enterprises to raise the eligibility criteria for foreign applicants, and the change requests for modifications in the recruitment process of the enterprise (see Figure 7.1). Based on these, an adequate action should tend towards modifying the recruitment process, so that eligibility criteria will be tightened for foreign applicants. Stakeholders can start to think about the various ways of modifying the recruitment process to raise eligibility criteria, and hence reduce the employment of foreign applicants.

The enterprise can carry out various tasks (actions) to tighten the recruitment criteria for foreign applicants. For instance, it can decide to redesign its employment business process to include psychometric tests for all foreign applicants. An alternative action to this can be to place an embargo on recruiting foreign applicants. It is advisable, where necessary, to always identify more than one action for responding to a given change or change driver; so that one action can be reserved as an alternative and used for contingency purposes, especially in case the initially selected action fails. This can save the cost, time, and effort, it would have taken to formulate a new action if the initially selected action fails. After identifying possible actions, stakeholders can compare and rank each of these actions in order of preference. The highest ranking one can be selected as the most viable action

to be implemented, while others can be kept as alternatives. The alternative action should be clearly marked as alternative, see Figure 7.1 and 7.2 and the next paragraph.

For *Case 1*, introducing a psychometric test to the recruitment process can lead to re-designing the business process which has cost and time implications. On the other hand, placing embargo on hiring foreign applicants can have adverse effect on the quantity and quality of employees, and can also affect the quality of projects, since employees work on projects. From the case study, the quality and skill of employees are vital to this enterprise. Thus it can be reasonable to decide that the most viable action, among the two, is to *re-design business process to include psychometric test for foreign applicants*. The re-designed recruitment process for *Case 1* is shown at the right hand side of Figure 7.1. For *Case 2*, the selected action requires that the organization *launch an international IT project department to attract customers abroad*. An alternative action to this has also been identified and modelled as shown in Figure 7.2. The selected action in the two cases should be broken down into achievable units or modules. The business process activities can substitute as the modules for *Case 1*. The action module for *Case 2* is at the extreme right of Figure 7.2. The module for a given action can be decided by the stakeholders responsible for change management.

Step 5–Identify and Model the Enablers : After identifying and selecting the adequate action for responding to a change, stakeholders should specify technologies or functions that can support the selected actions. The selected action in *Case 1* can be supported by technologies such as business process execution language. Similarly, information systems such as distributed database systems and web technologies can be used to support the selected action in *Case 2*.

Step 6–Specify and Model Change Data : Changes are generally expected to reference or relate to some information or property of an enterprise, which are represented as enterprise data. These can be identified by examining the change case. For instance, in *Case 1*, the change directly relates to the foreign applicant entity, which is a specialised type of the employee entity. Hence, this change directly references the employee and applicant entities and their attributes, see Figure 7.1. Similarly the change in *Case 2*, as shown in Figure 7.2 relates to the bid, location, and RFP entities.

Step 7–Identify the Change Impact : Change impacts are the potential implications of a given change to an enterprise. They can be identified by examining the selected actions for responding to the change. As shown in Figure 7.1, the change in *Case 1* requires an action to redesign the recruitment business process. The impact of this change can be derived by examining the potential implications of the re-designing the recruitment process. These can include the costs such as time and efforts involved in redesigning the business process. In this case, the impact can be taken to be a negative outcome to the enterprise as shown in Figure 7.1. The impact in *Case 2*, as shown in Figure 7.2, is the potential benefits of global business and the profits of establishing new business department. These can be taken to have positive outcomes to the enterprise. The impacts for *Cases 1 and 2* are shown below the action module bar in Figures 7.1 and 7.2 respectively.

Step 8–Identify and Model Change Indicator : This can be done by observing the environmental current affairs that led to the change driver. For *Case 1*, the change driver (government regulation) is in response to surge in immigration and high level of unemployment. While the competition change driver in *Case 2* is primarily due to new entrant and existing IT service providers. As shown in Figures 7.1 and 7.2, the change indicators are represented in the enterprise environment. The *degree* attribute of the change indicator should be determined by the enterprise change management team depending on the intensity and prevalence of the change indicator. Afterwards it can specified on top of the notation for change indicator, see Figures 7.1 and 7.2.

Change indicators such as surge in immigration and unemployment usually attract government laws that can affect business, thus the degree attribute for these are set to be high as shown in Figure 7.1. Similarly, competitors and new entrant change indicators can cause an enterprise to lose its customers or its competitive advantage. Hence the degree attribute for these are also set to be high. In addition, the threat or type of change driver change indicators index should also be represented. An arrow is can used to associate a change indicator with the type of change driver it can index. For example, as shown in Figure 7.1, the threat for surge in immigration and unemployment change indicators can result to government regulation change driver, which s is shown with an arrow.

7.3 Application Part 2: Change Management Proforma

The proposed change modelling language can also be used to implement a change management proforma, which can be used to capture and record enterprise changes and change management activities. The sample of a change management proforma implemented from the proposed change modelling language is shown in Figure 7.3.

CHANGE MODEL ELEMENT	ATTRIBUTE	DESCRIPTION
Change Driver	name	
	type	
Change	iDomain	
	request	
Action	name	
	module	
	alternative	
Goal	Conjunction Goal	
	Disjunction Goal	
	Atomic Goal	
	<i>Condition</i>	
Impact	outcome	
Business Process	Sequential Activity	
	Conditional Activity	
Data	entity	
	attribute	
Agility Enabler	name	
	details	
Change Indicator	name	
	threat	
	degree	
Agent	name	
	type	
	role	

FIGURE 7.3: Change Management Proforma

The first column of this proforma contains a list of the key model elements of the proposed change modelling language. The attributes of each model elements are

listed in the second column, and the third column provides a space for describing each attribute. This proforma can be useful to change management practice in a number of ways. Enterprise stakeholders with little or no modelling experiences can use it to capture and examine enterprise changes and change management activities.

In addition, this proforma can be used to capture, simulate, and reason about anticipated changes, *i.e.*, changes that have not yet occurred. For instance, assuming existing enterprises sense that new entrant is about enter the industry, or that a new government is likely to enact laws that would affect them. They can use this proforma to reason about the type of changes the anticipated change drivers can cause, the potential impact of those changes, the actions that should be taken to reduce the effects of the changes, and so on. In these ways, they can be prepared to pre-empt an anticipated change from hurting their enterprises.

Another useful benefit of this type of proforma, to an enterprise, is that it can allow some sort of comparative analysis among various changes over time. For instance, an enterprise executive may be interested in obtaining knowledge from previous change management activities, and hence can decide to ask questions such as: What change or change drivers are relatively easily managed? Which stakeholders have experience/expertise in managing and responding to a particular type of change driver? Which change drivers affect services and customers of the enterprise?

A collection of previously completed proformas, over a certain period of time, can be compared and then used to answer these questions. The knowledge derived from answering these type of questions can be used to support decisions about current or future change management initiatives. For instance, having the knowledge of stakeholders experience or expertise, from previous change drivers, can help enterprise managers to easily assign skilled stakeholders to work on a similar change driver in the future. Furthermore, since changes are known to be constant in enterprises, having a structure format, such as proforma, for capturing previous changes can be useful. For instance, strategies used in overcoming previous change drivers can be captured and stored using this proforma, so that they can be re-used in the future to support subsequent change management decisions.

7.4 Application Part 3: Modelling Enterprise and Deriving Domain Elements

Another application of the proposed framework is in deriving the set of enterprise domain elements (goals, business process activities, and data entities) that are affected by a given change, and hence need to be modified in order to adapt to that change. This section uses the two cases in the above case study to show how the proposed framework can be applied to derive these domain elements.

A stepwise procedure for deriving domain elements, affected by change, has been described in Chapter 6 and summarised in Figure 7.4. This procedure will now be applied to demonstrate how these domain elements can be derived. The first step is to model the change using the proposed change modelling language. The changes for both *Case 1* and *Case 2* have been modelled in Section 7.2 and shown in Figure 7.1 and Figure 7.2 respectively. Hence they will not be repeated in this section.

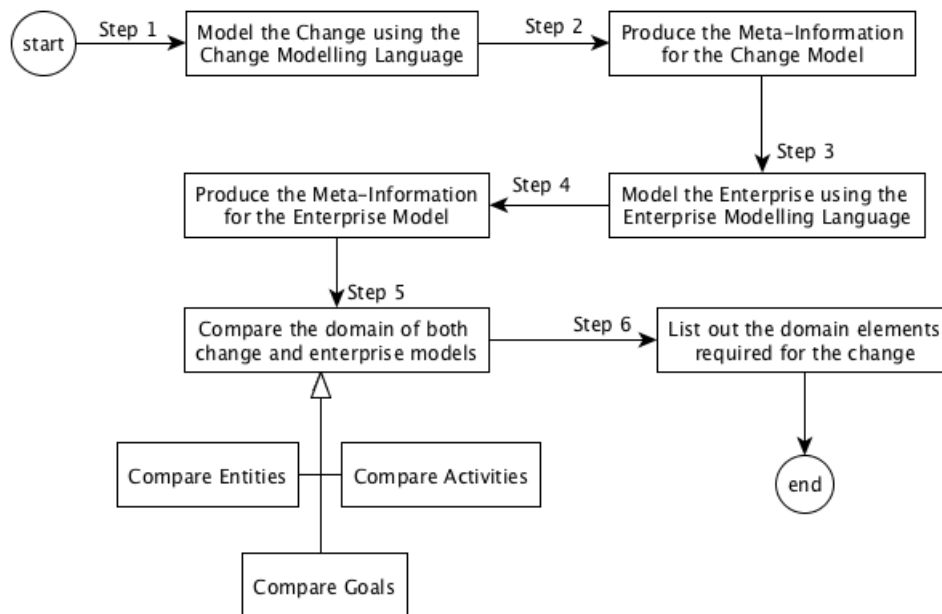


FIGURE 7.4: Procedures For Deriving Domain Elements

The second step is to create the meta-information for the change models in *Case 1* and *Case 2*. Meta-information is a table of three columns which contains the three

RefEntity	Activity	Goal
Applicant	enter applicant details	R, S1
Oversea	submit testScore	S2, S3
Home	get schRank	∅

RefEntity	Activity	Goal
RFP	n/a	Sd
Bid	n/a	R, Sa
Client	n/a	n/a
Old Client	n/a	Sb, Se, Sf
Global Client	n/a	Sc, Sd

(A) Meta-Information for Case 1

(B) Meta-Information for Case 2

FIGURE 7.5: Meta-Information for the 2 Change Models in the Case Study

core domain elements to be compared. See Chapter 6 for more details and examples about meta-information. The meta-information for the two change models in the case study is shown in Figure 7.5. The meta-information for *Case 1* is shown in Figure 7.5a, while that of *Case 2* is shown in Figure 7.5b.

As shown in Figure 7.4 the third step is to build the enterprise model using the proposed change modelling language. In order to do this, the the elements of the three core domains of an enterprise *data, business process, and goals* have to be identified and modelled. Existing enterprise models, particularly the data, business process, and goal models that concerns a change under consideration can be re-used. Assuming no such models exist, or if it is difficult to re-use existing models, then a new enterprise model (data, goal and business process models) can be developed. An enterprise model can be developed by analysing and studying a given enterprise to identify its objectives (goals), the information (data) it can use to achieve those goals, and the process/activities (business process) for achieving the goals. These can be done through various ways such as interviewing stakeholders, brainstorming, focus groups, etc.

In the case of this research, the collaborating enterprise has provided a case study that contain the necessary information required to build an enterprise model. Hence in order to identify the goals, data, and business process, the author examined description of the enterprise as provided in the case study in Section ???. First, the enterprise goals are identified and modelled using goal modelling techniques. As

shown in Figure 7.6, these goals are then decomposed subgoals, and some conditions are attached to the atomic goals. Each goal is also annotated with some alphabetical or alphanumeric symbols. These annotations would be useful when building Meta-information for the enterprise model and also during the comparison of enterprise and the change models.

After identifying and modelling the goals, the data (entities and attributes) required to operationalise or achieve the desired goals are specified and modelled. This can be done by examining the enterprise, described in Section ??, to identify the key business entities and their attributes. These can then be integrated into a conceptual data model using data modelling techniques. Figure 7.7 shows the relevant data model for the enterprise described in the case study.

Enterprise models should include some processes (business processes) or set of coherent activities for actualising enterprise goals. Two important processes can be identified from the enterprise description in Section ?. These are the bidding and recruitment processes. The bidding business process describes a set of consistent activities to be followed in order to actualise bidding goals. Similarly, the recruitment business process shows the activities for achieving the enterprise recruitment goals. Both processes are shown in Figure 7.8.

The next step in deriving the domain elements, as summarised in Figure 7.4, is to construct the Meta-information for the enterprise model. This is shown in Figure 7.9. The entire enterprise model is shown in Figure 7.10, this includes the goals, data, business processes, and the enterprise meta-information.

As shown in Figure 7.4, the next step after constructing the change model, enterprise model and the meta-information is to compare both models. This can be achieved using the rules proposed in Chapter 6. As discussed earlier, the essence of this comparison is to derive the set of enterprise domain elements that need to be modified in order to adapt a given change. The following sections demonstrate how these rules can be applied to compare and derive enterprise domain elements.

7.4.1 Applying Rule 1 to Derive Data Elements

Rule 1 can be used to compare entities of change model with those in enterprise model. So as to derive entities or data elements that are affected by a change and hence should be added to the TO-BE enterprise model to adapt that change. This rule states that: *If an entity exist in the change model, but similar entity do not exist in the corresponding enterprise model, then that entity is required to adapt to the change (represented by the change model).* This implies that a user should check through

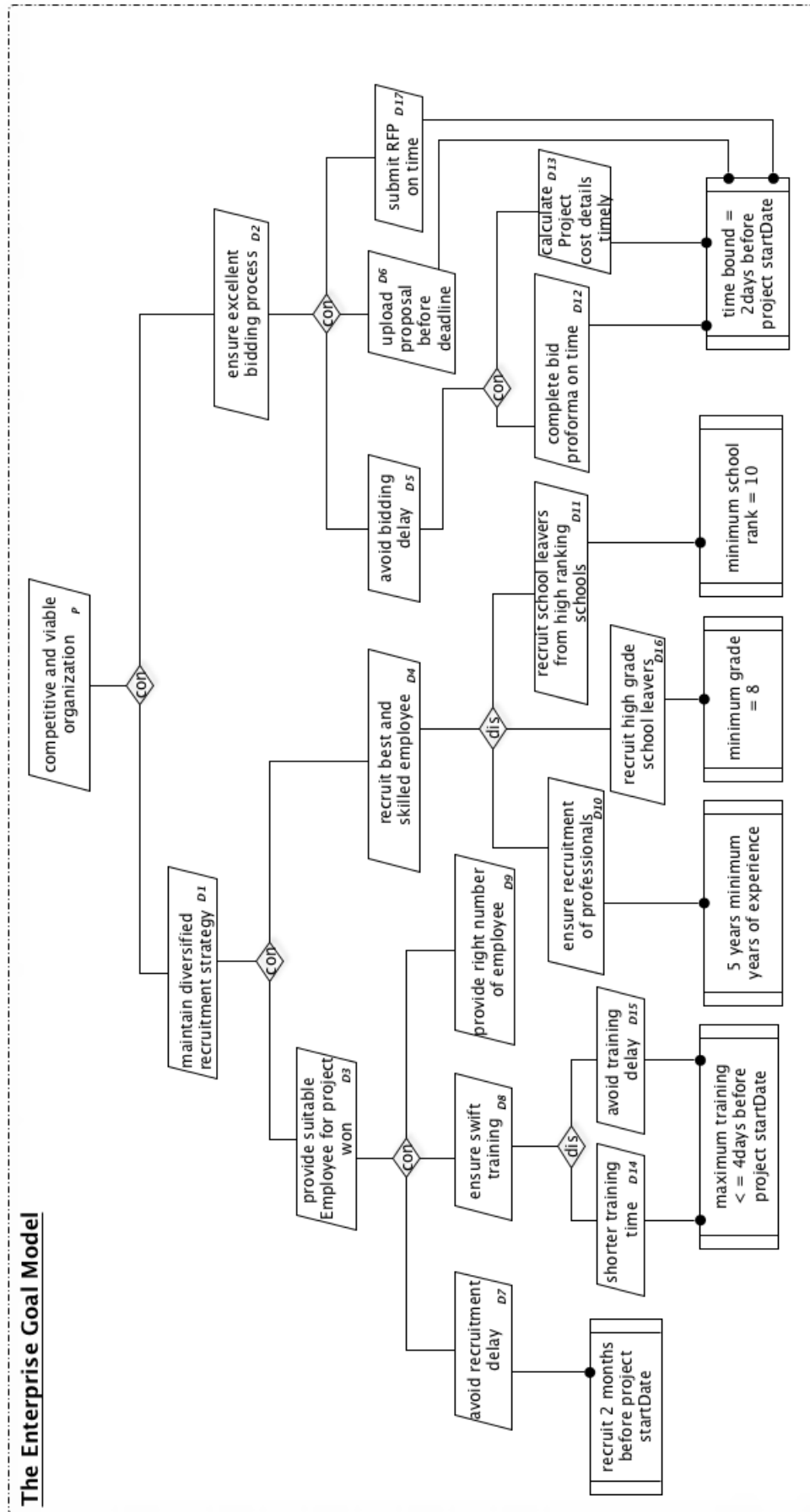


FIGURE 7.6: Case Study Enterprise Goal Model (Enlarged in Appendix D)

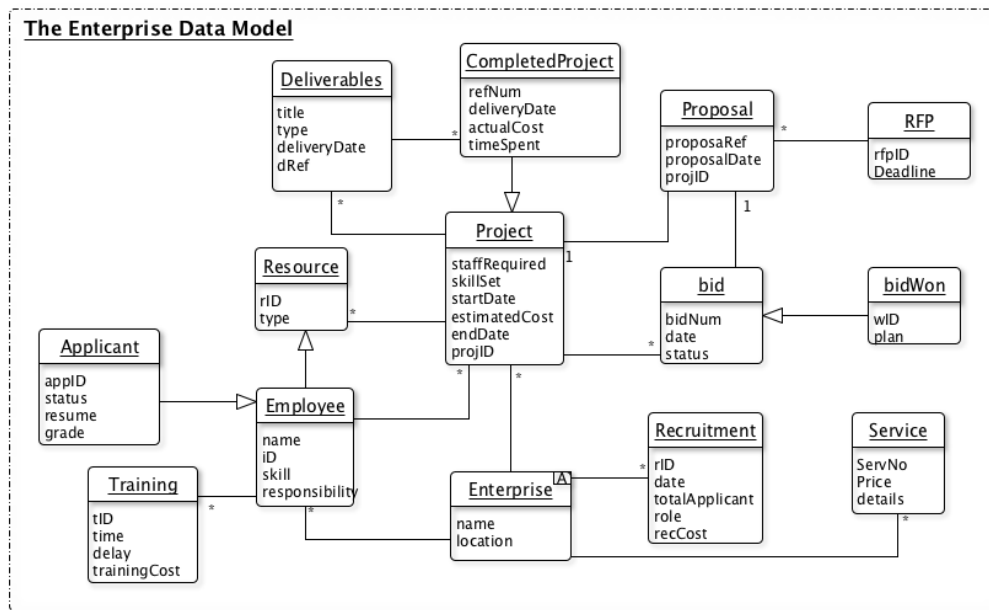


FIGURE 7.7: Case Study Enterprise Data Model

both enterprise and change model to look out for entities that are in change model but not in enterprise model. To ease this comparison, either the models or their meta-information can be used for this comparison.

Figure 7.11a and 7.11b respectively show the meta-information of the enterprise and change model for *Case 1*. The first entity in the change model is "Applicant" which relates to the "enter applicant details" activity. Since this same entity exists in the enterprise model, relates to the same activity and have same attributes as in the change model. It does not satisfy rule 1, and hence not affected by the change and need not be added to the TO-BE enterprise model to adapt to the change in *Case 1*.

