Solúbtha : A Model for Designing Flexible Business Transactions of Purchase Order Management Processes

A.K.M. Rafiqul Haque

M. Sc. in Computer Science
B. Sc. in Computer Science

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Supervised by
Dr. Ita Richardson
Dr. Eoin Whelan
Professor dr. Micheal P. Papazoglou
Professor dr. Willem-jan van den Heuvel
Dr. Yehia Taher

Submitted to the UNIVERSITY of LIMERICK, June 2014
Declaration


Author: A.K.M. Rafiqul Haque.

Award: Doctor of Philosophy.

Supervisor: Dr. Ita Richardson
Dr. Eoin Whelan
Professor dr. Micheal P. Papazoglou
Professor dr. Willem-jan van den Heuvel
Dr. Yehia Taher

I hereby declare that this thesis is entirely my own work, and does not contain material previously published by any other author, except where due reference or acknowledgement has been made. Furthermore, I declare that it has not previously been submitted for any other academic award.

4 June 2014

Except where excerpts from the contributions of others are explicitly noted, this text is copyright © 2011–2014 A.K.M. Rafiqul Haque.
Abstract

**Solúbtha : A Model for Designing Flexible Business Transactions of Purchase Order Management Processes**  

A.K.M. Rafiqul Haque

With the advent of Service Oriented Architecture (SOA), organizations tend to use service-based applications (SBAs) that are dynamic, distributed, and ubiquitous. In recent years, SBAs have become the state of the art technologies for building large-scale and cost-effective applications on the web. SBAs that automate end-to-end business processes typically involve well-defined standard composite activities that are hosted at the sites of business partners. These activities encapsulate one to many operations. A business transaction is the successful execution of these operations that must satisfy the trading partners agreements which contain the clauses agreed upon by the business partners. In order to ensure the agreement is satisfied during operations, business transactions have to be designed by correlating agreements. Furthermore, business transactions are collaborative, distributed, and long-running and thus they are prone to failure. Therefore, flexibility is a paramount importance for business transactions.

A business transaction model should provide design-elements and techniques to design highly flexible business transactions by correlating the trading partner agreements. Unfortunately, there is a dearth of a model that enables designing flexible business transactions by correlating the business perspective elements with the functional perspective elements. Existing business transaction models provide a limited number of design-elements and techniques. None of the existing business transaction models are able to provide the necessary and sufficient design-elements and techniques. Accordingly, there is a strong need for a business transaction model which is capable of addressing these shortcomings.

This research has developed Solúbtha business transaction model that enables designing business transactions by correlating the business and functional perspective of end-to-end purchase order management processes. Additionally, this thesis has extended the theory of classical atomic behaviour of business transactions to enable designing highly-flexible behaviour of business transactions. The model is built on five properties: fault-tolerance, failure-resiliency, operational autonomy, non-determinism, and extensibility, which are the basic building blocks of flexible behaviour of business transactions. The model is capable of providing recovery operators and other elements that are needed for implementing these properties.

**Supervisor:** Dr. Ita Richardson  
Dr. Eoin Whelan  
Professor dr. Micheal P. Papazoglou  
Professor dr. Willem-jan van den Heuvel  
Dr. Yehia Taher

**Keywords:** Business Transaction, Business Transaction Model, Flexible Behavior, Electronic Business, Service Oriented Architecture
Acknowledgements

I would like to express my deepest appreciation to my supervisor Dr. Ita Richardson for the continuous support of my Ph.D study and research and for her motivation. Without her guidance and persistent help this thesis could never be completed. All these years she always showed me the right path and kept me on track to reach my goal. She believed on me and also perused me to believe that I have the ability to finish my journey.

I would like to thank my other supervisor Dr. Eoin Whelan who was always an outstanding instructor. I also thank my external supervisors Prof. Mike Papazoglou, Prof. Willem-jan van den Heuvel, and Dr. Yehia Taher for their invaluable inputs and supports on various issues. A special thanks goes to my daily supervisor in Tilburg University, Dr. Yehia Taher who was a friend and at the same time was a Guru. Beside my Ph.D topic, he taught me ten million things about Computer Science.

My sincere thanks goes to my internal examiner Dr. Sarah Beecham for her high quality inputs and patience. In the last few months she went beyond her limit to help me finishing this thesis.

I owe thanks to my former colleagues in ERISS, Vasilios Andrikopoulos, Amal Elgammal, Michele Mancioppi, Michael Parkin, and Oktay Türetken. Also, I wish to thank my colleagues in Lero, Deva Kumar Deeptimahanti, and Anila Mjeda.

I was blessed to be friend of two wonderful persons Sajid Ibrahim Hashmi and Khoa Nguyen who helped me to survive in many tough situations. They have been always true friends by giving their effective advises and above all mental support. They always stood beside me to overcome certain circumstances that I was not able to handle by myself.

A special acknowledgment goes to Patsy Finn and Gerard Mulligan for their supports. I wish to thank my funding organizations for funding my research.

Since my childhood my parents had a strong belief that I will earn the top academic degree. Their inspiration never knew no bounds. My deepest gratitude goes to them. Words are not enough to thank my brothers.

Last but the most important person and also the best gift ever in my life Sumaiya Islam Chowdhury, my better half. In fact, I could not find words to acknowledge her contributions. In all these years I was always amazed by her dedication, patience, inspirational words, and above all her enthusiasm. I always felt she is doing eighty percent of my works. I thank her by dedicating this thesis.
# Contents

Table of Contents v

List of Figures xii

List of Tables xv

1 Introduction 1

1.1 Research Background and Contexts ............................................. 2
1.2 Motivation ............................................................................... 6
1.3 Research Scope ....................................................................... 9
1.4 Problem Definition ................................................................. 10
1.5 Research Objectives ................................................................ 12
1.6 Research Questions ................................................................ 13
1.7 Research Methodology .............................................................. 14
1.8 Contribution ........................................................................... 16
1.9 Limitations ............................................................................. 16
1.10 Outline of the Dissertation ...................................................... 18

2 Literature Review 20

2.1 Introduction ........................................................................... 20
2.2 Overview of Business Transaction ........................................... 21
2.3 Business Transaction Models .................................................. 24
   2.3.1 Nested Transaction Model .................................................. 24
   2.3.2 Saga .................................................................................. 27
   2.3.3 Cooperative Transaction Model ........................................... 29
   2.3.4 Split-Join Transaction Model .............................................. 31
   2.3.5 S Transaction Model .......................................................... 32
   2.3.6 ConTract Transaction Model .............................................. 33
   2.3.7 TMLA ............................................................................... 34
   2.3.8 WIDE .............................................................................. 34
## CONTENTS

2.3.9 Business-Aware Transaction Model ........................................... 35
2.3.10 TxQoS-Aware Approach .......................................................... 37
2.3.11 CoBTX-Net ................................................................. 37
2.3.12 Summary of Business Transaction Models ................................. 38

2.4 Technologies for Designing Business Transaction ......................... 40
2.4.1 ASSET ............................................................... 40
2.4.2 CheeTah ............................................................... 40
2.4.3 Java Activity Service ......................................................... 41
2.4.4 Web Service Transaction (WS-TX) ........................................ 42
2.4.5 Business Transaction Protocol ................................................ 43
2.4.6 Web Service Composite Application Framework (WS-CAF) .......... 44
2.4.7 UN/CEFACT Modeling Methodology (UMM) ............................ 45
2.4.8 Business Process Model And Notation .................................... 46
2.4.9 Business Process Execution Language .................................... 46
2.4.10 Commercial Business Transaction Software ............................ 48
2.4.11 Summary of Business Transaction Technologies ...................... 50

2.5 Summary .................................................................................. 51

3 Research Method ........................................................................... 53
3.1 Introduction ............................................................................... 53
3.2 Research Method ......................................................................... 54
3.2.1 Design-science Research Method ............................................ 55
3.2.1.1 Formal Method .......................................................... 57
3.3 Research design .......................................................................... 58
3.3.1 Problem Definition ............................................................... 58
3.3.2 Literature Review .................................................................. 61
3.3.2.1 Search ........................................................................ 62
3.3.2.2 Selection ...................................................................... 63
3.3.2.3 Store .......................................................................... 64
3.3.2.4 Extraction .................................................................... 64
3.3.2.5 Writing ........................................................................ 65
3.3.3 Development of Solúbtha BTM ............................................... 65
3.3.3.1 Initial Version of Solúbtha BTM .................................... 65
3.3.3.2 Interim Evaluation ........................................................ 65
3.3.3.3 Secondary Data Analysis of Business Cases .................... 67
CONTENTS

3.3.3.4 Study Trading Partner Agreements ............................ 71
3.3.3.5 Construction of Models ........................................... 72
3.3.4 Validation and Evaluation ............................................ 73
3.4 Research Validity ....................................................... 75
3.5 Summary ............................................................... 76

4 Data Collection and Analysis ........................................... 77
4.1 Introduction ............................................................. 77
4.2 Scope of the Study ....................................................... 77
4.3 Case 1: The Auto Inc. Supply Chain (Nguyen, 2009) ............... 78
  4.3.1 Overview ............................................................ 78
  4.3.2 Identify Design Elements ......................................... 81
  4.3.3 Identify Relationship between Design Elements ................. 83
  4.3.4 Identify Business Transaction Faults ........................... 83
4.4 Case 2: the AVERS OEM Supply Chain (Papazoglou et al., 2012) .... 84
  4.4.1 Introduction .......................................................... 85
  4.4.1.1 Identify Design Elements .................................... 87
  4.4.2 Identify Relationship Between Design Elements ................ 88
  4.4.3 Identify Business Transaction Faults ........................... 88
4.5 Case 3: Walmart Changes Tactics to Meet International Tastes (Simchi-Levy, 2008) ... 88
  4.5.0.1 Introduction ....................................................... 88
  4.5.1 Identify Business Transaction Faults ........................... 89
4.6 Business Partner Agreements ........................................... 89
  4.6.1 Identify Business Transaction Fault ............................. 90
4.7 Summary ............................................................... 90

5 Business Transaction Fault Models ................................. 91
5.1 Construction of Fault Models ......................................... 91
5.2 Business Related Fault Model ......................................... 92
5.3 Business Protocol Related Fault Model ........................... 99
5.4 Business Object Related Fault Model ................................ 100
5.5 Message Related Fault Model .......................................... 102
5.6 Application Related Fault Model ..................................... 103
5.7 Relationship between Faults .......................................... 104
5.8 Formal Representation of Business Transaction Faults ............ 105
<table>
<thead>
<tr>
<th>5.8.1</th>
<th>Formal Representation of BT Related Faults</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8.2</td>
<td>Formal Representation of Business Related Faults</td>
<td>106</td>
</tr>
<tr>
<td>5.8.2.1</td>
<td>Resource Unavailability</td>
<td>106</td>
</tr>
<tr>
<td>5.8.2.2</td>
<td>Business Policy Violation</td>
<td>106</td>
</tr>
<tr>
<td>5.8.2.3</td>
<td>Compliance Policy Violation</td>
<td>107</td>
</tr>
<tr>
<td>5.8.2.4</td>
<td>Security Policy Violation</td>
<td>107</td>
</tr>
<tr>
<td>5.8.2.5</td>
<td>Product Quality Violation</td>
<td>107</td>
</tr>
<tr>
<td>5.8.2.6</td>
<td>Uncertain Events</td>
<td>108</td>
</tr>
<tr>
<td>5.8.2.7</td>
<td>Compound Business Related Faults</td>
<td>108</td>
</tr>
<tr>
<td>5.8.3</td>
<td>Formal Representation of Business Protocol Related Faults</td>
<td>109</td>
</tr>
<tr>
<td>5.8.4</td>
<td>Formal Representation of BO Related Faults</td>
<td>109</td>
</tr>
<tr>
<td>5.8.5</td>
<td>Formal Representation of Message Related Faults</td>
<td>109</td>
</tr>
<tr>
<td>5.8.6</td>
<td>Formal Representation of Application Related Faults</td>
<td>109</td>
</tr>
<tr>
<td>5.8.7</td>
<td>Fault Chain in Business Transaction</td>
<td>110</td>
</tr>
<tr>
<td>5.9</td>
<td>Summary</td>
<td>111</td>
</tr>
</tbody>
</table>

6 Flexible Business Transaction – Concepts and Characteristics

6.1 Introduction | 113
6.2 A Brief Overview of Business Transactions Behavior | 113
6.2.1 Types of Business Transaction Behavior | 114
6.3 Characteristics of Flexible Business Transaction | 115
6.3.1 Fault-Tolerance | 116
6.3.2 Failure-Resilience | 116
6.3.3 Non-Determinism | 116
6.3.4 Operational Autonomy | 117
6.3.5 Extensible | 117
6.4 Summary | 118

7 Development of Solúbtha BTM

7.1 Introduction | 119
7.2 Solúbtha BTM | 119
7.2.1 Agreement | 121
7.2.1.1 Party | 124
7.2.1.2 Period | 124
7.2.1.3 Policy | 125
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2.1.4 BusinessPolicy</td>
<td>125</td>
</tr>
<tr>
<td>7.2.1.5 Compliance Policies</td>
<td>127</td>
</tr>
<tr>
<td>7.2.1.6 Security Policy</td>
<td>128</td>
</tr>
<tr>
<td>7.2.2 Transaction Role</td>
<td>133</td>
</tr>
<tr>
<td>7.2.3 Transaction Process Fragment</td>
<td>134</td>
</tr>
<tr>
<td>7.2.4 Business Protocol</td>
<td>138</td>
</tr>
<tr>
<td>7.2.5 Transaction Failure Recovery Handler</td>
<td>140</td>
</tr>
<tr>
<td>7.2.6 The Complete Solúbtha BTM</td>
<td>143</td>
</tr>
<tr>
<td>7.3 Summary</td>
<td>145</td>
</tr>
</tbody>
</table>

8 Formal Syntax of Solúbtha BTM | 146 |
| 8.1 Introduction | 146 |
| 8.2 Formal Definition of Solúbtha BTM | 146 |
| 8.2.1 Set Theory | 146 |
| 8.2.2 Abstract Syntax of Solúbtha BTM | 147 |
| 8.2.2.1 BusinessTransaction | 147 |
| 8.2.2.2 Agreement | 147 |
| 8.2.2.3 Period | 148 |
| 8.2.2.4 Party | 148 |
| 8.2.2.5 Policy | 149 |
| 8.2.2.6 BusinessPolicy | 149 |
| 8.2.2.7 CompliancePolicy | 150 |
| 8.2.2.8 SecurityPolicy | 151 |
| 8.2.2.9 TransactionProcessFragment | 151 |
| 8.2.2.10 Business Protocol | 154 |
| 8.2.2.11 TransactionFailureRecoveryHandler | 154 |
| 8.2.2.12 M-to-1 recovery function | 155 |
| 8.2.2.13 Transaction Role | 155 |
| 8.2.2.14 Business Transaction Model | 156 |
| 8.2.3 Relationships between Design Elements | 156 |
| 8.2.4 Well-Formedness | 160 |
| 8.3 Summary | 162 |

9 Formal Semantics of Solúbtha BTM | 163 |
<p>| 9.1 Introduction | 163 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>Mapping Solúbtha BTM Design-elements to TEFSM</td>
<td>164</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Business Transaction</td>
<td>165</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Transaction Process Fragment</td>
<td>166</td>
</tr>
<tr>
<td>9.2.3</td>
<td>Activity</td>
<td>167</td>
</tr>
<tr>
<td>9.3</td>
<td>Mapping Control Flow Patterns to TEFSM</td>
<td>167</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Sequence</td>
<td>167</td>
</tr>
<tr>
<td>9.3.2</td>
<td>XChoice</td>
<td>168</td>
</tr>
<tr>
<td>9.3.3</td>
<td>NChoice</td>
<td>170</td>
</tr>
<tr>
<td>9.3.4</td>
<td>XJoin</td>
<td>171</td>
</tr>
<tr>
<td>9.3.5</td>
<td>NJoin</td>
<td>172</td>
</tr>
<tr>
<td>9.3.6</td>
<td>Multiple Instances Without Synchronization</td>
<td>173</td>
</tr>
<tr>
<td>9.3.7</td>
<td>Interleave</td>
<td>174</td>
</tr>
<tr>
<td>9.3.8</td>
<td>Standard Discriminator</td>
<td>174</td>
</tr>
<tr>
<td>9.4</td>
<td>Formalization of Behavioral Properties</td>
<td>175</td>
</tr>
<tr>
<td>9.4.1</td>
<td>An Overview of Computational Tree Logic (CTL)</td>
<td>175</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Theory of Flexible Behavior</td>
<td>177</td>
</tr>
<tr>
<td>9.4.3</td>
<td>Preliminaries</td>
<td>178</td>
</tr>
<tr>
<td>9.4.4</td>
<td>Transactional Properties and Behavioral Models</td>
<td>180</td>
</tr>
<tr>
<td>9.4.4.1</td>
<td>Fault Tolerant Business Transaction</td>
<td>181</td>
</tr>
<tr>
<td>9.4.4.2</td>
<td>Failure-Resilient Business Transaction</td>
<td>186</td>
</tr>
<tr>
<td>9.4.5</td>
<td>Summary</td>
<td>205</td>
</tr>
<tr>
<td>10</td>
<td>Validation and Evaluation</td>
<td>206</td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>206</td>
</tr>
<tr>
<td>10.2</td>
<td>Validation Setting</td>
<td>207</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Business Setting</td>
<td>208</td>
</tr>
<tr>
<td>10.2.1.1</td>
<td>Overview of the Rhea Automotive Business Scenario</td>
<td>208</td>
</tr>
<tr>
<td>10.2.1.2</td>
<td>Agreements</td>
<td>209</td>
</tr>
<tr>
<td>10.2.1.3</td>
<td>Purchase Order Management Process</td>
<td>210</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Technological Setting</td>
<td>212</td>
</tr>
<tr>
<td>10.2.2.1</td>
<td>ARIS Express</td>
<td>212</td>
</tr>
<tr>
<td>10.2.2.2</td>
<td>UPPAAL</td>
<td>212</td>
</tr>
<tr>
<td>10.2.2.3</td>
<td>System Environment</td>
<td>213</td>
</tr>
<tr>
<td>10.3</td>
<td>Designing Rhea POM Business Transaction Model</td>
<td>213</td>
</tr>
<tr>
<td>10.3.1</td>
<td>The POM Business Transaction Model</td>
<td>213</td>
</tr>
<tr>
<td>CONTENTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.3.2 Specifying Agreement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4 POM Business Transaction Model Transformation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1 POM Automata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.1 Registration Automata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.2 Inventory Template</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.3 Invoice Automata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.4 Delivery Automata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.5 Payment Automata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.1.6 Compensation and Resume</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.2 Transaction Property Specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.3 Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.3.1 Correctness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.3.2 Correlation-ability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.3.3 Applicability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4.3.4 Flexibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.5 Comparative Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.5.1 Comparison with Business Related Design Elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.5.2 Comparison with Techniques</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.6 Summary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Conclusions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.1 Introduction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.2 Usage of the Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.3 Summary of the Research Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.4 Meeting the Research Objective</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.5 Summary of the Contribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.6 Limitations of the Work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.6.1 Limitations to the Research Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.6.2 Limitations to the Research Product</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7 Further Work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7.0.1 Feasibility and Scalability Study</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7.0.2 Extension of Solútha BTM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7.0.3 A Language for Designing Business Transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7.0.4 Integration with SOA Technologies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7.0.5 Tool Support for Business Transaction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1.1 An Example of Business Scenario. (Source: Hantry et al., 2010) .......................... 4
1.2 An example of the connection between business and functional perspective ............... 5
1.3 The Chain of Development of Transaction Approaches. ....................................... 5
1.4 The S-cube Research Framework (Source: (Papazolou et al., 2010)) ....................... 11
1.5 Research Design. .................................................................................. 15

2.1 Closed Nested Transaction Model that allows decomposing a large transaction into multiple units. ........................................................................................................... 25
2.2 Example of Compensation Transaction Associating Sub-transactions. ...................... 28
2.3 Cooperative transaction hierarchy model .............................................................. 30
2.4 Split-Join transaction model (Source (Pu et al., 1988)). ......................................... 31
2.5 Business-Aware Transaction Model (Source: (Papazoglou and Kratz, 2006)). ........ 36

3.1 Design-Science Research Framework for Information System Research (Source (Peffers, 2008)). .......................................................... 56
3.2 Research Design. .................................................................................. 59

4.1 Supply Chain Operations Reference-model (Source: Supply Chain Council, (2000)). ... 79
4.2 Automotive Supply Chain Scenario (Source: Nguyen, 2009). ............................... 80
4.3 The AVERS OEM Supply Chain Scenario ......................................................... 86
4.4 The Logistics Scenario of Wall-Mart ................................................................... 89