On the other hand, the "Oversea" and "Home" entities are in the change model but do not exist in the enterprise model. Hence, they satisfy rule 1, and are required to adapt to the change. In other words, to adapt to the change in *Case 1*, the data domain of the enterprise should be modified by adding these two entities. In this way, certain information such as "testScore", "schRank", and "nationality" required to adapt to the change can be captured in the enterprise model. Observe that these information are not available in the AS-IS enterprise Data model shown in Figure 7.7 and Figure 7.10. Thus in order to adapt this change they should be

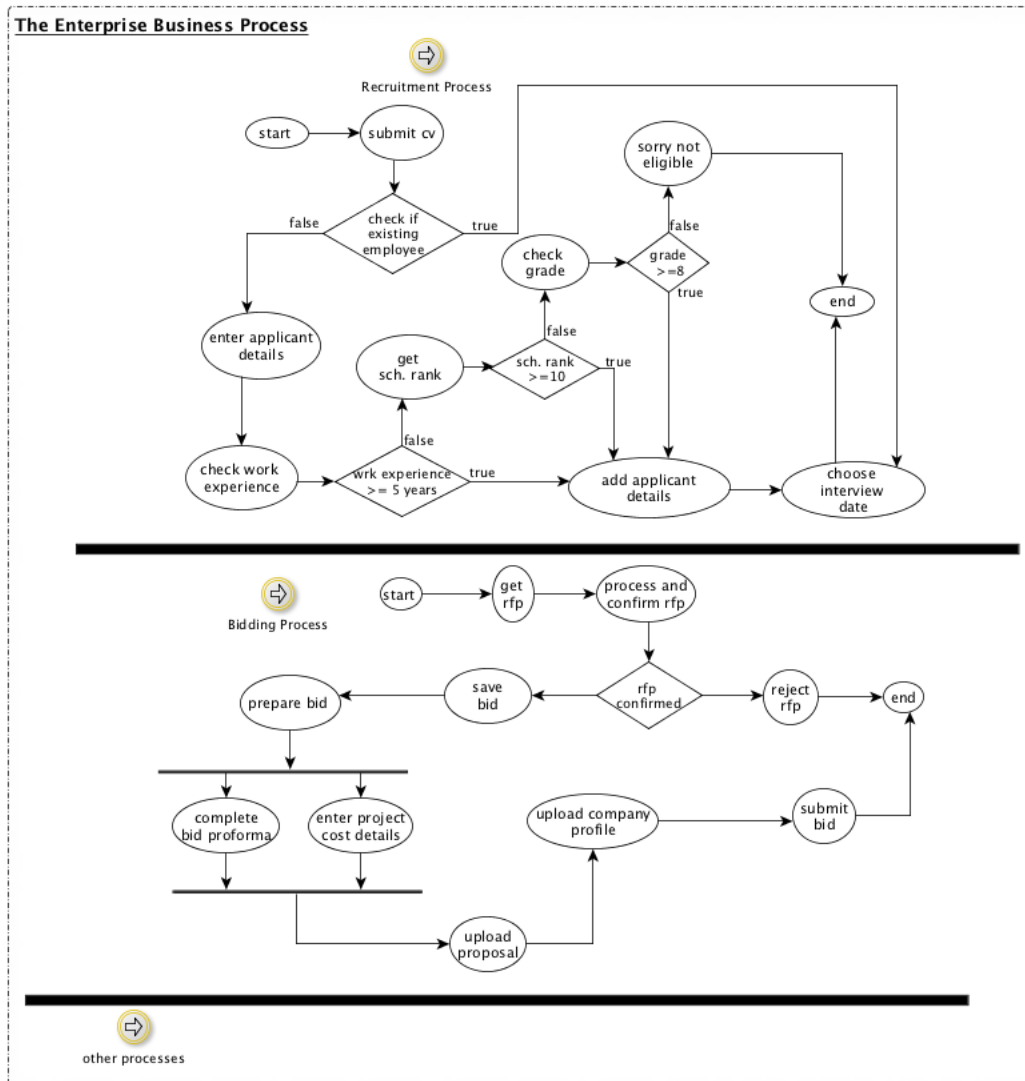


FIGURE 7.8: Case Study Enterprise Business Process Model

Enterprise Meta-Information		
RefEntity	Activity	Goal
Completed Project	n/a	n/a
Proposal	upload Proposal	D6
Bid	save bid prepare bid	D2, D5, D12
Project	enter project cost details	D13
Employee	check if existing employee	D3, D4, D9
Applicant	enter applicant details	D11, D16
Training	n/a	D8, D14, D15
Recruitment	∅	D1, D7, D10, D11, D16
Rfp	get rfp	D17

FIGURE 7.9: Case Study Enterprise Meta Information

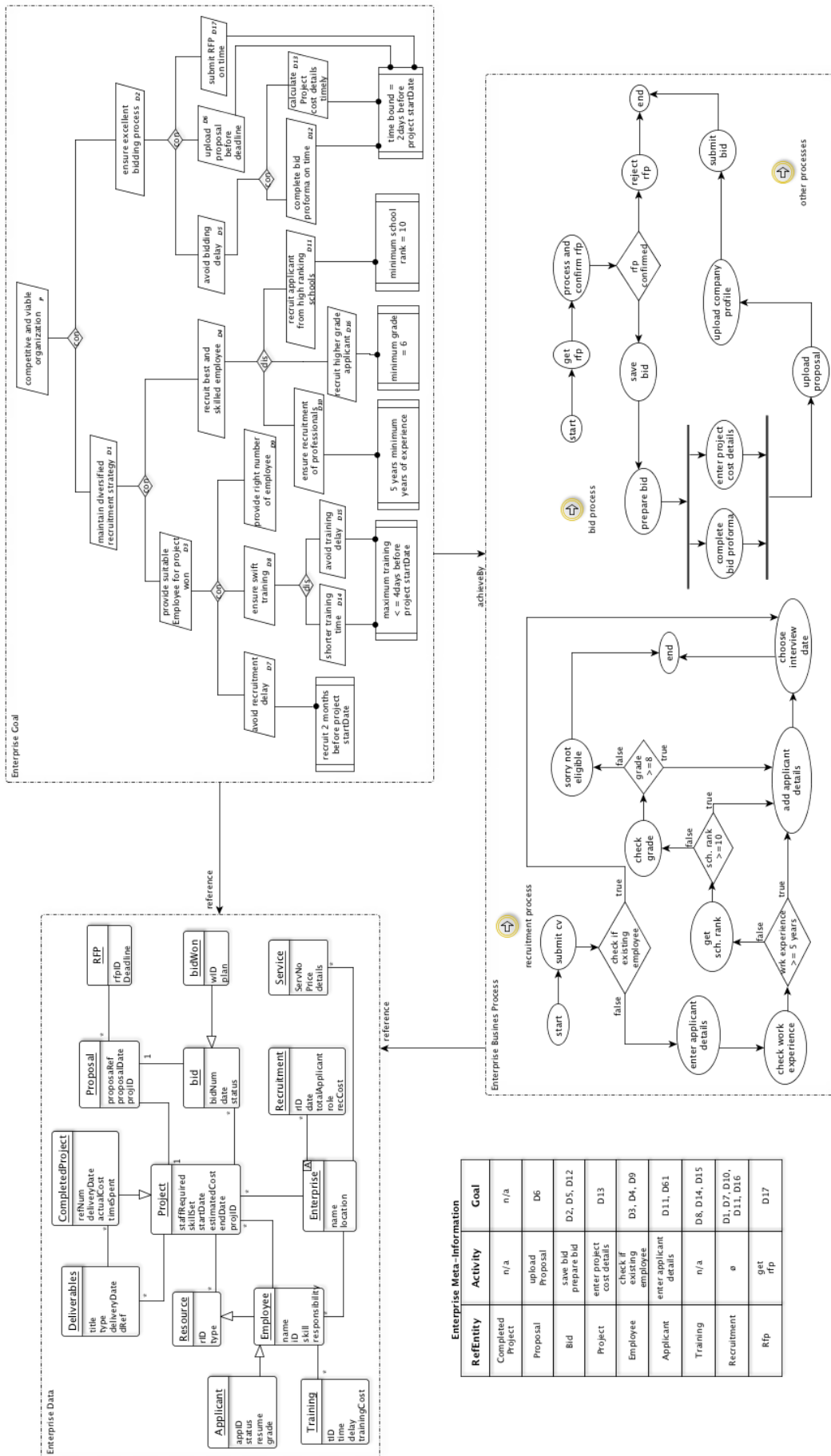


FIGURE 7.10: Case Study Enterprise Model (Enlarged in Appendix D)

Enterprise Meta-Information		
RefEntity	Activity	Goal
Completed Project	n/a	n/a
Proposal	upload Proposal	D6
Bid	save bid prepare bid	D2, D5, D12
Project	enter project cost details	D13
Employee	check if existing employee	D3, D4, D9
Applicant	enter applicant details	D11, D16
Training	n/a	D8, D14, D15
Recruitment	∅	D1, D7, D10, D11, D16
Rfp	get rfp	D17

Meta-Information For Case 1		
RefEntity	Activity	Goal
Applicant	enter applicant details	R, S1
Oversea	submit testScore	S2, S3
Home	get schRank	∅

(A) Enterprise Meta-Information (B) Meta-Information for Case 1

FIGURE 7.11: Comparing Enterprise and Change Entities For Case 1 using Meta-Information

added to the TO-BE enterprise change.

Enterprise Meta-Information

RefEntity	Activity	Goal
Completed Project	n/a	n/a
Proposal	upload Proposal	D6
Bid	save bid prepare bid	D2, D5, D12
Project	enter project cost details	D13
Employee	check if existing employee	D3, D4, D9
Applicant	enter applicant details	D11, D16
Training	n/a	D8, D14, D15
Recruitment	∅	D1, D7, D10, D11, D16
Rfp	get rfp	D17

(A) Enterprise Meta-Information

Meta-Information For Case 2

RefEntity	Activity	Goal
RFP	n/a	Sd
Bid	n/a	R, Sa
Client	n/a	n/a
Old Client	n/a	Sb, Se, Sf
Global Client	n/a	Sc, Sd

(B) Meta-Information for Case 2

FIGURE 7.12: Comparing Enterprise and Change Entities For Case 2 using Meta-Information

Rule 1 can also be applied in *Case 2* to derive the data elements or entities that are required to adapt the change. Figures 7.12a and 7.12b show, respectively, the meta-information of the enterprise and change model for *Case 2*. Notice that the activities and processes for *Case 2* are not the typical business process activities, thus *n/a* (not applicable) is written in the 'activity' column of the meta-information. In such cases, the attributes can be used to check if two entities are the similar. Even though the "RFP" and "Bid" entities exist in both enterprise and change model of *Case 2*, they contain some different set of attributes. They satisfy rule 1 and are hence required to adapt the change. The "Client", "Old Client", and "Global Client" entities are in change model but do not exist in the enterprise model, hence they satisfy rule 1 and are required to adapt the change. These entities should then be added to the TO-BE enterprise model in order to adapt to the change.

7.4.2 Applying Rule 2 to Derive Activities

Rule 2 can be used to compare activities of change model with those of enterprise model. So as to derive activities that are affected by a change and hence should be modified to adapt that change. This rule states that: *If an activity exist in the change model, but similar activity does not exist in the corresponding enterprise model, then that activity is required to adapt to the change (represented by the change model).* This implies that a user should check through both enterprise and change model to look out for activities that exist in the change model but not in enterprise model. These activities can be business process activities or any other logical activities as in *Case 2*, see Figure 7.2.

Figure 7.11 shows the meta-information of enterprise and change models for *Case 1*, which can be used to facilitate this comparison. The activity "enter applicant details" exists in the change model and also exists in the enterprise model. It implies that this activity does not satisfy rule 2. Hence it is not required and should not be added to the TO-BE enterprise model to adapt the change. But the other two activities i.e., "submit testScore" and "get schRank" exist in the change model but do not exist in the enterprise model, which implies that they satisfy rule 2. Therefore they are required and should be added to the TO-BE enterprise model so as to adapt to the change.

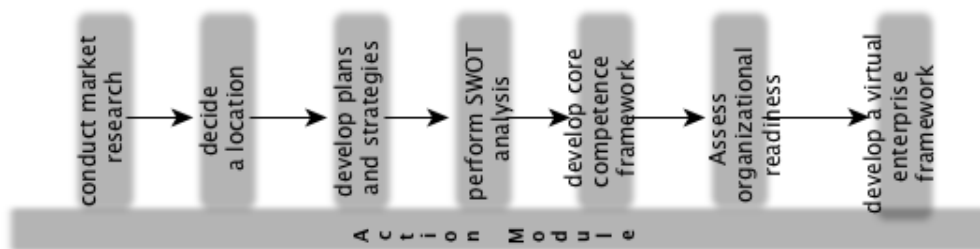


FIGURE 7.13: The Set of Activities for Case 2

This rule can also be applied in *Case 2* to derive a set of activities that are required to adapt the given change. The activities for *Case 2* are presented in Figure 7.13, these are obtained from the extreme right of the change model shown in Figure 7.2. Since these activities are not the typical business process activities and do not necessarily reference data entities. They may not be listed in the meta-information, so the cells in activity column of the meta-information for *Case 2*, shown in Figure 7.12b, are

marked with 'n/a' (not applicable). Clearly, these activities exist in the change model but do not exist in the enterprise model, they satisfy rule 2. Thus, they are required and should be added to the TO-BE enterprise model.

7.4.3 Applying Rule 3 to Derive Goals

The next enterprise domain element to derive, after deriving entities and activities, is the goal element. Rule 3 can be used to compare the goals in a change model with those in the enterprise model. This rule states that: *If a goal exist in the change model, but similar goal does not exist in the corresponding enterprise model, then that goal is required to adapt to the change (represented by the change model).* This rule can be applied to derive the goals that are needed to adapt to the change in *Case 1*. The meta-information in Figure 7.11 can be used to this end. Goals *R* and *S1* of the change model can be compared to goals *D11*, and *D16* of the enterprise model, since they reference the same entity *i.e.*, *applicant*. However, a closer examination reveals that *R* and *S1* relates to a specialised type of applicant *i.e.*, 'oversea applicant', and are thus not the same as *D11* and *D16*. Therefore, goals *R* and *S1* satisfies rule 3 and should be added to the TO-BE enterprise model in order to adapt to the change. This is also the same with the other goals in the change mode of *Case 1*, *i.e.*, *S2* and *S3*, since they do not exist in the enterprise model.

This rule can also be applied to *Case 2* using the meta-information shown in Figure 7.12. One can start by identifying the goals that should be compared. This can be done by looking at goals that references the same entities in both models. Goals *Sd*, *R*, and *Sa* of the change model can be compared with goals *D17*, *D2*, *D5*, and *D12* of the enterprise model, since they reference the same entities *i.e.*, *RFP*, and *Bid*. But a closer examination of the individual goals as shown in Figure 7.2 and Figure 7.6 shows that these goals are not the same. This also applies to the other goals in the change model of *Case 2*, since they do not exist in the enterprise model. Therefore all the goals in the change model of *Case 2* satisfies the rule 3, and hence should be added to the TO-BE enterprise model in order to adapt to the change.

7.5 Chapter Summary

This chapter demonstrates utility of the process (framework) proposed in this thesis, and shows how an enterprise model can be compared with a change model to derive the enterprise domain elements required to adapt to a given change. As mentioned earlier, the design science research (DSR) suggests that one way to

demonstrate utility of research contributions in Information Systems is by using them (contributions) to represent an aspect of a real world. Scenarios or case studies can be used to describe an aspect of a real world. Accordingly, utility of this research contributions has been demonstrated in this Chapter using a case study obtained from the Research Group's Industry Partners.

The three applicable areas of this research contributions have been discussed in Sections 7.2, 7.3, and 7.4. Results show that the proposed change modelling language can be used to represent, structure, and visualise enterprise changes. One essence of this representation is to understand and derive the domains of an enterprise required to adapt a given change. In other words, modelling enterprise changes and relating them to enterprise models is useful for discerning the goals, business process activities, and business entities (Data) that should be included in the target (TO-BE) enterprise model, so as to effectively respond to a given change.

In addition, the proposed change modelling language has been used to implement a Change Management Proforma to support enterprises in capturing, documenting, and analysing change management activities. Another important application of the proposed change modelling language has been discussed in Appendix E, which shows how it can be used to implement a database system to support enterprise change management initiatives. The next chapter brings this Thesis into conclusion, reflects on the research and discusses its limitations, as well as proposes further work to be carried out.

Chapter 8

Reflection and Conclusion

8.1 Chapter Overview

The overall aim of this research is to make contributions and extend what the Information Systems community already know about enterprise agility and change management. The key contribution here is a process/framework for representing an enterprise change and deriving the enterprise domain elements required to adapt the given change. This extends the body of knowledge by providing a novel modelling language for representing enterprise changes; a concise enterprise modelling language for describing an enterprise in terms of its goals, business processes, and data; and a set of procedures together with rules for comparing a change model with an enterprise model and deriving the set of domain elements required to adapt a given change. In addition, Appendix E shows how the proposed change conceptual model can be used to implement a database. This database could support the management of data and relevant information relating enterprise agility and change management.

In order to measure the success of these contributions and determine whether this research has achieved its aim, it is helpful to reflect on how well this research meets its objectives, as well as how the evaluation and validation criteria, stated in Chapter 1 of this thesis, have been satisfied. It is also useful to examine how the research question has been answered. Already, the research objectives and question have been stated in Sections 1.3 and 1.2 respectively, while the evaluation and validation criteria have been stated in Section 1.7, of the first chapter.

The remaining part of this chapter is organised as follows: Section 8.2 presents a discussion on how the research question is answered, while Section 8.3 reflects on how well the research objectives have been met. This is followed by Section 8.4 that discusses how the contributions met the evaluation and validation criteria.

Section 8.5 reflect on some lessons learned from applying the research contributions to industry case study. Then, Section 8.6 discusses some limitations of the research contributions. These limitations provides a scope for future work, which is discussed in Section 8.7. Finally Section 8.8 gives a general concluding remarks and summary of the research.

8.2 Reflecting on the Research Question

As stated in Chapter 1, this research aims to answer the following question:

What is the process for deriving the set of new enterprise domain elements required to adapt a given enterprise change?

To answer this question, this research develops a process or framework for relating and comparing a change to the enterprise it affects, and determining the goals, business process activities, and data entities required to adapt the given change.

This framework includes the following:

- i A conceptual structure as described in Figure 4.1. This conceptual structure includes a unique conceptual model for expressing enterprise changes, and a conceptual language for modelling an enterprise.
- ii Procedures for applying this conceptual structure in deriving the required domain elements. These procedures have been summarised as a stepwise approach in Figure 6.4.
- iii A set of rules for comparing a change model with an enterprise model. These rules are presented in Expressions F.1, F.2, and F.3

To demonstrate its utility, the developed framework is experimented on a real world industry case study. The results of this experiment produce a set of data entity, business process activities, and business goals, required to adapt the two given changes in the case study. Since this framework can be applicable in real world industry, it is considered successful in accordance to the well established guidelines for DSR proposed by Hevner et al [89, 90], and hence provide the answer to the research question.

8.3 Reflecting on the Research Objectives

The objectives followed to achieve this research aim and contributions are as follows: (a) To identify the minimal, necessary, and sufficient set of concepts required to represent enterprise changes, through a systematic literature review. (b) To design and develop a conceptual language for modelling enterprise changes, using the concepts identified from the systematic literature review. (c) To design and develop a concise conceptual language for modelling an enterprise, re-using concepts from existing enterprise models and architectures. (d) To provide a set of rules and procedures for deriving new domain requirements when changes occur in an enterprise. (e) To demonstrate the utility of the proposed process or framework using an industry case study. The sections below reflect on how each of these objectives are met.

8.3.1 Identification of Concepts

The first research objective is to identify the necessary and sufficient set of concepts required to model an enterprise change. In order to meet this objective, a systematic literature review (SLR) was conducted and discussed in Chapter 2 of this thesis. SLR provides a rigorous and reliable methodology for meeting stated research objective(s), solving given research problem(s), and answering research question(s) [107]. Based on these, SLR is selected over other literature review methods, such as systematic mapping study (SMS) and traditional literature review.

Another reason for selecting SLR is to ensure that a wider range of publications are systematically selected and reviewed, so that important concepts would not be left out. The concepts of enterprise agility and change management, required to model enterprise changes, have been presented in Table 2.5 and discussed in Section 2.7.1. Hence the first objective of this research is considered to be met.

8.3.2 Design and Develop Conceptual Model for Change

The second objective of this research is to design and develop a conceptual modelling language for enterprise change. This objective is met by using the concepts identified from the SLR to develop a change modelling language, see Chapter 5. To ensure that the modelling language is of a good quality and standard, a rigorous approach has been followed in its development. This approach involves using

existing meta-modelling techniques such as developing the abstract syntax, concrete syntax, and semantics for the proposed language. In addition, the modelling language is designed to conform to existing meta-modelling technologies such as the meta-object facility (MOF), UML, and the eclipse modelling framework (EMF).

8.3.3 Concise Enterprise Modelling Language

This objective has been met in Chapter 6, which presents a concise modelling language for representing an enterprise. The concepts used for this enterprise model are derived from existing enterprise modelling approaches and frameworks, which are also identified from the SLR. Similar to the change modelling language, good quality of this enterprise modelling language is ensured by applying existing techniques and technologies for developing conceptual modelling languages. For instance, MOF and UML concepts were re-used to develop the language. In addition, both the change and enterprise modelling languages are developed using the design science research (DSR) methodology, see Chapter 4. The DSR provides a rigorous approach to develop Information Systems contributions and artefacts, so as to ensure that they are of good quality as well as usable.

8.3.4 Rules and Procedures

This objective seeks to provide the rules and procedures that can be used to derive the set of domain requirements, *i.e.*, data, business process and goals, for adapting to a given change. To meet this objective, Chapter 6 of this thesis provides three rules and a stepwise procedure that can be used to relate and compare change and enterprise models. This procedure includes a *meta-information* which provides a consistent means to structure and compare model elements in both change and enterprise models. In addition, to ensure they can be used by a wide range of audience, these rules are also formalised using formal logic. To show that these rules and procedures can be useful, they have been tested with an industry case study, see Chapter 7.

8.3.5 Demonstrate Utility

The last objective is to demonstrate the utility of the proposed framework. To meet this objective, a real industry case study was obtained from the Research Group's industry collaborator, this case study has been reported in Sections 2.6 of

Chapter 7. The proposed change modelling language was used to construct the models of the two changes in the case study. Similarly, the proposed enterprise modelling language was used to construct the model of the enterprise. These show that both languages can be utilised in a real world enterprise. Afterwards, the rules and procedures were applied to derive the domain elements that are required to adapt changes. Hence this objective has been met.

8.4 Evaluation and Validation

Recall that in Chapter 1, some criteria for validating and evaluating the contributions of this research have been identified and stated. These include the following: (i) Represent/model the enterprise changes described in the case study. (ii) Represent/model the enterprise described in the case study. (iii) Derive new domain elements required to adapt to the change described in the case study. This section revisits these criteria and discusses how they have been satisfied by the research contributions through the experiments conducted with a case study in Chapter 7.

In Chapter 7 of this research, an industry case study is described and used to demonstrate the utility of the research contributions. This is divided into 3 parts which corresponds to the three key areas where the utility of this research contributions is demonstrated. In the first part or part 1 of Chapter 7, the proposed change modelling language is used to model or represent enterprise changes. Two enterprise changes caused by different change drivers, captured in an industry case study, are modelled using the proposed change modelling language. As discussed in Section 7.2, the proposed change modelling language sufficiently represents the changes, and features of these changes, described in the case study. It is therefore considered to satisfy the first evaluation and validation criteria.

Equally, part 3 of Chapter 7 shows how the proposed enterprise modelling language is used to represent or model the enterprise described in the case study. Additionally, part 3 shows how the proposed rules and procedures are applied to derive a set of new domain elements required to adapt the changes in the changes described in the case study. The experiment and discussion in Section 7.4 shows that the enterprise modelling language can be used to model an enterprise. It also shows that the rules and procedures can be used to derive the new enterprise domain elements required to adapt changes. Therefore, the second and third criteria for evaluation and validation have been met. Overall, the above discussions show

that the contributions of this research satisfy the afore stated evaluation and validation criteria. Hence, they are considered valid and can be used by enterprises that which to support their ability to manage changes.

8.5 Discussion of Lessons

During the application of the proposed framework on industry case study, some vital lessons are learned. It would be useful to present and discuss these lessons. The aim of this section is to present a discussion of the lessons learned by applying the proposed framework to model an enterprise, its changes, and derive the domain elements required to adapt those changes.

8.5.1 Lesson From Conceptual Modelling

By modelling a change driver, it is easier to clarify and express tacit and ambiguous information or knowledge about an enterprise change. These enhance the understanding and knowledge about the root cause or the origin of an enterprise change. As mentioned in part 1 of Chapter 7, these knowledge and understanding helped to reason about the various modifications that can be made to the current state of the enterprise, and the type of transition the enterprise should make in order to respond to a given change. In addition, a good representation, understanding, and knowledge of change drivers and change provide a means of identifying and deciding on the adequate actions for adapting a given a change, thus supporting enterprise agility and change management initiatives. In other words, if the change driver and change are clearly understood, represented, and known, then it is easier to formulate actions to adapt them.

The conceptual representation of an enterprise in terms of its goal, business process, and data, makes it easier to relate or compare an enterprise model with a change model. This type of comparison facilitates the derivation of new enterprise domain elements required to adapt the given changes. However, this comparison currently involves no form of automation. Instead it relies on the ability of the modeller to manually construct meta-information tables, and use these tables to carefully compare the elements of the enterprise model with those of the change model. Hence, enterprises modellers or change managers using this approach require extra care so as to ensure that there are minimal or no errors, and the resulting derived domain elements are precise as well as accurate.

Expressing a change as a goal, made it easier to decompose the change into achievable objectives. By so doing, it is easier to understand and communicate the requirements for adapting to a given change. For instance in Figure 7.1, the change is represented, broken down in subgoals (objectives) and finally attached to a condition. Also this helps to convey further knowledge about what tightening the recruitment process for foreign applicants would mean to the enterprise. In concrete terms, as shown in Figure 7.1 it means that the said enterprise should accept a minimum grade of 8 or pass score of 60% for foreign applicants.

The rules and procedures for deriving the set of enterprise domain elements, required to adapt changes, are easy to understand, and simple to apply. Therefore they may be easily used and adopted by a wide range of audience, particularly those who are new to conceptual modelling or those that have little or no experience of programming. The latter may apply to the majority of top enterprise change managers. At the same time, since these rules and procedures are described in natural language, it might be a little difficult to transmute them into codes or executable programs. However, this difficulty may not present a big problem given that most change management and enterprise agility initiatives and decisions are usually carried out by human rather than software programs.

Identifying and modelling the change indicator is a helpful step in change management and enterprise agility processes. It helps to clarify as well as enhance understanding of the environment issues that led or can lead to a change. In this way, stakeholders can be in constant alert and monitor the environment for the (re-)occurrence of similar issues in the future. If similar or any of such environmental issue becomes predominant/popular in a the future, stakeholders can capture such an issue and set the threat level to severe. Then they can start anticipating the type of change driver and change such captured change indicator can cause, as well as the impact the change can have. More so, by specifying and representing the change impact, as shown in Figures 7.1 and 7.2, it becomes easier to visualise, know, and understand the implications or potential consequences of a given change to enterprise. Such knowledge and understanding may be useful in performing change impact analysis and/or change risk assessment.

Actions, which specify what should be done to respond to a given change driver and remain agile, are clearly expressed and assigned to relevant agents or stakeholders. By so doing, other stakeholders can focus on many other enterprise activities that require their attention. This can help to minimize duplication of efforts and reduce unnecessary repetition (redundancy) of duties, as well as support the optimization of resources, cost and time spend during change management initiatives. Additionally, it can save a considerable amount of business time by reducing

ambiguity and making it easier for stakeholders to understand what they are required to do for their enterprises to improve agility.

8.6 Limitations

This section discusses some limitations of this research. Each limitation is stated and discussed in the paragraphs below.

No Automation: Although automation and software tools may not be mandatory prerequisites for supporting enterprise agility and change management. They can be useful and helpful to enterprise agility and change management initiatives, particularly in larger organizations and in a more dynamic business environment where changes are more frequent. As mentioned earlier, in its current state, the approach proposed by this research involves no form of automation and software tools. Instead it is limited to manual approach, in other words, users have to manually analyse the case study, represent the change and enterprise models, construct the meta-information tables for both models, and apply the rules to derive the new domain elements required to adapt the given changes. This manual approach may be cumbersome, boring, and can involve some errors, as most humans activities usually involve some percentage errors. Although the application of this approach to the case study in Chapter 7 proves successful with little or no errors. It can be possible to experience errors while modelling larger enterprises with a large quantity of data.

Validation and Evaluation Limitation: The case study method of evaluation, used in this research, is usually acceptable and have proved successful in validating and evaluating Information Systems (IS) contributions and artefacts [4, 89, 90]. However, there are certain limitations associated with the use of case study method of evaluation and validation. One of such limitations is the challenge of generalising the results or findings from a case study [45, 94]. The contributions of this research have been validated and evaluated against a case study collected from a single enterprise in an IT industry. Even though these contributions meet the stated criteria and are considered to be validated and evaluated, they have not been tested on enterprises in other industries. More so, since this case study is from a single enterprise in a particular industry, it may not be a representative of a wider range of enterprises. Based on these, it can be challenging to generalise these contributions on all industries.

Another limitation is that some case studies can be non-numerical and non-statistical [45, 94]. The case study used to validate this research contributions are not based on numerical or statistical figures or data. Instead, it is based on textual descriptions which can subject it to the interpretation, analysis, knowledge, and expertise of the researcher [45, 94]. In other words, various researchers can interpret this case study in various ways, and may arrive at different understanding of the results of the experiments conducted with the case study. This can introduce some elements of bias and raise questions or doubts about the precision or accuracy of the results obtained from experimenting with the research contributions. Besides, it can also make it difficult to quantify the results of the experiments conducted with the case study.

Furthermore, the experiments conducted with this case study are not based on empirical method, thus the research contributions have not been empirical validated and evaluated. In other words, other IS practitioners have not used and interacted with this research contributions. This can make it difficult to exactly ascertain how easy, simple, usable, and supportive the proposed contributions can be in supporting enterprise agility and change management. Although the method of validation and evaluation, used here, shows that the contributions of this research can be applicable in industry. Empirical validation and evaluation can further demonstrate utility and strengthen the argument about the industry applicability of this research contributions. However, empirical validations are beyond the scope of this research.

Enterprise Model Limitation: The enterprise modelling language proposed in this research considers an enterprise as an embodiment of its business objectives (goals), processes for achieving these objectives (business process), and the information required to achieve the objectives (data). Even though this modelling language is sufficient to capture an enterprise in a concise manner and is thus deemed to be a useful contribution. Most enterprises may want to consider other domains or viewpoints such as technology domain, and application domain. These other domains are not included in the proposed enterprise modelling language. Hence the proposed enterprise modelling language is limited to enterprises who desire a concise language to represent their goals, business process, and data. However, enterprises can overcome this limitation by applying model driven approaches to extend the proposed enterprise modelling language with any number of relevant domains.

8.7 Future Work

The research limitations presented in Section 8.6 above provide some scope to further improve the contributions of this research. There are three key areas in which the contributions of this research can be improved. The first is to carry out empirical evaluation and validation of the research contributions. This would involve some activities such as getting an enterprise to adopt or adapt the proposed contribution as its change management framework. The next activity would be to train and supervise the stakeholders of this enterprise on how to apply the proposed framework on their change management activities. Other activities include creating a feedback mechanism and observing the stakeholders as they apply this framework to manage changes. Finally, I would use these feedbacks for further enhancement and also to obtain a better version of the proposed framework.

The second area of improvement is in tooling or automation. Currently, the proposed conceptual modelling languages are not executable and a software tool has not been developed to support this framework and approach. It would be possible to partially or fully automate enterprise change management initiatives or implement a software tool to support enterprise agility. For instance model driven development approaches, such as model to model transformation or model integration, can be applied to implement a software tool that can automatically generate a change model given some parameters. Similarly a software tool can be implemented extract goals, business process activities, and data elements from both change and enterprise models, and thus automatically building the meta-information tables required to compare both models. These would be the focus of the future work for this research. Hence future work would focus on applying model driven development approaches to implement software tools to support enterprise agility and change management initiatives.

8.8 Concluding Remarks

It has been well acknowledged that uncertainties and spontaneous changes pose serious threats, problems, and challenges to modern day enterprises. Threats and problems can include loss of competitive advantage and market share, bankruptcy and liquidation, financial or profit loss, among others. Hence, effective change management is required for enterprises that desire to survive, as well as overcome these threats, problems, and challenges. Enterprise agility, *i.e.*, the ability of enterprises to respond to changes timely and effectively, is a core imperative for

change management and survival in the face of changes and uncertainties. But, at the same time, enterprise agility is too difficult to achieve and a major concern for business and IT executives. For this reason, enterprises are in search of techniques, methods and frameworks to support, enhance, or achieve enterprise agility.

The definition of enterprise agility clearly suggests that the ability to adapt to change is critical to achieving enterprise agility. A clear understanding and knowledge of enterprise changes can facilitate and enhance an enterprise to acquire the ability to adapt changes. Such understanding and knowledge can be achieved if changes together with its essential features and related concepts are expressed and represented explicitly. Therefore, in order to acquire the ability to adapt changes and thus facilitate enterprise agility, it is imperative to provide a means of expressing and modelling any given enterprise change, its essential features and related concepts in a coherent and understandable format. Conceptual modelling languages are known to provide the basis for describing, expressing, and modelling an aspect of the physical world, such as enterprise changes and agility, in a coherent and understandable format. Therefore Conceptual modelling language techniques can be useful to these ends.

In order to support enterprise agility and change management, this research proposes a conceptual framework to support the representing and modelling of enterprises together with the changes they can face. In addition, this framework includes a set of procedures and rules that are used to derive the new domain elements required to adapt changes. An industry case study has been used to test the utility of this framework. The results obtained from this case study show that this framework supports enterprise agility and change management in a number of ways. It provides an approach to represent an enterprise and its changes. Such representation proves to enhance the understanding of enterprise changes, and how changes can relate to an enterprise. In addition it provides a means to derive a set of new enterprise elements required to adapt a given change. Furthermore, the change conceptual modelling language has been used as the basis for developing a database system that can be used to capture, store and manage data and information relating to enterprise agility. Other benefits of the proposed conceptual model are discussed in the paragraphs below.

Conceptual modelling is known to support the capturing and sharing of information about a problem domain. Enterprise changes is a well known problem domain. Therefore, the change modelling language proposed in this research can be used to capture and share vital information about enterprise changes. Information sharing can help enterprise stakeholders to understand a problem domain. Such

understanding can motivate concerted efforts towards proffering a lasting solution to the problem. For instance, as mentioned in Section 8.5, the *change driver* model concept, included in the proposed change modelling language, is used to capture, understand, and share information about the origin and root cause of a change. These help to reason about the actions that can be suitable to adapt the change driver.

Furthermore, conceptual modelling languages are core prerequisites for information systems development, as they are used to capture and represent concepts required to build database and software systems [149, 169]. The conceptual modelling language proposed in this research is developed by gathering the essential features and concepts of change and integrating them into a meta-model. These concepts and features of change can be used to implement a database to support enterprise agility and change management initiatives. As demonstrated in [140] and discussed in Appendix E, this database can support enterprise agility in a number of ways. Apart from providing support for capturing, storing, and managing data, information, and knowledge about enterprise agility. SQL query functionalities can be used to obtain useful business intelligence to support change management efforts. Additionally, previous but successful change management initiatives, activities, and strategies, can be stored in a database system. These can be retrieved, using query languages, and re-used to support future change management decisions. In these ways, time, cost, and efforts spent in change management can be optimised and enterprise agility can be enhanced easily.

Sensing and early detection of environmental events are key facilitators of enterprise agility and change management [49, 146]. The *change indicator* model concepts, included in this change modelling language, can be used to these ends. The change indicator concept can be used to represent and document information about certain social affairs, within the enterprise environment, that are likely to instigate change drivers and changes. For instance, social affairs such as election can bring about new government laws. These laws can initiate regulatory compliance, which is a known change driver that can cause changes in an enterprise. The change indicator concept can be used to capture and documents this type of social events. So that they can be known and analysed by stakeholders to determine their likelihood of leading to changes. If such events are likely to lead to changes, then stakeholders can quickly make preparations towards pre-empting such changes. In this way, enterprise agility can be enhanced.

Besides, conceptual modelling languages support and facilitate requirements analysis and elicitation. For instance, conceptual modelling languages, e.g., *i-star (i*)*,

have been widely used to identify, analyse, specify, and elicit systems requirements. Similarly the concepts, conceptual modelling languages, and framework proposed in this research can be useful in eliciting and analysing the requirements for achieving or facilitating enterprise agility and effective change management. Additionally, they provide a route to clearly understand requirements strategies to enable an organisation to respond to change with enhanced agility. When the requirements are clearly identified, analysed, and understood, adequate action plans, strategies, and policies for pre-empting or reacting to changes can be designed and implemented, and enterprise agility can be enhanced.

Bibliography

- [1] M.S. Abdullah et al. "Modelling knowledge based systems using the executable modelling framework (XMF)". In: *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*. Vol. 2. 2004, pp. 1055–1060. DOI: [10.1109/ICCIS.2004.1460735](https://doi.org/10.1109/ICCIS.2004.1460735).
- [2] Ruth Sara Aguilar-Saven. "Business process modelling: Review and framework". In: *International Journal of production economics* 90.2 (2004), pp. 129–149. URL: http://secure.com.sg/courses/ICT353/Session_Collateral/TOP_04_ART_03_ARTICLE_AGUILAR_Biz_Proc_Modelling.pdf.
- [3] Judith D. Ahrens and Chetan S. Sankar. "Tailoring Database Training for End Users." In: *MIS Quarterly* 17.4 (1993), pp. 419–439. ISSN: 02767783. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=9410256179&site=ehost-live>.
- [4] Stephan Aier and Bettina Gleichauf. "Applying Design Research Artifacts for Building Design Research Artifacts: A Process Model for Enterprise Architecture Planning". In: *Global Perspectives on Design Science Research*. Ed. by Robert Winter, J.Leon Zhao, and Stephan Aier. Vol. 6105. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 333–348. ISBN: 978-3-642-13334-3. DOI: [10.1007/978-3-642-13335-0_23](https://doi.org/10.1007/978-3-642-13335-0_23). URL: http://dx.doi.org/10.1007/978-3-642-13335-0_23.
- [5] David Akehurst and Stuart Kent. "A relational approach to defining transformations in a metamodel". In: *UML 2002-The Unified Modeling Language*. Springer, 2002, pp. 243–258. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.2113&rep=rep1&type=pdf>.
- [6] Manoli Albert et al. "Generating operation specifications from {UML} class diagrams: A model transformation approach". In: *Data & Knowledge Engineering* 70.4 (2011), pp. 365–389. ISSN: 0169-023X. DOI: <http://dx.doi.org/10.1016/j.datak.2011.01.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0169023X11000048>.
- [7] Laden Aldin and Sergio de Cesare. "A literature review on business process modelling: new frontiers of reusability." In: *Enterprise Information Systems* 5.3 (2011), pp. 359–383. ISSN: 17517575. URL: <http://search>.

- ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=62610243&site=ehost-live.
- [8] A. Aleti et al. "Software Architecture Optimization Methods: A Systematic Literature Review". In: *Software Engineering, IEEE Transactions on* 39.5 (2013), pp. 658–683. ISSN: 0098-5589. DOI: [10.1109/TSE.2012.64](https://doi.org/10.1109/TSE.2012.64).
- [9] Thomas Allweyer. *BPMN 2.0: introduction to the standard for business process modeling*. BoD–Books on Demand, 2010. URL: <http://www.bpmn-introduction.com/BPMN20AllweyerExcerpt.pdf>.
- [10] J. Andrade et al. "Definition of a problem-sensitive conceptual modelling language: foundations and application to software engineering". In: *Information and Software Technology* 48.7 (2006), pp. 517–531. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2005.05.009>. URL: <http://www.sciencedirect.com/science/article/pii/S095058490500087X>.
- [11] Ian O. Angell. "Systems thinking about information systems and strategies." In: *Journal of Information Technology (Routledge, Ltd.)* 5.3 (1990), p. 168. ISSN: 02683962. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=cph&AN=5418415&site=ehost-live>.
- [12] B.M. Arteta and R.E. Giachetti. "A measure of agility as the complexity of the enterprise system". In: *Robotics and Computer-Integrated Manufacturing* 20.6 (2004). <ce:title>13th International Conference on Flexible Automation and Intelligent Manufacturing</ce:title>, pp. 495–503. ISSN: 0736-5845. DOI: <http://dx.doi.org/10.1016/j.rcim.2004.05.008>. URL: <http://www.sciencedirect.com/science/article/pii/S0736584504000717>.
- [13] C. Atkinson and T. Kuhne. "Model-driven development: a metamodeling foundation". In: *Software, IEEE* 20.5 (2003), pp. 36–41. ISSN: 0740-7459. DOI: [10.1109/MS.2003.1231149](https://doi.org/10.1109/MS.2003.1231149).
- [14] John Bailey et al. "Evidence relating to Object-Oriented software design: A survey". In: *null*. IEEE. 2007, pp. 482–484.
- [15] J. Barjis. "Automatic business process analysis and simulation based on DEMO." In: *Enterprise Information Systems* 1.4 (2007), pp. 365–381. ISSN: 17517575. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=27601314&site=ehost-live>.
- [16] Carlo Batini, Stefano Ceri, and S Navathe. *Entity Relationship Approach*. Elsevier Science Publishers BV (North Holland), 1989. URL: <http://tocs.ulb.tu-darmstadt.de/2107187X.pdf>.
- [17] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. "A comparative analysis of methodologies for database schema integration". In: *ACM computing surveys (CSUR)* 18.4 (1986), pp. 323–364. URL:

- pubs.dbs.uni-leipzig.de/files/Batini1986AComparativeAnalysisof.pdf.
- [18] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. "Reasoning on {UML} class diagrams". In: *Artificial Intelligence* 168.1-2 (2005), pp. 70 – 118. ISSN: 0004-3702. DOI: <http://dx.doi.org/10.1016/j.artint.2005.05.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0004370205000792>.
- [19] Birol Berkem. "From The Business Motivation Model (BMM) To Service Oriented Architecture (SOA)." In: *Journal of Object Technology* 7.8 (2008), pp. 57–70. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.3295&rep=rep1&type=pdf>.
- [20] Meral Binbasioglu and Elaine Winston. "SYSTEMS THINKING FOR IDENTIFYING UNINTENDED CONSEQUENCES OF IT: PACKAGED SOFTWARE IMPLEMENTATION IN SMALL BUSINESSES." In: *Journal of Computer Information Systems* 45.1 (2004), pp. 86 –93. ISSN: 08874417. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=16997627&site=ehost-live>.
- [21] Michael R Blaha, William J Premerlani, and James E Rumbaugh. "Relational database design using an object-oriented methodology". In: *Communications of the ACM* 31.4 (1988), pp. 414–427. URL: <http://sims.monash.edu.au/subjects/ims2501/seminars/oomodelling.pdf>.
- [22] Constantin Blome, Tobias Schoenherr, and Daniel Rexhausen. "Antecedents and enablers of supply chain agility and its effect on performance: a dynamic capabilities perspective." In: *International Journal of Production Research* 51.4 (2013), pp. 1295 –1318. ISSN: 00207543. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=83776087&site=ehost-live>.
- [23] Team Bmm Version 1.2. *Business motivation model (bmm) Version 1.2 Beta 2 Specification*. Tech. rep. Technical Report dtc/2013-08-24, Object Management Group, Needham, Massachusetts, 2013. URL: <http://www.omg.org/spec/BMM/1.1/PDF>.
- [24] Francois Bodart et al. "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests." In: *Information Systems Research* 12.4 (2001), pp. 384 –405. ISSN: 10477047. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=5647429&site=ehost-live>.
- [25] F.S. de Boer et al. "Change impact analysis of enterprise architectures". In: *Information Reuse and Integration, Conf, 2005. IRI -2005 IEEE International Conference on*. 2005, pp. 177–181. DOI: [10.1109/IRI-05.2005.1506470](https://doi.org/10.1109/IRI-05.2005.1506470).