5.1 The Causal Tree of Business Related Faults. ....................................................... 92
5.2 The Complete Causal Tree of Business Related Fault. ......................................... 93
5.3 The Causal Tree of Business Protocol Related Fault. .......................................... 99
5.4 The Causal Tree of Business Object Related Fault. .......................................... 101
5.5 The Causal Tree of Message Related Fault. .................................................... 102
5.6 The Causal Tree Shows the Application Related Faults. ................................... 103
5.7 The Business Transaction Fault Chain. ............................................................. 111
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>The Core Packages of Solúbtha BTM</td>
<td>120</td>
</tr>
<tr>
<td>7.2</td>
<td>The Agreement Construct of Solúbtha BTM</td>
<td>122</td>
</tr>
<tr>
<td>7.3</td>
<td>The Policy of Solúbtha BTM</td>
<td>124</td>
</tr>
<tr>
<td>7.4</td>
<td>The Transaction Role Construct of Solúbtha BTM</td>
<td>126</td>
</tr>
<tr>
<td>7.5</td>
<td>The Transaction Process Fragments of Solúbtha BTM</td>
<td>134</td>
</tr>
<tr>
<td>7.6</td>
<td>Business Protocol Concept of Solúbtha BTM</td>
<td>136</td>
</tr>
<tr>
<td>7.7</td>
<td>Transaction Failure Recovery Handler Concept of Solúbtha BTM</td>
<td>139</td>
</tr>
<tr>
<td>7.8</td>
<td>The Complete Solúbtha Business Transaction Model</td>
<td>141</td>
</tr>
<tr>
<td>9.1</td>
<td>State transition of business transaction state machine.</td>
<td>166</td>
</tr>
<tr>
<td>9.2</td>
<td>State Transition of Transaction Process Fragment.</td>
<td>166</td>
</tr>
<tr>
<td>9.3</td>
<td>Formal Representation of Activity in Business Transaction State Machine</td>
<td>167</td>
</tr>
<tr>
<td>9.4</td>
<td>Formal Representation of Sequential Behavioral Model.</td>
<td>168</td>
</tr>
<tr>
<td>9.5</td>
<td>XChoice Automation</td>
<td>169</td>
</tr>
<tr>
<td>9.6</td>
<td>Example of XChoice Automation</td>
<td>170</td>
</tr>
<tr>
<td>9.7</td>
<td>Example of NChoice Automata</td>
<td>171</td>
</tr>
<tr>
<td>9.8</td>
<td>Example of XJoin Automata</td>
<td>172</td>
</tr>
<tr>
<td>9.9</td>
<td>Example of NJoin Automata</td>
<td>173</td>
</tr>
<tr>
<td>9.10</td>
<td>Multiple Instances Without Synchronization Automata</td>
<td>174</td>
</tr>
<tr>
<td>9.11</td>
<td>Standard Discriminator Automation</td>
<td>175</td>
</tr>
<tr>
<td>9.12</td>
<td>An Example of Eventual Failure Atomicity</td>
<td>178</td>
</tr>
<tr>
<td>9.13</td>
<td>Transaction Scope and Sub-Scope.</td>
<td>179</td>
</tr>
<tr>
<td>9.14</td>
<td>Fault Tolerant Behavioral Model - I defined using Ignore Recovery Policy</td>
<td>182</td>
</tr>
<tr>
<td>9.15</td>
<td>Fault Tolerant Behavioral Model - II defined using Ignore Recovery Policy</td>
<td>184</td>
</tr>
<tr>
<td>9.16</td>
<td>Failure Resilient Behavioral Model defined using Substitute Recovery Policy</td>
<td>189</td>
</tr>
<tr>
<td>9.17</td>
<td>Failure Resilient Behavioral Model based on Repair Recovery Policy.</td>
<td>193</td>
</tr>
<tr>
<td>9.18</td>
<td>Failure Resilient Behavioral Model based on Mutualize Recovery Policy.</td>
<td>196</td>
</tr>
<tr>
<td>9.19</td>
<td>Failure Resilient Behavioral Model based on Retry Policy.</td>
<td>198</td>
</tr>
<tr>
<td>9.20</td>
<td>Failure-Resilient Behavioral Model based on Compensation Recovery Policy</td>
<td>201</td>
</tr>
<tr>
<td>9.21</td>
<td>Failure-Resilient Behavioral Model based on Resume Recovery Policy.</td>
<td>204</td>
</tr>
<tr>
<td>10.1</td>
<td>The Research Validation Phases.</td>
<td>208</td>
</tr>
<tr>
<td>10.2</td>
<td>The Rhea Automotive Business Scenario.</td>
<td>209</td>
</tr>
<tr>
<td>10.3</td>
<td>Purchase Order Management Business Transaction Model.</td>
<td>214</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.4</td>
<td>Purchase Order Fulfillment Business Transaction</td>
<td>216</td>
</tr>
<tr>
<td>10.5</td>
<td>Registration Automata</td>
<td>221</td>
</tr>
<tr>
<td>10.6</td>
<td>Inventory Automata</td>
<td>222</td>
</tr>
<tr>
<td>10.7</td>
<td>Invoice Automata</td>
<td>223</td>
</tr>
<tr>
<td>10.8</td>
<td>Delivery Automata</td>
<td>224</td>
</tr>
<tr>
<td>10.9</td>
<td>Payment Automata</td>
<td>225</td>
</tr>
<tr>
<td>10.10</td>
<td>Compensation and Resume Template</td>
<td>226</td>
</tr>
<tr>
<td>10.11</td>
<td>Simulation of Business Transaction of Rhea Automotive PMO Business Case</td>
<td>233</td>
</tr>
<tr>
<td>10.12</td>
<td>The Simulation of Compensation and Resume</td>
<td>234</td>
</tr>
<tr>
<td>1</td>
<td>Process P, Q, R Execute Sequentially (Left Hand Side Box) and Concurrently (Right Hand Side Box)</td>
<td>268</td>
</tr>
<tr>
<td>2</td>
<td>The Initial Version of Solúbtha Business Transaction Model</td>
<td>281</td>
</tr>
<tr>
<td>3</td>
<td>Solúbtha Business Transaction Model V1.</td>
<td>290</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>List of Design Elements</td>
<td>39</td>
</tr>
<tr>
<td>2.2</td>
<td>List of Techniques for Flexible Business Transactions</td>
<td>39</td>
</tr>
<tr>
<td>2.3</td>
<td>List of Design Elements</td>
<td>50</td>
</tr>
<tr>
<td>2.4</td>
<td>List of Techniques for Flexible Business Transactions</td>
<td>51</td>
</tr>
<tr>
<td>2.5</td>
<td>List of Design Elements</td>
<td>51</td>
</tr>
<tr>
<td>2.6</td>
<td>List of Techniques for Flexible Business Transactions</td>
<td>52</td>
</tr>
<tr>
<td>3.1</td>
<td>Participants and their Expertise</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>Participants and Duration of Discussion</td>
<td>68</td>
</tr>
<tr>
<td>4.1</td>
<td>A List of Identified Design Element</td>
<td>81</td>
</tr>
<tr>
<td>4.2</td>
<td>List of Faults Identified in the Auto Inc. Business Scenario</td>
<td>83</td>
</tr>
<tr>
<td>4.3</td>
<td>List of Design Element identified in the AVERS Supply Chain</td>
<td>87</td>
</tr>
<tr>
<td>4.4</td>
<td>List of Business Related Faults Identified in Business Partner Agreement</td>
<td>90</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary of the Business Related Faults</td>
<td>94</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of the Business Protocol Related Faults</td>
<td>100</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary of the Business Object Related Faults</td>
<td>101</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary of the Message Related Faults</td>
<td>102</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary of the Application Related Faults</td>
<td>104</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of the Agreement Design-element of Solúbtha BTM</td>
<td>123</td>
</tr>
<tr>
<td>7.2</td>
<td>The Table Summarizes the Policy Construct of Solúbtha BTM</td>
<td>128</td>
</tr>
<tr>
<td>7.3</td>
<td>The Table Summarizes the Transaction Process Fragment Construct of Solúbtha BTM</td>
<td>137</td>
</tr>
<tr>
<td>10.1</td>
<td>List of Properties Implemented for the Rhea Automotive POM Business Transactions</td>
<td>226</td>
</tr>
<tr>
<td>10.2</td>
<td>Comparison between Existing Business Models and Technologies and Solúbtha BTM</td>
<td>236</td>
</tr>
<tr>
<td>10.3</td>
<td>Comparison between Existing Business Models and Technologies and Solúbtha BTM</td>
<td>238</td>
</tr>
<tr>
<td>1</td>
<td>List of Excessive Design-elements</td>
<td>269</td>
</tr>
<tr>
<td>2</td>
<td>Actions on Relationships between Design Elements and Status</td>
<td>272</td>
</tr>
<tr>
<td>Table</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3</td>
<td>The List of Design-elements and Actions</td>
<td>273</td>
</tr>
<tr>
<td>4</td>
<td>The Design-elements and causes for including them in the Initial Version of Solúbtha BTM</td>
<td>275</td>
</tr>
<tr>
<td>5</td>
<td>The List of Design-elements and Actions</td>
<td>282</td>
</tr>
<tr>
<td>6</td>
<td>List of the new Design-elements that were added and causes of add operation</td>
<td>284</td>
</tr>
<tr>
<td>7</td>
<td>List of Design-elements that were renamed and causes of rename operations</td>
<td>285</td>
</tr>
<tr>
<td>8</td>
<td>The list of design-elements that were merged and joined and causes of these operations</td>
<td>286</td>
</tr>
<tr>
<td>9</td>
<td>Description of design elements identified through studying secondary data of Auto Inc. Automotive Supply Chain Scenario.</td>
<td>287</td>
</tr>
<tr>
<td>10</td>
<td>Design-elements and cause of actions</td>
<td>292</td>
</tr>
<tr>
<td>11</td>
<td>Design-elements and causes of actions</td>
<td>293</td>
</tr>
<tr>
<td>12</td>
<td>Design-elements and cause of actions</td>
<td>295</td>
</tr>
<tr>
<td>13</td>
<td>Description of the Newly Added Design-elements</td>
<td>296</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Servitization of software components is one of the recent major breakthroughs in the area of Information System (IS). This has given the rise to a new paradigm, Service Oriented Computing (SOC), for building Service-based Applications (SBAs) that is essentially composed of software components provided as software services by third party software providers. Because of its low-cost and effective features, such as self-containment and loose coupling, software services have become popular for building large-scale complex distributed SBAs. Enterprises today are moving from their legacy applications to SBAs as the SOC technologies adequately enable legacy components to be transformed into services that are integrated into SBAs. However, since it is a relatively new and emerging area, several issues have emerged that have led to many research initiatives. The Software Services and System Network (S-Cube) is such an initiative that is heavily focused on software services. However, S-Cube covers broad areas from requirement engineering to software infrastructure. S-Cube is a European Union (EU) funded 7th framework program (FP7) Network of Excellence (NoE) project. The main research objective of the project is as follows, the objective of the S-Cube project encompassed the research on engineering, quality assurance and adaptation concepts and techniques for service-based applications, as well as research on service technology foundations to realize service-based applications. (Source: S-Cube final report).

This objective was refined to many objectives and grouped them into several research fragments. One of the key fragments was Business Process Management (BPM) and the central focus in this was Business Transaction that simply put, is the automation of end-to-end business processes. The key concerns in this research fragment were developing efficient techniques, methods, etc. that support efficient composition of software services to build robust SBAs. The main reason to concentrate on the business transaction is straightforward: in a service-oriented environment, electronic business transactions of large-scale end-to-end business processes (e.g., purchase order management processes in a supply chain management system) that include tens of hundreds of activities, are enormously challenging. Because, such processes involve partners from different regions, software services that perform the activities are distributed, and also there are business perspective challenges. Technically speaking, business transaction is the core of process automation because its failure affects the entire process chain.

---

1 http://www.s-cube-network.eu/
2 http://europa.eu/index_en.htm
The research presented in this thesis is about business transaction and is the key part of the BPM research fragment of the S-Cube project. As part of it, my focus was to develop efficient methods and techniques for business transactions of end-to-end business processes to enable SBAs to execute and manage business transactions efficiently.

1.1 Research Background and Contexts

The origin of modern transaction

It had begun with database transaction followed by several transaction paradigms: long-running transaction, distributed transaction, web service transaction, and business transaction. Being an ancestor, the database transaction influenced the underlying techniques and methods of successive transaction approaches, for which it is considered the foundation of all transaction models and technologies. The database transaction comprises two key operations, READ and WRITE. The WRITE refers to data insertion whereas READ means retrieving data from the databases. These operations rely on four: properties Atomicity, Consistency, Isolation, and Durability which are better known as ACID (Haerder and Reuter, 1983) that relies on an all or nothing principle (a detailed description of these properties is provided in Appendix A). Although ACID the main property in database transactions to preserve data integrity, it was realized by many, e.g., Gray and Reuter (1989), Molina and Salem (1987), that its strictness is applicable to certain applications but not all.

The advent of long-running transaction

With the advent of long-running applications, the concept of database transaction had changed. The new transaction approaches were attributed by novel application technologies. Specifically, the long-running application led to an extension of database transaction because this promoted new requirements in terms of transaction techniques and methods.

In database transactions, the operations such as update database were short-lived. However, it is opposite in long-running operations that had introduced in long-running applications (Molina and Salem, 1987). Long-running operations are a chain of operations that take a substantial amount of time. This gave rise to the notion of long-running transactions that access to many database objects. In this approach, the atomicity property has been relaxed to make the long-running operations flexible to prevent transaction operations.

Yet another transaction approach

The limitation of centralized applications led to the development of the distributed application paradigm. In this, applications are decomposed into several components that are distributed to multiple sites. This essentially promoted the need for new techniques and methods to perform transactions. The distributed transaction was developed to design and implement the operations performed by the distributed applica-
tions in a coherent manner. Operations within the scope of this transaction model must access databases that are located in different locations. A coordinator manages the traversing of transactions across the locations. This makes the transactions long-running that is, they require a significant amount of time to be completed. However, in certain cases distributed transactions can be short-lived.

The change continues
The biggest change in transaction design had happened with the advent of the Service Oriented Architecture (SOA) paradigm. SOA is an approach to designing a system that can provide services to end users and/or other applications in the network. It gave rise to the idea of providing components as services which are also known as service components as well as Web Services. The Web Services enabled building service-based applications (SBAs) by composing service components. This new application paradigm gave freedom to use third party software components in SBAs. However, it promoted the key question of how to design and manage transactions. It is worth mentioning that the operations in SBAs are long-running and distributed. Without having efficient approaches, managing transactions of SBAs is non-trivial for a system because, in SBAs both data and computations are distributed; thus, transactions are performed autonomously that need special failure management treatments. Also, the web services are provided by third party software providers, which promote the challenge of the availability of service components. These issues led the development of Web Service Transaction (Papazoglou, 2003). The key purpose of this transaction approach is to design business transactions efficiently to make transactions fault-tolerant. Several technologies (discussed in Chapter 2) were developed to design and manage web service transactions.

Transactions are process-aware and business-driven
Organizations no longer act as independent or isolated entities, rather they collaborate to deal with today’s dynamic and competitive business environment (Hantry et al., 2010). Figure 1.1 illustrates a simple business scenario where business partners perform business operations collaboratively by exchanging messages between them.\(^5\)

One prominent approach for automating such business scenarios is capturing it into a process and then automating the process. A business process consists of two layers: global layer where public processes are defined and local layer where private processes are defined. The public processes are composed of messages and activities that are publicly visible. The composition of interactions is called service choreography (Decker et al., 2006). On the other hand, private processes contain activities and controls choreography. The composition of activities is called orchestration. The private processes should be aligned with the public ones because the actual operations are performed by the business activities contained in private business process spaces. It is worth noting that a collaborative business environment engages various trading partners

\(^5\)Reproduced from Hantry et al., 2010
who share business information amongst themselves, nevertheless, the actual operations are performed autonomously by the respective business partners.

The automation of the end-to-end business process gave rise to the latest transaction approach *Business Transaction* which from the technical standpoint refers to the successful execution of each activity contained in business processes. The execution of a large number of process activities takes a significant amount of time. Thus, business transactions are long-running. In addition, business transaction is complex because it includes tens of hundreds of process activities that are distributed to various sites.

Furthermore, a formal approach to establishing collaborations between trading partners is *agreement*, which brings business partners into a common space. An agreement enables organizations to define common goals and to share business information and operational activities. Typically trades are governed by the agreements; however, since it is purely a business perspective entity, it was never used in process automation. Rather, the clauses of agreements were executed manually. In fact, there was no suitable approach for doing it automatically. Recently, agreement has become a critical concept for business processes. Several initiatives have been taken to build technologies for managing business processes from business perspectives. So far, Business Process Management (BPM) is the most widely known initiative for integrating business into technologies. BPM technologies enable the combination of the knowledge from information technology, management science, and real-world business perspectives and their application to end-to-end business processes (Van der Aalst, 2003). BPM is a commitment to expressing, understanding, representing and managing a business (or the portion of business to which it is applied) in terms of a collection of business processes that are responsive to the business environment of internal or external

---

6In this dissertation, a business entity denotes a business concept
events (McGovernan, 2008). Initially, BPM was focused on managing business processes. However, in the business issues of business processes in particular, agreement was overlooked. The current wave of BPM correlates business related issues with business processes to enable SBAs to perform business transactions from the business perspective. The idea of binding the business aspect to business processes promoted the concept of business-driven transaction. A vertical integration between business and functional perspectives of end-to-end business processes need for business-driven transactions. Figure 1.2 illustrates a connection between business and functional perspective elements. It shows that business perspective element agreement contains clauses which are constraints. The clauses are enforced on activities and business interactions that are functional perspective elements. The activities are business operations and the business interactions are messages exchanged between partners (The interactions are shown in Figure 1.1 by dotted arrow lines).

![Figure 1.2: An example of the connection between business and functional perspective](image)

To sum up, this section presented the transaction stack where business transaction is on the top since it is the latest development in this area. Figure 1.3 shows a chain of development of transaction approaches.

![Figure 1.3: The Chain of Development of Transaction Approaches.](image)

This section describes the evolution of the business transaction. It explains how the advent of different
technologies promoted different application environments and different requirements. Also, the section explains how all these issues drove a multitude of changes in transactions. The focus of this research is the process-aware and business-driven transaction.

1.2 Motivation

Business operations are driven by business entities

Consider a simple business-to-business (B2B) case where a trading partner trades with another trading partner. There is a preparation stage before the actual operation starts. In this stage, the trading partners negotiate on several issues such as the payment method, the deliver channel etc... etc... If one of these partners is from different country or continent then it becomes more complicated, for instance, the compliance policies become a critical issue. Through negotiation, the trading partners reach a set of common clauses that are stipulated in an ‘agreement’. These clauses govern business operations such as making payments, delivery of goods etc... This indicates that there is correlation between the business processes activities and agreements.

After the preparation stage, the actual operations start. With the advent of information technology (IT), business activities that do not require manual intervention are performed by an automated system. Ideally, the automation of such business activities should be done in conformance with the agreement, where the clauses are specified with the corresponding activities that compose end-to-end business processes. The agreements can be managed manually; however, it will slow the execution time because frequent manual intervention will be required while business transactions are running. To automate everything, including agreement, a formal approach is required to translate the agreements expressed in natural language into a machine-readable codes. This will make the systems capable of maintaining consistency between operations and agreement while controlling the operations.

Evidently, there is a strong need for technology that enables correlating the critical business entities with the functional entities so that the business transactions happen automatically by correlating trading partner agreements. Lately, this has become a significant research issue to the information system researchers. Research conducted by Papazoglou and Kratz (2006) highlighted the need for an explicit correlation of business perspective with the functional perspective. The S-Cube consortium reported a gap between the business and functional perspective in JRA (2009). Hantry et al. (2010) have explained why correlation is important and how it would add value in business transactions. The authors said that “as organizations are becoming more interested in applications that can drive the operations from business perspective, the correlation between business and functional perspective is indispensable.” According to Doug McClure (2008) “it does not matter how powerful an application is because it cannot do anything in terms of meeting business goal just sitting there. The integration of business and application perspectives can be more
1.2. MOTIVATION

effective to meet the business goals and objectives.” It is not possible to reap the benefits of IT in the context of business applications without a seamless correlation between these two perspectives.

In addition, Business Transaction Management (BTM) software is becoming popular. Business transaction management (BTM) software is considered the potential candidate for bridging two major trends in the IT management space; the trends are: the need for more granular application management and the need to manage a business perspective. According to IDC (2007) the business transaction management software market is at an early stage and it will grow over the years because it is able to give better visibility than traditional application performance management (APM) software. According to IDC, BTM is aimed at detecting and resolving problems at the granular level of interactions between IT elements that form a business transaction (McClure 2009). Also, IDC believes that BTM will likely become a core offering of established IT system management - ranging from performance management, SLA management, capacity planning, and to change and configuration management (Shacham, 2011).

Business Transactions are prone to failure

Business transactions are prone to risk of failure caused by various faults. There are a number of issues that may promote business transaction failure at runtime. Some of these are found in literature.

- A business transaction takes days or months (Garcia-Molina and Salem, 1987) to be completed. Such a long duration increases the possibility of the occurrence of a fault that may lead to business transaction abortion (Cfarku et al., 2012).

- In a collaborative business environment, an end-to-end business process integrates a number of partner business processes. A successful business transaction of an end-to-end business process depends on the successful execution of the business processes that contain a significant amount of activities that are performed autonomously by the partner applications. This increases the risks of occurring faults which results business transaction abortion, (Papazoglou and Kratz, 2006), (Sun and Yang, 2008).

- A business transaction of an end-to-end business process is performed by accessing partner applications. Business transactions have to traverse ‘N’ (>1) number of distant paths and access several distributed hosts. This increases the risk of business transaction failures (Helland, P., 2007).

- Lastly, the correlation between business and functional perspectives give rise to a challenge in committing business transactions successfully, which is tackling the business-oriented faults, such as violation of agreement, that may lead to business transaction abortion (Hantry et al., 2010). In Chapter 5, several business-oriented transaction faults are presented.

In addition, in Oracle (2010), Elias (2009), and McClure (2008) it has been pointed out that many
business transaction management software providers, including the large ones, are facing challenges in managing business transactions efficiently because of failures. Cappelli, W. from Gartner, states a shortcoming of application performance management software as follows: “it is less helpful in determining the root cause of failure. This makes failure prevention difficult for the application users”.\(^7\) Hence, an efficient approach for highly flexible business transactions is of paramount importance to prevent frequent business transactions abortion which is computationally expensive.

**Atomicity is too restrictive**

This is a well-known issue. A business transaction system that relies on the atomicity property consumes a significant amount of time to finish operations, and they are not fault-tolerant because any failure would force the system to rollback the entire transaction. The rollback simply put is a technique that invalidates all successfully completed operations and forces the system to return to the initial state. Atomicity is an efficient technique for preserving data integrity and thus it is the best fit for database transactions. However, since business transactions are prone to failure (Garcia-Molina and Salem, 1987), high flexibility is needed especially to reduce computation time and prevent business transaction failure or abortion.

**Design-time verification enables re-engineering instance-models**

The availability of simulation tools has enabled verifying a system before deploying it for real-world use. In a service-oriented environment, modeling is a popular approach because it gives better visibility to business processes. The wider adoption of BPMN (OMG, 2011) is an example of the popularity of this approach. As mentioned in the previous sections, business transactions of large-scale end-to-end business processes are complex, which promotes the risk of design-fault in business transaction instance-models. An instance-model is a concrete business transaction model that contains actual user-defined data. Design-faults would affect business transaction performance in terms of producing correct results. Verification would be an effective approach to sidestep the design-faults because it would enable detecting the inaccuracies and re-engineering the instance-models.

To sum up, the two most dominating factors that motivate this research are: the need of an explicit correlation between business and functional perspectives of end-to-end business processes and highly flexible behaviour of business transactions. The flexibility relates to the operational behavior of business transactions. More specifically, the operations within the scope of business transactions can be performed in a flexible manner unlike the restrictive approach of atomic transactions. In addition, the need for design-time verification motivated the research presented in this dissertation.

\(^7\)www.gartner.com
1.3 Research Scope

Business transactions are applied in different business scenarios that include business partners, the business operations, the process chain, and so on. With the advent of electronic business (eBusiness), businesses have been categorized into many types of which two predominate: business-to-business (B2B) and business-to-consumer (B2C). The Supply Chain Council (SCC, 2000) has defined a framework called Supply Chain Operation Reference (SCOR) which is a de facto standard for building processes for these businesses. It defines five main processes: plan, source, make, deliver, return, that cover all business activities carried out in a large-scale business scenarios. A business transaction encompasses one to many of these processes.

Although the research presented in this thesis is scientific research, the business issues are important because the effort here is to establish integration for business-driven transactions. Since it is not possible to provide a completely abstract solution to cover the entire sets of business issues, the research is scoped as following.

- **eBusiness and B2B**: The advent of IT and low-cost Internet have enabled organizations to perform businesses electronically. As a matter of fact, organizations are moving from manual business operations to system-driven business operations. In eBusiness, the interactions between the trading partners, and business activities, such as customer credential check, sending notification, etc., are carried out electronically. Since the research is about electronic business transactions, it is limited to eBusinesses only.

Although business processes of different business scenarios can contain common business activities such as make payment, deliver goods or services; however, there are elements such as payment means and delivery channel that are specific to business type. This may change the workflow. The solution provided in this research may serve other business types such as B2C or Business to Government (B2G); however, since these businesses were not studied, they are not within the scope of the solution. Therefore, the scope is limited to B2B electronic business only.

- **Business cases and processes**: Every business has a purpose and therefore, has a specific set of business processes. Business processes cover a specific business scenario, such as logistics management, purchase order management etc.. In this thesis, I mainly investigated purchase order management business cases. It is worth noting that I analysed secondary data of these business cases published in literature (Nguyen, 2009), (Papazoglou et al., 2010), and (Simchi-Levi, 2008). Although, a logistics management process was studied, the investigation was about faults not how business is done. Therefore, the research in this thesis is limited to the processes of a purchase order management scenario.