- [26] Erich Bornberg-Bauer and Norman W. Paton. "Conceptual data modelling for bioinformatics." In: *Briefings in Bioinformatics* 3.2 (2002), p. 166. ISSN: 14675463. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=6865507&site=ehost-live>.
- [27] Randy V. Bradley et al. "Enterprise architecture, IT effectiveness and the mediating role of IT alignment in US hospitals". In: *Information Systems Journal* 22.2 (2012), pp. 97–127. ISSN: 1365-2575. DOI: [10.1111/j.1365-2575.2011.00379.x](https://doi.org/10.1111/j.1365-2575.2011.00379.x). URL: <http://web.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=16&sid=2c8f7850-a902-493c-8e46-b0768f285ede%40sessionmgr111&hid=108>.
- [28] Tim Bray et al. "Extensible markup language (XML)". In: *World Wide Web Consortium Recommendation REC-xml-19980210*. <http://www.w3.org/TR/1998/REC-xml-19980210> 16 (1998).
- [29] Pearl Brereton et al. "Lessons from applying the systematic literature review process within the software engineering domain". In: *Journal of Systems and Software* 80.4 (2007). Software Performance 5th International Workshop on Software and Performance, pp. 571–583. ISSN: 0164-1212. DOI: <http://dx.doi.org/10.1016/j.jss.2006.07.009>. URL: <http://www.sciencedirect.com/science/article/pii/S016412120600197X>.
- [30] Manuel Brhel et al. "Exploring principles of user-centered agile software development: A literature review". In: *Information and Software Technology* 61 (2015), pp. 163–181. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2015.01.004>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584915000129>.
- [31] Barrett R Bryant et al. "Challenges and directions in formalizing the semantics of modeling languages". In: *Computer Science and Information Systems* 8.2 (2011), pp. 225–253. URL: <http://www.doiserbia.nb.rs/img/doi/1820-0214/2011/1820-02141100012B.pdf>.
- [32] Liliana Cabral, Barry Norton, and John Domingue. "The Business Process Modelling Ontology". In: *Proceedings of the 4th International Workshop on Semantic Business Process Management*. SBPM '09. Heraklion, Greece: ACM, 2009, pp. 9–16. ISBN: 978-1-60558-513-0. DOI: [10.1145/1944968.1944971](https://doi.org/10.1145/1944968.1944971). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/1944968.1944971>.
- [33] M. Mercedes Canavesio and Ernesto Martinez. "Enterprise modeling of a project oriented fractal company for {SMEs} networking". In: *Computers in Industry* 58.8-9 (2007), pp. 794–813. ISSN: 0166-3615. DOI: <http://dx.doi.org/10.1016/j.compind.2007.02.005>. URL: <http://www.sciencedirect.com/science/article/pii/S0166361507000322>.

- [34] Evellin Cristine Souza Cardoso, João A Paulo Almeida, and Renata SS Guizardi. "On the support for the goal domain in enterprise modelling approaches". In: *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International*. IEEE. 2010, pp. 335–344. URL: https://www.researchgate.net/profile/Joao_Almeida16/publication/224193265_On_the_Support_for_the_Goal_Domain_in_Enterprise_Modelling_Approaches/links/02e7e53457eb700738000000.pdf.
- [35] Joana Amorim Carvalho and Rui Dinis Sousa. "Enterprise architecture as enabler of organizational agility: a municipality case study". In: *20th Americas Conference on Information Systems*. Association for Information Systems. 2014. URL: <http://repositorium.sdum.uminho.pt/bitstream/1822/31340/1/Carvalho%20%26%20Sousa%202014%20AMCIS%20Enterprise%20Architecture.pdf>.
- [36] Kai Chen et al. "Semantic anchoring with model transformations". In: *Model Driven Architecture—Foundations and Applications*. Springer. 2005, pp. 115–129. URL: <http://131.107.65.14/en-us/um/people/ejackson/publications/ecmda2005.pdf>.
- [37] Peter Pin-Shan Chen. "The Entity-relationship Model—Toward a Unified View of Data". In: *ACM Trans. Database Syst.* 1.1 (Mar. 1976), pp. 9–36. ISSN: 0362-5915. DOI: 10.1145/320434.320440. URL: http://delivery.acm.org.ezproxy.mdx.ac.uk/10.1145/330000/320440/p9-chen.pdf?ip=158.94.0.157&id=320440&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5.319303F0AC56D129.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=554232749&CFTOKEN=97253590&__acm__=1445261198_da6ce05f93de9d3f59712dd7d4ddf889.
- [38] Samira Si-Said Cherfi, Jacky Akoka, and Isabelle Comyn-Wattiau. "Conceptual Modeling Quality - From EER to UML Schemas Evaluation". English. In: *Conceptual Modeling ER 2002*. Ed. by Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi. Vol. 2503. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 414–428. ISBN: 978-3-540-44277-6. DOI: 10.1007/3-540-45816-6_38. URL: http://dx.doi.org/10.1007/3-540-45816-6_38.
- [39] Roger HL Chiang, Terence M Barron, and Veda C Storey. "Reverse engineering of relational databases: Extraction of an EER model from a relational database". In: *Data & Knowledge Engineering* 12.2 (1994), pp. 107–142. URL: https://www.researchgate.net/profile/Roger_Chiang/publication/223098619_Reverse_engineering_of_relational_databases_Extraction_of_an_EER_model_from_a_relational_database/links/02e7e52713686e9f52000000.pdf.

- [40] Lawrence B. Chonko and Eli Jones. "THE NEED FOR SPEED: AGILITY SELLING." In: *Journal of Personal Selling and Sales Management* 25.4 (2005), pp. 371–382. ISSN: 08853134. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=eoah&AN=8263468&site=ehost-live>.
- [41] Yang Chyan and Liu Hsian-Ming. "Boosting firm performance via enterprise agility and network structure." In: *Management Decision* 50.6 (2012), pp. 1022–1044. ISSN: 00251747. URL: <http://www.emeraldinsight.com/journals.htm?issn=0025-1747&volume=50&issue=6&articleid=17038577&show=html>.
- [42] James Clark et al. "Xsl transformations (xslt)". In: *World Wide Web Consortium (W3C)*. URL <http://www.w3.org/TR/xslt> (1999), p. 103.
- [43] Tony Clark, Paul Sammut, and James Willans. "Applied metamodelling: a foundation for language driven development". In: *arXiv preprint arXiv:1505.00149* 1 (2015), pp. 1–244. URL: <http://arxiv.org/pdf/1505.00149.pdf>.
- [44] Khanh Hoa Dam and Michael Winikoff. "Comparing agent-oriented methodologies". In: *Agent-Oriented Information Systems*. Springer. 2004, pp. 78–93. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.9346&rep=rep1&type=pdf>.
- [45] Peta Darke, Graeme Shanks, and Marianne Broadbent. "Successfully completing case study research: combining rigour, relevance and pragmatism". In: *Information systems journal* 8.4 (1998), pp. 273–289.
- [46] Giorgio De Michelis et al. "A Three-faceted View of Information Systems". In: *Commun. ACM* 41.12 (Dec. 1998), pp. 64–70. ISSN: 0001-0782. DOI: 10.1145/290133.290150. URL: <http://doi.acm.org/10.1145/290133.290150>.
- [47] Alan Dearle. "Software Deployment, Past, Present and Future". In: *2007 Future of Software Engineering*. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 269–284. ISBN: 0-7695-2829-5. DOI: 10.1109/FOSE.2007.20. URL: <http://dx.doi.org.ezproxy.mdx.ac.uk/10.1109/FOSE.2007.20>.
- [48] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. "Semantics and analysis of business process models in BPMN". In: *Information and Software Technology* 50.12 (2008), pp. 1281–1294. URL: <http://eprints.qut.edu.au/7115/1/7115c.pdf>.
- [49] Seo Dongback and Ariel I. La Paz. "Exploring the Dark Side of IS in Achieving Organizational Aagility." In: *Communications of the ACM* 51.11 (2008), pp. 136–139. ISSN: 00010782. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=35211915&site=ehost-live>.

- [50] Rick Dove. "Agile Enterprise Cornerstones: Knowledge, Values, and Response Ability". English. In: *Business Agility and Information Technology Diffusion*. Ed. by RichardL. Baskerville et al. Vol. 180. IFIP International Federation for Information Processing. Springer US, 2005, pp. 313–330. ISBN: 978-0-387-25589-7. DOI: [10.1007/0-387-25590-7_20](https://doi.org/10.1007/0-387-25590-7_20). URL: http://dx.doi.org/10.1007/0-387-25590-7_20.
- [51] Yves Ducq, David Chen, and Bruno Vallespir. "Interoperability in enterprise modelling: requirements and roadmap". In: *Advanced Engineering Informatics* 18.4 (2004), pp. 193–203.
- [52] Tore Dyba and Torgeir Dingsoyr. "Empirical studies of agile software development: A systematic review". In: *Information and Software Technology* 50.9-10 (2008), pp. 833–859. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2008.01.006>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584908000256>.
- [53] Gregor Engels, Reiko Heckel, and Stefan Sauer. "UML: A Universal Modeling Language?" English. In: *Application and Theory of Petri Nets 2000*. Ed. by Mogens Nielsen and Dan Simpson. Vol. 1825. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 24–38. ISBN: 978-3-540-67693-5. DOI: [10.1007/3-540-44988-4_3](https://doi.org/10.1007/3-540-44988-4_3). URL: http://dx.doi.org/10.1007/3-540-44988-4_3.
- [54] Wilco Engelsman and Roel Wieringa. "Goal-Oriented Requirements Engineering and Enterprise Architecture: Two Case Studies and Some Lessons Learned". In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Bjorn Regnell and Daniela Damian. Vol. 7195. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 306–320. ISBN: 978-3-642-28713-8. DOI: [10.1007/978-3-642-28714-5_27](https://doi.org/10.1007/978-3-642-28714-5_27). URL: http://dx.doi.org/10.1007/978-3-642-28714-5_27.
- [55] Wilco Engelsman et al. "Extending enterprise architecture modelling with business goals and requirements." In: *Enterprise Information Systems* 5.1 (2011), pp. 9–36. ISSN: 17517575. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=55816051&site=ehost-live>.
- [56] S Yu Eric. "Social Modeling and i*". In: *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 99–121. URL: <http://mis.sauder.ubc.ca/files/2011/08/JMfest09-soc-modeling-istar.pdf>.
- [57] Ozgur Erol, Brian J. Sausser, and Mo Mansouri. "A framework for investigation into extended enterprise resilience." In: *Enterprise Information Systems* 4.2 (2010), pp. 111–136. ISSN: 17517575. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=49459865&site=ehost-live>.

- [58] A.S. Evans. "Reasoning with UML class diagrams". In: *Industrial Strength Formal Specification Techniques, 1998. Proceedings. 2nd IEEE Workshop on.* 1998, pp. 102–113. DOI: [10.1109/WIFT.1998.766304](https://doi.org/10.1109/WIFT.1998.766304).
- [59] T. Fallmyr and B. Bygstad. "Enterprise Architecture Practice and Organizational Agility: An Exploratory Study". In: *System Sciences (HICSS), 2014 47th Hawaii International Conference on.* 2014, pp. 3788–3797. DOI: [10.1109/HICSS.2014.471](https://doi.org/10.1109/HICSS.2014.471).
- [60] Roxana Fekri, Alireza Aliahmadi, and Mohammad Fathian. "Predicting a model for agile NPD process with fuzzy cognitive map: the case of Iranian manufacturing enterprises". English. In: *The International Journal of Advanced Manufacturing Technology* 41.11-12 (2009), pp. 1240–1260. ISSN: 0268-3768. DOI: [10.1007/s00170-008-1565-7](https://doi.org/10.1007/s00170-008-1565-7). URL: <http://dx.doi.org/10.1007/s00170-008-1565-7>.
- [61] L. Ferrarini et al. "Applied meta-modelling to convert IEC 61499 applications into Step7 environment". In: *Control Conference (ECC), 2007 European.* 2007, pp. 2059–2064.
- [62] Mark S Fox and Michael Grüninger. "Ontologies for enterprise modelling". In: *Enterprise Engineering and Integration.* Springer, 1997, pp. 190–200. URL: <http://eil2.mie.utoronto.ca/wp-content/uploads/enterprise-modelling/papers/fox-eimt97.pdf>.
- [63] L. Garcia-Borgonon et al. "Software process modeling languages: A systematic literature review". In: *Information and Software Technology* 56.2 (2014), pp. 103–116. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2013.10.001>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584913001894>.
- [64] V. Garcia-Diaz et al. "MCTest: towards an improvement of match algorithms for models". English. In: *IET Software* 6 (2 2012), 127–139(12). ISSN: 1751-8806. URL: <http://web.a.ebscohost.com.ezproxy.mdx.ac.uk/ehost/pdfviewer/pdfviewer?sid=01650e74-05cc-4ab5-9232-417290afd7bf%40sessionmgr4003&vid=0&hid=4212>.
- [65] Hamada Ghenniwa, Michael N. Huhns, and Weiming Shen. "eMarketplaces for enterprise and cross enterprise integration". In: *Data & Knowledge Engineering* 52.1 (2005). Collaborative business process technologies, pp. 33–59. ISSN: 0169-023X. DOI: <http://dx.doi.org/10.1016/j.datak.2004.06.005>. URL: <http://www.sciencedirect.com/science/article/pii/S0169023X04000874>.
- [66] David M. Gligor, Mary C. Holcomb, and Theodore P. Stank. "A Multidisciplinary Approach to Supply Chain Agility: Conceptualization and Scale Development." In: *Journal of Business Logistics* 34.2 (2013), pp. 94–108. ISSN: 07353766. URL: <http://search.ebscohost.com.ezproxy.mdx>.

- ac.uk/login.aspx?direct=true&db=bth&AN=88106789&site=ehost-live.
- [67] Paulo B. Goes. "Design Science Research in Top Information Systems Journals." In: *MIS Quarterly* 38.1 (2014), pp. iii–viii. ISSN: 02767783. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=94003226&site=ehost-live>.
- [68] Thomas Goldschmidt, Steffen Becker, and Axel Uhl. "Classification of concrete textual syntax mapping approaches". In: *Model Driven Architecture—Foundations and Applications*. Springer, 2008, pp. 169–184. URL: <https://sdqweb.ipd.kit.edu/publications/pdfs/goldschmidt2008b.pdf>.
- [69] Arturo Gonzalez, Sergio Espana, and Oscar Pastor. "Towards a Communicational Perspective for Enterprise Information Systems Modelling". English. In: *The Practice of Enterprise Modeling*. Ed. by Janis Stirna and Anne Persson. Vol. 15. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2008, pp. 62–76. ISBN: 978-3-540-89217-5. DOI: [10.1007/978-3-540-89218-2_5](https://doi.org/10.1007/978-3-540-89218-2_5). URL: http://dx.doi.org/10.1007/978-3-540-89218-2_5.
- [70] Dale L. Goodhue et al. "Addressing Business Agility Challenges with Enterprise Systems." In: *MIS Quarterly Executive* 8.2 (2009), pp. 73–87. ISSN: 15401960. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=42093790&site=ehost-live>.
- [71] Hans Grönninger et al. "Textbased modeling". In: *arXiv preprint arXiv:1409.6623n/a* (2014), pp. 1–9. URL: <http://arxiv.org/ftp/arxiv/papers/1409/1409.6623.pdf>.
- [72] Michael Gruninger and Mark S Fox. "The logic of enterprise modelling". In: *Modelling and Methodologies for Enterprise Integration*. Springer, 1996, pp. 140–157. URL: https://www.researchgate.net/profile/Mark_Fox5/publication/2520448_The_Logic_of_Enterprise_Modelling/links/0046353861eed59a33000000.pdf.
- [73] A Gunasekaran. "Agile manufacturing: A framework for research and development". In: *International Journal of Production Economics* 62.1-2 (1999), pp. 87–105. ISSN: 0925-5273. DOI: [http://dx.doi.org/10.1016/S0925-5273\(98\)00222-9](https://doi.org/10.1016/S0925-5273(98)00222-9). URL: <http://www.sciencedirect.com/science/article/pii/S0925527398002229>.
- [74] A. Gunasekaran. "Preface". In: *Agile Manufacturing: The 21st Century Competitive Strategy*. Ed. by A. Gunasekaran. Oxford: Elsevier Science Ltd, 2001, pp. v–vi. ISBN: 978-0-08-043567-1. DOI: [http://dx.doi.org/10.1016/B978-008043567-1/50000-0](https://doi.org/10.1016/B978-008043567-1/50000-0). URL: <http://www.sciencedirect.com/science/article/pii/B9780080435671500000>.

- [75] Farshad Hakimpour and Andreas Geppert. "Resolving Semantic Heterogeneity in Schema Integration". In: *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*. FOIS '01. Ogunquit, Maine, USA: ACM, 2001, pp. 297–308. ISBN: 1-58113-377-4. DOI: [10.1145/505168.505196](https://doi.org/10.1145/505168.505196). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/505168.505196>.
- [76] D. Harel and A. Pnueli. "On the Development of Reactive Systems". English. In: *Logics and Models of Concurrent Systems*. Ed. by Krzysztof R. Apt. Vol. 13. NATO ASI Series. Springer Berlin Heidelberg, 1985, pp. 477–498. ISBN: 978-3-642-82455-5. DOI: [10.1007/978-3-642-82453-1_17](https://doi.org/10.1007/978-3-642-82453-1_17). URL: <http://www.wisdom.weizmann.ac.il/~dharel/SCANNED.PAPERS/ReactiveSystems.pdf>.
- [77] D. Harel et al. "STATEMATE: a working environment for the development of complex reactive systems". In: *Software Engineering, IEEE Transactions on* 16.4 (1990), pp. 403–414. ISSN: 0098-5589. DOI: [10.1109/32.54292](https://doi.org/10.1109/32.54292).
- [78] David Harel. "Statecharts: a visual formalism for complex systems". In: *Science of Computer Programming* 8.3 (1987), pp. 231–274. ISSN: 0167-6423. DOI: [http://dx.doi.org/10.1016/0167-6423\(87\)90035-9](http://dx.doi.org/10.1016/0167-6423(87)90035-9). URL: <http://www.sciencedirect.com/science/article/pii/0167642387900359>.
- [79] David Harel. "Statecharts in the Making: A Personal Account". In: *Commun. ACM* 52.3 (Mar. 2009), pp. 67–75. ISSN: 0001-0782. DOI: [10.1145/1467247.1467274](https://doi.org/10.1145/1467247.1467274). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/1467247.1467274>.
- [80] David Harel and Hillel Kugler. "The rhapsody semantics of statecharts (or, on the executable core of the UML)". In: *Integration of Software Specification Techniques for Applications in Engineering*. Springer, 2004, pp. 325–354. URL: <http://research.microsoft.com/pubs/148761/Charts04.pdf>.
- [81] David Harel and Amnon Naamad. "The STATEMATE Semantics of Statecharts". In: *ACM Trans. Softw. Eng. Methodol.* 5.4 (Oct. 1996), pp. 293–333. ISSN: 1049-331X. DOI: [10.1145/235321.235322](https://doi.org/10.1145/235321.235322). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/235321.235322>.
- [82] David Harel and Bernhard Rumpe. "Meaningful modeling: what's the semantics of" semantics"?" In: *Computer* 37.10 (2004), pp. 64–72. URL: http://people.irisa.fr/Jean-Marc.Jezequel/enseignement/M2RI/MDE/Articles/HR_ModSemantics_IEEEComp_04.pdf.
- [83] David Harel and Bernhard Rumpe. "Modeling Languages: Syntax, Semantics and All That Stu". In: *N/A n/a* (2000), pp. 1–28. URL: <http://www4.in.tum.de/publ/papers/HR00.pdf>.