- **Modeling business transaction**: Business transactions can be designed using modeling approaches
or hard-coding. The latter is not a viable option due to high complexity and the need for proficiency in coding. On the other hand, modeling is a lingua franca approach which means it is not tied to any platform specific language and thus business transactions at execution level can be implemented using any language. This influenced the research to focused on a modeling approach in lieu of hard-coding.

- **Modeling and verification at design-time:** Business transaction modeling is a design-time issue. The design-time refers to a phase where business transaction systems are designed and verified. In addition, if necessary then business transactions are re-engineered at design-time. This research is concerned with designing and verifying instance-models of business transactions.

- **Service-oriented application:** The underlying technology of business processes is an SBA. It automates business processes. A traditional object-oriented application can be used; however, it requires extensive manual coding, changing the control flow of the business process needs enormous effort, and time. This research is concerned with business transactions of end-to-end business processes. Thus, it is scoped to service based applications.

1.4 Problem Definition

The thesis deals with two issues: explicit correlation between business and functional perspective elements and highly flexible behaviour of business transactions. The latter is a well-known issue whereas the former was outlined by the S-Cube consortium. The business transaction problems regarding these issues are discussed in the following.

**Correlation Problem**

The S-Cube research framework (Papazoglou et al., 2010) comprises three vertical layers: business process management, composition and coordination, and infrastructure. Figure 1.4 shows these layers.

The research is part of the BPM layer where transactional end-to-end business processes are discovered, designed, simulated, analyzed, and optimized (Hantry et al., 2010). The S-Cube consortium conducted a survey (the report can be found in (S-Cube, 2008)) and pointed out some critical factors at this layer need to addressed. The one this research is concerned with is an explicit correlation between business and functional perspectives. The consortium emphasized that business transactions should be governed from the business perspective, which is an efficient approach to integrate business and information technology. The consortium reviewed existing business transaction technologies, in particular, the SOA based technologies. They found that these technologies do not provide elements that can be used to design business transactions. It is worth noting that these elements are called design-elements in this thesis because they are used in designing business transaction models. They stated the problem as follows:

“The current business transaction technologies do not provide design-elements which can be used to
1.4. PROBLEM DEFINITION

![The S-cube Research Framework (Source: (Papazolou et al., 2010))](image)

**Figure 1.4: The S-cube Research Framework (Source: (Papazolou et al., 2010))**

Design business transactions by correlating the business perspective with the functional perspective so that at runtime the transactions can be governed from the business perspective.

Since the consortium defined this by reviewing only SOA-based technologies, in my research I decided to re-review the literature with extended search space (Chapter 3 describes the methods used for reviewing literature). I review existing business transaction models, open source technologies, and commercial products available in the market (Chapter 2 presents the review). It is worth mentioning that I reviewed the literature not only to ratify the problem stated by the consortium but also to identify the reusable design-elements (if there is any) provided by the state of the art. I found several reusable design-elements, which implies that the previous problem statement is not completely true. I reformulated the problem as follows,

“The current business transaction models or technologies do not provide a complete set of design-elements which can be used to design business transactions by correlating the business perspective with the functional perspective.”

**Flexible Behaviour Problem**

Flexibility is a well-known business transaction problem discussed in the literature as Greenfield, P. *et al.* (2001). While studying secondary data of the business cases (the cases are presented in Chapter 4 and Chapter 3 describes the methods and techniques used to study the cases), I realized that flexible behaviour is a real-world business transaction problem, especially for the business transactions of end-to-end business processes that involve multiple trading partners. The main reason flexibility is a critical issue for business
transactions is that business processes are prone to failure because of various runtime faults (presented in
Chapter 5) while cause business transaction abortion. One approach to prevent abortion is flexible oper-
ations behaviour that are scoped in business transactions. This needs efficient techniques. I reviewed the
literature and the commercial BTM software again. However, this time I concentrated on finding techniques
or design-elements, specifically the operators which can be used to define flexible operations in business
transaction models. Like the previous study, I reviewed literature to identify the techniques the state of art
models and technologies provide and to check whether the techniques provided are reusable. I found a few
reusable techniques; however, these are not sufficient because these are too basic ones for defining highly
flexible business transactions. This gave rise to the second research problem stated below,

The current business transaction models and technologies do not provide adequate techniques to define
the highly flexible business transactions of end-to-end business processes.

These two problems can be combined together into a single problem which can be stated as follows,

"The current business transaction models or technologies do not provide a complete set of design-
elements and techniques which can be used to design highly flexible business transactions by correlating
the business perspective with the functional perspective. ”

In addition to these problems, there is the design-time verification problem. The business transaction
models and technologies do not support design-time verification of the correctness of business transactions
because these lack formal underpinnings. This is a secondary problem and the solution of the problem is
produced in a natural manner in my thesis where I give a formal foundation to my model.

1.5 Research Objectives

The goal of the research is to develop a business transaction model to address the problems described
in the previous section. I develop the model because it is a more suitable option. Models are platform-
agnostic, that is, they are independent of any particular language for implementation. Also with the model
the interoperability issue can be easily avoided. Furthermore, it is a design-time approach with graphical
elements and is therefore it is easy to use for the non-technical experts.

Additionally, a model in computer science is a language which describes a particular kind of model.
Since models underlie a formal language, it can be translated into machine-readable codes. The business
transaction model that is developed in the research is called Solúbtha\(^8\). The model will enable describing the
business transactions instance-models. The instance-models are the instances of Solúbtha and it conforms
to the syntax and semantics of the the model. Solúbtha is an abstract model and thus it does not contain any
Concrete data.

\(^8\)Solúbtha is a Gaelic word which means flexible
1.6. RESEARCH QUESTIONS

The goal of this research is presented below:

“The goal of the research is to develop Solúbtha business transaction model that enables designing the verifiable and flexible business transactions for long-running complex purchase order management processes by correlating the business perspective design-elements with the functional perspective design-elements.”

Distilled from the goal, the objectives are more specific. Below the three objectives are presented:

- **Explicit Correlation**: Develop Solúbtha BTM with a complete set of design-elements that are used to model business transactions by correlating explicitly the business and functional perspectives of end-to-end business processes. The design-elements include the functional perspective design-elements, the business perspective design-elements, and various operators including the system level operators. The business perspective design-elements refers to the elements that cover the business domain and they are sourced from this domain. The functional perspective design-elements cover the operational aspect of end-to-end business processes.

- **Flexible Behaviour**: Develop Solúbtha BTM with a set of robust, efficient, and effective techniques to define the flexible behavior of business transactions. The techniques could be included in the model as operators or using any other suitable approach.

- **Design-time Verification**: The capability to verify the instance-models of business transactions at design-time. The verification will allow the detection of the design-flaws and let users re-engineer the particular models.

1.6 Research Questions

Several research questions are answered throughout the thesis and these are listed below. Essentially, the questions guide the research to meet the research objectives:

- Does any existing business transaction model or technology provide design-elements that can be used to define or describe business transaction instance-models by correlating the business and functional perspectives? If **yes** then,

  - does the model or technology provide all necessary design-elements to design business transactions of end-to-end business processes? If **not** then,

    * does the model or technology provide one or more design-elements that can be reused to design business transaction instance-models?
CHAPTER 1. INTRODUCTION

The literature was reviewed (in Chapter 2) with respect to this question (including its sub-questions).

- Does any existing business transaction model or technology provide techniques to define the flexible business transactions? If yes then
  
  * does the model or technology provide all required techniques to define the behavioural properties which can prevent faults at runtime and can guarantee highly flexible behaviour of business transactions? If not then,

* does the model or technology provide one or more techniques that can be reused to define the flexible behaviour of business transactions?

Like the previous question, in the second iteration, the literature were reviewed for this question.

What are the real-world business entities that influence business activities? Do the entities have relations between or among them? These questions guide the investigation of business cases and a business entity in Chapter 4.

What are business transaction faults? How the faults lead a business transaction to failure or abortion? The questions guide building of the fault models in Chapter 5. Additionally, they guide the investigation of business-related faults (in Chapter 4).

What are the behavioral characteristics of a flexible business transaction? This questions leads to the development a set of characteristics (presented in Chapter 6) of flexible business transactions.

How to construct the business transaction model by composing the design-elements logically? The construction of Solúbtha business transaction model (presented in Chapter 7) is guided by this question.

How to support the design-time verification of an instance-model? Solúbtha business transaction model is formalized (in Chapter 8 and 9) to answer this question.

1.7 Research Methodology

Since Chapter 3 is exclusively dedicated to describing the research methodology, this section provides a very brief description of the research methodology. This research is designed using the design-science research methodology developed by Hevner et. al, (2004). It comprises four main phases: problem definition, literature review, development of the business transaction model, and validation and evaluation. Figure 1.3 shows these phases.

This research project started by formulating the research problem. One problem was defined through reviewing the literature related to business transaction models and technologies and the other was found by studying business cases. Then, in the next phase Solúbtha business transaction model was constructed.
Three versions of the model were constructed in this phase. The Initial Version of the model was constructed from the literature. The design-elements of the Initial Version were elicited from the literature. In Figure 1.5, the literature review phase is positioned between problem definition and construction of the initial version of the model because it serves both steps.

An interim evaluation of the Initial Version of the model was performed by three participants. This was followed by the analysis of secondary data of the first business case. The evaluation and the outcome of the analysis led to customization of the Initial Version. This resulted in the next version of model called Solúbtha BTM V1. While studying secondary data of second and third cases and the trading partner agreements, I identified design-elements, relation between them, and business transaction faults. The second customization was carried out on Solúbtha BTM V1 to incorporate the data extracted from this study. This customization produced the final version of the business transaction model ‘Solúbtha BTM’. In the last phase, the model was validated and evaluated.
1.8 Contribution

The core contribution of this research project is Solúbtha business transaction model (Solúbtha BTM). The contribution of the research presented in this thesis is explained by splitting it into several units:

- **A business transaction model for designing a flexible business transaction instance-model by correlating explicitly the business and functional perspectives of end-to-end business processes.** The model is developed to enable designing business transactions by explicitly correlating the business entities so that the business operations can be governed from the business perspective. The model will assist in defining techniques for defining the behavioural properties of flexible business transactions. The behavioural properties are the logical statements that are refined to business transaction rules for controlling the execution and guaranteeing that the operations are flexible. Several operators are provided in Solúbtha BTM to define the techniques for flexible behaviour operations that are performed in business transactions.

- **Business Transaction Fault Models.** Several fault models have been constructed in this thesis. These models were developed to serve three purposes. First, these proved very effective in developing techniques for flexible behavior of business transactions. Second, the fault models can be defined in the business transaction instance-model to detect the transaction faults at runtime automatically. Third, the fault logic can be used in the decision support module of the business transaction system to select and execute the best failure prevention technique to prevent abortion or failure. Since in this research the prevention of failure or abortion is the key approach to guaranteeing the flexibility, the fault models are critical.

- **Eventual Failure Atomicity.** This is the fundamental principle of the flexible behavior of business transactions. Being developed from the classical notion of atomicity, it preserves atomicity to the lowest extent yet guarantees highly flexible operations in business transactions. This is the core of the notion of flexibility in the research presented in this thesis because it addresses the fundamental requirement of flexible business transactions. In this contribution the classical atomic principle was extended.

- **Formalization of the business transaction model.** Solúbtha business transaction model is formalized. The formalization will enable the business transaction systems to verify the business transaction model to guarantee the correctness of the instance models.

1.9 Limitations

The research presented in this thesis has certain limitations that are presented in the following:
1.9. LIMITATIONS

• The cases studied in the research are secondary, that is, the study makes use of publicly available data extracted in previous research. The data collected from the business cases Auto Inc. Automotive Supply Chain (Nguyen, 2009) and AVERS OEM Supply Chain (Papazoglou, et al., 2012) were primarily researched by the S-Cube consortium. Although these cases were accepted by the European Union (EU) review committee and also used in research validation in the S-Cube project, I cannot confirm that the real-world business scenarios are the same. Since I had no access to companies mentioned in the report or other documents except for a technical report, the quality of the primary research cannot be verified. The two other cases, Walmart Changes Tactics to Meet International Tastes (Simchi-Levy, 2008) and Rhea Automotive Supply Chain (Manzouri et al., 2011), were taken from the literature as well. The latter was used in validating the model. Although these cases were well-written, these secondary studies may not meet the real-world requirements by Solúbtha BTM.

• The business process models provided in the case reports do not include the return processes. Although I believe that the design-elements that are included in Solúbtha BTM can provide adequate support for designing the workflow, activities, and messages of return processes, yet I do not claim that Solúbtha BTM can provide all the necessary elements to design the business transactions of the processes. There may be specific design-elements needed to define one or more functions or techniques necessary to define flexible operations.

• Of the four business cases analysed in this thesis, three are about the automotive supply chain. This indicates that the model is of specific use, that is, business transactions of B2B supply chains in the automotive industry. However, the business cases include the Auto Inc., the AVERS OEM, and the Rhea automotive, are limited to a few things. Firstly, the business scenarios are limited to purchase order management business processes. The down-side chain does not include end-customer and the upside-chain includes only two-tier suppliers. Very limited information about the warehouses was provided in the case reports. Additionally, I cannot confirm that the business activities included in the business scenarios cover all business methods. For instance, the payment activity in all business processes in these business scenarios consider payment on delivery; however, there is a method called letter of intent which basically is an authorization of a percentage of payment before the delivery of goods, with the remaining payment made after the delivery. The business transaction models provide a design-element to split workflow which I believe can be used in such situations; however, I cannot guarantee that it is the best fit element because it was not tested.

• Businesses are very diverse and each business organization has its own set of operational strategies which are applied to business operations. Additionally, with change happening over time, the operational strategies are being changed to gain competitive advantages or to gain better market share. Solúbtha BTM has been abstracted to make it a generic model. I strongly believe that the model
provides sufficient elements to define business activities, workflow, define clauses of agreements etc.., etc.. However, the operators provided by Solúbtha BTM to define flexible behaviour is limited. Therefore, if there is any failure situation that I was not aware of while conducting this research, it cannot be dealt with by the model.

• I do not claim or argue that Solúbtha BTM is complete because it needs several experiments with real-world business scenarios. The experiments need access to business process management systems of the host organizations. Additionally, it is a time consuming job. In fact, no technology was/is complete in its first release. A technology matures over time by being used in several applications. The extension is done based on experience it gaining in certain environments. I strongly believe that Solúbtha BTM is a rich model for designing highly flexible business transactions, yet it is not complete maturer. Further extensions to increase its maturity level have been left as future work.

• The model developed in this thesis could be validated by developing a prototype which could execute a business transaction of an industrial case. This is an ideal validation approach for such a research project. However, it was not possible because of a few barriers. Several companies were contacted but most of them were reluctant to provide access to their business process kernel and databases, and also providing operational details was a security issue for them. Therefore, I decided to validate the research outcome using simulation which is an approach suggested by the design-science methodology. The model’s correctness was validated by mathematical formalism which is an efficient approach in computer science. However, for simulating behaviour the model was used in a automotive business case taken from literature. This is a weaker side of the validation because a real-world industrial case would make validation more rigorous. Additionally, the agreements used in the validation are sample agreements found on the web. All these issues prompt the question would Solúbtha BTM perform the same in real-world business scenarios.

1.10 Outline of the Dissertation

The thesis is structured as followed:

Chapter 2 (Literature Review): The literature review comprises four sections. The first section gives an overview of business transactions. The business transaction models and technologies are reviewed in the next two sections. In the final section, a commercial business transaction software are reviewed.

Chapter 3 (Research Methodology): This chapter provides a detailed description of the methodology used in this research. In this chapter I defend the selection of the methodology that is used in designing this research. The details of each step of the research process are provided in this chapter.

Chapter 4 (Study Business Cases and Business Entity): The secondary data of three business cases
are studied in this chapter. The business transaction design elements, relationships between elements, and business transaction faults are elicited in this chapter.

**Chapter 5 (Fault Models):** The faults elicited in Chapter 4 are presented as the fault models in this chapter. This chapter introduces different types of fault models covering business, application, business object and communication domains.

**Chapter 6 (Flexible Business Transaction: Concepts and Characteristics):** This chapter provides a description of the flexible business transaction. The behavioral concepts of the flexible business transactions are described in this chapter. Additionally, the characteristics of the flexible business transaction are described in this chapter.

**Chapter 7 (Development of the Solúbtha BTM):** The construction process of Solúbtha BTM is described in this Chapter. It provides the detail of how the business transaction elements are composed in Solúbtha BTM.

**Chapter 8 (Formal Syntax of Solúbtha BTM):** Solúbtha BTM is formalized in this chapter. The formal syntax of the business transaction model is presented in this chapter. The formal syntax is defined using set theory - a mathematical approach for defining the design constructs.

**Chapter 9 (Formal Semantics of Solúbtha BTM):** The formal semantics of Solúbtha BTM’s design elements and the semantics of the behavioral properties are presented in this chapter.

**Chapter 10 (Validation):** The business transaction model is validated in this chapter using Solúbtha BTM in a business case that is taken from the literature. Solúbtha BTM is used in designing the business transaction instance-model to validate the applicability of the model. Additionally, the UPPAAL model checker is used to verify behavioral flexibility of that instance-model. Also, the evaluation results are reported in this chapter.

**Chapter 11 Conclusion and Future Work:** This collates various strands of this research and draws the conclusion in terms of what has been achieved, what could not be achieved, and lesson learnt in this research. Finally, the directions for future work are outlined in this chapter.


Chapter 2

Literature Review

2.1 Introduction

The business transaction has been viewed in many ways by different researchers. This chapter discusses the views to find similarities and differences and also describes how business transaction is viewed in this research. Several models and technologies are available to design business transactions of end-to-end business processes. The strengths and limitations of these are discussed in this chapter. Some fundamental concepts of business transactions are provided in Appendix A.

An initial review of business transaction models and technologies was done by the S-Cube consortium (the review report can be found in (S-Cube, 2008)). The review focused on investigating the SOA-based business transaction technologies. It is worth mentioning that the notion of flexible behaviour was not within the scope of their study. The review concluded that a shortcoming of state of the art was that the current technologies do not provide design-elements to design business transactions of end-to-end business processes by correlating the business and functional perspectives of end-to-end business processes.

In my research I re-reviewed literature because the scope of the initial study was limited. The consortium reviewed only the well-known technologies that include the following:

- Open Electronic Data Interchange (Open EDI)
- the UN/CEFACT Modeling Methodology (UMM)
- electronic Business Markup Language (ebXML)
- Business Process Execution Language (BPEL)
- Java Transaction Activity
- WS-Transaction
- WS-Atomic Activity
- WS-Coordination Protocol.

No evidence was found that the business transaction models were reviewed. In my review I study both business transaction models and technologies. The review is conducted in two phases over two research questions presented in the Chapter 1 (Section 1.6). The review in the first phase focused on correlation and
in the second phase, on the behavioural flexibility of business transactions. I repeat the research questions below:

- **RQ 1:** Does any existing business transaction model or technology provide design-elements that can be used to define or describe business transaction instance-models by correlating the business and functional perspectives? If **yes** then,
  
  - *does the model or technology provide all necessary design-elements to design business transactions of end-to-end business processes? If **not** then,*
  
  *does the model or technology provide one or more design-elements that can be reused to design business transaction instance-models?*

- **RQ 2:** Does any existing business transaction model or technology provide techniques to define flexible business transactions? If **yes** then
  
  - *does the model or technology provide all required techniques to define the behavioural properties which can prevent faults at runtime and can guarantee highly flexible behaviour of business transactions? If **not** then,*
  
  *does the model or technology provide one or more techniques that can be reused to define the flexible behaviour of business transactions?*

Regarding existing models and technologies, my assumption is as follows, **there is at least one business transaction model that provides all necessary design-elements and techniques to design flexible business transactions by correlating the business perspective of end-to-end business processes with the functional perspective.** I will validate the assumption by reviewing the current business transaction models and technologies.

### 2.2 Overview of Business Transaction

The concept of business transaction has been viewed in various contexts. It has been well documented in the literature where some researchers share similar meanings and others argue its meaning within different contexts.

The research on business transactions in the beginning was heavily focused on the **choreography layer**, the **orchestration layer**, and the **application layer** of SOA (Papazoglou, 2008). Choreography defines the interactions between the trading partners that are involved in business transactions. Orchestration is the composition of business transaction activities and the application layer integrates the applications that are accessed by the business transaction instances.
According to Kim H. (2002), the choreography defines the sequences of business transaction activities at the global level. Medjahed et al., (2003) share the same view and outline the importance of designing business transactions in this layer. In UMM (2006), choreography is viewed as the basic building blocks of business transactions. The authors state that “a business transaction defines a simple choreography of exchanging business information between two authorized roles and an optional response”. According to Kratz B. (2004 ) “business transactions are complex and multilevel, span many organizations since multiple parties with heterogeneous systems are involved, usually have a long duration and optional parts are automatically executed”. Papazoglou and Kratz (2006) points out that in multilevel business transactions, the interactions between partners are structured at the choreography level and the transaction activities that are internal to the business partners are defined at the orchestration level.

Hofman et al., (1994), proposed an interactive approach at the choreography level for designing business transactions. The authors define business transactions as “sequence of messages” which suggests that a transaction is triggered through the exchange of messages within a business management system (i.e. the initiator). Kambil and Heck (1997) share the same view. However, the authors combine information about changes of application states with business transactions. This is called the operational view. They emphasize significant information processing and communication (interaction) to reduce uncertainties for buyers and sellers, i.e. quality, commitment, and protocols in place for the resolution of conflicts. Clark et al. (2001), realize the importance of commitment. Essentially, the authors argue for the advantage of the notion of agreement in business transaction. The authors state that a business transaction is a set of business information and business signals which are exchanged between two commercial partners that must occur in an agreed format, sequence and time period. If agreement is violated then a business transaction is terminated and all business information and business signals which were exchanged must be discarded. The document flow structure (time, format, and sequence) which exists between parties is important for business transactions.

According to Kim H. (2002), a business transaction consists of two predefined business document flows and additional business signals”. Aissi et al. (2002) shared the same view. They focus on document flow and define a business transaction as “an atomic unit of work between trading partners. Each business transaction has one requesting (incoming) document and an optional responding (outgoing) document”. Essentially, by this definition, the authors give importance to the atomic characteristic of business transactions.

The OASIS (2002) consortium define business transactions as follows, “a business transaction is a consistent change in the state of a business relationship between two or more parties. A business relationship is any distributed state held by the parties which is subject to contractual constraints agreed by these parties”. In this, agreement is an important entity of business transactions. However, Papazoglou (2003) realized that in this definition the scope of agreement is delimited to stipulating the document flow structure; other
attributes of agreements such as service quality (which is the key concern in dynamic business processes (Boulding et al., 1993)) have not been taken into account. Papazoglou extends the scope of the agreements in business transaction by incorporating quality related characteristics of services in business transaction. According to Papazoglou and Kratz (2006), a business transaction is “a trading interaction between possibly multiple parties that strives to accomplish an explicitly shared business objective, which extends over a possibly long period of time and which is terminated successfully only upon recognition of the agreed conclusions between the interacting parties.” This indicates business transactions should be atomic over agreements between the partners. More specifically, the predefined objectives should be met through business transactions, and upon failure, the transaction should roll back.