- [84] Majid Hashemipour, Omer Anlagan, and Sinan Kayaligil. "A computer-supported methodology for requirements modelling in CIM for small and medium size enterprises: a demonstration in Apparel Industry." In: *International Journal of Computer Integrated Manufacturing* 10.1-4 (1997), pp. 199 – 211. ISSN: 0951192X. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=cph&AN=7614124&site=ehost-live>.
- [85] Florian Heidenreich et al. "Derivation and refinement of textual syntax for models". In: *Model Driven Architecture-Foundations and Applications*. Springer. 2009, pp. 114–129. URL: https://www.researchgate.net/profile/Christian_Wende/publication/220989580_Derivation_and_Refinement_of_Textual_Syntax_for_Models/links/09e415094f36b6370b000000.pdf.
- [86] Richard Heining. "Requirements for Business Process Management Systems Supporting Business Process Agility". English. In: *S-BPM ONE - Education and Industrial Developments*. Ed. by Stefan Oppl and Albert Fleischmann. Vol. 284. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012, pp. 168–180. ISBN: 978-3-642-29293-4. DOI: 10.1007/978-3-642-29294-1_12. URL: http://dx.doi.org/10.1007/978-3-642-29294-1_12.
- [87] Brian Henderson-Sellers. "Models". English. In: *On the Mathematics of Modelling, Metamodeling, Ontologies and Modelling Languages*. SpringerBriefs in Computer Science. Springer Berlin Heidelberg, 2012, pp. 31–39. ISBN: 978-3-642-29824-0. DOI: 10.1007/978-3-642-29825-7_3. URL: http://dx.doi.org/10.1007/978-3-642-29825-7_3.
- [88] Wolfgang Hesse. "More matters on (meta-) modelling: remarks on Thomas Kuhnes matters". In: *Software & Systems Modeling* 5.4 (2006), pp. 387–394. URL: <http://homepages.mcs.vuw.ac.nz/~tk/publications/papers/MoreMattersOfMetaModeling.pdf>.
- [89] Alan R Hevner. "A three cycle view of design science research". In: *Scandinavian journal of information systems* 19.2 (2007), p. 4. URL: <http://community.mis.temple.edu/seminars/files/2009/10/Hevner-SJIS.pdf>.
- [90] Alan R. Hevner et al. "DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH." In: *MIS Quarterly* 28.1 (2004), pp. 75 –105. ISSN: 02767783. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=12581935&site=ehost-live>.
- [91] Jean-Marc Hick and Jean-Luc Hainaut. "Database application evolution: A transformational approach". In: *Data & Knowledge Engineering* 59.3 (2006). Including: {ER} 2003 Selection of papers presented at the 22nd International Conference on Conceptual Modeling 22nd International Conference

- on Conceptual Modeling, pp. 534 –558. ISSN: 0169-023X. DOI: <http://dx.doi.org/10.1016/j.datak.2005.10.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0169023X05001631>.
- [92] C. A. R. Hoare. *Essays in Computing Science*. Ed. by C. B. Jones. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989, pp. 193–216. ISBN: 0-13-284027-8. URL: http://delivery.acm.org.ezproxy.mdx.ac.uk/10.1145/70000/63445/cb-ecs-hoare.pdf?ip=158.94.0.157&id=63445&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5.319303F0AC56D129.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=721187359&CFTOKEN=10378744&__acm__=1444730891_a0597ed3e1f658324
- [93] CA Hoare. *Hints on programming language design*. Tech. rep. DTIC Document, 1973. URL: <http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=AD0773391>.
- [94] Phil Hodgkinson and Heather Hodgkinson. “The strengths and limitations of case study research”. In: *Learning and Skills Development Agency Conference at Cambridge*. Vol. 1. 1. 2001, pp. 5–7.
- [95] Nick Horney et al. “From Change Projects to Change Agility.” In: *People & Strategy* 37.1 (2014), pp. 40 –45. ISSN: 19464606. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=97590297&site=ehost-live>.
- [96] C. Huizing and W.P. de Roever. “Introduction to design choices in the semantics of Statecharts”. In: *Information Processing Letters* 37.4 (1991), pp. 205 –213. ISSN: 0020-0190. DOI: [http://dx.doi.org/10.1016/0020-0190\(91\)90190-S](http://dx.doi.org/10.1016/0020-0190(91)90190-S). URL: <http://www.sciencedirect.com/science/article/pii/002001909190190S>.
- [97] T. Hussain. “Revisiting quality metrics for conceptual models”. In: *TENCON 2014 - 2014 IEEE Region 10 Conference*. 2014, pp. 1–6. DOI: [10.1109/TENCON.2014.7022370](https://doi.org/10.1109/TENCON.2014.7022370).
- [98] ISO/19440. *Enterprise integration: Constructs for enterprise modelling*. International Organization for Standardization. 2007. URL: <http://www.evsee/preview/iso-19440-2007-en.pdf>.
- [99] S. Izza et al. “An Approach for the Evaluation of the Agility in the Context of Enterprise Interoperability”. English. In: *Enterprise Interoperability III*. Ed. by Kai Mertins et al. Springer London, 2008, pp. 3–14. ISBN: 978-1-84800-220-3. DOI: [10.1007/978-1-84800-221-0_1](https://doi.org/10.1007/978-1-84800-221-0_1). URL: http://dx.doi.org/10.1007/978-1-84800-221-0_1.
- [100] Caron H St. John, Alan R Cannon, and Richard W Poudier. “Change drivers in the new millennium: implications for manufacturing strategy research”. In: *Journal of Operations Management* 19.2 (2001), pp. 143 –160. ISSN: 0272-6963. DOI: [http://dx.doi.org/10.1016/S0272-6963\(00\)00054-](http://dx.doi.org/10.1016/S0272-6963(00)00054-)

1. URL: <http://www.sciencedirect.com/science/article/pii/S0272696300000541>.
- [101] Henk Jonkers, Erik Proper, and Mike Turner. "TOGAFTM and ArchiMate[®]: A Future Together". In: *A White Paper Published by The Open Group* 192 (2009), pp. 1–15. URL: http://www.itilbookshop.org/Player/eKnowledge/togaf_and_archimate_a_future_together.pdf.
- [102] Frederic Jouault and Jean Bezivin. "KM3: A DSL for Metamodel Specification". English. In: *Formal Methods for Open Object-Based Distributed Systems*. Ed. by Roberto Gorrieri and Heike Wehrheim. Vol. 4037. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 171–185. ISBN: 978-3-540-34893-1. DOI: [10.1007/11768869_14](https://doi.org/10.1007/11768869_14). URL: http://dx.doi.org/10.1007/11768869_14.
- [103] D. R. Kalbande, G. T. Thampi, and N. T. Deotale. "e-Procurement for Increasing Business Process Agility". In: *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*. ICWET '11. Mumbai, Maharashtra, India: ACM, 2011, pp. 761–764. ISBN: 978-1-4503-0449-8. DOI: [10.1145/1980022.1980187](https://doi.org/10.1145/1980022.1980187). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/1980022.1980187>.
- [104] Gabor Karsai et al. "Design guidelines for domain specific languages". In: *arXiv preprint arXiv:1409.2378* n/a (2014), pp. 1–7. URL: <http://arxiv.org/ftp/arxiv/papers/1409/1409.2378.pdf>.
- [105] Vipul Kashyap and Amit Sheth. "Semantic and Schematic Similarities Between Database Objects: A Context-based Approach". In: *The VLDB Journal* 5.4 (Dec. 1996), pp. 276–304. ISSN: 1066-8888. DOI: [10.1007/s007780050029](https://doi.org/10.1007/s007780050029). URL: <http://dx.doi.org.ezproxy.mdx.ac.uk/10.1007/s007780050029>.
- [106] William J. Kettinger and Varun Grover. "Special Section: Toward a Theory of Business Process Change Management." In: *Journal of Management Information Systems* 12.1 (1995), pp. 9–30. ISSN: 07421222. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=19077878&site=ehost-live>.
- [107] Barbara Kitchenham. "Procedures for performing systematic reviews". In: *Keele, UK, Keele University* 33.2004 (2004), pp. 1–26.
- [108] Dimitrios Kolovos et al. "The epsilon book". In: *Structure* 178 (2010), pp. 1–10. URL: <https://git.eclipse.org/c/epsilon/org.eclipse.epsilon.git/plain/doc/org.eclipse.epsilon.book/EpsilonBook.pdf>.

- [109] Dimitrios S Kolovos et al. "Requirements for domain-specific languages". In: *Proceedings of the First ECOOP Workshop on Domain-Specific Program Development*. 2006. URL: https://www-users.cs.york.ac.uk/~tpk/req_dspls.pdf.
- [110] K. Kotiadis and S. Robinson. "Conceptual modelling: Knowledge acquisition and model abstraction". In: *Simulation Conference, 2008. WSC 2008. Winter*. 2008, pp. 951–958. DOI: [10.1109/WSC.2008.4736161](https://doi.org/10.1109/WSC.2008.4736161).
- [111] John Krogstie, Odd Ivar Lindland, and Guttorm Sindre. "Towards a deeper understanding of quality in requirements engineering". In: *Advanced Information Systems Engineering*. Springer. 1995, pp. 82–95. URL: <http://www.idi.ntnu.no/~krogstie/publications/1995/CAISE95/fulltext.pdf>.
- [112] Thomas Kuhne. "Matters of (meta-) modeling". In: *Software & Systems Modeling* 5.4 (2006), pp. 369–385. URL: <http://homepages.ecs.vuw.ac.nz/~tk/publications/papers/kuehne-matters.pdf>.
- [113] R. Kumar, C. Bhattacharyya, and V. Varshneya. "Discovering Business Process Model from Unstructured Activity Logs". In: *Services Computing (SCC), 2010 IEEE International Conference on*. 2010, pp. 250–257. DOI: [10.1109/SCC.2010.78](https://doi.org/10.1109/SCC.2010.78).
- [114] Axel van Lamsweerde and Emmanuel Letier. "Handling obstacles in goal-oriented requirements engineering". In: *Software Engineering, IEEE Transactions on* 26.10 (2000), pp. 978–1005. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=879820>.
- [115] Marc M. Lankhorst. "Enterprise architecture modelling-the issue of integration". In: *Advanced Engineering Informatics* 18.4 (2004). Enterprise Modelling and System Support, pp. 205–216. ISSN: 1474-0346. DOI: <http://dx.doi.org/10.1016/j.aei.2005.01.005>. URL: <http://www.sciencedirect.com/science/article/pii/S1474034605000054>.
- [116] Marc M Lankhorst, Henderik Alex Proper, and Henk Jonkers. "The architecture of the archimate language". In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2009, pp. 367–380.
- [117] Ramon Lawrence and Ken Barker. "Integrating Relational Database Schemas Using a Standardized Dictionary". In: *Proceedings of the 2001 ACM Symposium on Applied Computing*. SAC '01. Las Vegas, Nevada, USA: ACM, 2001, pp. 225–230. ISBN: 1-58113-287-5. DOI: [10.1145/372202.372327](https://doi.org/10.1145/372202.372327). URL: <http://doi.acm.org/10.1145/372202.372327>.
- [118] Xavier Le Pallec and Sophie Dupuy-Chessa. "Support for Quality Metrics in Metamodelling". In: *Proceedings of the Second Workshop on Graphical Modeling Language Development*. GMLD '13. Montpellier, France: ACM, 2013,

- pp. 23–31. ISBN: 978-1-4503-2044-3. DOI: [10.1145/2489820.2489825](https://doi.org/10.1145/2489820.2489825). URL: <http://doi.acm.org/10.1145/2489820.2489825>.
- [119] Ching-Torng Lin, Hero Chiu, and Yi-Hong Tseng. “Agility evaluation using fuzzy logic”. In: *International Journal of Production Economics* 101.2 (2006), pp. 353–368. ISSN: 0925-5273. DOI: <http://dx.doi.org/10.1016/j.ijpe.2005.01.011>. URL: <http://www.sciencedirect.com/science/article/pii/S0925527305000514>.
- [120] O.I. Lindland, G. Sindre, and A. Solvberg. “Understanding quality in conceptual modeling”. In: *Software, IEEE* 11.2 (1994), pp. 42–49. ISSN: 0740-7459. DOI: [10.1109/52.268955](https://doi.org/10.1109/52.268955).
- [121] Pericles Loucopoulos and Evangelia Kavakli. “Enterprise modelling and the teleological approach to requirements engineering”. In: *International Journal of Cooperative Information Systems* 4.01 (1995), pp. 45–79. URL: http://patrologia.ct.aegean.gr/people/vkavakli/publications/pdf_files/IJICS95_kavakli.pdf.
- [122] Richard A Martin. “International standards for system integration”. In: *Rochester, NY: International Council on Systems Engineering (INCOSE2005)* 1 (2005), pp. 1–48. URL: http://www.tinwisle.com/iso/RM_OMG_BEIDTF_Plenary05Slides.pdf.
- [123] Sabine Matook and Likoebe M. Maruping. “A Competency Model for Customer Representatives in Agile Software Development Projects.” In: *MIS Quarterly Executive* 13.2 (2014), pp. 77–95. ISSN: 15401960. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=96300139&site=ehost-live>.
- [124] Kashif Mehmood and Samira Si-Said Cherfi. “Evaluating the Functionality of Conceptual Models”. English. In: *Advances in Conceptual Modeling - Challenging Perspectives*. Ed. by Carlos Alberto Heuser and Gunther Pernul. Vol. 5833. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 222–231. ISBN: 978-3-642-04946-0. DOI: [10.1007/978-3-642-04947-7_27](https://doi.org/10.1007/978-3-642-04947-7_27). URL: http://dx.doi.org/10.1007/978-3-642-04947-7_27.
- [125] S.J. Mellor, A.N. Clark, and T. Futagami. “Model driven development: Guest editor’s introduction”. In: *Software, IEEE* 20.5 (2003), pp. 14–18. ISSN: 0740-7459. DOI: [10.1109/MS.2003.1231145](https://doi.org/10.1109/MS.2003.1231145).
- [126] Bart Meyers and Hans Vangheluwe. “A framework for evolution of modelling languages”. In: *Science of Computer Programming* 76.12 (2011). Special Issue on Software Evolution, Adaptability and Variability, pp. 1223–1246. ISSN: 0167-6423. DOI: <http://dx.doi.org/10.1016/j.scico.2011.01.002>. URL: <http://www.sciencedirect.com/science/article/pii/S0167642311000141>.

- [127] Daniel L Moody. "Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions". In: *Data & Knowledge Engineering* 55.3 (2005), pp. 243–276. URL: http://ac.els-cdn.com/S0169023X04002307/1-s2.0-S0169023X04002307-main.pdf?_tid=89dfbf34-0dbd-11e5-b4b4-0000aacb362&acdnat=1433754521_adf536f3b4ab16d58058e75aefa722ce.
- [128] Daniel L Moody and Graeme G Shanks. "Improving the quality of data models: empirical validation of a quality management framework". In: *Information Systems* 28.6 (2003), pp. 619–650. ISSN: 0306-4379. DOI: [http://dx.doi.org/10.1016/S0306-4379\(02\)00043-1](http://dx.doi.org/10.1016/S0306-4379(02)00043-1). URL: <http://www.sciencedirect.com/science/article/pii/S0306437902000431>.
- [129] Daniel L. Moody. "Metrics for Evaluating the Quality of Entity Relationship Models". English. In: *Conceptual Modeling ER'98*. Ed. by Tok-Wang Ling, Sudha Ram, and Mong Li Lee. Vol. 1507. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 211–225. ISBN: 978-3-540-65189-5. DOI: [10.1007/978-3-540-49524-6_18](https://doi.org/10.1007/978-3-540-49524-6_18). URL: http://dx.doi.org/10.1007/978-3-540-49524-6_18.
- [130] Liping Mu et al. "Specification of Modelling Languages in a Flexible Meta-model Architecture". In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ECSA '10. Copenhagen, Denmark: ACM, 2010, pp. 302–308. ISBN: 978-1-4503-0179-4. DOI: [10.1145/1842752.1842807](https://doi.org/10.1145/1842752.1842807). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/1842752.1842807>.
- [131] Pierre-Alain Muller, Frédéric Fondement, and Benoît Baudry. "Modeling Modeling." In: *MoDELS*. Vol. 5795. Springer. 2009, pp. 2–16. URL: <https://hal.inria.fr/inria-00477528/document>.
- [132] John Mylopoulos. *Conceptual modelling and Telos 1*. 2008. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.3647&rep=rep1&type=pdf>.
- [133] John Mylopoulos. "Information modeling in the time of the revolution". In: *Information Systems* 23.3-4 (May 1998), pp. 127–155. ISSN: 03064379. DOI: [10.1016/S0306-4379\(98\)00005-2](https://doi.org/10.1016/S0306-4379(98)00005-2). URL: <http://www.cs.toronto.edu/~jm/2507S/Readings/Survey.pdf>.
- [134] Agnes Nakakawa, Patrick Van Bommel, and H.A. Proper. "Definition and Validation of Requirements for Collaborative Decision-making in eEnterprise Architecture Creation." In: *International Journal of Cooperative Information Systems* 20.1 (2011), pp. 83–136. ISSN: 02188430. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=74219669&site=ehost-live>.

- [135] Majid Nejatian and Mohammad Hossein Zarei. "Moving Towards Organizational Agility: Are We Improving in the Right Direction?." In: *Global Journal of Flexible Systems Management* 14.4 (2013), pp. 241–253. ISSN: 09722696. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=91928637&site=ehost-live>.
- [136] J.C. Nwokeji, T. Clark, and B.S. Barn. "Towards a comprehensive Meta-Model for KAOS". In: *Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on*. 2013, pp. 30–39. DOI: [10.1109/MoDRE.2013.6597261](https://doi.org/10.1109/MoDRE.2013.6597261).
- [137] Joshua C. Nwokeji, Tony Clark, and Balbir S. Barn. "A proposal for consolidated intentional modeling language". In: *Proceedings of the Second Workshop on Graphical Modeling Language Development*. GMLD '13. Montpellier, France: ACM, 2013, pp. 12–22. ISBN: 978-1-4503-2044-3. DOI: [10.1145/2489820.2489826](https://doi.org/10.1145/2489820.2489826). URL: <http://doi.acm.org/10.1145/2489820.2489826>.
- [138] Joshua Chibuikwe Nwokeji. "A Framework for Enterprise Agility". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 1249–1250. ISBN: 978-1-4503-3196-8. DOI: [10.1145/2695664.2696062](https://doi.org/10.1145/2695664.2696062). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/2695664.2696062>.
- [139] Joshua Chibuikwe Nwokeji et al. "A Conceptual Framework for Enterprise Agility". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 1242–1244. ISBN: 978-1-4503-3196-8. DOI: [10.1145/2695664.2699495](https://doi.org/10.1145/2695664.2699495). URL: <http://doi.acm.org.ezproxy.mdx.ac.uk/10.1145/2695664.2699495>.
- [140] Joshua Chibuikwe Nwokeji et al. "A Data-centric Approach to Change Management". In: *Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International*. 2015, pp. 185–190. DOI: [10.1109/EDOC.2015.34](https://doi.org/10.1109/EDOC.2015.34).
- [141] Joshua C. Nwokeji et al. "Automated Completeness Check in KAOS". English. In: *Advances in Conceptual Modeling*. Ed. by Marta Indulska and Sandeep Puroo. Vol. 8823. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 133–138. ISBN: 978-3-319-12255-7. DOI: [10.1007/978-3-319-12256-4_14](https://doi.org/10.1007/978-3-319-12256-4_14). URL: http://dx.doi.org/10.1007/978-3-319-12256-4_14.
- [142] DD Object-Management-Group. *Diagram Definition Version 1.1*. Object Management Group. 2015. URL: <http://www.omg.org/spec/DD/1.1/>.
- [143] Objectiver. *A Kaos Tutorial Version 1.0*. Tech. rep. Respect-IT, 2007. URL: <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>.

- [144] Object Management Group Specification OMG. "Business Process Model and Notation 2.0". In: *OMG Specification, Object Management Group 2* (2011), pp. 1–600. URL: <http://www.omg.org/spec/BPMN/2.0>.
- [145] Marcel van Oosterhout, Eric Waarts, and Jos van Hillegerberg. "Change factors requiring agility and implications for IT." In: *European Journal of Information Systems* 15.2 (2006), pp. 132–145. ISSN: 0960085X. URL: <http://fkaouane.free.fr/EJIS/3000601a.pdf>.
- [146] Eric Overby, Anandhi Bharadwaj, and V Sambamurthy. "A framework for enterprise agility and the enabling role of digital options". In: *Business agility and information technology diffusion*. Springer, 2005, pp. 295–312. URL: <http://opendl.ifip-tc6.org/db/conf/ifip8-6/ifip8-6-2005/OverbyBS05.pdf>.
- [147] Eric Overby, Anandhi Bharadwaj, and V. Sambamurthy. "A Framework for Enterprise Agility and the Enabling Role of Digital Options". English. In: *Business Agility and Information Technology Diffusion*. Ed. by RichardL. Baskerville et al. Vol. 180. IFIP International Federation for Information Processing. Springer US, 2005, pp. 295–312. ISBN: 978-0-387-25589-7. DOI: 10.1007/0-387-25590-7_19. URL: http://dx.doi.org/10.1007/0-387-25590-7_19.
- [148] Eric Overby, Anandhi Bharadwaj, and V. Sambamurthy. "Enterprise agility and the enabling role of information technology." In: *European Journal of Information Systems* 15.2 (2006), pp. 120–131. ISSN: 0960085X. URL: <http://fkaouane.free.fr/EJIS/3000600a.pdf>.
- [149] R.F Paige, J.S Ostroff, and P.J Brooke. "Principles for modeling language design". In: *Information and Software Technology* 42.10 (2000), pp. 665–675. ISSN: 0950-5849. DOI: [http://dx.doi.org/10.1016/S0950-5849\(00\)00109-9](http://dx.doi.org/10.1016/S0950-5849(00)00109-9). URL: <http://books.genems.com/journals/it/2-2/design%20and%20analysis%20of%20algorithms/1-s2.0-S0950584900001099-main.pdf>.
- [150] Ken Peffers, Marcus Rothenberger, and William Kuechler. *Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012. Proceedings*. Vol. 7286 2012. LNCS. Springer, 2012. URL: <http://link.springer.com/book/10.1007/978-3-642-29863-9>.
- [151] Carla Marques Pereira and Pedro Sousa. "A Method to Define an Enterprise Architecture Using the Zachman Framework". In: *Proceedings of the 2004 ACM Symposium on Applied Computing*. SAC '04. Nicosia, Cyprus: ACM, 2004, pp. 1366–1371. ISBN: 1-58113-812-1. DOI: 10.1145/967900.968175. URL: <http://doi.acm.org/10.1145/967900.968175>.

- [152] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. "Guidelines for conducting systematic mapping studies in software engineering: An update". In: *Information and Software Technology* 64 (2015), pp. 1–18. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2015.03.007>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584915000646>.
- [153] Kai Petersen et al. "Systematic mapping studies in software engineering". In: *12th international conference on evaluation and assessment in software engineering*. Vol. 17. 1. sn. 2008, pp. 1–10.
- [154] Daniel Pfeiffer. "Constructing Comparable Conceptual Models with Domain Specific Languages." In: *ECIS*. 2007, pp. 876–888. URL: <http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1042&context=ecis2007>.
- [155] Daniel Pfeiffer and Andreas Gehlert. "A framework for comparing conceptual models". In: *Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2005)*. Citeseer. 2005, pp. 108–122. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.6840&rep=rep1&type=pdf>.
- [156] Terry Quatrani. *Visual modeling with rational rose 2002 and UML*. Ed. by Quatrani. Addison-Wesley Longman Publishing Co., Inc., 2002. URL: ftp://210.212.172.242/Digital_Library/CSE/Computer,%20Technology%20and%20Engineering%20eBooks/Books6/Addison%20Wesley%20-%20Visual%20Modeling%20with%20Rational%20Rose%202000%20and%20UML.pdf.
- [157] Agnar Renolen. "Modelling the Real World: Conceptual Modelling in Spatiotemporal Information System Design." In: *Transactions in GIS* 4.1 (2000), p. 23. ISSN: 13611682. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=3163063&site=ehost-live>.
- [158] M. Roque, B. Vallespir, and G. Doumeingts. "Interoperability in enterprise modelling: Translation, elementary constructs, meta-modelling and {UEML} development". In: *Computers in Industry* 59.7 (2008). Enterprise Integration and Interoperability in Manufacturing Systems, pp. 672–681. ISSN: 0166-3615. DOI: <http://dx.doi.org/10.1016/j.compind.2007.12.017>. URL: <http://www.sciencedirect.com/science/article/pii/S0166361508000420>.
- [159] B Rubenstein-Montano et al. "A systems thinking framework for knowledge management". In: *Decision Support Systems* 31.1 (2001). Knowledge Management Support of Decision Making, pp. 5–16. ISSN: 0167-9236. DOI:

- [http://dx.doi.org/10.1016/S0167-9236\(00\)00116-0](http://dx.doi.org/10.1016/S0167-9236(00)00116-0). URL: <http://www.sciencedirect.com/science/article/pii/S0167923600001>
- [160] M. Sabetzadeh et al. "Consistency Checking of Conceptual Models via Model Merging". In: *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*. 2007, pp. 221–230. DOI: [10.1109/RE.2007.18](https://doi.org/10.1109/RE.2007.18). URL: <http://www.cs.toronto.edu/~chechik/pubs/re07.pdf>.
- [161] Mohamad Sadegh Sangari, Jafar Razmi, and Saeed Zolfaghari. "Developing a practical evaluation framework for identifying critical factors to achieve supply chain agility". In: *Measurement* 62 (2015), pp. 205 –214. ISSN: 0263-2241. DOI: <http://dx.doi.org/10.1016/j.measurement.2014.11.002>. URL: <http://www.sciencedirect.com/science/article/pii/S0263224114005375>.
- [162] Hans-Jurgen Scheruhn, Mark von Rosing, and Richard L. Fallon. "Information Modeling and Process Modeling". In: *The Complete Business Process Handbook*. Ed. by Mark von Rosing, August-Wilhelm Scheer, and Henrik von Scheel. Boston: Morgan Kaufmann, 2015, pp. 511 –550. ISBN: 978-0-12-799959-3. DOI: <http://dx.doi.org/10.1016/B978-0-12-799959-3.00025-2>. URL: <http://www.sciencedirect.com/science/article/pii/B9780127999593000252>.
- [163] Giovanni Schiuma, Daniela Carlucci, and Francesco Sole. "Applying a systems thinking framework to assess knowledge assets dynamics for business performance improvement". In: *Expert Systems with Applications* 39.9 (2012), pp. 8044 –8050. ISSN: 0957-4174. DOI: <http://dx.doi.org/10.1016/j.eswa.2012.01.139>. URL: <http://www.sciencedirect.com/science/article/pii/S0957417412001571>.
- [164] R. Seethamraju and J. Seethamraju. "Enterprise Systems and Business Process Agility - A Case Study". In: *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*. 2009, pp. 1–12. DOI: [10.1109/HICSS.2009.196](https://doi.org/10.1109/HICSS.2009.196).
- [165] Ed Seidewitz. "What models mean". In: *IEEE software* N/A.5 (2003), pp. 26–32. URL: <http://ieeexplore.ieee.org.ezproxy.mdx.ac.uk/stamp/stamp.jsp?tp=&arnumber=1231147&tag=1>.
- [166] Bohdana Sherehiy, Waldemar Karwowski, and John K. Layer. "A review of enterprise agility: Concepts, frameworks, and attributes". In: *International Journal of Industrial Ergonomics* 37.5 (2007), pp. 445 –460. ISSN: 0169-8141. DOI: <http://dx.doi.org/10.1016/j.ergon.2007.01.007>. URL: <http://www.sciencedirect.com/science/article/pii/S0169814107000236>.
- [167] Alberto Rodrigues da Silva. "Model-driven engineering: A survey supported by the unified conceptual model". In: *Computer Languages, Systems*

- & Structures* 43 (2015), pp. 139–155. ISSN: 1477-8424. DOI: <http://dx.doi.org/10.1016/j.cl.2015.06.001>. URL: <http://www.sciencedirect.com/science/article/pii/S1477842415000408>.
- [168] Christian Soltenborn and Gregor Engels. “Using rule overriding to improve reusability and understandability of Dynamic Meta Modeling specifications”. In: *Journal of Visual Languages & Computing* 22.3 (2011). Special Issue on Visual Analytics and Visual Semantics, pp. 233–250. ISSN: 1045-926X. DOI: <http://dx.doi.org/10.1016/j.jvlc.2010.12.005>. URL: <http://www.sciencedirect.com/science/article/pii/S1045926X10000807>.
- [169] Veda C. Storey, Juan C. Trujillo, and Stephen W. Liddle. “Research on conceptual modeling: Themes, topics, and introduction to the special issue”. In: *Data & Knowledge Engineering* In press (2015), In press. ISSN: 0169-023X. DOI: <http://dx.doi.org/10.1016/j.datak.2015.07.002>. URL: <http://www.sciencedirect.com/science/article/pii/S0169023X15000476>.
- [170] Sagar Sunkle, Hemant Rathod, and Vinay Kulkarni. “Practical Goal Modeling for Enterprise Change Context: A Problem Statement”. English. In: *Advances in Conceptual Modeling*. Ed. by Marta Indulska and Sandeep Purao. Vol. 8823. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 139–144. ISBN: 978-3-319-12255-7. DOI: [10.1007/978-3-319-12256-4_15](https://doi.org/10.1007/978-3-319-12256-4_15). URL: http://dx.doi.org/10.1007/978-3-319-12256-4_15.
- [171] Paul P. Tallon and Alain Pinsonneault. “Competing perspectives on the link between strategic information technology alignment and organizational agility: Insights from a mediation model.” In: *MIS Quarterly* 35.2 (2011), pp. 463–486. ISSN: 02767783. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=60461965&site=ehost-live>.
- [172] A.S.M. Tam, L.K. Chu, and D. Sculli. “Business process modelling in small to medium sized enterprises”. In: *Industrial Management & Data Systems* 101.4 (2001), pp. 144–152. DOI: [10.1108/02635570110390107](https://doi.org/10.1108/02635570110390107). eprint: <http://dx.doi.org/10.1108/02635570110390107>. URL: <http://dx.doi.org/10.1108/02635570110390107>.
- [173] WB Teeuw and H Van den Berg. “On the quality of conceptual models”. In: *Proceedings of the ER 97* (1997), pp. 6–7. URL: <http://osm7.cs.byu.edu/ER97/workshop4/tvdb.html>.
- [174] Toby J Teorey. *Database modeling & design*. Morgan Kaufmann, 1999. URL: <http://libvolume2.xyz/biotechnology/semester7/fundamentalsofosanddbms/datadictionaryandquerydesign/datadictionaryandquerydesigntutorial2.pdf>.

- [175] Trinh-Phuong Thao, Alemayehu Molla, and Konrad Peszynski. "Enterprise Systems and Organizational Agility: A Review of the Literature and Conceptual Framework." In: *Communications of the Association for Information Systems* 31 (2012), pp. 167–193. ISSN: 15293181. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=86652303&site=ehost-live>.
- [176] TOGAF The Open Group. *The Open Group Standard, TOGAF Version 9.1: Evaluation Copy*. Van Haren Publishing, 2011. URL: <http://pubs.opengroup.org/architecture/togaf9-doc/arch/>.
- [177] Frank SC Tseng and Chia-Wei Chen. "Integrating heterogeneous data warehouses using XML technologies". In: *Journal of Information Science* 31.3 (2005), pp. 209–229.
- [178] Yi-Hong Tseng and Ching-Torng Lin. "Enhancing enterprise agility by deploying agile drivers, capabilities and providers". In: *Information Sciences* 181.17 (2011), pp. 3693–3708. ISSN: 0020-0255. DOI: <http://dx.doi.org/10.1016/j.ins.2011.04.034>. URL: <http://www.sciencedirect.com/science/article/pii/S0020025511002088>.
- [179] A. Van Lamsweerde. "Goal-oriented requirements engineering: a roundtrip from research to practice [engineering read engineering]". In: *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. 2004, pp. 4–7. DOI: [10.1109/ICRE.2004.1335648](https://doi.org/10.1109/ICRE.2004.1335648).
- [180] A. Van Lamsweerde, R. Darimont, and E. Letier. "Managing conflicts in goal-driven requirements engineering". In: *Software Engineering, IEEE Transactions on* 24.11 (1998), pp. 908–926. ISSN: 0098-5589. DOI: [10.1109/32.730542](https://doi.org/10.1109/32.730542).
- [181] Daniel Vazquez-Bustelo and Lucia Avella. "Agile manufacturing: Industrial case studies in Spain". In: *Technovation* 26.10 (2006), pp. 1147–1161. ISSN: 0166-4972. DOI: <http://dx.doi.org/10.1016/j.technovation.2005.11.006>. URL: <http://www.sciencedirect.com/science/article/pii/S0166497205001768>.
- [182] Daniel Vazquez-Bustelo, Lucía Avella, and Esteban Fernández. "Agility drivers, enablers and outcomes: empirical test of an integrated agile manufacturing model". In: *International Journal of Operations & Production Management* 27.12 (2007), pp. 1303–1332. URL: <http://www.emeraldinsight.com/doi/pdfplus/10.1108/01443570710835633>.
- [183] Riccardo Vecchiato and Claudio Roveda. "Strategic foresight in corporate organizations: Handling the effect and response uncertainty of technology and social drivers of change". In: *Technological Forecasting and Social Change* 77.9 (2010). Strategic Foresight, pp. 1527–1539. ISSN: 0040-1625. DOI: <http://dx.doi.org/10.1016/j.techfore.2009.12.003>.

- URL: <http://www.sciencedirect.com/science/article/pii/S0040162509002091>.
- [184] F. Vernadat. "UEML: towards a unified enterprise modelling language." In: *International Journal of Production Research* 40.17 (2002), pp. 4309–4321. ISSN: 00207543. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=8952691&site=ehost-live>.
- [185] Yair Wand et al. "Theoretical foundations for conceptual modelling in information systems development". In: *Decision Support Systems* 15.4 (1995), pp. 285–304. ISSN: 0167-9236. DOI: [http://dx.doi.org/10.1016/0167-9236\(94\)00043-6](http://dx.doi.org/10.1016/0167-9236(94)00043-6). URL: <http://www.sciencedirect.com/science/article/pii/0167923694000436>.
- [186] H. Wang et al. "An enterprise modelling method based on an extension mechanism." In: *International Journal of Computer Integrated Manufacturing* 22.9 (2009), pp. 836–846. ISSN: 0951192X. URL: <http://search.ebscohost.com.ezproxy2.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=43840687&site=ehost-live>.
- [187] Zheng Wang et al. "Achieving IT-Enabled Enterprise Agility in China: An IT Organizational Identity Perspective". In: *Engineering Management, IEEE Transactions on* 61.1 (2014), pp. 182–195. ISSN: 0018-9391. DOI: [10.1109/TEM.2013.2259494](https://doi.org/10.1109/TEM.2013.2259494).
- [188] Lorraine Warren and Peter Adman. "The use of critical systems thinking in designing a system for a university information systems support service." In: *Information Systems Journal* 9.3 (1999), pp. 223–242. ISSN: 13501917. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=5608015&site=ehost-live>.
- [189] Ron Weber. "Are Attributes Entities? A Study of Database Designers' Memory Structures." In: *Information Systems Research* 7.2 (1996), pp. 137–162. ISSN: 10477047. URL: <http://search.ebscohost.com.ezproxy.mdx.ac.uk/login.aspx?direct=true&db=bth&AN=4430740&site=ehost-live>.
- [190] R. Wendler. "Development of the organizational agility maturity model". In: *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. 2014, pp. 1197–1206. DOI: [10.15439/2014F79](https://doi.org/10.15439/2014F79). URL: http://annals-csis.org/Volume_2/pliks/79.pdf.
- [191] Martin Wolf, Jens Vykoukal, and Roman Beck. "Services Grid Competence as Driver of Business Agility in Turbulent Environments A Conceptual Model in the Financial Services Industry". English. In: *43RD HAWAII INTERNATIONAL CONFERENCE ON SYSTEMS SCIENCES VOLS 1-5 (HICSS 2010)*. Proceedings of the Annual Hawaii International Conference on System Sciences. 43rd Hawaii International Conference on Systems Sciences

- (HICSS 2010), Honolulu, HI, JAN 05-08, 2010. Univ Hawaii, Shidler Coll Business. 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA: IEEE COMPUTER SOC, 2010, 3731–3740. ISBN: 978-1-4244-5509-6. URL: <http://www.computer.org/csdl/proceedings/hicss/2010/3869/00/09-04-06.pdf>.
- [192] Michael Wooldridge, Nicholas R Jennings, and David Kinny. “The Gaia methodology for agent-oriented analysis and design”. In: *Autonomous Agents and multi-agent systems 3.3* (2000), pp. 285–312. URL: <http://eprints.soton.ac.uk/253748/1/jaamas2000.pdf>.
- [193] Michael Wooldridge and Paolo Ciancarini. “Agent-oriented software engineering: The state of the art”. In: *Agent-oriented software engineering*. Springer, 2001, pp. 1–28. URL: <http://www2.hawaii.edu/~nreed/ics606/papers/wooldridge01aose1957.pdf>.
- [194] E. Yu, M. Strohmaier, and Xiaoxue Deng. “Exploring Intentional Modeling and Analysis for Enterprise Architecture”. In: *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW '06. 10th IEEE International*. 2006, pp. 32–32. DOI: [10.1109/EDOCW.2006.36](https://doi.org/10.1109/EDOCW.2006.36).
- [195] Eric Yu. “Agent-oriented modelling: software versus the world”. In: *Agent-Oriented Software Engineering II*. Springer, 2002, pp. 206–225. URL: <ftp://learning.cs.toronto.edu/dist/eric/eric/AOSE01.pdf>.
- [196] Eric Yu. “Modelling strategic relationships for process reengineering”. In: *Social Modeling for Requirements Engineering 11* (2011), p. 2011. URL: <http://ftp.cs.toronto.edu/pub/eric/DKBS-TR-94-6.pdf>.
- [197] Xiaodan Yu and Stacie Petter. “Understanding agile software development practices using shared mental models theory”. In: *Information and Software Technology 56.8* (2014), pp. 911–921. ISSN: 0950-5849. DOI: <http://dx.doi.org/10.1016/j.infsof.2014.02.010>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584914000524>.
- [198] J. A. Zachman. “A framework for information systems architecture”. In: *IBM Systems Journal 26.3* (1987), pp. 276–292. ISSN: 0018-8670. DOI: [10.1147/sj.263.0276](https://doi.org/10.1147/sj.263.0276).
- [199] J.A. Zachman. “A framework for information systems architecture”. In: *IBM Systems Journal 38.2.3* (1999), pp. 454–470. ISSN: 0018-8670. DOI: [10.1147/sj.382.0454](https://doi.org/10.1147/sj.382.0454).
- [200] John Zachman. “The zachman framework for enterprise architecture”. In: *Zachman International 4* (2002), p. 33. URL: http://www.businessrulesgroup.org/BRWG_RFI/ZachmanBookRFIextract.pdf.
- [201] Pamela Zave. “Classification of Research Efforts in Requirements Engineering”. In: *ACM Comput. Surv.* 29.4 (Dec. 1997), pp. 315–321. ISSN: 0360-0300.

DOI: 10.1145/267580.267581. URL: <http://doi.acm.org/10.1145/267580.267581>.