Ciao et al., (2012) concentrate on business transactions at the composition layer. They define business transaction as “a composition of the web services (WSs) provided by multiple service providers”. This indicates that a business transaction is complex to manage because the asynchronous, statelessness, distributive and opaqueness natures of WSs (Yun, 2008) may promote the complexity of managing business transactions.

According to Hantry et al., (2010), business transactions must be governed by the business perspective. They define business transactions as “a series of collaborative activities that explicitly enforces the achievement of an agreed-upon business objective in end-to-end processes. The objective is subject to service-level agreements that govern the choreographed/orchestrated behavior, non-functional and timing requirements, correlated exchange of information, and control flow of composed services.” The successful completion of a business transaction implies that the activities (services) that are contained in an end-to-end business process have performed the operations successfully and the service level agreements between the trading partners are met; otherwise, the business transactions have not successfully committed.

In summary, as organizations and technology continued to evolve, what is understood by the concept of ‘business transaction’ also underwent a multitude of changes. The application landscape underwent several changes, for example, the traditional tightly coupled monolithic application changed to SBAs. Several definitions of business transactions have been provided by the researchers to adapt to such changes. Among these, Medjahed et al., Kratz (2004), Kim, H. (2002), and Papzoglou and Kratz share the same view which is, business transactions in service oriented environments reside at the choreography layer and they define the choreography between the trading partners. Hofman et al., (1994), and Kambil and Heck (1997) share almost the same opinion–with a small extension. According to them, business transactions should be interactive, that is, they should be done by exchanging a sequence of messages between the trading partners. Clark et al., (2001), proposed the notion of commitment in business transactions. The OASIS consortium shares the same idea but they call it ‘agreement’ instead of commitment. Papazoglou (2003) and Hantry et al., (2010), proposed business transactions with SLA, which is a subset of agreement containing the clauses
of service related commitments agreed on by the trading partners. The view of Ciao et al., (2012) is different from others. They view business transactions at the composition layer. A business transaction to the authors is the composition of WSs.

After critically analyzing the definitions discussed in this section, this thesis adopts the definition of business transaction provided by Hantry et al, (2010). The main reason for choosing this definition is it focuses on the correlation between business and functional perspectives of end-to-end business processes. Also, the way the business transaction has been viewed by the authors fits the contexts as well as the objectives of this thesis. However, a slight modification is made to adapt the notion of flexible behavior in business transactions. In this thesis, the business transaction is defined as follows,

“A business transaction is a series of transaction fragments that are distributed across a network of business partners and performed collaboratively in a flexible manner by accomplishing the commitments agreed upon by the partners”.

2.3 Business Transaction Models

In many articles (e.g. Gray and Reuter, (1989), Leymann and Roller (2000), Papazoglou (2003)), the researchers argue that transaction models that rely on classical ACID properties are not suitable for long-running applications. This shortcoming led to the advent of the business transaction models. In this research, I investigate existing business transaction models. In my investigation, I focus on the two main features of business transaction models: ability to correlate business entities with the functional perspective elements and ability to provide techniques or mechanism to enable users to define the flexible behaviour.

2.3.1 Nested Transaction Model

The shortcomings of single layered transaction models gave rise to the multilevel transaction models that enable parallelising the execution of transactional activities. This technique was introduced in the Nested transaction model (Moss, J.E., 1981). The key idea of this model is to allow the transaction designers to design complex and functionally decomposable transactions from top-down (Papazoglou, P.M., 2008). It enables business transaction systems to perform transaction activities concurrently which enhance operational flexibility over the flat transaction model. The multilayer architecture has been adopted in all the business transaction models existing today because it enables running a wide array of (sub-) transactions in parallel.

The top-down decomposition of transactional activities in multilevel transaction models creates a hierarchical structure which can be viewed as a tree of sub-transactions. The child sub-transactions in this structure are nested within the parent transactions. A nested transaction model allows these child sub-
transactions to commit or abort independently, which is a flexible approach that is useful for designing a flexible business transaction.

The nested transaction models are categorized as closed nested transaction models and open nested transaction models. Structurally both models are similar but they are different from the behavioral point of view. I analysed both because the investigation was necessary to uncover the techniques they provide in terms of handling the transaction behaviour.

Staring with the closed nested transaction model, which was pioneered by Moss, J.E. (1981), he views a transaction as a tree of sub-transactions organized hierarchically. In closed nested transaction model, the smaller execution granules are allowed for partial rollback, and therefore, finer error handling results in less work being repeated. The model inherently supports reuse of transactions as modules, since they can be combined as sub-transactions into a nested transaction. Furthermore, the nested transaction model allows sub-transactions to be executed in parallel, while ensuring correct synchronization within a transaction (Moss, J.E.). Figure 2.1 depicts the closed nested transaction model.

![Closed Nested Transaction Model](image)

**Figure 2.1:** Closed Nested Transaction Model that allows decomposing a large transaction into multiple units.

In Figure 2.1, sub-transactions, for instance $T_{11}$ and $T_{12}$, are nested within transaction $T_1$, which in turn is a child of the outermost root (also known as the parent) transaction $T$. The innermost transactions are known as leaf transactions, e.g., $T_{11}$ and $T_{21}$ in Figure 2.1 are the children of the parents $T_1$ and $T_2$, respectively. The children of the same parent are called siblings, e.g., $T_{31}$ and $T_{32}$ are siblings. Noticeably, the depth of the tree of transactions in a nested model is arbitrary. It usually depends on the scale of the applications.

Parallelisation and partial rollback enable transactions to perform in a flexible manner. Although, in
principle, parallelization should help in designing transaction activities non-deterministically and allow the transaction model to be extended as required, the closed nested model does not provide either of these features. I concluded that they were not within the scope this model. Furthermore, Leymann and Roller (2000) and Lewis et al. (2002) state that the parent nodes in nested transaction models comply with the ACID principle, this is a serious drawback of this model for designing business transactions. With these properties, it is not possible to design fault-tolerant and failure-resilient business transactions. Another important drawback is the lack of visibility of outcomes produced through committing a child sub-transaction. The closed nested transaction model does not allow any modification in a child sub-transaction to be visible to other sub-transactions that originate from the same parent. In indicates that child nodes are not autonomous. The classical serializability technique provides full isolation at the global level. Therefore, a closed nested transaction model is incapable of designing business transactions for process-based applications (Papazoglou, 2003).

In order to overcome the problems of a closed nested transaction model, Moss, J. E. (2006) proposed the open nested transaction model. The open nested transaction model relinquishes the strict isolation property of the closed nested transaction model. Giving up the isolation property while retaining the intra-transaction parallelism, promises to increase the throughput within the overall environment (Leymann and Roller, 2000), (Bukhres et al., 1992). From the structural point of view, both closed and open nested transaction models are the same; however, they are different from the behavioral point of view. In an open nested transaction model, the sub-transactions can commit or abort independently of the status of parent transactions. In other words, there are no restrictions with respect to the semantic relationships between the parent and child transactions. It increases the operational flexibility and hence the behavioural flexibility of business transactions.

The open nested transaction model relaxes isolation which is another form of enhancing flexibility of business transactions. It allows circulating effects of sub-transactions across the transaction tree as soon as they commit. This means the effects of sub-transactions are not only visible to parents but to all other (sub)transactions in the tree. The demerit of this approach is that the consistency of the database can be compromised, especially for the business transactions of specific business activities such as inventory checking that rely on shared database accesses by the parties involved in business transactions. Also, consistency can be compromised if there is a data dependency between the sub-transactions. Inconsistent outcomes may lead to failure of one or more sub-transactions, which may cause business transaction abortion. For instance, company ‘X’ has an inventory application whose database is shared by many parties, and hence many process instances share it. An instance (i) accesses that database, (ii) READs and WRITEs the data, and then (iii) the transaction is committed that results in changing number of data items in the database, and finally (iv) the outcome is published to the outside world immediately. If, for instance, another transaction,
2.3. BUSINESS TRANSACTION MODELS

...delivery transaction, of that instance fails (for any reason such as shipment failed), then it promotes inconsistency in the inventory database because the transaction results in a mismatch between the number of items shown in the database and actual number of items stored in the inventory.

An open nested transaction model offers a very effective technique called compensation to avoid such inconsistency. Compensation undoes the effects logically instead of the classical way of undoing the transaction. This is a widely used technique and very effective for business transactions because it enables avoiding the classical rollback that leads a business transaction to the initial state when a sub-transaction fails. Compensation reduces the computational cost of business transactions.

However, Gray and Reuter called the open nested transaction model an anarchic version of multilevel transactions because the sub-transactions within the tree cannot be controlled. Although ‘autonomous’ is an important characteristic for business transactions, yet a certain level of control is necessary because, by the definition provided in this thesis (see Section 2.2), business transactions should be governed by agreements. This implies that a sub-transaction cannot leave the transaction scope whenever it wants. In business transactions, each transaction participant performs sub-transactions locally and autonomously, but they are controlled by agreements.

In summary, both the closed and open nested transaction models have several advantages. However, there are several disadvantages. First and foremost, these models focus only at the lowest level of abstraction. More specifically they provide techniques to design flexible operations at the functional level, yet do not provide any design-element. No business perspective elements were found in either of the models. Therefore, to answer the first part of the research question RQ 1, these models are not capable of correlating business and functional perspectives. Consequently, the remaining parts becomes void.

Considering RQ 2, the nested transaction models provide techniques including parallelization, compensation, and relax isolation that enables designing flexible behaviour to some extent. However, these techniques are not sufficient to design highly flexible behaviours which can handle failures that are caused by faults related to business activities (Chapter 5 describes the business related faults). It is worth noting that relaxing isolation is not within the scope of the research. While parallelization only increases the concurrency, the technique does not deal with transaction faults. The open nested transaction model provides ‘compensation’; however, according to Greenfield, P. (2006), compensation enables the design of failure-resilient business transaction models but it is not enough.

2.3.2 Saga

The ‘Saga transaction model’ was introduced by Garcia-Molina and Salem (1987), to enable applications to: (i) determine compensation transactions and (ii) invoke compensation transactions automatically. It was the first commercial model for designing long-running transactions.
The key idea of the Saga transaction model is similar to several existing models, e.g., nested transaction models. It is a two-level transaction model. Therefore, the basic structure of Saga is similar to multilevel transaction models. However, a ‘Saga’ permits only two levels of nesting: the top level Sagas and simple transactions. This is essentially a limitation from the structural point of view: it cannot be used to design business transactions of complex end-to-end business processes because these processes are composed of several activities which again can be composed of activities.

A ‘Saga’ comprises a set of independent sub-transactions (also called component transactions) that can be executed sequentially as well as concurrently. Each sub-transaction in a Saga model is a flat transaction that relies on ACID properties and can interleave arbitrarily with other sub-transactions. Interleaving is definitely an effective and efficient technique because it allows designing and executing the transaction activities in any order. However, as mentioned in the beginning of this section, ACID properties are not suitable for business transactions. Essentially, due to these properties the Saga model cannot be used for designing business transactions of end-to-end business processes. However, similar to the open nested transaction model, ‘Saga’ provides ‘compensation’. In the Saga transaction model, each sub-transaction, $T_i$ (where $i = 1 \ldots n$), is associated with a compensation transaction $C_i$, (where $i = 1 \ldots n$). Upon failure of a sub-transaction, the transaction system invokes the associated compensation transaction to revert the transaction effects from the semantic point of view. It is worth mentioning that the compensation activities in Saga are invoked automatically. Figure 2.2 shows the compensations associated with sub-transactions.

![Figure 2.2: Example of Compensation Transaction Associating Sub-transactions.](image)

In addition to compensation, Saga introduces a new technique called the Savepoint, which is an intermediate state in a ‘Saga’ transaction. The Savepoint prevents rolling back the transaction control to the initial state. Rather, it takes the control to the most recent state. To explain, upon failure transactions rollback to the the last saved transaction activity and continue from there. Garcia-Molina and Salem called it forwards.

---

1 Reproduced from Garcia-Molina and Salem (1987)
In summary, ‘Saga’ does not provide any modeling elements that can be used to design business transactions by correlating business and functional perspective elements. Essentially, this model concentrates on techniques for designing flexible operations. Although Saga provides techniques such as ‘savepoint’ and compensation to define the flexible behaviour of business transactions; however, like the nested transaction model these are inadequate for designing flexible business transactions. I already discussed the shortcomings of compensation in the previous section. The ‘savepoint’ is a new technique but it was proposed only to complement compensation operation. In addition, the Saga transaction model does not allow partial execution. Therefore, it cannot ignore any failure. If it is not possible to repair a failure of the running transaction, the transaction has to be compensated and rolled back to the last ‘savepoint’. Also, the top level transactions in Saga models rely on the ACID properties. Therefore, failure of a child transactions is not allowed. This approach is definitely not suitable for designing fault-tolerant business transactions.

2.3.3 Cooperative Transaction Model

Cooperative transaction hierarchy (Nodine and Zdonik, 1992) was developed to overcome the strict correctness criteria of the traditional serialization technique. Nodine and Zdonik (2002) argue that serialization is an efficient technique, yet the serialized transactions consume significant amount of time due to its strict criteria. The cooperative transaction model addresses this shortcoming by providing protocols to design cooperative transactions.

A cooperative transaction hierarchy is a structured set of cooperative transactions, where the structure of interactions among the cooperative transactions reflects the underlying hierarchical decomposition of the tasks they are working on together (Nodine and Zdonik, 1992). The model comprises internal nodes called transaction groups that contain a set of member transactions called cooperative transactions and the root nodes. The roots maintain the database, while the actual transactions are carried out by the cooperative transactions that interact with each other. Since the transactions are performed in cooperation, each cooperative transaction knows what other is doing. Figure 2.3 illustrate the structure of the cooperative transaction model.\(^2\)

The model employs a protocol that is a set of rules for constraining the operation of a transaction group. The rules must be satisfied while performing transactions. Rules that are applied in the root nodes guide what operations on root’s objects are allowed, whereas rules in ‘transaction groups’ control interactions among cooperative transactions. Unlike the previous multilevel transaction models, in particular, Nested and Saga, cooperative transactions rely on these protocols to maintain transaction integrity.

The cooperative transaction model is extensible. The model can be tailored dynamically. Transaction

\(^2\)Reproduced from (Nodine and Zdonik, 1992)
groups and cooperative transactions can join and leave the hierarchy as the overall task progresses (Nodine and Zdonik, 1992). This type of structural flexibility is very effective for business transactions as sub-transactions may leave and join on the fly at runtime.

To summarise, the cooperative transaction model is yet another multilevel transaction model and an effort to enhance flexibility in transactions. Additionally, the model concentrates on guaranteeing transaction integrity, for which it provides internal and external protocols. However, the business perspective of transactions is beyond its scope. There is no element which can be used to design business transactions by correlating the business perspective with the functional perspective.

Considering RQ 2, the only technique the model offers is extensible which is certainly not sufficient to design the flexible behaviour of business transactions. Moreover, in the cooperative transaction model, there is a strong sphere of control between parent and child transactions. No child transaction is able to commit until the respective parent commits. This strict atomic principle forces abortion if the parent transaction fails. The operational semantics (in terms of atomicity) of a cooperative transaction—in this regard similar to the closed nested transaction model. Therefore, this approach is not suitable for business transactions. Additionally, in cooperative transactions any object produced through a sub-transaction cannot be visualized to the outside world immediately. It follows two steps: the transaction group delivers the newly created object to the parent transaction then, in the second step, the parent decides whether or not the object should be made available for other transactions. This means, visualization is controlled by the parent transaction of the tree. Such a strict principle will surely degrade the performance of a business transaction at runtime.
2.3. BUSINESS TRANSACTION MODELS

2.3.4 Split-Join Transaction Model

The Split-Join transaction (Pu et al., 1988) model was proposed to design uncertainty regarding (a) time and actions, and (b) interactions among concurrent activities. The key purpose of the model is to relax isolation for uncertain activities that are called open-end activities. Figure 2.4 shows the Split-Join transaction model.3

![Split-Join transaction model (Source (Pu et al., 1988)).](image)

A Split-Join transaction is processed in two steps: split first and then join. In this model, a transaction is split into two or more transactions that commit independently and join to a target transaction. In the above Figure, T splits into $T_A$ and $T_B$ (each contains a set of operations) that commit independently and join to transaction S subsequently.

In the Split-Join model, transactions $T_A$ and $T_B$ are serial, which implies that they commit sequentially. Serialization is an efficient approach for database transactions, for instance, performing WRITE and READ after read operations but it is a highly restrictive technique for business transactions (Hantry et al., 2010).

Pu et al., (1988), argue that the original transaction T (in Figure 2.4) and the target transaction S are siblings but T can be thought of as a sub-transaction of transaction S. This definitely contradicts the structural principle of a multilevel application. According to the principle of multi-level transaction, siblings lie at the same level in a tree-like structure of a transaction. If ‘T’ is a sub-transaction then it must be a child of S.

The model relies on the atomic principle. The atomic relation between $T_A$ and $T_B$ (shown in Figure 2.4) prescribes that if $T_A$ fails to commit, $T_B$ has to abort. The failure of either of these transactions affects the subsequent join transaction as well. For example, T (in Figure 2.4) cannot join to the state S unless $T_A$ and $T_B$ commit successfully.

Relaxing isolation is the key purpose of splitting a transaction. The Split-Join model provides the commit serializability technique that allows transactions to visualize the results and release the resources immediately after the commit. To explain more, consider two transactions $T_1$ and $T_2$. $T_1$ splits into A and B and delegates operations such as read and write to these transactions. If A commits then it immediately makes its result public and releases its resources; therefore, its results are accessible by the other transaction.

3reproduced from (Pu et al., 1988)
$T_2$ while B continues its operations. In the Split-Join model, the subsequent transaction $T_2$ does not need to wait for B to commit to access A's result. Additionally, the Split-Join model introduces participation domain to relax the classical serializability for user-centric transactions. The users are the participants and the domain (D) is the union of a set of transactions (T) and a set of participants (P) such that,

$$D = \{T\} \cup \{P\}$$

The users are also called the observers. The key idea behind the domain based approach is in eliminating serializability between transactions. This indicates that transactions in the same domain are not serialized. Transactions can commit and abort without caring whether or not their succeeding or preceding one has committed. Such a non-serialized model should reduce transaction processing time and, therefore, should improve system performance.

In summary, like the previous models, the Split-Join transaction model concentrates on operational flexibility. It does not provide any modeling elements. Therefore, this model cannot be used to design business transactions. It provides techniques for defining autonomous and extensible properties of flexible business transactions. However, the properties can not be used to handle certain type of failure failures. Therefore, these properties are not sufficient for designing highly flexible business transactions.

### 2.3.5 S Transaction Model

The S Transaction model was developed by Veijalainen and Eliassen (1991) to design transactions for large-scale inter-organizational autonomous environments where cooperation and autonomy are the primary concerns. It is yet another model of the multilevel transaction family type. The key objectives of this model are similar to Saga – relaxing atomicity and isolation. The model employs the notion of compensation for relaxing atomicity. For relaxing isolation, upon commit, transactions are allowed to make the effects visible to outside world.

The S Transaction model introduces a new failure prevention technique called alternative component system. This is the novel aspect of this model and an effective technique that can be used for defining flexible behaviour. In the model, a system component performs a sub-transaction independently out of the influence of other system component. Upon failure of a sub-transaction, the component system associated with that failed sub-transaction sends a request to another system component to perform the sub-transaction. This technique reduces the failure rate in business transaction significantly. In addition, S Transaction model supports both backward and forward recovery modes which can be selected dynamically.

To sum up, business perspective design-elements were not found in this transaction model. The correlation therefore is outside of its scope. However, the S Transaction model is richer in terms of failure prevention techniques than the models discussed in the previous sections. Those models rely mainly on ‘Compensation’ and ‘Savepoint’ techniques to prevent failures. In addition to these, the S Transaction
model provides a new technique called *alternative component system*. It is effective for designing flexible behaviour. However, the model provides techniques which can handle faults that are related to operations from the functional perspective. If business related faults (*e.g.*, violation of agreement) occur, these techniques are not sufficient. The business related failure needs different treatments. For instance, if agreement violation happens, then applying backward or forward recovery is not computationally cost effective, because for every violation the transactions compensation will execute transactions in a reverse order and restart from the last savepoint. The alternative components can be effective in such cases; however, it needs supplementary recovery operators because a business transaction fault may create a chain of faults (See Chapter 5, Section 5.8.7).

### 2.3.6 ConTract Transaction Model

The ConTract model was proposed by Reuter and Wächter (1991) to provide a generalized mechanism for controlling business transactions; more specifically, for controlling the interrelations among transactions. A ConTract transaction model is composed of multiple predefined *steps* which are sequential programs (Reuter and Wächter, 1991). Each step is an integral part of a transaction or it can be a transaction itself. These steps are interrelated. The interrelation (control flow) is defined declaratively by a script. The scripts describe the control flow and all other dependencies among the steps. The scripts allow more complex procedures related to workflows. They allow flexible error handling that can be realized by calling alternative steps dynamically or rolling the entire ConTract back. The notion of the alternative step is similar to the S Transaction model.

To handle system errors, such as a crash, the execution environment of the ConTract has to be restorable. The context of the ConTract is therefore persisted, which involves saving states of global variables, intermediate states of the control flow, screen contents etc. After resuming from a crash, the ConTract can be rebuilt using these data.

Relaxing isolation is another goal of the ConTract transaction model. In order to relax isolation, it supports *synchronization from the semantic point of view* which depends on the invariants. The invariants prescribe the application-dependent state constraints which are to be hold during the execution of a ConTract, *e.g.* the restriction of a variable to a specific range. With this concept, locking resources is not required. Obviously, this is a significant achievement from the technical standpoint. Applying this idea, serializability may not be guaranteed, but, correctness of the process execution can be ensured semantically and also the possibility of conflicts reduced.

In the ConTract model, each step can have an incoming and an outgoing invariant. Incoming invariants have to be true when starting the step, whereas outgoing invariants have to be true when the step is finished. For instance, in a purchase order business transaction, an invariant can ensure that the initially
chosen product remains available until the order is finally placed. This techniques — using invariants for constraining transaction — ensures the correctness and, therefore, it is promising for business transactions.

To sum up, in addition to the fact that this model does not provide any design-element related to the business perspective of an end-to-end business processes, it guarantees atomicity (Haerder and Reuter, 1983). As already explained as with any transaction model that relies on ACID principle, it cannot be considered for modeling business transactions.

2.3.7 TMLA

Transaction Model for Long-running Activities was proposed by Dayal et al., (1991). The model comprises activity and transaction. Activity is essentially a long-running business process comprising steps that are either activities or transactions or a combination of both that is referred to as action in the model. The model supports designing the tree-like structure of business transactions where the top level transactions are the root transactions and top level activities are the root activities. The model provide techniques to relax the strict relation between the parent and child nodes by allowing parent transactions to remain active upon failure of their child transactions. This approach is similar to the Open Nested Transaction Model.

The model provides two techniques for preventing business transaction failures: alternative sub-transaction and retry, which is a new approach. Additionally, TMLA provides the ‘compensation’ for undoing the results upon failure of a transaction. Compensation is necessary, as root transaction cannot be failed due to the failure a child transaction. The results of root transactions are retained; this is why, the compensation activity may be executed to undo the results logically.