Appendix A

Abstract Syntax Model of the Change Modelling Language

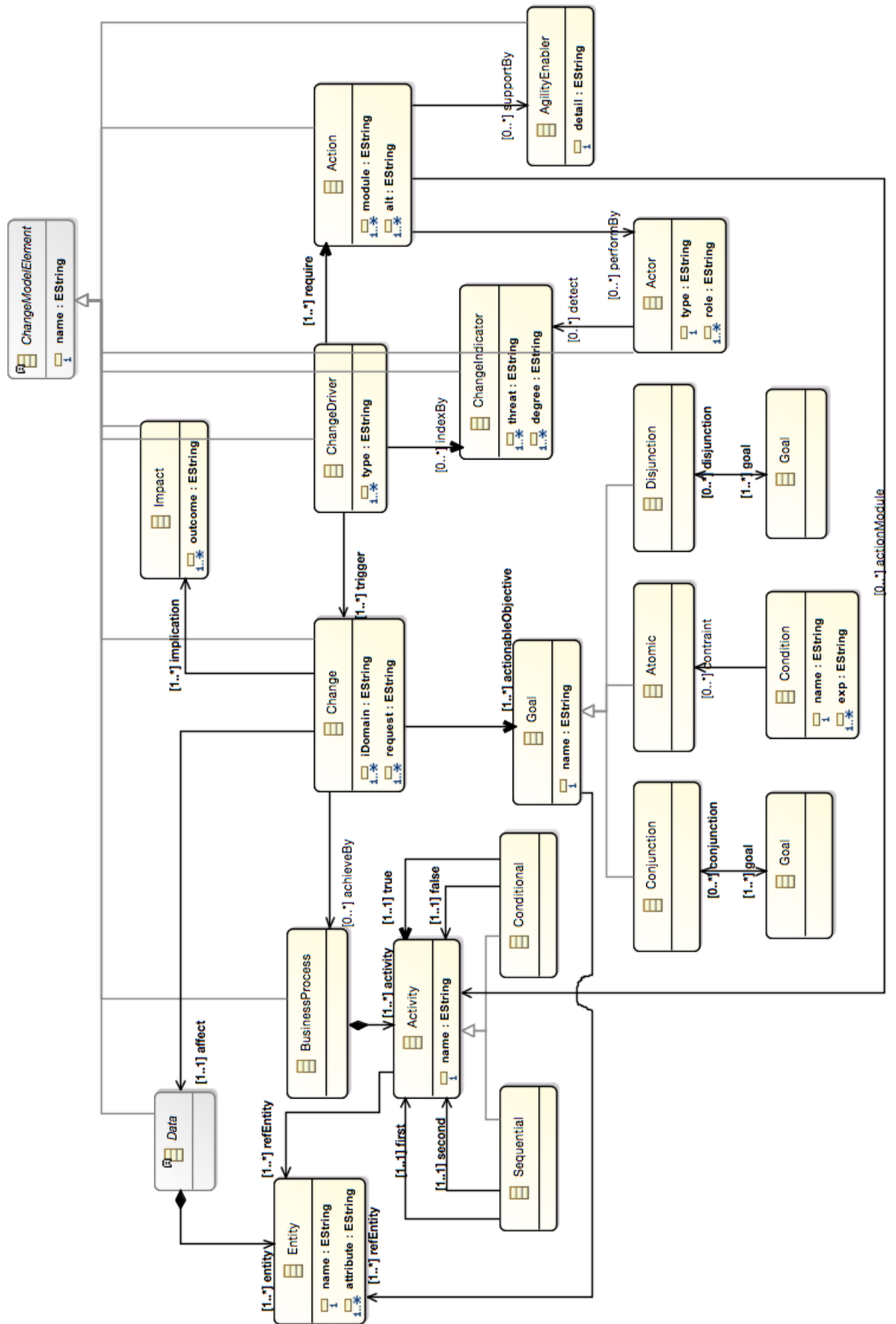


FIGURE A.1: Abstract Syntax Model of the Proposed Modelling Language

Appendix B

Abstract Syntax Model of Enterprise

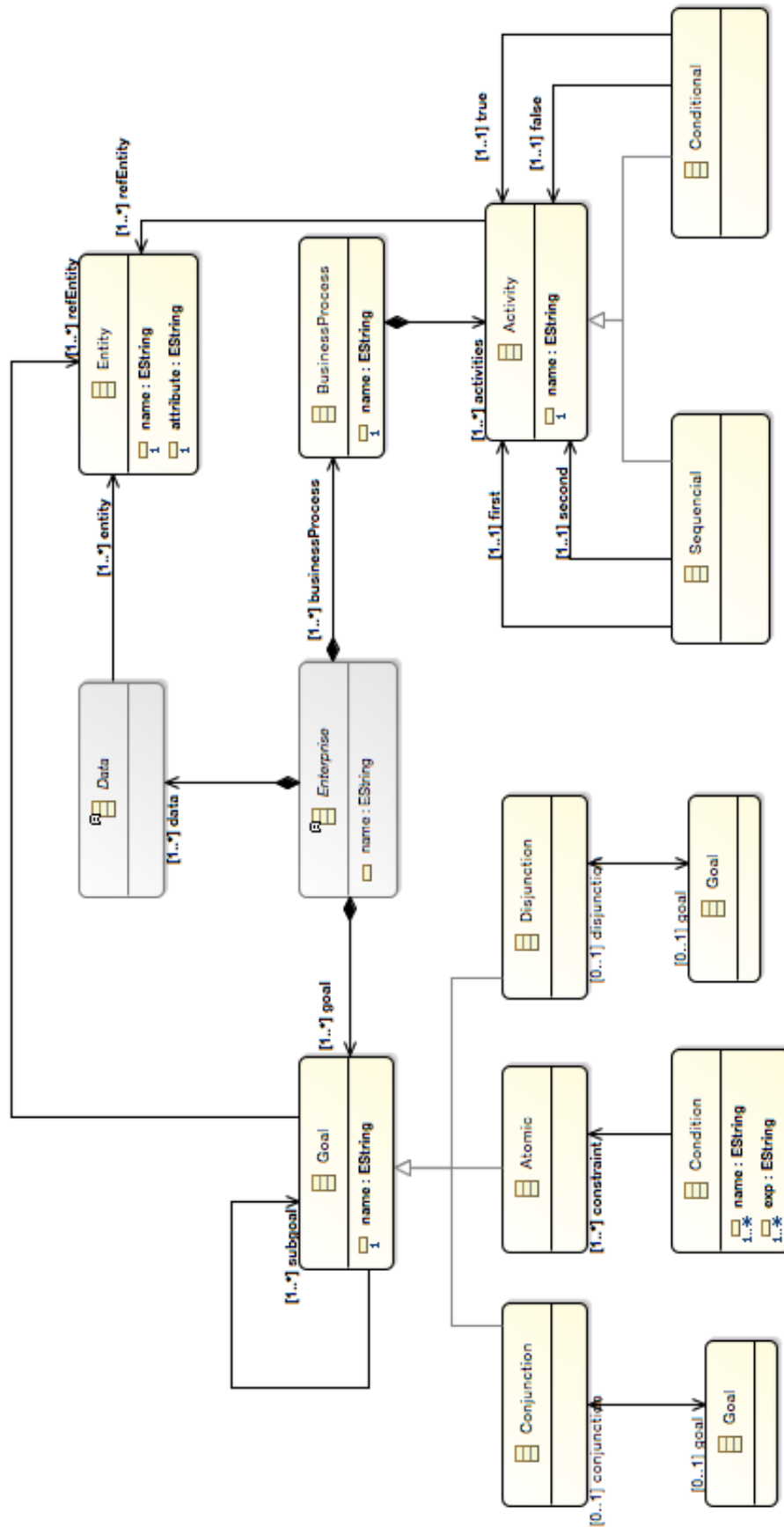


FIGURE B.1: Abstract Syntax Model of Enterprise

Appendix C

Enlarged Models of Case Study

C.1 Enlarged Change Model From Case Study: Case 1

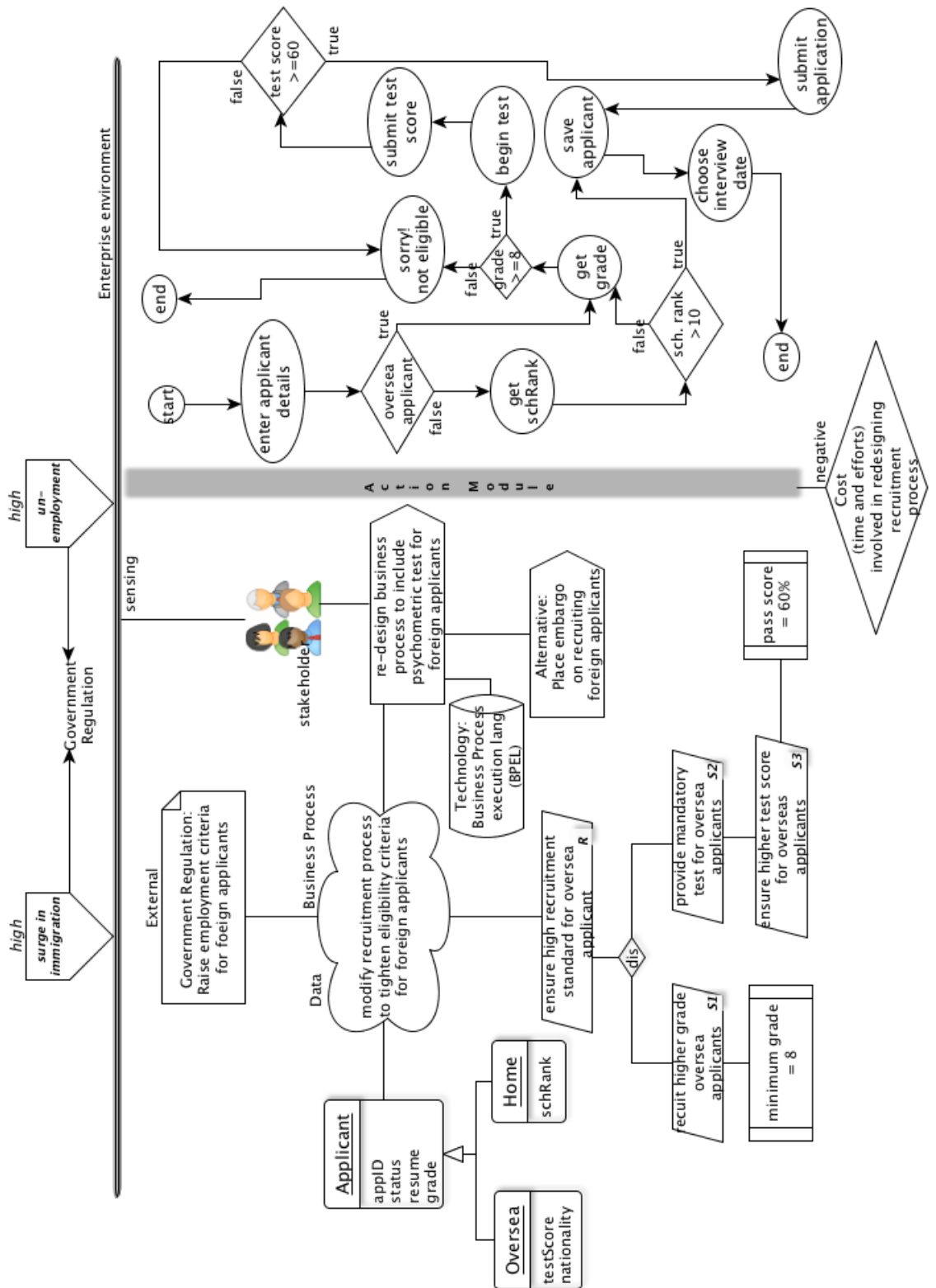


FIGURE C.1: Enlarged Change Model of the case Study: Case 1

C.2 Enlarged Notation for Change Modelling










Construct	Symbol	Construct	Symbol	Construct	Symbol
<i>Agility enabler</i>		<i>Change</i>		<i>impact</i>	
<i>Change driver</i>		<i>Tasks</i>		<i>Change Indicator</i>	
<i>Condition</i>		<i>Action</i>		<i>Enterprise boundary</i>	

FIGURE C.2: Notations used for Change Modelling

C.3 Enlarged Change Model From Case Study: Case 2

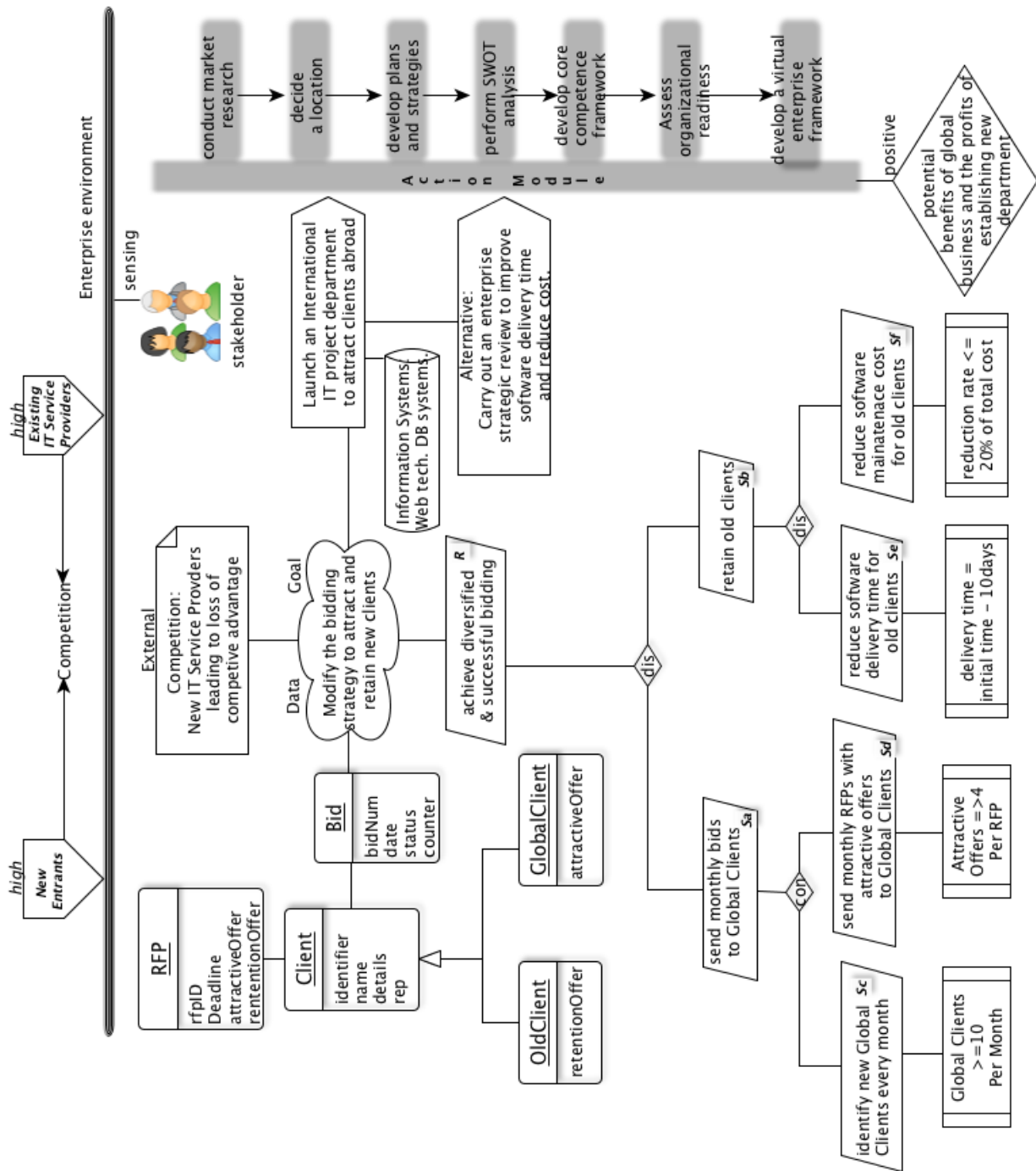


FIGURE C.3: Enlarged Change Model of the case Study: Case 2

Appendix D

Enlarged Case Study Goal Model

D.1 Enlarged Case Study Goal Model

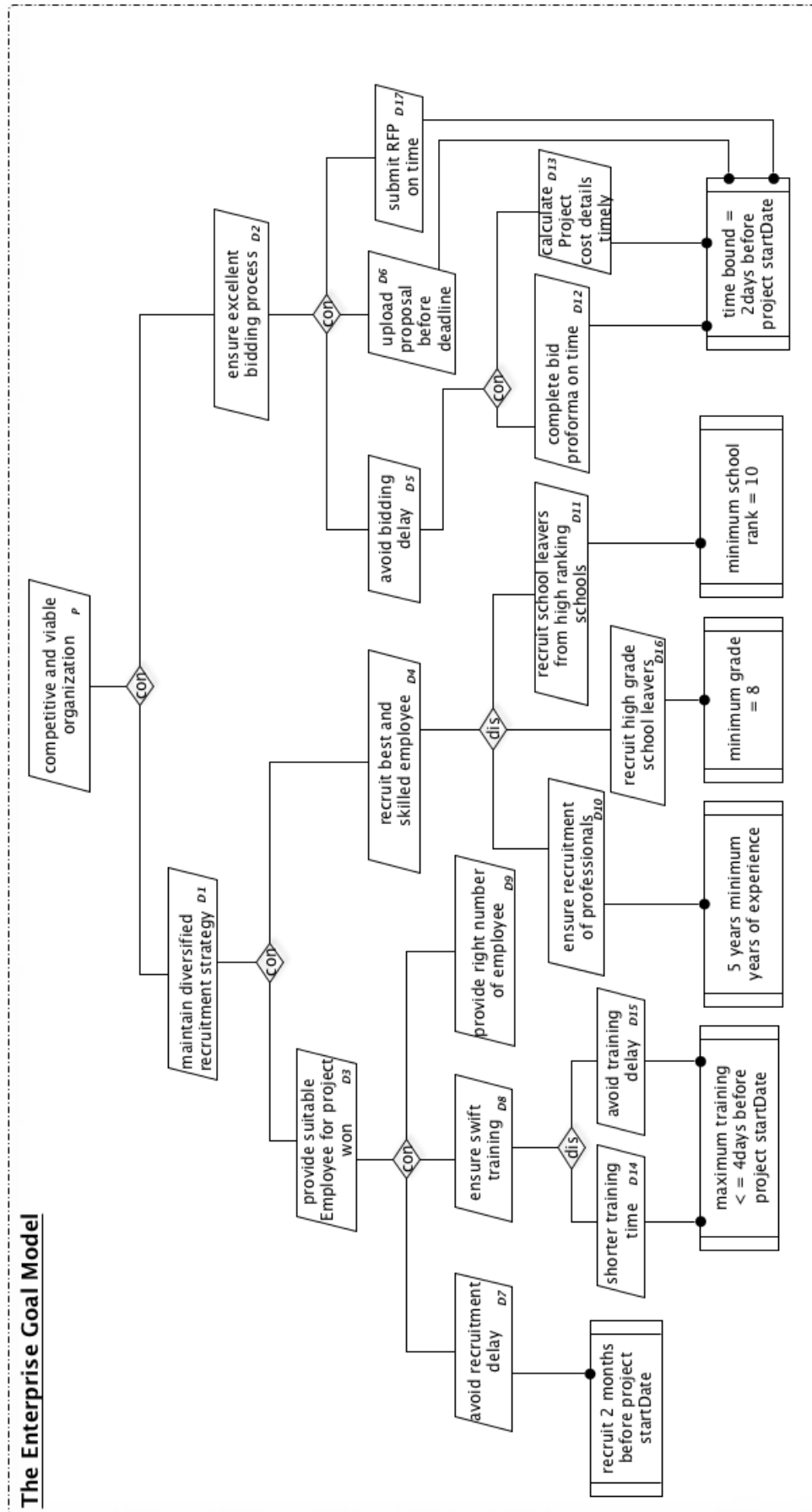
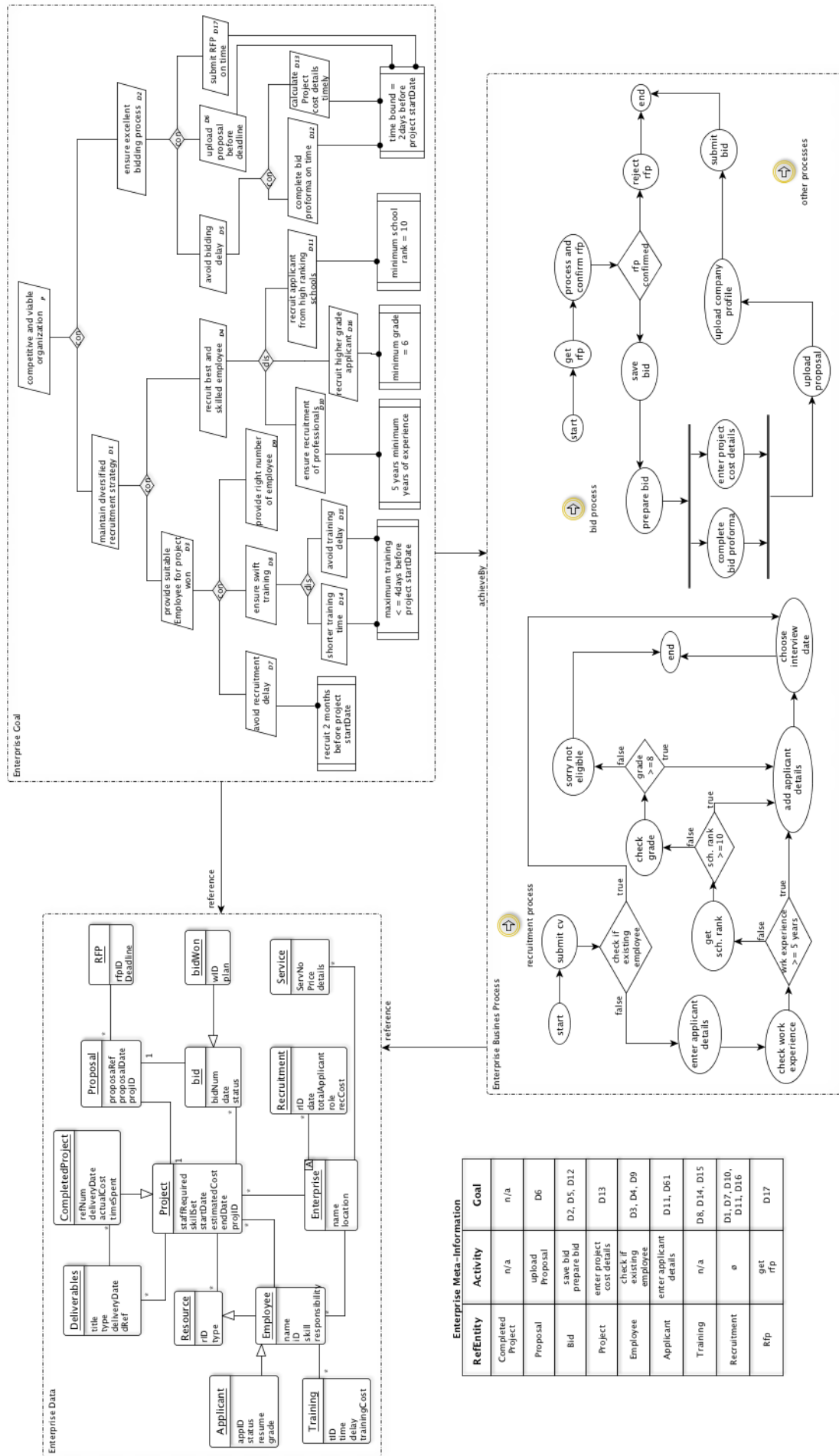


FIGURE D.1: Case Study Enterprise Goal Model

D.2 Enlarged Case Study Enterprise Model



Appendix E

Application of the Proposed Change Modelling Language in Database Implementation

E.1 Application Part 4: Database Implementation

The conceptual modelling language proposed in this thesis can also be applied to implement a database that can be used to support enterprise agility and change management. Information systems, such as databases, are known to developed from conceptual schemas or modelling language. Hence, the conceptual modelling language proposed in this research can be used as a conceptual schema to implement a database. Such database would be useful to change management and enterprise agility in a number of ways.

Database systems are well known to provide the means to capture, document, and manage vital data in a given domain. Therefore, a change management database can be used to capture, document, and manage enterprise agility and change management data. For instance, the origin, causes, and impact of change, as well as the activities, actions, and initiatives carried to manage changes can be documented and managed using a change management database system. By capturing, documenting, and managing these data, enterprise stakeholders can easily access data and information required to manage changes and support enterprise agility initiatives.

This sections shows how the proposed conceptual modelling language can be used to implement a database system. The well established steps to be followed while implementing database systems have been described in [16, 174]. These steps are adapted and used to implement the proposed database system. The first step is to construct the conceptual model, this is then used to produced the relational model. The relational is then implemented as table *i.e.*, the physical database. Data can then be inserted into the table, and queried over time to obtain useful business intelligence. Each of these steps are described below.

The Conceptual Model: Conceptual models used for database implementations are usually developed by gathering concepts of interest in a given universe of discourse. These concepts are then integrated into a conceptual modelling language. The concepts used for implementing the proposed database have been identified and discussed in Chapter 2 of this thesis. These concepts have also been integrated into a conceptual modelling language in Chapter 5. Hence, the abstract syntax model of the proposed change conceptual modelling language is used as the conceptual database model. This has been discussed in Chapter 5.