To sum up, the model was developed to define specifically the retry, alternative, and compensation logic to prevent failure and hence to provide runtime flexibility in business transactions. The model does not provide any elements for designing business transactions. Considering the techniques, in addition to compensation and alternative sub-transaction, it provides retry which can be very useful to prevent the failure of retryable sub-transactions contained in a business transaction. However, these do not suffice to prevent failures that result from business-oriented faults.

2.3.8 WIDE

A WIDE business transaction model (Ceri et al., 1997) is developed by combining short-lived transactions and business transactions. The model enables designing business transactions at the global layer and ACID transactions at the local layer. Such a decoupled two-layered business transaction model aids in efficiently managing complex business transactions (Ceri et al., 1997).

The local layer ACID transactions are subsumed in the business transaction. An ACID transaction is purely atomic, whereas a business transaction follows the principle of the ‘Saga transaction model’ with
an extension, such that the atomicity is relaxed by allowing the partial rollback upon failure of a sub-transaction. The WIDE model rolls the transaction back to the last savepoint. The Saga’s savepoint has been adopted in the WIDE model but it has been called ‘safe point’.

The WIDE workflow model is driven by Event-Condition-Action (ECA) rules that drive the business transaction flow of applications. While running business transactions, if the ECA rule of a flow fails, a sub-flow is invoked for preventing the abortion of business transactions. It relaxes strict the atomic nature of transactions. Isolation is relaxed in this model by allowing the intermediate steps visibility outside of the scope of a sub-transaction. Many transaction models already offer this technique.

The WIDE business transaction model concentrates on combining business and ACID transaction models and separating them for efficient management. It offers some basic recovery features include backward and forward recovery. These techniques, however, are not sufficient for complex business transactions. In addition, in the WIDE model, a transaction at the local level is atomic which is not an efficient approach. If a model relies on the atomicity principle at any level in its hierarchy, designing fault tolerance and failure-resilient business transaction becomes impossible.

2.3.9 Business-Aware Transaction Model

The business-aware transaction (BAT) model was proposed by Papazoglou and Kratz (2006). The main focus of the model is to establish a correlation between the business and functional perspectives of end-to-end business processes. It concentrates on providing operational business principles for modeling business transactions by specifying their Quality of Service characteristics. Figure 2.5 shows the BAT model.

The model contains some design-elements that enable specifying the business aspect of end-to-end business processes. For instance, it contains ‘Party’ and ‘Role’ that are used for specifying the trading partners involved in business transactions and their roles. Also, it contains BusinessConstraints that composes different types of constraints that are called InterParty, Invariant, and SectorialInvariant. Each constraint type is sourced from different entities. For instance, InterParty originates from agreement, Invariant is from regulation, and SectorialInvariants are from the specific business sector such as telecommunication, banking etc. Additionally, the model contains functional perspective elements such as Activity.

Essentially, BAT was the first business transaction model that provides business related modeling elements. However, the model is missing explicitness in terms of business perspective elements such as agreement. It is not clear how a clause of trading partner agreement can be defined using this model. Additionally, there are some flaws in the model. The association relation shown in the model between BusinessOperation and Activity is a design flaw. Because, according to the definition provided by Leymann and Reuter (2000), an activity is contained in the process rather than it associates BusinessOperation. In addition, SimpleOperation and Process are presented as subtypes of business operation which is a clear design flaw. If activities
perform business operations, then \textit{BusinessOperation} should be contained in the element \textit{Activity} which by definition is a member of \textit{Process}. This implies that the \textit{Process} can never be a subtype of \textit{BusinessOperation}. Essentially, the semantics of \textit{BusinessOperation} in the model is not clear. Its relation with other elements implies that it is a business function; however, there already is an element called \textit{BusinessFunction} in the model which creates confusion. If \textit{BusinessFunction} and \textit{BusinessOperation} are semantically the same then one of them is redundant in the model. Moreover, since activity is shown as the smallest unit in the model, \textit{BusinessOperation} and \textit{SimpleOperation} are invalid elements.

BAT introduces business level atomicity criteria, which essentially is a form of atomic business transaction taking strategic constraints as critical conditions, which if violated result in the abortion of a business transaction. This is not the traditional atomicity but certainly promotes the atomic nature of business transactions. It has already been explained in the previous sections why the atomicity property is not suitable for business transactions.

To Summarise, this is the first business transaction model which provides business perspective design-elements and, thus enables to the design of business transactions by correlating the business perspective elements with the functional perspective elements. However, the model lacks explicitness. The definition of many business perspective design-elements of the model are not clear. In addition, several redundant elements and logical flaws have been found in the model. If the redundant elements are taken out of the model, then it might result in incompleteness. The flexible behaviour of business transactions is the outside
of the scope of this model. It only provides *contingency transaction* technique.

### 2.3.10 TxQoS-Aware Approach

The Transactional Quality of Service (TxQoS) (Wang, 2011) is a specification for transactions in service-oriented environments. The specification is used for designing business transactions of service oriented applications. Wang (2011) defines TxQoS as the agreed reliability between a service provider and a service user with regard to the transaction performance of executing services. The TxQoS approach enables a business transaction system to interpret the term *reliability* from the business perspective to make it comprehensive for the business parties. It enables encapsulating a business transaction specification as an *e-contract*. However, the concept contract in TxQoS has a limitation. It is a service contract, which is plainly focused on services such as IT services. There is an intersection between service contract and trading partner agreement, however, they are different from many perspectives. One obvious difference is a service contract contains the obligations corresponding to the services such as performance time, and reliability, while a business contract contains the delivery method, delivery means, delivery point, packaging specification and so on. The common aspects of these two contract types can be payment means, payment date, penalty *etc.*. Therefore, although the TxQoS specification can be used for specifying constraints with service oriented applications, it is not suitable for designing the business transactions of end-to-end business processes which are highly business focused.

In summary, the model enables the design of business transactions by correlating partially the business perspective elements with the functional perspective elements. It is partial because the model enables specifying only the subset of ‘agreement’ which is in the QoS clauses. This implies the model lacks a complete set of design-elements. Considering techniques, the model offers ‘compensation’ and ‘forward recovery’, like Saga, which are not sufficient for designing a flexible business transaction as these cannot prevent failures related to the business aspect of transactions.

### 2.3.11 CoBTX-Net

The Choreographical Business Transaction (CoBTX-Net) model was proposed by Sun and Yang (2008) to guarantee reliability in collaborative business transactions. According to the authors, collaborative business transactions are more prone to unreliability than a single processed transactions due to its peer-based nature. They defined reliability in the context of collaboration as follows, a business transaction is reliable if the collaboration is carried out as planned, each partner behaves as agreed, and the transaction completes at the right time and in the right place. The model supports designing collaboration between the partners.

The CoBTX-Net model was developed by combining the individual Business Transaction–Nets (BTx-Nets) (Sun and Yang, 2007a) where the business transaction network has been broken down into three
layers: Execution subnet, Abstraction subnet, and Communication subnet. At execution level the internal activities are formed as Exe-subnet, at abstract level the messages are defined as Abs-subnet, and at the communication level, communication patterns such as request-response and notify are defined as Com-subnet.

Essentially, the CoBTX-Net is an extension of BTx-Net. It builds business transaction model by composing individual BTx-Net. The CoBTX-Net adds two layers: ExCom-net and ExAbs-net. The ExCom-net is a composition of interactions between the partners, and the ExAbs-net represents the publicly visible interfaces of BTX-Nets. In ExAbs-net, the CoBTX-Net binds the internal processes of involved business partners.

The CoBTX-Net facilitate observing the public behaviors of business partners involved in business transactions. It enables detecting the logical conflicts regarding collaboration so that the deadlock can be avoided. It also enables healing any improper termination as the behavior of business partner is visible.

To sum up, the CoBTX-Net enables defining high-level behaviors such as the interaction behavior of the business partners. The interaction behaviour means the messages exchange between the trading partner systems – which is the functional aspect of business transactions. The model does not provide business perspective design-elements like the Business-Aware Transaction model does. Additionally, flexibility in terms of operational behavior of business transactions has not been discussed at all in CoBTX-Net model.

### 2.3.12 Summary of Business Transaction Models

I discussed existing business transaction models in this section. The strength and weaknesses of each model was studied in the context of business transaction to find the answer to the research questions presented in the Introduction as well as in the beginning of this Chapter.

It is worth noting that this section studied business transaction models, the technologies are studied in the next sections. The study has revealed that there are models which offer both business and functional perspective design-elements. Also, these models provide techniques for defining the flexible behavioural properties of business transactions. However, there is no single model which provides the necessary design-elements and techniques for designing flexible business transactions by correlating business and functional perspectives. The implies the assumption regarding the current business transaction models and technologies does not hold true.

Considering RQ 1, there are a few models that provide one or more design-elements and techniques for designing flexible business transactions by correlating the business and functional perspective of end-to-end business processes. However, unfortunately, none of these models alone is capable of providing all necessary design-elements or techniques. Considering the third part, Table 2.1 and 2.2 show the reusable design-elements and techniques. All the design-elements and techniques provided by the models are
2.3. BUSINESS TRANSACTION MODELS

Table 2.1: List of Design Elements

<table>
<thead>
<tr>
<th>Design Elements</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Perspective Elements</strong></td>
<td></td>
</tr>
<tr>
<td>Quality of Service</td>
<td>BAT, TxQoS</td>
</tr>
<tr>
<td>Strategic Constraints</td>
<td></td>
</tr>
<tr>
<td><strong>Functional Perspective Elements</strong></td>
<td></td>
</tr>
<tr>
<td>Business Function</td>
<td>BAT</td>
</tr>
<tr>
<td>Business Object</td>
<td>BAT</td>
</tr>
<tr>
<td>Protocol</td>
<td>BAT</td>
</tr>
<tr>
<td>Role</td>
<td>BAT</td>
</tr>
<tr>
<td>Activity</td>
<td>BAT</td>
</tr>
<tr>
<td>Process</td>
<td>BAT</td>
</tr>
<tr>
<td>Actor</td>
<td>BAT</td>
</tr>
</tbody>
</table>

Table 2.2 shows the techniques that were found in the literature.

Table 2.2: List of Techniques for Flexible Business Transactions

<table>
<thead>
<tr>
<th>Technique</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensation</td>
<td>Open Nested, Sagas, S, TMLA, TxQoS</td>
</tr>
<tr>
<td>Forward Recovery</td>
<td>TxQoS</td>
</tr>
<tr>
<td>Retry</td>
<td>TMLA</td>
</tr>
<tr>
<td>Contingency Transaction</td>
<td>S, BAT, ConTract</td>
</tr>
</tbody>
</table>

Table 2.1 reveals that many of these models do not provide any design-elements. BAT is the only model which provides a few elements for modeling business transactions by incorporating both business and functional perspectives of end-to-end business processes. TxQoS provides only one business perspective design-element. Although, BAT is a potential candidate, however, the model is missing some critical elements such as agreement. Additionally, the model does not consider the flexibility issue within its scope. It provides only one technique for defining flexible behaviour (See Table 2.2).

Moreover, Table 2.2 shows that although many of these models focused on defining the flexible behaviour of business transactions, however, they do this by mainly relaxing the atomicity by preventing failure at functional level which is required but not sufficient for business transactions. The models must provide techniques to deal with a variety of faults including business-related faults, system-related faults etc.. This is critical weakness of the state of the art.
2.4 Technologies for Designing Business Transaction

Many technologies are available for designing and implementing business transactions. Most of these are open source, however, there are also commercial software products. I review both types in this research. Similar to the models, business transaction technologies were reviewed over the research questions RQ 1 and RQ 2 presented in Section 2.1.

2.4.1 ASSET

The ASSET (stands for A System for Supporting Extended Transactions) (Biliris et al., 1994) was proposed to implement long-running transaction models for business applications. The ASSET provides a set of transaction primitives that are influenced by ACTA (Chrysanthis and Ramamritham, 1990). The primitives are used to design application specific transaction models. In addition, the primitives enable specifying activities or workflows that are typically composed of transaction-like components. The ASSET primitives enable the design and implementing of different long-running transaction models including nested transaction models, Split-Join Transaction Model and Saga Transaction Model. In ASSET, transaction primitives are categorized into basic and new primitives. Basic primitives includes initiate, begin, commit, wait, abort, self, and parent whereas new primitives are delegate, permit, and form-dependency. It is worth noting that, these primitives cannot be used directly. Primitives are used by the code that is translated from a high level description of the transaction model. The data structure algorithms are used to implement the ASSET primitives. There are different data structures including transaction descriptor, object descriptor, lock request descriptor, permit descriptor, and transaction dependency graph for implementing transaction primitives. Although, ASSET is a promising transaction system that provides an opportunity to select the suitable long-running transaction model for business applications, the performance of this system has never been evaluated because ASSET was never implemented (Wang and Kaiser, 1999).

ASSET is essentially a library for implementing the long-running transactions. It provides some system level primitives and algorithms, however, ASSET does not provide design-elements. Two primitives provided by ASSET Contingency Transaction and wait can be useful for designing the recovery of business transactions upon occurrence of faults. However, these two are not adequate to repair a wide variety of faults (Chapter 5 describes the faults).

2.4.2 CheeTah

CheeTah (Pardon and Alonso, 2000) was developed to support transactions in a distributed and dynamic environment where a collection of heterogeneous and autonomous information systems interact with each other. The systems are called composite systems in CheeTah. The objective is to provide a lightweight architecture to support transactions of interactive composite systems. CheeTah supports the implementation
of the open nested transaction model for processing transactions of composite systems. It covers two aspects of transactions: flexibility and collaboration, which are critical to service-oriented business environments.

A transaction in CheeTah starts locally when a service has been invoked. Each local transaction is a thread contained in a global composite transaction that is built by the system using information from all threads. For controlling concurrency and scheduling, the call level locking is used. This locking mechanism checks for conflicts at service level but not at the operational level. It checks the correctness of transactions. Moreover, upon failure of a transaction, the undo operation of each component acts to recover the failed transaction by rolling back the database. Since the undo operation is local, it compensates only the local database.

To summarise, it is yet another technology that concentrates on the functional level of business transactions and therefore overlooks the need of business perspective elements. In fact, CheeTah is merely the technology to implement the open nested transaction model. It does not provide any design-elements. Considering techniques, the model provides only the undo operation which is essentially the compensation operation. I have explained several times in previous sections that ‘Compensation’ alone is not sufficient to prevent failures of business transactions. Additionally, the locking mechanism offered by CheeTah guarantees isolation to control concurrency, however, it is not a suitable approach for business transactions as it may degrade performance.

2.4.3 Java Activity Service

Sun Microsystems™ developed J2EE Activity Service (Java™ Activity Service) (Oracle Inc., 2006) for building long-running transactions in Java™ based applications. JAS is a generic middleware framework for constructing long-running transaction models. It provides an extended Unit of Work (UoW) that groups a related set of tasks as short-lived transactions within a longer activity. This technique relaxes isolation. To explain further, consider a business transaction that comprises several related tasks; JAS enables invoking resources for these tasks that release the resources immediately upon commit. Consequently, the isolation of resources is relaxed. However, if the transaction would access a resource as a single block then, it would lock the resource until all of its tasks carried out the READ and WRITE operations. JAS supports implementing the features of open nested and several other transaction models that prescribe visualizing resources upon commit in order to relax isolation.

The primary components of JAS are activity service and High Level Service (HLS). HLS is an implementation of a specific extended UoW that allows the interfaces and API to interact with corresponding applications and applications servers. Furthermore, activity service manages the service context of high-level services that are essentially activities from the business process perspective. Activity service provides the interfaces to the high-level services for supporting the context demarcation and coordination of HLS
specific objects. Upon failure of transactions, the activity services make use of the persistent information to recover that transaction. The decision to commit or rollback transactions depends on the information retrieved from storage.

To sum up, JAS was heavily focused on relaxing isolation. The correlation between business and functional perspectives was not within its scope. Yet, a functional perspective design-element called ‘Activity’ was found. Additionally, JAS does not provide any technique for preventing failures to increase flexibility in business transactions. Also, it implements the classical atomic principle which is not suitable for business transactions.

2.4.4 Web Service Transaction (WS-TX)

The Web Services Transaction (WS-TX) is a service-oriented technology. It is a framework consisting of WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity. The framework enables the design of both atomic transaction and business transactions for the web services. It provides two major components: WS-AtomicTransaction for designing the atomic transactions and WS-BusinessActivity for designing business transactions. The WS-Coordination specification describes models for organizing and coordinating Web Services (WSs) that are software components hosted at distributed locations. The coordination model organizes the independent services to operate in a consistent context to achieve a specific goal. Coordinators are typically used to disseminate information to the participants of a joint activity such as reaching consensus on a decision or to simply ensure that all participants obtain a specific message. The main work of the coordinator is to keep the track of transactions and various Web services involved in a transaction. WS-Coordination carries out its jobs in conjunction with WS-AtomicTransaction and WS-BusinessActivity.

Since atomic transaction is outside of the scope of the research, the WS-AtomicTransaction (WS-AT) (OASIS, 2009a) was not studied. The WS-BusinessActivity (WS-BA) (OASIS, 2009) is a protocol to define business transactions in web-based environments. It enables WS-Coordination to support business activities which require tentative operations visible to third parties before the final outcome of an activity, and also the case where exclusively locking of resources is impractical. Therefore, WS-BA can be used for applications needing long-running transactions and facilitates the application of business logic when exceptions occur.

WS-BA can be used to design a business activity as a hierarchy of scopes, where each scope includes those operations that may span different participants and are needed to drive completion of tasks. The notion of nested scope gives the opportunity to a parent of a scope to handle exceptions that may arise from its children and continue processing and to select which children will be included in the completion protocol. The state of the business activity is saved between steps in order to reach a desired goal, even if exceptions occur. When a child task is completed, the child decides whether to leave the business activity
2.4. TECHNOLOGIES FOR DESIGNING BUSINESS TRANSACTION

or to declare to its parent that the work is completed which can be compensated later if needed. In such a way, the WS-BusinessActivity relaxes atomicity based on business-level decisions.

To sum up, WS-BA is merely a protocol that implements the open nested business transaction model. It provides two functional perspective design-elements, *activity* and *process*, which are inadequate for modeling business transactions. In addition, no business perspective design-elements was provided by WS-BA. Considering the techniques, WS-BA provides ‘compensation’ for defining the flexible behaviour of business transactions.

2.4.5 Business Transaction Protocol

The Business Transactions Technical Committee (BTTC) of the Organization for the Advancement of Structured Information Standards (OASIS) developed Business Transaction Protocol (BTP) in 2004. The main purpose of BTP is to coordinating interactions between business partners from different organizations.

BTP incorporates *actors*, roles and their relationships. A relationship is the connection between two actors such as a client system and a remote system that represent two business partners. In addition, BTP defines state tables that specify the transitions between roles, detailing the exchanged messages, and internal decisions that affect the state. A participant in a business transaction is represented by two elements: an *application element* and a *BTP element*. The application elements exchange the information and cause the associated business functions to be performed, whereas the BTP elements send and receive messages.

BTP contains a set of specific messages exchanged between the systems supporting a business application to achieve interoperability between web service enabled transactions (Papazoglou, 2003). It is based on the two-phase commit for short-duration interactions known as atoms, which can be aggregated into larger non-ACID transactions known as cohesions (Bunting et al., 2001). The Atom was not studied because it relies on the ACID principle. The ‘Cohesions’ are business transactions. In Cohesions, a subset of involved participants have to agree to commit changes before a transaction is committed. This subset is called the *confirm-set* and is determined by initiating an application element that provides selection to the superior. Atomicity and isolation are relaxed in Cohesion.

Like WS-TX, BTP is a protocol for implementing atomic and business transactions. It provides business perspective design-elements, *Role* and *Actor* and two functional perspective elements *Activity*, and *Business Function*. These elements are not adequate to design business transactions. The critical business perspective design-elements such as ‘agreement’ are missing in BTP. It provides only three operators ‘compensation’, *contingency Transaction* and ‘wait’, which are not sufficient for defining flexible properties of business transactions.
2.4.6 Web Service Composite Application Framework (WS-CAF)

The Web Service Composition Framework (WS-CAF) (Bunting et al., 2003) was developed by OASIS to assist in designing different transaction models and architectures for composite applications. WS-CAF is a suite comprising three specifications that can be implemented incrementally to address a range of requirements needed to support a variety of complex composite applications. These specifications are Web Service Context (WS-Context), the Web Service Coordination Framework (WS-CF) and the Web Service Transaction Management (WS-TXM). WS-Context is a lightweight framework for simple context management; WS-Coordination Framework provides an extensible framework to support various transactional protocols and defines the behavior of a coordinator with which Web Services can be registered for context augmentation and result propagation; WS-Transaction Management includes three transaction protocols to model three different types of transaction models: ACID transaction model, long running action model (also called compensation), and business transaction model.

The ACID transaction model in WS-CAF supports designing short-lived transactions and the long running action (LRA) model is used to design the triggers for compensation actions. LRA designs the conditions upon which these triggers are executed. Each LRA is attached to the scope of an activity that reflects business interactions and all works that occurs within a scope is required to be compensable. The LRA coordinator protocol is used to ensure that an activity is completed successfully or compensated. Each activity is bound to the scope of a compensation interaction. The Business Transaction model aims to combine heterogeneous transaction domains together into a single business-to-business interaction. In this model, all parties involved in a business process reside within business domains. A business transaction is used to design interactions between business domains. A business process is split into business tasks. Each task executes within a specific business domain. A business domain may be divided into business domains in a recursive manner. An individual task may require multiple services to work. Each task is assumed to be a compensable unit of work. The business transaction model supports both synchronous and asynchronous interaction patterns.

In summary, WS-CAF is a protocol which can be used to implement three different transaction models. One of them is a business transaction model. It provides only two functional perspective elements business task which is essentially the same as Activity (provided by the BAT model discussed Section 2.3.8) and Protocol. Also, the only business perspective elements it provides is Actor. Essentially, the main focus of WS-CAF is designing the message exchanges between transaction participant applications that are hosted at different sites. In fact, in WS-CAF business transactions have been viewed as exchanging messages between the applications. LRA complements the business transaction model by offering a compensation model to design the logical revert upon failure of business transactions. This is the only technique provided by WS-CAF to prevent transaction abortion. Clearly, it lacks the ability to design flexible business transactions.
2.4.7 UN/CEFACT Modeling Methodology (UMM)

The UN/CEFACT Modeling Methodology (UMM) (UMM, 2006) was developed by the United Nations Center of Trade Facilitation and Electronic Business (UN/CEFACT). It focuses on designing a global choreography of inter-organizational business processes. Also, it is an effort to integrate the business domain with the choreography domain view, which is somewhat similar to the interests of this research. However, the transaction in UMM was seen as interaction between two partners.

Business transactions are the basic building blocks to define the choreography between two business partners appointed to authorize roles by describing the exchange of a business document and an optional response and/or business signals that acknowledge the correctness of business documents. UMM does not support multi-party collaboration which is a shortcoming for business transactions.

In UMM, business transactions align the business information systems of the collaborating business partners. Aligning refers to the action of maintaining and ensuring business objects (e.g., purchase orders, line item, etc.) are in the same state in each information system. If a partner recognizes an event that changes the state of a business object, it initiates a business transaction to synchronize with the collaborating business partner.

Two types of business transactions are defined in UMM: one-way and two-way. The former describes the synchronization process in a unidirectional way in which business information flows from the initiating business partner to the responding one. The latter describes the business information flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change (e.g., search for products, request for registration). In a business context, irreversible refers to returning to an original state that requires compensation by another business transaction e.g., once a purchase order is agreed, the rollback is not allowed, instead a cancel order operation is invoked to logically undo the result.

To sum up, it is yet again a model that views business transactions as a set of messages exchanged between the trading partner systems. UMM provides several functional perspective design-elements, business process, activity, actor and business object. Also, it provides a number of business perspective design-elements including Business Area, Process Area, Business Collaboration, Business Object, and Operational Constraints. Yet, it is missing a critical business perspective element ‘Agreement’. Although, it provides relatively more design-elements than the technologies discussed in the previous sections, yet, like most of them, it only supports designing business transactions as interactions between business partners. Additionally, it cannot be used for designing multiparty business transactions.

Considering techniques for defining flexible behaviour, it provides only compensation. More importantly, UMM relies on the atomicity property. In UMM, a business transaction is an atomic unit that leads to a synchronized state in both information systems. If an information system fails to synchronize the state of an object, the others must be aborted, even though those had completed successfully. The atomic principle
is not suitable for business transactions.

### 2.4.8 Business Process Model And Notation

The Business Process Model and Notation (BPMN 2.0) (OMG, 2010) is an emerging standard for modeling business processes. It is perhaps one of the most widely adopted technologies for modeling business processes. BPMN is a graphical language that provides a large number of notations for modeling business processes. The notation inherits and combines elements from a number of previously proposed notations for business process modeling (Dijkman, 2007), including XPDL (WFMC, 2002), and UML (OMG, 2005). BPMN supports designing activities, different types of events, different types of gateways, and message flows.

It is a rich language for modeling business processes. It provides all the elements required for modeling complex business processes.

IN summary, BPMN was developed specifically for designing business processes, however, the core business perspective was not considered in this language. It provides several functional perspective elements include Process, Transactional Workflow Handler, Activity, business Object and business function. However, the only notation it provides for designing the business perspective of end-to-end business processes is the Role. Considering the techniques for defining flexible behaviour, it provides notations for specifying Contingency Transaction which can be used to define the non-deterministic fragments of business processes and consequently, the operations becomes flexible. Additionally, it provides notations for defining compensation activity. Clearly, BPMN lacks the ability to support designing flexible business transactions.

### 2.4.9 Business Process Execution Language

Business Process Execution Language (BPEL) (OASIS, 2007) is used for the formal specifications of business processes and business interaction protocols. It was proposed by IBM, BEA Systems and Microsoft and submitted to the WS-BPEL Technical Committee of OASIS for standardization. It is designed to describe, coordinate and execute business processes by combining Web services with workflow concepts. It enables interactions between heterogeneous business domains based on a common set of specifications. Besides the language constructs for defining navigation in a process and constructs for interactions with Web services, BPEL provides a long-running transactional model (LRT) based on compensation-based recovery. In BPEL, this transactional model is described in terms of WS-BusinessActivity (WS-BA). BPEL uses one specific variant of WS-BA, the Business Agreement with Participant Completion coordination protocol to describe interactions between nested BPEL scopes. Therefore, BPEL extends the Web services interaction model and enables it to support business transactions.
The compensation activity in BPEL refers to the idea of invoking explicitly coded business logic to undo the effects of a successfully committed action or transaction. The compensation approach of LRT is based on Saga; however, BPEL provides a variant of such a compensation protocol by providing the ability to define fault handling and compensation in an application specific manner. LRT is meant to be used purely within a platform specific implementation. There is no prescribed requirement that the business process be distributed or span multiple vendors and platforms. Furthermore, LRT is purely local within a single business process instance. There is no distributed coordination regarding an agreed upon outcome for multiple-participant services.

In BPEL, activities are grouped into scopes to define transactional boundaries. In transaction processing, these scopes define the transactional context in the process definition. Handlers can be attached to scopes to provide actions for particular situations. These handlers are categorized into event handlers, fault handlers, a termination handler and a compensation handler. Scopes with compensation and fault handlers can be nested without constraint to an arbitrarily depth. Event handlers react to timing or message events and fault handlers cope with faults occurring inside a scope. A termination handler is called upon to force termination of a scope. A compensation handler is simply a wrapper of a compensation activity. When a scope completes successfully, its compensation handler is installed. Therefore, scopes are reachable even if they have finished their execution. A compensation handler can be called from the enclosing scope if the enclosing scope encounters a fault or is compensating itself. The compensation handler is called using compensate activity or compensateScope activity. A compensate activity specifies the compensation, in reverse order, of all nested scopes that have completed and compensateScope activity specifies one nested scope that has to be compensated through its target attribute. Compensation handlers use the current state of process at the time they are called. This state comes from its associated scope and the enclosing scopes and includes the state of variables, partner links and correlation sets.

To sum up, like BPMN, BPEL is focused on the functional perspective of end-to-end business processes. It provides all functional perspective elements provided by BPMN; however, it provide some additional elements that are: Event handler and Fault Handler. Like BPMN, it provides only one business perspective element which is Role. With this one business perspective element, it is not possible to make an explicit correlation between the business and functional perspectives of business transactions.

It provides Fault Handler to define the fault handling logic in business processes; however, it does not provide any specific technique which the users can use to define the flexible properties. It provides Compensation Handler which logically undoes business transactions upon failure. Additionally, BPEL provides the forward recovery technique. Although it is a rich language, like all other technologies, it lacks the ability to provide all necessary techniques to define flexible behaviour of business transactions. Compensation Handler, Fault Handler, and forward recovery are not sufficient to deal with different types
of faults that have been identified in this research.

2.4.10 Commercial Business Transaction Software

This section discusses commercial business transaction software. It is worth noting that I did not find any commercial business transaction model. However, there is a list of business transaction management (BTM) software. I analyzed the software products to find whether they provide design-elements or any technique for designing flexible business transactions. It is worth noting that these products are proprietary and therefore, a very little information was made public. The information was collected mostly from the company websites, specifically, from the product feature page. I cannot guarantee the accuracy of information because, companies in many occasions create hype for their products. The only white paper found was about the OpTier BTM yet it contains just basic technical information with more on the business side of the technology.

TransactionVision from HP is a Business Transaction Management (BTM) solution which supports managing business transactions at runtime holistically from both business and system perspectives. It provides convergence between low-level systems and network monitoring and high-level business process monitoring to a complete business service management strategy, and links IT performance with business performance (HP, 2011). It enables tracking and measuring individual transactions as they traverse through numbers of applications across a network. Some other notable features of TransactionVision include capturing key transaction performance metrics, providing early warning and real-time alert, and enabling auto-discovery of transaction flow. Alongside the integration with HP OpenView, TransactionVision can be integrated with HP Mercury Interactive products to provide complete platforms for managing business transactions.

BMC is a software vendor that integrates components for root cause analysis with its existing transaction management solution to diagnose transaction failure efficiently. BMC promises to (i) provide deep insight across the distributed n-tier architecture for synthetic (involves system and infrastructure) and real transaction (involves end-users), (ii) delivers diagnostics for root-cause analysis, and (iii) improve problem isolation and resolution for organization. BMC’s transaction management solution is composed of four components: Transaction Management Application Response Time, Transaction Management Real-Experience, MainView Transaction Analyzer, and Transaction Management Root Cause Analyzer which is based on AppSight technology. Each of these components provides a set of functionalities to manage the business transactions of large scale complex applications.

IBM has a large family of monitoring components. Tivoli Monitoring for Transaction Performance (TMTP) and Tivoli Composite Application Manager (TCAM) for SOA are the components developed for monitoring transactions. These two components jointly monitor end-to-end business transactions but reside
at two different layers. At the service layer, IBM TCAM monitors the overall health of services, message broker flows, and DataPower multiprotocol gateway support. In addition, it measures service unavailability. This specialized component is designed for monitoring various facets of services because services are the central components in service based applications and business transactions of these applications depend on the status of services. IBM TCAM gives full details about the status of services which may help the transaction processing system to be proactive to resist the potential failures. Furthermore, TMTP provides functions to monitor business transactions performance in a variety of situations. It enables the system to monitor the performance of inter-enterprise web-based transactions initiated by a client application from outside of the organization and also to monitor intra-enterprise business transaction initiated by an application within an organization.

A list of start-ups have also been investing in business transaction products to capitalize market share. Companies such as OpTier, Correlix, B-hive Network, and Correlsense are BTM solution providers. OpTier is the veteran among them and has a BTM solution called CoreFirst which provides a list of functionalities: detecting, tracking, and analyzing transactions across all tiers of the system architecture, and also creates a topology map of the business services at the transaction level, essentially linking the business context with underlying IT components (OpTier, 2010). Correlix and B-hive Network offer transaction functionalities at network level. Correlix’s Transaction Platform is specialized for discovering transactions and analyzing performance at different tiers and root cause analysis for transaction failures. B-Hive’s Conductor is connected with network devices (e.g., switches) and monitors the flows between end-users, web applications, and backend servers. Additionally, it gives real-time alerts of service availability problems. Conductor can also be used to optimize transaction performance such as transaction routing and prioritization. Another start-up Correlsense’s transaction management solution, SLAce, was designed to inspect network traffic to detect business transactions and monitor their performance across all tiers. It identifies deviation by creating a normal transaction behavior. In addition, it facilitates prioritizing transactions proactively by optimizing resource allocation according to policies.

The study concludes that the commercial software discussed in the above are not at all concerned with modeling business transactions. These are software products that deals with the application and system level operations. Designing business transactions is a higher level abstraction than these. No technique for defining the flexible behaviour of business transaction behaviour were found. Since the formal foundation of these technologies are not publicly available I cannot guarantee that these technologies do not provide any technique for defining flexible behaviour. However, from the information available, I conclude that they do not provide necessary and sufficient techniques.
2.4.11 **Summary of Business Transaction Technologies**

The result of the study is the same as the previous one. The technologies provide design-elements and techniques, however, none of these technologies is adequately rich to provide all necessary design-elements and techniques. Consequently, like the models, my assumption does not hold true. Considering RQ 2, the business transaction technologies provide design-elements of both business and functional perspectives. Some of these technologies in particular, BPMN, UMM, BPEL provide relatively more design-elements than any model discussed in the previous sections, however, unfortunately, none of these alone is able to provide all the elements required for modeling business transactions of end-to-end business processes.

Regarding techniques for defining flexible behaviour, the study reveals that most of the technologies provide the same of techniques provided by the models. *Wait* is the only new technique found in the study.

Several design-elements which were found in this study can be reused. The elements are presented in the Table 2.3.

<table>
<thead>
<tr>
<th>Table 2.3: List of Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Elements</strong></td>
</tr>
<tr>
<td>Business Perspective Design-Element</td>
</tr>
<tr>
<td>Business Collaboration</td>
</tr>
<tr>
<td>Business Area</td>
</tr>
<tr>
<td>Process Area</td>
</tr>
<tr>
<td>Operational Constraints</td>
</tr>
<tr>
<td>Business Perspective Design-Element</td>
</tr>
<tr>
<td>Process</td>
</tr>
<tr>
<td>Activity</td>
</tr>
<tr>
<td>Business Function</td>
</tr>
<tr>
<td>Business Object</td>
</tr>
<tr>
<td>BTP, WS-CAF, BPEL</td>
</tr>
<tr>
<td>Role</td>
</tr>
<tr>
<td>Fault Handler</td>
</tr>
<tr>
<td>Event Handler</td>
</tr>
</tbody>
</table>

The Table shows that BPEL and BPMN provide most of the functional-perspective design-elements. On the other hand, UMM provides more business perspective elements. Table 2.4 presents the reusable techniques found in the study.

Table 2.4 shows that these technologies provide only four techniques including compensation and forward recovery, contingency transaction, and wait. These basic techniques can deal with a very limited number of faults; however, faults such as messaging fault, customer credential fault, policy violation, security constraint violation *etc.* *etc.* cannot be dealt with these techniques.
Table 2.4: List of Techniques for Flexible Business Transactions

<table>
<thead>
<tr>
<th>Technique</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensation</td>
<td>BPEL, WS-TX, WS-BA, CheTah, BTP, UMM, BPMN</td>
</tr>
<tr>
<td>Forward Recovery</td>
<td>BPEL</td>
</tr>
<tr>
<td>Wait</td>
<td>Asset, BTP</td>
</tr>
<tr>
<td>Contingency Transaction</td>
<td>Asset, BPMN, BPEL, BTP</td>
</tr>
</tbody>
</table>

2.5 Summary

In this chapter, business transaction models and technologies were reviewed. Additionally, commercial business transaction software products were investigated. The results were summarised in section 2.3.12 and 2.4.11. The models and technologies provide various design-elements for modeling business transactions. Some elements are common to both models and technologies. Tables 2.5 combines the design-elements provided by both models and technologies.

Table 2.5: List of Design Elements

<table>
<thead>
<tr>
<th>Design Elements</th>
<th>Models and Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Perspective Design-Element</strong></td>
<td></td>
</tr>
<tr>
<td>Business Collaboration</td>
<td>UMM</td>
</tr>
<tr>
<td>Business Area</td>
<td>BAT, BPEL, BPMN</td>
</tr>
<tr>
<td>Process Area</td>
<td>BAT, BPEL, BPMN</td>
</tr>
<tr>
<td>Operational Constraints</td>
<td>BAT, BPEL, BPMN</td>
</tr>
<tr>
<td>Quality of Service</td>
<td>BAT, TxQoS</td>
</tr>
<tr>
<td>Strategic Constraints</td>
<td>BAT, BPEL, BPMN</td>
</tr>
<tr>
<td><strong>Business Perspective Design-Element</strong></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>WS-BA, BPEL, BPMN, WS-CAF, BAT</td>
</tr>
<tr>
<td>Activity</td>
<td>WS-BA, WS-CAF, JAS, WS-TX, BTP, BPMN, BPEL, BAT</td>
</tr>
<tr>
<td>Business Function</td>
<td>BTP, BPEL, BPMN, BAT</td>
</tr>
<tr>
<td>Business Object</td>
<td>BPMN, BPEL, UMM, BAT</td>
</tr>
<tr>
<td>Actor</td>
<td>BTP, WS-CAF, BPEL, BAT</td>
</tr>
<tr>
<td>Role</td>
<td>BPEL, BPMN, BAT</td>
</tr>
<tr>
<td>Fault Handler</td>
<td></td>
</tr>
<tr>
<td>Event Handler</td>
<td>BPEL</td>
</tr>
</tbody>
</table>

Combining the results, Table 2.5 shows a good number of design-elements were found in the models and technologies. However, using all these models and technologies together to design one business transaction will be unrealistic and impossible. The table reveals a few interesting results. For instance, the technologies provide more design-elements than the models. Especially, BPMN, BPEL, and UMM provide many
functional perspective design-elements. However, none of them alone is able to provide all the required design-elements to design business transactions by correlating the business perspective with the functional perspective. Some critical business perspective design-elements, in particular, agreement and policy were not found in any model or technology.

Since the models and technologies provide a number of design-elements covering both the business and functional perspectives of end-to-end business processes, I should adjust the problem defined by the S-Cube consortium. Based on the study I reformulate the research problem defined by the S-Cube consortium,

“The current business transaction technologies do not provide design-elements which can be used to design business transactions by correlating the business perspective with the functional perspective so that at runtime the transactions can be governed from the business perspective.”

to the following,

“The current business transaction models or technologies do not provide a complete set of design-elements which can be used to design business transactions by correlating the business perspective with the functional perspective.”

Similarly, the techniques provided by the models and technologies are combined in the Table 2.6.

Table 2.6: List of Techniques for Flexible Business Transactions

<table>
<thead>
<tr>
<th>Technique</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensation</td>
<td>BPEL, WS-TX, WS-BA, CheTah, BTP, UMM, BPMN, Open Nested, Sagas, S, TMLA TxQoS</td>
</tr>
<tr>
<td>Forward Recovery</td>
<td>BPEL, TxQoS</td>
</tr>
<tr>
<td>Wait</td>
<td>Asset, BTP</td>
</tr>
<tr>
<td>Retry</td>
<td>TMLA</td>
</tr>
<tr>
<td>Contingency Transaction</td>
<td>Asset, BPMN, BPEL, BTP, BAT, ConTract</td>
</tr>
</tbody>
</table>

To summarise, the most critical weakness of the current business transaction models and technologies, which is, they provide a very limited number of techniques for defining the flexible behavior of business transactions. Only five techniques (combining both models and technologies) were found in the study and none of them provide all five. There are various faults such as policy violation, uncertain events, policy violation, security policy violation etc.. that cannot be handled with these basic techniques because these faults are business-related faults which require techniques such as suspend, postpone, mutualization etc... Clearly, the models and technologies are not able to support defining highly flexible business transactions.
Chapter 3

Research Method

3.1 Introduction

A methodology is a set of guidelines for solving a problem with methods, techniques, and tools (Irny and Rose, 2005). Research methodology explains how the research is done, methods of data collection, materials used, subjects interviewed, or places visited (Jabar et al., 2009). It provides a methodical and a rigorous way to design a research roadmap. Thus, choosing an appropriate methodology for conducting research is critical to its success.

Choosing a method depends on several attributes such as research objective, research context, research area etc.. Such attributes are essentially the driving factors to select a research method. The research presented in this thesis falls within the computer science and information systems domain. The research is about the business transactions of end-to-end supply chain business processes, particularly the purchase order management processes. Business transactions automation involves several applications that are distributed to different locations. This research is concerned with the SBAs that rely on an SOA paradigm.

Designing/modeling business transactions of end-to-end business processes is complex and heavily challenging because the transactions are distributed and they cannot be atomic because atomic behaviour can be computationally expensive. The existing business transaction models were reviewed in Chapter 2. Several limitations of current solutions have been outlined. The research objective became clearer following the review:

"Develop a business transaction model that enables designing the verifiable flexible business transactions for long-running complex end-to-end business processes by correlating the real-world business entities."

It is worth mentioning that in computer science (CS) the notion of model is different from the conventional models that essentially can be described by texts or block diagrams. In CS, a model is described by formal language designed specifically to describe particular kind of models (Amaral). The model this research aims to develop is a formal model to define the logical construction of flexible business transactions by incorporating business perspective and functional perspective design-elements. Since correlating the business perspective with the functional perspective is a part of the research objective, the method for conducting this research should guide investigation of business scenarios for collecting design-elements.
This chapter discusses the research methods used in this research project. Also, the research project is designed in this chapter.

3.2 Research Method

Qualitative and quantitative are two widely used research methods. Quantitative methods quantify data and generalize results from a sample of the population of interest whereas qualitative methods provide insights into a problem, generating ideas and/or hypotheses (McDonald and Headlam). Information system researchers use both qualitative and quantitative research methods; in some cases, they prefer an eclectic approach by combining both. Quantitative research methods are very useful for investigating social phenomena using mathematical or statistical techniques. They rely on numerical analysis of collected data.

In contrast, qualitative research applies interpretative techniques, which are not mathematical. According to Krauss, qualitative research denies the existence of objective reality – this implies that subjectivism is the philosophical foundation of the qualitative research. Qualitative research is applied as a vehicle for studying the empirical world from the perspective of the research subject (Duffy, 1987). Benoliel (Benoliel, 1985) expanded this view by describing qualitative research methods as follows, 'modes of systematic enquiry with understanding human beings and the nature of their transactions with themselves and with their understandings'. Some prominent qualitative research methods are: Ethnography, Action Research, Grounded Theory, Case Study, and Survey.

Furthermore, design science is an emerging research method which the researchers employ in information system research. The design science paradigm seeks to extend the boundaries of human and organizational capabilities (Hevner et al., 2004). Simon (1996) conceptualized the design science method and noted that it is rooted in engineering and the sciences. According to him design-science supports positivity that calls for the creation of innovative artifacts to solve real-world problems. The following section gives a brief introduction of design science research method.

I choose the design science method for my research. Below I present the list of reasons for choosing this method:

- In my research, analysing business cases is important, however, building the logical construction with formal language is relatively more critical. To the best of my knowledge, only the design science method can guide research using business case and mathematical formalism. Essentially, this has been the main reason for choosing the design science research method for my study.

- This research is about the integration of business and functional perspectives of business processes. The design science method was developed for guiding such researches.
3.2. RESEARCH METHOD

- Design science was specifically developed for information system research. Evidently, it suits this research more than other methods, whether qualitative or quantitative.

3.2.1 Design-science Research Method

In this section, I summarise the design-science research method. Design denotes to invent and bring into being.¹ According to Takeda et al. (1990), in a perspective analogous to considering design of artifacts as the crafting of an interface between inner and outer environment, design can be thought of as a mapping from a functional requirement constituting a point of multidimensional space – to attribute space, where an artifact satisfying the mapping constitutes a point in that space. Simon defined the inner environment is the set of components that create the artifact whereas the outer environment is the total set of external forces and events that act on the artifact. Design-science (in the literature the term was spelled “design-science” and “design science”, however, semantically these two are exactly the same) is knowledge in the form of constructs, techniques, and methods for performing the mapping, models, theory – the know-how for creating artifacts that satisfy given sets of functional requirements (Vaishnavi and Kuechler, 2013). It is a problem-solving paradigm. Simon also called this research method science of the artificial and defined it as ‘a body of knowledge about the design of the artificial objects and phenomena –artifacts– designed to meet certain desired goal.’

According to Hevner et al. (2004), design-science has a strong relevance to information systems research because it directly addresses two of the key issues of this discipline:

- the central, albeit controversial, role of the IT artifact, information system research (Weber, 1997), (Orlikowski and Iacono, 2001), (enbasat and Zmud, 2003) and

- the perceived professional relevance of information system research.

The authors explain more of the relevance – design-science creates artifacts that are essentially products through which the analysis, design, implementation, and use of information system can be accomplished effectively and efficiently. The research method is used for developing and evaluating artifacts such as method, theory, construct, model, instruments, framework, and instantiation.

The design-science research process model consists of five phases (Peffers, 2008): Identify Problem and Motivation, Define Objectives of Solution, Design and Development, Demonstration and Evaluation, and Conclusion. The research process starts with defining the problem followed by defining the objectives of the solution. Then, the artifact (research product) is designed and developed. In the next phase, the artifact is used in a suitable context to demonstrate and evaluate its effectiveness and efficiency. Finally, the conclusion is drawn. I designed this research adapting these phases.

¹Webster’s Disctionary and Thesaurus
Additionally, the thesis followed the guidelines suggested by Henver et al., (2004). They developed a conceptual framework of the design-science research method. The framework was developed to conduct research on information systems focusing on the integration between business and information technology (For more detail of the framework, see Hevner et al., (2004)). Figure 3.1 shows the design-science research framework for information system research.

They proposed seven guidelines to enable researchers to conduct research in a structured way to produce effective and efficient results. I summarise the guidelines below in the context of my research:

- **Guideline 1 (Design as an Artifact)**: Design-science research must produce a viable result in the form of construct, model, method, instantiation, and theories. This research will produce a business transaction model for designing business transactions of large-scale business processes.