The Relational Model: The relational database model is used to convert each concept in the conceptual model into data element that can be stored in a database. Data elements extends concepts with additional details such as the data type of each attribute. In addition the data elements include the primary and foreign keys for each table, and the generally constraints to guarantee the integrity of tables in the database [21, 39].

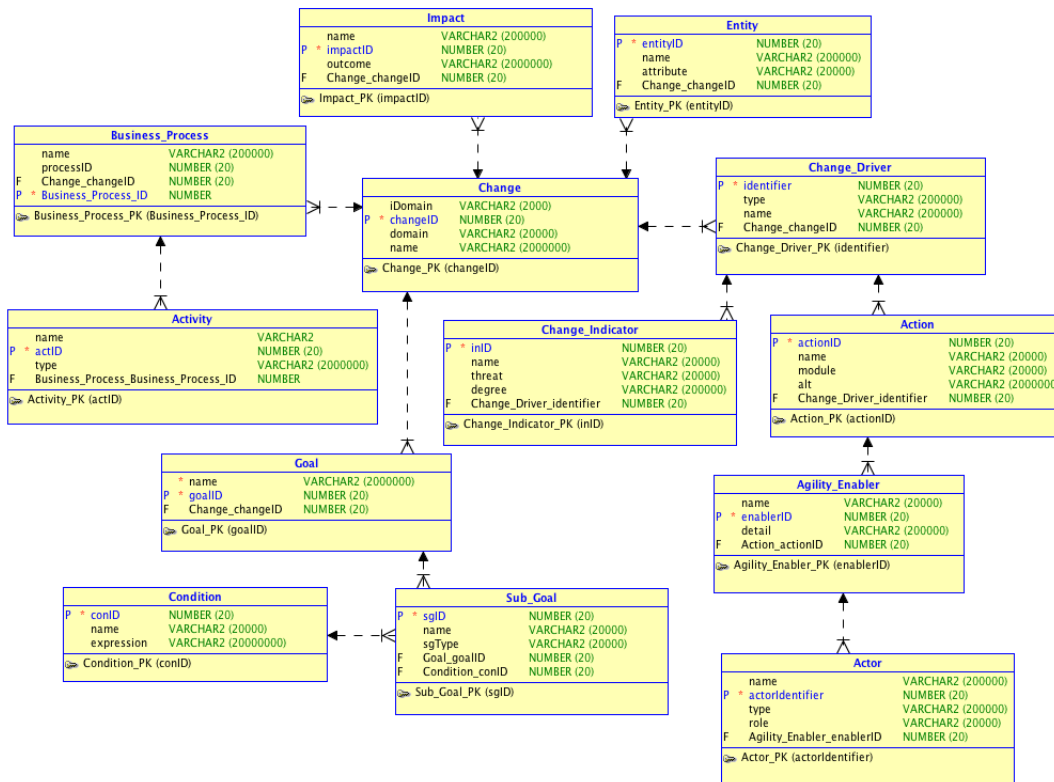


FIGURE E.1: Relational Database Model

The relational database model used to implement the proposed database is shown in Figure E.1. As seen in this figure, the attributes in each data element include data types such as *VARCHAR* and *NUMBER*. Each data type has a precision, *i.e.*, the allowable character for that attribute. Defining data type and precision for each attribute helps to constraint the database to ensure data integrity. In addition, a unique identifier called the primary key is included in each data element, this is to ensure the uniqueness of each table in the database. More so, the relational model shows how one (parent) data element relates or references another (referent) data

element by assigning an attribute, usually the primary key, of the parent table to the referent table. The assigned attribute is called the foreign key.

The Physical Database or Table: The next step after constructing the relational model is to create the physical database. The physical database is in form of a table with columns and rows. Each table corresponds to a data element in the relational database model, while the columns of the table are the attributes of corresponding data elements in the relational model. Usually, a structured query language is used to implement the physical database.

The structured query language (SQL) is known to be the de facto standard for implementing and managing information in a database. It includes a data definition language and a data manipulation language. The data definition language can be used to describe or create the tables in the physical database. After creating the database, the data manipulation language (DML) can be used to insert, update, and manage the data in the database. The tables in the database proposed in this research is created using the data definition language of the SQL. The SQL codes, shown in Listing E.1, are used to create these tables.

LISTING E.1: SQL Codes For Creating Change-Driver & Change-Indicator Tables

```
1 CREATE TABLE Change_Driver
2 (
3     identifier      NUMBER (20) NOT NULL ,
4     type            VARCHAR2 (200000) ,
5     name            VARCHAR2 (200000) ,
6     Change_changeID NUMBER (20)
7 ) ;
8 ALTER TABLE Change_Driver ADD CONSTRAINT Change_Driver_PK PRIMARY KEY ( identifier ) ;
9
10
11
12 CREATE TABLE Change_Indicator
13 (
14     inID            NUMBER (20) NOT NULL ,
15     name            VARCHAR2 (20000) ,
16     threat          VARCHAR2 (20000) ,
17     degree          VARCHAR2 (200000) ,
18     Change_Driver_identifier NUMBER (20)
19 ) ;
20 ALTER TABLE Change_Indicator ADD CONSTRAINT
21     Change_Indicator_PK PRIMARY KEY ( inID ) ;
22
```

```
23 ALTER TABLE Change_Driver ADD CONSTRAINT Change_Driver_Change_FK FOREIGN KEY
24 ( Change_changeID ) REFERENCES Change ( changeID ) ;
25
26 ALTER TABLE Change_Indicator ADD CONSTRAINT Change_Indicator_Change_Driver_FK FOREIGN KEY
27 ( Change_Driver_identifier ) REFERENCES Change_Driver ( identifier ) ;
28
29
30 CREATE TABLE Action
31 (
32     actionID          NUMBER (20) NOT NULL ,
33     name              VARCHAR2 (20000) ,
34     module            VARCHAR2 (20000) ,
35     alt               VARCHAR2 (2000000) ,
36     Change_Driver_identifier NUMBER (20)
37 ) ;
38 ALTER TABLE Action ADD CONSTRAINT Action_PK PRIMARY KEY ( actionID ) ;
39
40
41 CREATE TABLE Activity
42 (
43     name              VARCHAR2 ,
44     actID             NUMBER (20) NOT NULL ,
45     type              VARCHAR2 (2000000) ,
46     -- ERROR: Column name length exceeds maximum allowed length(30)
47     Business_Process_Business_Process_ID NUMBER
48 ) ;
49 ALTER TABLE Activity ADD CONSTRAINT Activity_PK PRIMARY KEY ( actID ) ;
50
51 CREATE TABLE Actor
52 (
53     name              VARCHAR2 (200000) ,
54     actorIdentifier   NUMBER (20) NOT NULL ,
55     type              VARCHAR2 (200000) ,
56     role              VARCHAR2 (20000) ,
57     Agility_Enabler_enablerID NUMBER (20)
58 ) ;
59 ALTER TABLE Actor ADD CONSTRAINT Actor_PK PRIMARY KEY ( actorIdentifier ) ;
60
61
62 CREATE TABLE Agility_Enabler
63 (
64     name              VARCHAR2 (20000) ,
65     enablerID         NUMBER (20) NOT NULL ,
66     detail            VARCHAR2 (200000) ,
67     Action_actionID  NUMBER (20)
68 ) ;
69 ALTER TABLE Agility_Enabler ADD CONSTRAINT Agility_Enabler_PK PRIMARY KEY ( enablerID ) ;
70
```

```
71 CREATE TABLE Business_Process
72 (
73     name          VARCHAR2 (200000) ,
74     processID     NUMBER (20) ,
75     Change_changeID NUMBER (20) ,
76     Business_Process_ID NUMBER NOT NULL
77 ) ;
78 ALTER TABLE Business_Process ADD CONSTRAINT Business_Process_PK PRIMARY KEY ( Business_Process_ID
79
80
81 CREATE TABLE Change
82 (
83     iDomain VARCHAR2 (2000) ,
84     changeID NUMBER (20) NOT NULL ,
85     domain  VARCHAR2 (20000) ,
86     name    VARCHAR2 (2000000)
87 ) ;
88 ALTER TABLE Change ADD CONSTRAINT Change_PK PRIMARY KEY ( changeID ) ;
89
90 CREATE TABLE Change_Driver
91 (
92     identifier    NUMBER (20) NOT NULL ,
93     type          VARCHAR2 (200000) ,
94     name          VARCHAR2 (200000) ,
95     Change_changeID NUMBER (20)
96 ) ;
97 ALTER TABLE Change_Driver ADD CONSTRAINT Change_Driver_PK PRIMARY KEY ( identifier ) ;
98
99
100 CREATE TABLE Change_Indicator
101 (
102     inID          NUMBER (20) NOT NULL ,
103     name          VARCHAR2 (20000) ,
104     threat        VARCHAR2 (20000) ,
105     degree        VARCHAR2 (200000) ,
106     Change_Driver_identifier NUMBER (20)
107 ) ;
108 ALTER TABLE Change_Indicator ADD CONSTRAINT Change_Indicator_PK PRIMARY KEY ( inID ) ;
109
110
111 CREATE TABLE Condition
112 (
113     conID        NUMBER (20) NOT NULL ,
114     name         VARCHAR2 (20000) ,
115     expression   VARCHAR2 (2000000)
116 ) ;
117 ALTER TABLE Condition ADD CONSTRAINT Condition_PK PRIMARY KEY ( conID ) ;
118
```



```
119
120 CREATE TABLE Entity
121 (
122     entityID      NUMBER (20) NOT NULL ,
123     name          VARCHAR2 (200000) ,
124     attribute     VARCHAR2 (20000) ,
125     Change_changeID NUMBER (20)
126 ) ;
127 ALTER TABLE Entity ADD CONSTRAINT Entity_PK PRIMARY KEY ( entityID ) ;
128
129
130 CREATE TABLE Goal
131 (
132     name          VARCHAR2 (2000000) NOT NULL ,
133     goalID        NUMBER (20) NOT NULL ,
134     Change_changeID NUMBER (20)
135 ) ;
136 ALTER TABLE Goal ADD CONSTRAINT Goal_PK PRIMARY KEY ( goalID ) ;
137
138
139 CREATE TABLE Impact
140 (
141     name          VARCHAR2 (200000) ,
142     impactID      NUMBER (20) NOT NULL ,
143     outcome       VARCHAR2 (2000000) ,
144     Change_changeID NUMBER (20)
145 ) ;
146 ALTER TABLE Impact ADD CONSTRAINT Impact_PK PRIMARY KEY ( impactID ) ;
147
148
149 CREATE TABLE Sub_Goal
150 (
151     sgID          NUMBER (20) NOT NULL ,
152     name          VARCHAR2 (20000) ,
153     sgType        VARCHAR2 (20000) ,
154     Goal_goalID   NUMBER (20) ,
155     Condition_conID NUMBER (20)
156 ) ;
157
158
159 ALTER TABLE Sub_Goal ADD CONSTRAINT Sub_Goal_PK PRIMARY KEY ( sgID ) ;
160
161 ALTER TABLE Action ADD CONSTRAINT Action_Change_Driver_FK FOREIGN KEY
162 ( Change_Driver_identififier ) REFERENCES Change_Driver ( identififier ) ;
163
164 ALTER TABLE Activity ADD CONSTRAINT Activity_Business_Process_FK FOREIGN KEY
165 ( Business_Process_Business_Process_ID ) REFERENCES Business_Process ( Business_Process_ID ) ;
166
```

```

167 ALTER TABLE Actor ADD CONSTRAINT Actor_Agility_Enabler_FK FOREIGN KEY
168 ( Agility_Enabler_enablerID ) REFERENCES Agility_Enabler ( enablerID ) ;
169
170 ALTER TABLE Agility_Enabler ADD CONSTRAINT Agility_Enabler_Action_FK FOREIGN KEY
171 ( Action_actionID ) REFERENCES Action ( actionID ) ;
172
173 ALTER TABLE Business_Process ADD CONSTRAINT Business_Process_Change_FK FOREIGN KEY
174 ( Change_changeID ) REFERENCES Change ( changeID ) ;
175
176 ALTER TABLE Change_Driver ADD CONSTRAINT Change_Driver_Change_FK FOREIGN KEY
177 ( Change_changeID ) REFERENCES Change ( changeID ) ;
178
179 -- ERROR: FK name length exceeds maximum allowed length(30)
180 ALTER TABLE Change_Indicator ADD CONSTRAINT Change_Indicator_Change_Driver_FK FOREIGN KEY
181 ( Change_Driver_identifier ) REFERENCES Change_Driver ( identifier ) ;
182
183 ALTER TABLE Entity ADD CONSTRAINT Entity_Change_FK FOREIGN KEY
184 ( Change_changeID ) REFERENCES Change ( changeID ) ;
185
186 ALTER TABLE Goal ADD CONSTRAINT Goal_Change_FK FOREIGN KEY
187 ( Change_changeID ) REFERENCES Change ( changeID ) ;
188
189 ALTER TABLE Impact ADD CONSTRAINT Impact_Change_FK FOREIGN KEY
190 ( Change_changeID ) REFERENCES Change ( changeID ) ;
191
192 ALTER TABLE Sub_Goal ADD CONSTRAINT Sub_Goal_Condition_FK FOREIGN KEY
193 ( Condition_conID ) REFERENCES Condition ( conID ) ;
194
195 ALTER TABLE Sub_Goal ADD CONSTRAINT Sub_Goal_Goal_FK FOREIGN KEY
196 ( Goal_goalID ) REFERENCES Goal ( goalID ) ;
197
198 CREATE SEQUENCE Business_Process_Business_Proc START WITH 1 NOCACHE ORDER ;
199 CREATE OR REPLACE TRIGGER Business_Process_Business_Proc BEFORE
200 INSERT ON Business_Process FOR EACH ROW WHEN (NEW.Business_Process_ID IS NULL)
201 BEGIN :NEW.Business_Process_ID := Business_Process_Business_Proc.NEXTVAL;
202 END;
203 /
204
205
206 -- Oracle SQL Developer Data Modeler Summary Report:
207 --
208 -- CREATE TABLE 13
209 -- CREATE INDEX
210 -- ALTER TABLE 25
211 -- CREATE VIEW
212 --

```

```
212 -- CREATE PACKAGE
    0
213 -- CREATE PACKAGE BODY
    0
214 -- CREATE PROCEDURE
    0
215 -- CREATE FUNCTION
    0
216 -- CREATE TRIGGER
    1
217 -- ALTER TRIGGER
    0
218 -- CREATE COLLECTION TYPE
    0
219 -- CREATE STRUCTURED TYPE
    0
220 -- CREATE STRUCTURED TYPE BODY
    0
221 -- CREATE CLUSTER
    0
222 -- CREATE CONTEXT
    0
223 -- CREATE DATABASE
    0
224 -- CREATE DIMENSION
    0
225 -- CREATE DIRECTORY
    0
226 -- CREATE DISK GROUP
    0
227 -- CREATE ROLE
    0
228 -- CREATE ROLLBACK SEGMENT
    0
229 -- CREATE SEQUENCE
    1
230 -- CREATE MATERIALIZED VIEW
    0
231 -- CREATE SYNONYM
    0
232 -- CREATE TABLESPACE
    0
233 -- CREATE USER
    0
234 --
235 -- DROP TABLESPACE
    0
```

```

236 -- DROP DATABASE
      0
237 --
238 -- ERRORS
      2
239 -- WARNINGS
      0

```

Inserting Data: SQL data manipulation language can be used to insert change management data into the created tables. For instance, Listing E.2 shows how data can be inserted into the change driver table. These data can be captured using proforma in Figure 7.3. The proforma can be manually completed by enterprise stakeholders responsible for change management. It can also be possible to implement this proforma as web form using web development technologies such as the hypertext mark-up language (HTML).

LISTING E.2: SQL Codes For Inserting Data into the Change-Driver Table

```

1 INSERT INTO Change_Driver
2 VALUES ('cd0001', 'external', 'government regulation: increase
3 corporation tax by 10% for all electronic goods', 'c2337a');
4 VALUES ('cd0034', 'external', 'competition: new entrant into
5 the industry offering discounts to new customers', 'c2345a');

```

Querying the Database: Apart from its applications in data management, this type of database can be used to obtain useful information, knowledge, and business intelligence to support change management initiatives. For instance, let's say the enterprise, described in Section ??, experiences a new change driver, say *regulatory compliance*. Assuming this enterprise has a functional database for storing and managing previous change management activities; and has stored previous change management initiatives for this regulatory compliance change driver.

Then, this database can be queried, using SQL data manipulation language, to obtain previously stored information about regulatory compliance change driver. For example, the enterprise change managers may want to know the actions that were previously used to manage this type of change driver. This type of information can be obtained from the database using the following SQL query in Listing E.3 below:

LISTING E.3: SQL Codes For Inserting Data into the Change-Driver Table

```
1 SELECT action.ACTIONID,  
2   action.NAME,  
3   action.MODULE,  
4   action.CHANGE_DRIVER_IDENTIFIER,  
5 FROM action  
6 INNER JOIN change_driver  
7 ON action.CHANGE_DRIVER_IDENTIFIER  
8 = change_driver.IDENTIFIER  
9 AND change_driver.NAME = "Regulatory Compliance" ;
```

This type of information can make it easier for enterprise stakeholders to examine the actions used to manage previous regulatory compliance change drivers. Then based on this information, they can make adequate decision on the suitable action to select for managing the current change driver. This can also save the time, efforts, and cost associated with formulating or making decisions about the actions to adopt for change management and enterprise agility.

E.1.1 Lessons From Database Implementation

A database has been implemented using the conceptual modelling language proposed in this research. SQL queries have been used to demonstrate the the implementation details of this database, as well as how change management data can be inserted into it. The various ways in which this database can be useful to enterprise agility and change management are discussed in the paragraphs below.

As an enterprise interacts with its environment, respond to change drivers, and manage changes, data are either being used or generated. Examples of such data could be the root cause of a change, the actions carried to manage a change, etc. These data should not be discarded, since they can be used to derive knowledge that would be useful in subsequent change management initiatives. Instead, they can captured and stored using this database implemented with the proposed change modelling language. SQL data manipulation languages can then be used to query this database to obtain useful information and knowledge that can support effective change management. For instance, previously stored change management strategies and actions, which are successful, can be retrieved and re-used to support current and future change management initiatives. This can make the information required to make change management decisions handy and readily available when needed. Thereby shortening the time and effort required to manage changes and achieve enterprise agility.

Furthermore, since database systems provide the means of managing, organising, and structuring data coherently, as well as make data easily accessible. The

database implemented the proposed conceptual model can make it easier to access and share information about enterprise changes, agility, and change management. Information sharing as well as easy access to information are known to enhance enterprise agility and change management [65]. Therefore the implemented database can facilitate enterprise agility change management by providing the means to easily access and share data, information, and knowledge generated from managing enterprise changes. More so, over time, these data can be used to build a data warehouse for enterprise agility. Data mining, big data, and data analytic techniques and technologies can then be used to obtain useful business intelligence about change management and enterprise agility.

Finally, the implemented database can be improved in the following ways: The change management proforma presented in Section 7.3 can be implemented as a web form. This can be done using any web development technology such as the hyper text mark-up language (HTML). Then this web form can be connected to this implemented database using any server side web technology, such as the hypertext preprocessor (PHP) or java database connectivity (JDBC). These can be implemented as an information system that can allow enterprises capture and store change management data. This type of information system can provide an easier to use and a more interactive user interface than using SQL codes to insert data into the database. The would also be part of the future work of this research.

Appendix F

Formalising the Rules

F.1 Rule 1

Expression **F.2** below shows an example of how *Rule 1* can be formalised using formal logic.

RULE 1

Predicate: Let P represent the predicate: "is entity of Change Model" · Q represents: "is entity AS-IS Enterprise Model" · R represents: "entity of TO-BE Enterprise Model"

Expression.

$$\top(\exists_x R(x)) \iff \exists_x P(x) \wedge \neg(\exists_x Q(x)) \quad (\text{F.1})$$

Explanation F.1. It is true that there exist any x , such that x is an entity of the TO-BE Enterprise Model, if and only if, x exist as an entity of Change Model and x does not exist as an entity of the AS-IS Enterprise Model.

F.2 Rule 2

Expression **F.2** below shows an example of how *Rule 2* can be formalised using formal logic.

RULE 2

Predicate. Let S represents the predicate "is activity of Change Model" · T represents: "is activity of AS-IS Enterprise Model" · U represents: "is activity TO-BE Enterprise Model"

Expression.

$$\forall_y U(y) \implies \top[\exists_y S(y) \wedge \nexists_y T(y)] \quad (\text{F.2})$$

Explanation F.2. For all y , such that y is an activity of the TO-BE enterprise model, implies that it is true that y exist as an activity of the Change Model but does not exist as an Activity of the AS-IS Enterprise Model.

F.3 Rule 3

Expression **F.3** below shows an example of how *Rule 3* can be formalised using formal logic.

RULE 3

Predicate. Let V represent the predicate: "is goal of Change Model" · W represents: "is goal of AS-IS Enterprise Model" · X represents: "goal of TO-BE Enterprise Model"

Expression.

$$\exists_z X(z) \iff \exists_z V(z) \wedge \neg(\exists_z W(z)) \quad (\text{F.3})$$

Explanation F.3. It is true that there exist any z , such that z is a goal of the TO-BE Enterprise Model, if and only if, z exist as a goal of the Change Model and z does not exist as a goal of the AS-IS Enterprise Model.