- **Guideline 2 (Problem Relevance)**: The solution within the scope of design-science research should be technology-based, should have relevance for business, and should be significant to business. Solúbtha BTM partly uses the Model-Driven Engineering (MDE) (Schmidt, C.D., 2006) paradigm. It is a technical solution with a formal foundation. The design of the instance-model of Solúbtha
BTM can be done using the MDA technologies such as BPMN at the business domain layer. The simulation of the instance-models can be implemented using programming languages. In short, as a solution, Solúbtha BTM is purely technology-based. As mentioned earlier, the model covers the business aspects since it correlates the entities that belong to the business domain. Clearly, Solúbtha BTM has strong relevance for the business domain and it enables the integration of the business perspective of an end-to-end business process with its functional perspective.

- **Guideline 3 (Research Contribution):** Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundation, and/or design methodologies. The research presented in this thesis produces Solúbtha BTM. The elements of this model are defined using formal languages and thus the model enables users to verify the instance-models.

- **Guideline 4 (Design Evaluation):** The utility, quality, and efficacy of design artifacts must be rigorously demonstrated via well-executed evaluation method. Solúbtha BTM is evaluated using model checker that takes formal specifications of instance-models as input and simulate the behaviour of business transactions. In the evaluation process the instance-model is designed from a real-life business scenario.

- **Guideline 5 (Research Rigor):** Design-science research relies on the application of rigorous methods in both the construction and evaluation of the research artifact (Hevner et al., 2004). The structure of Solúbtha BTM is formalized using Set Theory – a mathematical formalism to define the syntax of real-world objects. Additionally, formal languages are used to define the semantics of the design elements and behavioral semantics. In the evaluation phase these formal definitions are used to validate the correctness, effectiveness, and efficiency of the model.

- **Guideline 6 (Communication of Research):** The design-science research should be presented effectively to both technology-oriented and management-oriented audiences. Since this research covers the business perspective and has a solid technological foundation, this thesis is presented to both technology and management readers.

An important aspect of the design-science research methodology is it guides the formal foundation of contributions. This not only makes the contribution verifiable but also increase the soundness. A brief description of formal methods is presented in the following.

### 3.2.1.1 Formal Method

Design science research requires the application of rigorous methods in both the construction and evaluation of the design artifact (Hevner et al., 2004). Design science relies heavily on mathematical formalisms (or

---

2 Papazoglou (2008) describes the SOA layers where the business domain layer is at the top.
formal methods) for research rigor. The use of formal methods in the context of computer science research is inevitable. It is used for various purposes such as designing the logical structure of artifacts and providing the precise meaning of the entities used in the constructed artifact. Additionally, formal methods are used to develop the automated artifacts.

Formalization is vital for this research because without a formal foundation the instances of Solúbtha BTM developed in this thesis cannot be verified at design-time and also simulation of instance-models is not possible. The model checkers that simulate a business transaction system (can also be called state machines) parses the high-level syntax into formal logic and compiles the system definition and properties based on formal semantics. Therefore, an instance-model without basis of formal techniques cannot be compiled. In this research, the formal method is used to define the syntax and semantics of the constructs of Solúbtha business transaction model. Additionally, the formal semantics of behavioral properties of the flexible business transactions are defined using formal language.

3.3 Research design

I choose the design-science method designing this research. In section 3.2.1, I explained the reasons for choosing the method. This section explains how I conducted my research. I describe all the methods I used for conducting the research.

A research process is the logical order of different phases, which rely on a method. The research process of this thesis is adopted from the design-science method. The process consists of four phases: problem definition, literature review, development of Solúbtha BTM, and Validation and Evaluation. The research design presented in Figure 3.2 depicts these phases.

The research problem was identified in the first phase. In figure 3.2, the literature review phase is placed between the development and problem definition phases because the literature were reviewed to define the research problem, as well as to develop the Initial Version of the model. The next subsection explains more of this sequence. Three different versions of the business transaction were constructed in the development of Solúbtha BTM phase. In the final phase, the model was validated and evaluated.

3.3.1 Problem Definition

A problem is the heart of any research. Therefore, it needs to be defined explicitly and precisely. In this section, I give details of how the problems of this research were identified and formulated.

This research addresses two problems: establish an explicit correlation between the business and functional perspective elements of end-to-end business processes, particularly the purchase order management processes, and enable users to define the properties of flexible business transactions. The former was
3.3. RESEARCH DESIGN

defined by the S-Cube consortium. The problem was reported in the S-Cube project proposal. The need of a correlation between the business and functional perspective elements can be found in the first version of the description of work (DoW). Essentially, the consortium first identified the need of such correlation in business process management and then they conducted an initial study of literature related to business transactions. The study revealed the shortcoming of the state of the art technologies. Then the problem was formulated based on the study. The problem is stated below:

*The current business transaction technologies do not provide design-elements which can be used to design business transactions by correlating the business perspective with the functional perspective so that at runtime the transactions can be governed from the business perspective.*

The objective of my research was to address this problem. However, before going into the development phase, I re-reviewed the literature related to business transaction models and technologies to clarify and ratify the research problem. Notably, in my study I include business transaction models which are important but were overlooked in the previous study conducted by the consortium. From my study, I found that there are models (e.g., (Papazoglou and Kratz, 2006)) which provide design-elements for designing business transactions, however, none of them provides a complete set of elements. This means, several models
are needed for designing a business transaction which is practically impossible for many reasons such as complexity, disjoint features, etc.. Based on my study I reformulated the research problem as follows,

“The current business transaction models or technologies do not provide a complete set of design-elements which can be used to design business transactions by correlating the business perspective with the functional perspective.”

After an interim evaluation of the Initial Version of Solúbtha BTM, I investigate secondary data of a business case. The primary study of the business case was conducted by the S-Cube consortium. I studied the technical report produced by the consortium. During the investigation – which is described later in this section – I found that business transactions are long-running, distributive, and involve many parties who perform local operations autonomously. Therefore, business transactions be fail or be aborted due to different faults such as communication faults. Essentially, I discovered several faults in that study. In order to prevent failure or abortion the runtime behaviour of business transactions must be flexible. Business transaction models or technologies that rely on classical atomic principal (in Chapter 2, I described the limitation of atomicity principle in the context of business transactions) cannot be used to define the behavioural properties of flexible business transactions. Then I revisited the state of the art. Very few models and technologies were found which provide elements such as operators to define the flexible behaviors of business transactions. In other words, when I found the same result produced in my first visit to the literature which is, some business transaction models and technologies provide a limited number of techniques to define flexible operations, however, they are not sufficient for complex and distributed business transactions. This gave rise to the second part of the research problem:

The current business transaction models and technologies do not provide adequate techniques to define highly flexible business transactions of end-to-end business processes.

I combined these two problem statements into a single research problem. It is defined as follows:

There are no business transaction models nor technology that provide design-elements or techniques to design highly flexible business transactions by correlating explicitly the business and functional perspectives of end-to-end purchase order management processes.

In addition to these problems, there is the design-time verification problem. The business transaction models and technologies do not provide support design-time verification of the correctness of business transactions because they lack formal underpinnings. This is a secondary problem and the solution of the problem is produced in a natural manner in my research because I provide the formal foundation of my model which addresses the problem.
### 3.3. RESEARCH DESIGN

#### 3.3.2 Literature Review

The literature review should describe, summarise, evaluate, and clarify the literature and should give a theoretical basis for the research and help researchers to determine the nature of their own research (Boote and Beile, 2005). According to Okoli and Schabram (2010), in a doctoral dissertation the literature review serves as a justification for the novelty of the student’s work. The authors present two purposes of the literature review: *a theoretical background for subsequent research and learning the breadth of research on a topic of interest; or answering practical questions by understanding what existing research has to say on the matter.* However, Hart (1999) presents four purposes for the literature review in a thesis: it synthesizes the understanding a student has of their particular subject matter; it stands as a testament to the student’s rigorous research dedication, it justifies future research (including the thesis itself), and it welcomes the student into scholarly tradition and etiquette. Considering rigorousness, Fink’s (2005) definition highlights the essence of a rigorous literature review and includes: *systematic, explicit, comprehensive,* and *reproducible.*

She essentially focuses on systematic literature review (SLR) (Kitchenham and Charters, 2007). According to Levy and Ellis (2006) and Webster and Watson (2002), information systems scholars tend to be unaware of the need for structure in reviews.

I did not apply the SLR tool in this thesis however, I conducted a very focused review. The scientific or information system topics are *focused* on a particular, theory, model, etc., of a specific topic. Therefore, it is easier to conduct a focused literature review which is not rigorous as SLR, even so the rigorousness is maintained if the focused study covers a large-scale knowledgebase.

As mentioned in the previous section, the initial literature reviewed was conducted by the S-Cube consortium. This gave me an opportunity to be specific in terms of my need or what exactly I had to look for. I reviewed various business transaction model and technology related literature. The study was conducted to answer the following questions:

- **RQ 1**: Does any existing business transaction model or technology have the business perspective design-elements that can be used to define real-world business entities in a business transaction instance-model? If yes then
  - *is the model or technology adequately rich to enable users to design business transactions of an end-to-end business process?* (the required business perspective design-elements of a purchase order fulfillment scenario is reported in the technical report JRA-2.1.3 (Haque et al., 2009))

- **RQ 2**: Does any existing business transaction model or technology provide techniques to define the flexible business transactions? If yes then
  - *is the model or technology adequately rich to enable users to define properties which can prevent faults at runtime and can guarantee flexible business transactions?*
The review process consisted of five main steps: search, selection, store, extraction, and write. In order to carry out each step I found Fink’s attributes, explicit and comprehensive, very effective. She defines ‘explicit’ as a clearly expressed procedure by which the study was conducted and ‘comprehensive’ refers to all relevant materials being included within the scope of the review. I explain these in the context of my literature review:

3.3.2.1 Search

The search strategy was mixed combining automated and manual. In order to query the databases I specified several keywords ‘business transaction’, ‘model’, ‘technologies’, ‘business perspective’, ‘functional perspective’, ‘design-element’, ‘flexible behaviour’, ‘flexible technique’, ‘fault’ etc... I combined these terms using the Boolean conjunction (‘AND’) operator. Below are the examples of some compound queries I conducted in this research:

- (business Transaction)*
- ((business Transaction) AND Model)*
- ((business Transaction AND Technologies)*
- ((business Transaction) AND (Business Perspective) AND Design Element)*
- ((business Transaction) AND (flexible behaviour))*
- etc.

I queried open access databases ‘GoogleScholar’ and specific subject databases, and bibliography ‘DBLP’. I queried following specific subject databases:

- Elsevier
- IEEE Digital Library
- ACM Digital Library
- Citeseer
- Springer

---

5 GoogleScholar: http://scholar.google.fr/
6 DBLP: http://dblp.uni-trier.de/db/
7 Elsevier: http://www.elsevier.com/
8 IEEEDigitalLibrary: http://ieeexplore.ieee.org/Xplore/home.jsp
9 ACMDigitalLibrary: http://dl.acm.org/
10 Citeseer: http://citeseerx.ist.psu.edu/index
11 Springer: http://link.springer.com/
3.3. RESEARCH DESIGN

Additionally, the S-Cube deliverable and publication databases were searched. This research mainly used two search engines ‘Google’ and CiteseerX. Additionally, I used the Tilburg University library search engine and University of Limerick (UL) library search engine.

For manual searches the priority was those conferences and journals that include the topics business transaction, business process management, service composition and choreography, system modeling, operational flexibility, behavioural modeling, modeling flexible systems, service based, etc. I identified several conferences including ICSOC, ICWS, EC-WEB, etc. and journals including SOCA, IEEE Data Engineering, etc.

The queries returned different results. For instance, the first query returned papers related to business transactions in different contexts such as business process, choreography etc. It returned business transaction management related papers as well. The second query returned business transaction model related papers for example, A Conceptual Model of Business Transaction Management (Hofman, 1994) which was not found in the list of results produced by the first query. It is worth noting that different queries sometimes returned common results too. For instance, each of the first and second queries returned a paper relating to business-aware transaction model (Papazoloug and Kratz, 2006). These common results are expected because of the common tokens between the queries. Business and transaction are the two common tokens in all queries. Thus, theoretically, it is possible that the search engine produces one or more common results.

3.3.2.2 Selection

Typically, search engines produce – except the library engines – an exhaustive number of results. The selection strategy was straightforward. The articles were filtered based on their full or partial relevance to the research question. If the contribution of the articles was a model or technology for designing business transactions by correlating business aspects or defining the flexible behaviour of transactions then it was considered a fully relevant article. The partially relevant articles are those that discuss the research themes including end-to-end business processes, management of business processes, transaction faults, operational fault, however, do not contribute towards modeling techniques or modeling elements. It is worth noting that both fully or partially relevant articles are important for the research. Any article that did not have partial or full relevance was excluded.


CiteseerX: [http://citeseerx.ist.psu.edu/index](http://citeseerx.ist.psu.edu/index)


http://www2.ul.ie/web/WWW/Services/Library


ICWS: [conferences.computer.org/icws](http://conferences.computer.org/icws)


IEEE data Engineering Bulletin: [http://www.informatik.uni-trier.de/~ley/db/journals/debu/index.html](http://www.informatik.uni-trier.de/~ley/db/journals/debu/index.html)
3.3.2.3 Store

I created a database of literature for this thesis using Mendeley\textsuperscript{22}. It is a bibliographic tool that has desktop and web interfaces for creating database and generating bibliography files. I added the articles to the database that is partitioned into: business transaction models, business transaction technologies, and others.

3.3.2.4 Extraction

This is the crucial step where data (design-elements, techniques for flexible business transactions, and faults) were elicited. The data elicitation strategy was as follows. While I reviewed an article, the first step was identifying the contribution. If the contribution was a model or technology I analysed it to find an answer of the first part of the ‘RQ 1’. Then, I extracted the functional perspective design-elements which related to the operational aspects of business transactions. For instance, activities are functional perspective design-elements because they are abstractions of business operations. Essentially, identifying this type of elements was easier. The semantics of the design-elements that are not related to business operations were analysed. If it was found to be business specific then it was listed as a business perspective design-element.

I used an online business dictionary\textsuperscript{23} to check the meanings.

The next step was to find whether the model is complete which essentially answers the second part of ‘RQ 1’. In order to find the completeness of a model or a technology, I compared the list of design-elements (that were found through the study) with the the ones that were reported in Haque et al., (2009). It is worth noting that this step was carried out if and only if the business transaction models or technologies contained business perspective design-elements. Moreover, the design-elements were compared with the definition of a business transaction (my definition in Section 2.2) developed in this thesis. Because, the definition implicitly points to the business perspective elements that must be present in the model to design business transactions of end-to-end business processes.

Additionally, literature were investigated to find entities that are directly or indirectly related to business transactions. If any such entity was found then it was analyzed to identify its type (business or functional) using the same strategy.

As mentioned earlier the related literature were reviewed iteratively. I revisited literature after analysing data of the first business case. The main purpose of the revisit was to answer the ‘RQ 2’. This step was to find the techniques provided by the business transaction models and technologies. This was a straightforward investigation. If a transaction model or technology provides any technique for operational flexibility or to deal with a fault to prevent failure or abortion, it was reported in the literature with a comprehensive explanation. Therefore, no further analysis was required to find these techniques. To answer the second part of ‘RQ 2’, I investigated whether the techniques provided in a business transaction model or technology

\textsuperscript{22}Mendeley: http://www.mendeley.com
\textsuperscript{23}online business dictionary http://www.businessdictionary.com/
can deal with the faults that I found in the literature and secondary data analysis of business cases (Chapter 5 describes the faults). For my analysis, I constructed a logical statement – using IF..ELSE keywords – that comprise a fault and a technique. With this I analyzed whether the technique can repair the fault. This approach was used for every fault I found in this thesis. Lastly, the reusable design-elements and techniques are listed.

3.3.2.5 Writing

In the final step the findings were reported clearly and comprehensively. The review of a business transaction model or technology was written in three parts: a brief overview, the strengths, and the shortcoming.

3.3.3 Development of Solúbtha BTM

Solúbtha BTM was developed in this phase. Three versions of the model were produced: the Initial Version, Solúbtha BTM V1, and Solúbtha BTM. In the following, I describe the construction process of these versions and I especially discuss the methods I used for constructing them.

3.3.3.1 Initial Version of Solúbtha BTM

The Initial Version of Solúbtha BTM was constructed based on the data that were found through the review of the literature. In this version, I concentrated on constructing the model to enable users to designing business transactions by correlating business and functional perspective of end-to-end business processes. In addition to the modeling elements I found in literature, I proposed several design-elements in this version.

In the previous subsection, I described how the design-elements were extracted while reviewing literature. I constructed the model by composing them logically. The Unified Modeling Language 2.0 (UML 2.0) (OMG, 2005) was used for constructing Solúbtha model. Since the design-elements were categorised into business and functional perspectives, I introduced two packages in the model. Then the design-elements were added to these packages. The logical composition of a formal model relies on the relation between or among its elements. This is an important step because the syntactic correctness of formal models is determined by the correct definition of the relationships between the elements. During the investigation, in addition to design-elements, I identified the relations between them. Sometimes the relations were described in formal language, however, in most cases the natural expressions were not used. In this case, I interpreted them in formal language. For instance, the formal interpretation of the term ‘contain’ is ‘compose’.

3.3.3.2 Interim Evaluation

The S-Cube is a large consortium comprising 16 partners and 17 associate members from academia and industry (In Chapter 1 I described the project). The experts were from various domains such as service en-
gineering, business process, formal method etc.. I found it an appropriate platform to conduct an evaluation of the Initial Version of the model. I used an expert opinion technique. It is an informal technique that can be used to serve a variety of purposes, and may be used to assist in problem identification, clarifying the issues relevant to a particular topic, and the evaluation of the product (Expert Opinion, Para 1). Below, I describe the steps that I followed to conduct the evaluation.

- **Plan:** In this step I planned how to select experts, the number of experts, outlining issues for discussion, how to record the suggestions and opinions.

- **Expert Selection:** An expert, more generally, is a person with extensive knowledge or ability based on research, experience, or occupation and in a particular area of study (Wikipedia). The research presented in this thesis covers four main research areas, business process management, complex business processes, service engineering (service composition and choreography), and computational theory (logic and data structure). Therefore in order to evaluate the model, an expert needs to have expertise in these areas.

S-Cube was a network of excellence project that involved famous researcher in the area of theoretical computer science, business process management, requirements engineering, software architecture, service engineering etc.. Therefore, it was not difficult to find experts within the consortium.

I scanned the description of work to find S-Cube research fragments and responsible organizations. A list of researchers was found working in relevant domains. I searched their personal homepages to find details of their expertise. Additionally, I scanned their publications. I listed five potential participants and sent invitations. Three out of five became interested in participating in the evaluation process. They were from three different institutes. These participants are from computer science, business process, and service network domains. Table 3.1 shows the detail of the participants. The left hands side column shows the participant, and the right hand side column shows their expertise.

<table>
<thead>
<tr>
<th>Participants</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant 1</td>
<td>Service Network, business process</td>
</tr>
<tr>
<td>Participant 2</td>
<td>Service Composition, business process, Service based application</td>
</tr>
<tr>
<td>Participant 3</td>
<td>Service Composition, Service based Application, Process designing, Formal Methods, and Business Process</td>
</tr>
</tbody>
</table>

- **Outlining Issues:** I identified five critical issues: complexity or simplicity, complete or incomplete, over modeled or under modeled, degree of correlation, and correctness of relationships. The Initial Version of Solúbtha was evaluated over these. They are briefly explained in the following:
3.3. RESEARCH DESIGN

- Complete or Incomplete: Solúbtha BTM was a formal model for designing business transaction models of end-to-end business processes. Therefore, the model must be complete in terms of design-elements for modeling business transactions. An incomplete model is of no use to the users.

- Over Modeled: Over-modeled refers to a model that contains unnecessary elements. A model must not be over modeled because it will increase unwanted complexity.

- Complex or Simplex: The simple technologies are easy to use. Therefore, simplicity is an essential quality for Solúbtha BTM to enable both expert and non-expert users to design business transactions. The design simplicity was measured by analyzing the design-elements and the logical relationships between the elements.

- Degree of Correlation: Essentially, this issue address a research question of this thesis. It refers to measuring the correlation between business and functional perspective elements. The degree of relationship was not quantified, rather it was measured by a fuzzy criteria which is the degree to which the logical connection between business and functional perspective elements were established. It was measured by three units high, medium, and low.

- Syntactic Correctness: This refers to the correctness of the relations between the design-elements of the model. A relation is a function that logically connect two or more elements so that the instances at runtime can be executed coherently.

### Recording

- I used notepad24 software to record the opinions and suggestions of the participants. During the conversations I took notes only of the core of the experts’ opinions, and in fact, the opinions were short, simple, and instructive for the improvement of the model, which was essentially what I required. No long sentence was recorded.

I spoke with each participant separately. The model was sent to the participants in advance. During the sessions with participants, they asked questions such as why did you add this element? why is the relationship composition instead of aggregation? etc.. Table 3.2 shows the time that was spent with each participant. The left hand side shows the participant and right column shows the time spent with participant.

The outcomes of the interim evaluations are presented and discussed in Appendix B.

### Secondary Data Analysis of Business Cases

I analyzed secondary data from three business cases: Auto Inc Automotive Supply Chain (Nguyen, 2009), AVERS OEM Supply Chain (Papazoglou, 2012), and Walmart Changes Tactics to Meet International Tastes
Table 3.2: Participants and Duration of Discussion

<table>
<thead>
<tr>
<th>Participants</th>
<th>Duration (Minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant 1</td>
<td>45</td>
</tr>
<tr>
<td>Participant 2</td>
<td>45</td>
</tr>
<tr>
<td>Participant 3</td>
<td>70</td>
</tr>
</tbody>
</table>

(Simchi-Levy, 2008). According to Libby (2007), in the secondary analysis of good documentations cannot be underestimated as it provides necessary background and much needed context both of which make re-use a more worthwhile and systematic endeavour.

The initial version of the model was constructed based on the outcomes of the literature reviewed. However, there was no evidence whether the reviewed models or technologies were constructed from business scenarios. This gave rise to a research question given below:

What are the business entities in a real-world business scenario that influence the business transactions of the business processes of that scenario?

The key reason behind analysing the secondary data of the business cases was to address this research question. Furthermore, on the completeness issue, two of the experts opined that "the completeness of the functional perspective of the model can be evaluated since the operational aspect of end-to-end business processes are a well-defined area. However, it is difficult to evaluate the completeness of the business perspective of the model". They suggested a business case study could be a suitable option. Also, they opined that the study may prompt changes to the functional aspect.

The research presented in this thesis produces a business transaction model that enables the design of business transactions of end-to-end purchase order management business processes. There are different business scenarios e.g., online airlines reservation and online shopping. This thesis was seeking of a scenario within the business-to-business (B2B) domain because the processes in this are highly complex, long-running, and purely distributive (Papazoglou, 2008). Additionally, the volume of business-to-business transaction is very high (Sandhusen, 2008). I choose the purchase order management scenario because it involves tens of processes. SCC (Supply Chain Council, 2000) categorises the processes into five: plan, source, make, deliver, and return. SCC defined a framework called Supply Chain Operation Council (SCOR) (Supply Chain Council, 2000) that guides the construction of these processes. The scenarios investigated in this thesis do not include return processes within the scope of the study. Additionally, the planning processes were not integrated in purchase order management process automation.
3.3. RESEARCH DESIGN

Selection of Cases

Since I studied secondary data, the business cases were selected carefully based on the following criteria:

- **Relevance**: This is an important criterion for selecting the business cases. In this thesis I choose purchase order management business processes because these processes are long-running and encompass *plan, source, make*, and *deliver* processes defined by the SCC (Supply Chain Council, 2000). For the analysis, in this thesis I selected only purchase order management business cases. The ‘Walmart Changes Tactics to Meet International Tastes’ focuses on purchase order delivery, which is a part of purchase order management.

- **Complex and long running**: The goal was to study complex and long-running business processes because business transactions are long-running and complex. The complex business scenario encompasses tens of hundreds of business activities and also involves different business partners from different regions or countries. The selected business cases are complex. The business processes found in the published documents contain tens of process activities. Additionally, the business scenarios involve partners from different countries. For instance, the Auto Inc. Automotive Supply Chain (Nguyen, 2009) involves partners from south-east Asia and Europe. This makes the transactions long-running and complex.

- **Well-Documented**: This is another important criterion for selecting business cases. Since it is secondary data analysis, a comprehensive description of end-to-end business processes is of critical importance. The document should explain the workflow and the business activities. The document should contain a graphical model of the process workflows. Additionally, the document should detail the business partners involved in the business processes. I selected business cases that were well-written.

Secondary Data Analysis of Case 1 (Nguyen, 2009)

The Auto Inc. Automotive Supply Chain (Nguyen, 2009) was prepared within the scope of the S-Cube project. The case was chosen for the study because it fits the business scenario which the research was looking for. Since this research studied secondary data, the quality of documentation was an important concern. The Auto Inc. case was reviewed and accepted internally by the S-Cube partners and externally by the European Union (EU) review committee, thus justifying the quality of the report. Additionally, validation of previous research (Andrikopoulos, 2010) on the Auto Inc. business scenario was a success.

The data source I accessed during the study is the technical report (Nguyen, 2009). While analysing data, the focuses were to find entities that influence business operations and the elements used for conducting business operations. The data were analysed in three steps: *study documentation, extract data, and data*
**CHAPTER 3. RESEARCH METHOD**

**categorization.** The steps are described below:

- **Study Documentation:** In this step Auto Inc.’s organizational structure, its business scenario, and business processes were studied. This study was more structured than reviewing literature because the business scenario was clearly specified and business processes were presented graphically. It was easier to identify business and functional perspective elements. It is worth noting that the return processes were outside the scope of the case and no explicit connection between the planning process and business transactions was found. Therefore, these were not studied. Furthermore, this study concentrated more on business processes because the operational details of the business scenario were presented graphically and comprehensively. Therefore, it was easy to deduce the logical execution of process activities that are parts of business transactions.

- **Data Extraction:** The data were extracted using the same approach used for extracting data from the literature. However, in this study, the data were extracted *if and only if* they were fully relevant to business transactions. The relevance were determined by the explicit explanation of data and its relation with business processes. For instance, while analysing business processes, several annotations were found. These were coded using the traditional conditional operators `IF...ELSE`. Essentially these annotations were of different aspects related to the business process such as delivery location, time, and process performance. These are the constraints for business processes. It was clear that the constraints influence the business operations and hence the business transactions.

Furthermore, different types of activities were observed in the business processes and also the relational schema was presented graphically on the process space. This means the representation of business processes was a clear demonstration of design-elements and their relationships. Therefore, data extraction in this study was much easier than it was in the literature.

Business transaction faults at this stage was a new research issue for this thesis because it was not planned before the study. During the study, I discovered various potential faults (related to business transactions) in the business process. For instance, *customer credential faults* i.e., if a customer credential verification fails then the purchase order is canceled. This cancellation results in abortion of business transactions. I found several such faults that lead to the business transaction abortion whereas if they could be prevented, it would significantly reduce the *computational cost* of business transaction systems. I found it motivating and it provided a clear scope of improving the state of the art. This study gave rise to the research issue *flexible behaviour of business transactions* or *flexible business transactions* which is another dimension of this thesis. In order to address this research issue, my new research task was to collect faults which essentially started from this study. Faults collection was straightforward. Faults were reported comprehensively either in natural language or graphically in the process.
3.3. RESEARCH DESIGN

• **Data Categorization**: The extracted data were categorised into business and functional perspectives. The data categorization method was the same as the one used in literature review.

**Secondary Data Analysis of Case 2 (Papazoglou, 2012)**

The S-Cube consortium extended the Auto Inc. Automotive Supply Chain business scenario. It is called ‘AVERS OEM Supply Chain’ (Papazoglou, 2012). The reason I studied this business scenario was to collect more data to ensure that the model covers all required design-elements for modeling business transactions of a purchase order management scenario. Additionally, since from the previous study, business transaction behaviour became an important research issue, collecting faults (that have impact on business transactions) was another reason for the second study. Similar to the ‘Case Study 1’, I analyzed secondary data of this case.

The reason for choosing this case is two-fold: it was an extension of the previous case with different business functions and it was a well-documented. The latter was more important because I had no access to other sources. The data collection and analysis procedure was the same as the previous study.

**Secondary Data Analysis of Case 3 (Simchi-Levy, 2008)**

This was a focused study to collect specifically data on business transaction faults related to delivery processes. Since Walmart is the largest retailer in the world and has hundreds of thousands of retail outlet, its delivery chain is global and more complex. The Walmart Changes Tactics to Meet International Tastes (Simchi-Levy, 2008) was chosen mainly to study the faults that occur in the cross-border delivery of purchase orders. Like the second study, the data collection and the analysis procedure in this study was same as ‘Case Study 1’.

3.3.3.4 **Study Trading Partner Agreements**

While studying secondary data of business cases, I found that ‘agreement’ is the central business perspective element and it heavily influences the business activities. Especially, it influences the final outcomes such as abortion of business transactions. For defining new techniques to design flexible behaviour of business transactions, collecting data on faults related to partner agreements was of critical importance to this research. Therefore, I had decided to investigate a few business partner agreements. The data collection and analysis procedure consists of three steps: Search, Selection, Data Extraction. Below, these steps are explained.

• **Search**: The web-based databases were the main source of finding sample agreements. I used ‘Google’, and ‘GoogleScholar’ search engines to conduct the search. The search strings ‘trading
partner agreement’, ‘agreement’, ‘trading partner agreement of automotive industry’, and ‘automotive business partner agreement’ were used in the search. The Boolean query inserted into the data engine as as follows:

\[-((\text{trading partner agreement})* \text{ OR } (\text{agreement})* \text{ OR } (\text{trading partner agreement of automotive industry})* \text{ OR } (\text{automotive business partner agreement})*)]

• **Selection:** The search produced an exhaustive number of agreements. Two criteria for including them in study are as follows:

  - Since it is a secondary study, the agreement must be well-written.
  - Since the cases studied are of automotive business, the agreement should be for the same business.

• **Data Extraction:** In this study, data extraction was much simpler than was expected because the conditional clauses contained in the trading partner agreements were similar to condition statements. Agreements were studied by tokenizing the clauses. A clause was broken into tokens then each token was investigated. The most important tokens are the modal verbs such as *shall*, *will*, *should*. The semantics of the modals including *shall*, *will*, *should*, and *must* were interpreted as obligation. The negation of these modals *should not*, *must not*, *shall not* and *will not* were interpreted as prohibition. These verbs were used to determine whether violation of a clause is a fault. Additionally, these were used to discover whether a violation of a clause is tolerable or if it must be repaired. Furthermore, the modal verbs *may* and *might* were interpreted as possibility and *permission* based on the overall meaning of a clause. It is worth mentioning that I interpret the modal verbs from the linguistic perspective and also from the operational standpoint. Both are necessary for a rigorous and meaningful interpretation.

Furthermore, the conditional tokens ‘IF’, ‘ELSE’, and ‘THEN’ were useful in identifying faults. Also the tokens, shipment, delivery, and purchase order were used to discover a fault of the specific activity mentioned in that clause.

3.3.3.5 **Construction of Models**

In this phase, two versions of the model were developed: *Solúbtha BTM V1* and *Solúbtha BTM*.

**Solúbtha BTM V1**

The Initial Version of the model was modified based on suggestions and opinions from experts and also on the data collected through studying the Auto Inc. business case. The newly found design-elements were
3.3. RESEARCH DESIGN

added. Also, some were deleted, merged, renamed, and joined. The modification resulted in Solúbtha BTM V1. Additionally, Solúbtha BTM V1 integrated techniques and methods for flexible business transactions. To this end, the state of the art was revisited to extract data on fault prevention.

**Solúbtha BTM**

Solúbtha BTM is the final version which was constructed by incorporating outcomes that were discovered by studying the secondary data of business cases 2 and 3 and the trading partner agreements. Solúbtha BTM V1 was modified to produce the final version ‘Solúbtha BTM’.

The researcher formalized Solúbtha BTM using formal methods. ‘Set Theory’ was used to define the logical structure of the model. The formal semantics of the modeling elements were provided using Time Extended Finite State Machine (TEFSM). In addition, the flexible behavioral properties of business transactions were described using the Computational Tree Logic (CTL). Formal representation of the business transaction model, the semantics of modeling elements, and their behavioral properties ensure the rigorousness of Solúbtha BTM.

### 3.3.4 Validation and Evaluation

The final step of the research process is validation and evaluation. The design-artifact was validated by a demonstration which is evidence that the novel design artifact (models, methods etc..) works (Walls et al., 1992). Peffer et al., (2007), Eekels et al., (1991), and Nunamaker et al., (1991) prescribed demonstrating the use of the artifact to solve at least one or more instance of the problem. According to those authors, demonstration could involve the use of design artifacts in experimentation, simulation, case study, proof or other appropriate activity. In this thesis, Solúbtha BTM was used in simulation and also was used in formal method. These were chosen because the approaches are formal and thus lead to a rigorous validation.

Solúbtha BTM was validated over four criteria: Correctness, Flexibility, Applicability, and Ability to Correlate Business and Functional Perspective. They are explained below:

- **Correctness** was chosen to prove that the model is correct from the syntactic and semantic point of view. Solúbtha BTM is a model that is used to design instance-models (the concrete models used in actual transaction processes) of particular cases. Since it is a logical model, its correctness is of utmost importance to preserve the correctness of its instance-models. Incorrect models promote inaccurate modeling and irrational behaviour of transactions at runtime such as inaccurate transaction flow and, operational failure etc.. etc..

- **Flexibility** was chosen to prove that the techniques provided by Solúbtha BTM can define a highly flexible business transactions and also to show that these techniques are effective and guarantee flex-
ibility at runtime by preventing business transaction failures as well as abortions. Essentially, this is one of the objectives of this research. This is a critical criteria for Solútha BTM.

- **Applicability** was chosen to validate the completeness of the model. Completeness of the model is important to prove that it provides sufficient and necessary design-elements, operators, and other keywords to design business transactions of end-to-end business processes of a particular domain.

- **Correlation** is the key to governing business transactions from the business perspective. It was chosen to prove that the model has the ability to establish a strong and explicit correlation between the business and functional perspectives. Essentially, this is the other objective of this research.

The formal methods ‘Set Theory’ and ‘Time Extended Finite State Machine’ were used in validating the correctness of the model. In addition, the model used in a business scenario called Rhea Automotive POM (Manzouri et al., 2011), to demonstrate the applicability of the model in end-to-end business processes. The primary study of Rhea Automotive POM was conducted by Manzouri et al. (2011). I use secondary data of this business case for validating my model. The case was chosen because: (i) the case includes source, make, deliver processes, (ii) the case is about a purchase order management scenario in the automotive business domain, and (iii) it is well-documented. The model checking tool UPPAAL was used in during the demonstration procedure. The inputs specifically the transactional business process was defined using TEFSM and the behavioural properties were defined using ‘Computational Tree Logic’(Clarke et al., 1999). Additionally, Tenneco’s Purchase Order Agreement, (2012), and Tower International, (2011), were used in the validation.

Concerning evaluation, Peffer et al., (2007) suggest to observe and measure how well the artifacts supports a solution of the the problem. They suggested comparison as a method of measuring the performance of the design artifacts. The evaluation can take different forms based on the nature of the problem and the design-artifacts. Peffer et al. defined two forms: comparison of the artifact’s functionality with the solution objectives and objective quantitative performance measures, such as budgets or items produced, the results of satisfaction surveys, client feedback, or simulations. Adapting these suggestions this thesis was evaluated in two steps:

- **Comparison of simulation outcomes with evaluation criteria**: A list of criteria was defined. The criteria point to the research objectives. These were defined to evaluate the performance of Solútha BTM. The outcomes were observed through simulating the Rhea purchase order business transaction model. Then the outcomes were compared with the listed criteria.

- **Comparison of Solútha BTM with the state of the art**: Solútha BTM was compared with the existing business transaction models with respect to the functional characteristics and the design-elements.  

25[http://uppaal.org]
3.4 Research Validity

Research validity depends on the nature of research. Joppe (2000) defines validity in quantitative research as follows: *it determines whether the research truly measures what it was intended to measure or how truthful the research results are.* Wainer and Braun (1998) describe it as “construct validity”; the construct is the initial concept, notion, question or hypothesis that determines which data is to be gathered and how it is to be gathered.

There are researchers who argue that the term validity is not applicable to qualitative research (Golafshani, 2003). However, some, *e.g.*, Creswell Miller (2000) argue that validity is affected by the researcher’s perception of validity in the study and his/her choice of paradigm assumption. According to Golafshani, many researchers have developed their own concepts of validity and have often generated or adopted what they consider to be more appropriate terms, such as, quality, rigor and trustworthiness (Davies Dodd, 2002; Lincoln Guba, 1985).

This is neither quantitative nor qualitative research. It is design-science research. According to Juhani (2007), “the essence of Information Systems as design science lies in the scientific evaluation of artifacts”. However, the secondary data of business cases were analyzed for data collection. For this research, it raises a threat to external validity. Write *et al.*, defines external validity as follows, “*it is the applicability of study or experimental results to realms beyond those under immediate observation*”. In other words, it measures the degree to which the results of an investigation can be generalized. Excluding the threat to external validity, there are more considerations which are discussed in the following sections.

**Generalization**

The secondary data of business cases (Nguyen, 2009), (Papazoglou, 2012), and (Simchi-Levy, 2008) analyzed in this research are of the purchase order management scenarios in the automotive industries. The business processes of these cases were investigated to collect design-elements. It is worth mentioning that the design-elements that were collected from the processes are not concrete process elements. Rather, they are abstractions of the concrete elements. Therefore, although the business processes studied are context-specific, Solubtha BTM is a generic model to certain cases. This is a limitation. The purchase order management scenario studied in this research comprises to source, make, and delivery processes. In addition, the model was validated over the business scenario which was taken from the literature (Manzouri *et al*, 2011). Therefore, the model may need further improvements to be used in generic purchase order management scenarios that involve *return* processes.
Data Collection

The data were collected from the literature and business cases. The business cases were found in technical reports. I did not conduct primary study of the business cases. I used the data produced through the primary study. I analyzed secondary data to identify the potential design-elements for constructing my model. Although the business cases from where I collected data were well documented, yet the data extracted from the cases are secondary for this research. This promotes various threats. For instance, the data could be misinterpreted by the primary researchers; the business processes presented in the documents or literature could be incorrectly modeled. Any minor mistake in business processes model will affect the analysis of secondary study. In addition, the primary researcher could miss data in his study. This may result incompleteness of the business process model presented in the documents that I studied.

3.5 Summary

In this chapter, the research approach undertaken was described. This chapter described the design-science research method which has been used for designing the research. The reason for choosing the search method was explained. The steps of the research process were detailed in this chapter. The selection of each method was rationalized. In addition, the threats to external validity of the research were described.
Chapter 4
Data Collection and Analysis

4.1 Introduction

In this chapter, I discuss the findings of studying secondary data of business case studies (Nguyen, 2009), (Papazoglou, 2009), (Simchi-Levy, 2008) and trading partner agreements (Governatory Milesovic, 2006), (ArceloMittal, 2013), (Tower International, 2011), and (Tenneco’s Purchase Order Agreement, 2012). My study covers two business areas: purchase order fulfillment and logistics because the model I have developed in this thesis is specifically for designing the business transactions of ‘purchase order management’ business processes which essentially belong to these business areas. The outcomes of the study are reported in this chapter. Also, the chapter presents the analysis of trading partner agreements.

In my study, I investigated end-to-end purchase order management business processes that were found in the literature. The business processes present the functional aspects of a business particularly, the business operations and their dependencies with business related issues such as agreements (which are the main concern of the research presented in this thesis). Additionally, processes represent the characteristics of business transactions. Therefore, understanding and extracting transaction behaviour can be straightforward. In this study, I analyzed each business function contained in the business processes of the Auto Inc. and the AVERS OEM business cases. I studied the interconnections and interactions between the business functions. Additionally, I studied the sources of constraints that control business functions or business process activities. Also, I analysed the fault handling activities (that were incorporated in the business processes) to discover the faults and how they were dealt with in these processes.

In short, in my investigation I performed three tasks: (i) identify design-elements (ii) Identify relations between design-elements, and (iii) identify business transaction faults. The results of the studies are discussed in the next sections.

4.2 Scope of the Study

The supply chain systems of the Auto Inc. and the AVERS OEM business cases rely on the Supply Chain Operation Reference (SCOR) (Bolstorff and Rosenbaum, 2003) reference model. SCOR is the most widely used reference-model for designing and implementing the supply chain management systems of organ-
The model consists of three levels of process details. The SCOR-model suggests five distinct reference processes plan, source, make, deliver, and return for the supply chain systems. In the first level, companies define the scope of their supply chain systems; more specifically, they decide the processes required to compose the end-to-end supply chain processes. The processes are configured at level-2 based on the strategies of the companies. In the next level, they identify and define the process elements, in particular the business functions of the corresponding processes; also, they identify the performance metrics in this level. More detail on these levels can be found in Supply Chain Council, (2000). The implementation of business processes is done in the fourth level however it is not within the scope of the SCOR-model because implementations are specific to the organizations which essentially vary. Figure 4.1 shows the SCOR-model.

The first level of the SCOR-model and the plan and return processes were outside of the scope of this study. The planning processes were not studied because it is barely related to the notion of business transactions. The return processes were not scoped in my study because the business scenarios do not incorporate the return processes. In my study, I analysed the following secondary data:

- the business process elements
- the configuration of business processes

These are critical because the design-elements, the relations between them, and faults can be identified through investigating business functions of business processes.

In addition, I analysed the secondary data of several trading partner agreements. The study covers the international trading agreements for automotive businesses.

### 4.3 Case 1: The Auto Inc. Supply Chain (Nguyen, 2009)

The Auto Incorporation Automotive Supply Chain case was documented by the S-Cube consortium (S-Cube, 2008). The business case was published in Nguyen (2009). In this section, I analyze secondary data of this case published in Nguyen (2009). I provide a brief overview of the Auto Inc. supply chain scenario and then in the next sub-sections I present the results of my analysis.

#### 4.3.1 Overview

In this section I summarise the description of the supply chain scenario reported by Nguyen (2009). Auto Inc. is located in South East Asia. It is a local branch of a large enterprise in the automotive industry in Europe. It comprises a regional headquarters in Singapore and a manufacturing factory in Vietnam. Auto Inc. collaborates with several regional distribution centres, logistics providers, and warehouses located
in different countries in South East Asia. Figure 4.2 depicts the supply chain scenario and shows where business partners from various locations are involved. In this business scenario, the purchasing department of the manufacturing factory is responsible for sourcing materials from several suppliers. The main automobile parts are sourced from the headquarters supplier which is called the main source, while other parts and accessories are sourced from the best regional suppliers, selected through an assessment of the price, availability, business rules, regulatory rules, performance, and business relationship. The regional suppliers are called local sources. The local sourcing strategy is efficient in reducing costs and risks.
The manufacturing factory is assembling CKD (Completely Knocked-Down) automobile body parts and other parts such as wheels, brakes, chairs, etc.; painting; integrating accessories (e.g., air conditioner, CD player, etc.); testing and releasing the final products. Depending on the product specifications, the assembling, integrating, and painting tasks use varying materials and products, and might be executed in different ways as well.

Auto Inc. employs Stock-keeping Unit (SKU) and Warehouse Distribution strategies to efficiently manage the product inventory and enhance delivery to the customers. A unique SKU number is assigned to each finished product and thus it can be managed by the inventory management system. Auto Inc. can track the product units that are held in the manufacturing factory and warehouses or delivered to the retailers. Distributing products to different warehouses according to sales forecasts speeds up the delivery time and helps in better analysis of the sales operations in each country.

![Automotive Supply Chain Scenario](Source: Nguyen, 2009)

The logistics providers are the collaborators of Auto Inc. providing the transportation services to deliver the finished products from the manufacturing factory to the warehouses and from the warehouses to the retail customers. The selection of provider depends on transportation routes and rules.

The planning, coordinating, and controlling of the entire supply chain of Auto Inc. is carried out by
the headquarters based in Singapore. In addition, the Singapore headquarters defines product development strategies, performs sales and operation research, and manages relationships with customers.

Auto Inc. and the partners involved in the business scenario carry out most of their business operations and interactions with partners electronically. The company has a large-scale supply chain management software suite that integrates an Enterprise Resource Planning (ERP), Sales and Distribution, Logistics Management etc., etc.. They use RosettaNet Partner Interface Processes (ROSETTANET, 2012) for aligning their business processes. The Partner Interface Processes (PIPs) define the business processes between business partners involved in the Auto Inc. business scenario. They include business logic, message flows, and message contents.

4.3.2 Identify Design Elements

The order processing business process of Auto Inc. is composed of plan, source, make, and deliver processes. The Auto Inc. supply chain process covers the reference processes of the SCOR model. I investigated the make, source, delivery processes. These processes were found to be decomposed into many business processes: sales, CRM, ERP, Logistics, financial, and shipment processes. Each business process is composed of business functions. I studied several business functions such as inventory, payment, renegotiation, etc., to extract design-elements.

The design-elements are categorized into Business and Functional perspectives. The design-elements which belong to the business domain are the business perspective elements whereas the elements which belong to the functions of business processes are the functional perspective elements. Table 4.1 shows the list. The left column shows perspective, the right column shows design-elements.

Table 4.1: A List of Identified Design Element

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Design-elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Constraints</td>
</tr>
<tr>
<td></td>
<td>Spatial Constraints</td>
</tr>
<tr>
<td></td>
<td>Temporal Constraints</td>
</tr>
<tr>
<td></td>
<td>Quality of Service Constraints</td>
</tr>
<tr>
<td>Functional</td>
<td>Business Activities</td>
</tr>
<tr>
<td></td>
<td>Atomic Activity</td>
</tr>
<tr>
<td></td>
<td>Alternative Set</td>
</tr>
<tr>
<td></td>
<td>Transaction Process Fragment</td>
</tr>
<tr>
<td></td>
<td>Standard Discriminator</td>
</tr>
</tbody>
</table>

While investigating the order processing business process I found that the customer and supplier processes were constrained over some attributes. More specifically, the delivery function was constrained over time and location. From my study I understood that these constraints determine the behaviour of busi-
ness transactions at runtime. The basis of my understanding was an activity called 'cancel shipper'. This would be executed if the shipper failed to ship the ordered goods to customers. Consequently, the business transaction would be aborted.

I found that the 'shipment time' was agreed on by Auto Inc. and the shipper. This time constraint contained in the ‘Service Level Agreement’ is a business perspective design-element. Thus, I categorised 'constraint' as a ‘business perspective’ design-element. I found various time constraints such as Delivery Time and Order fulfillment Cycle Time. These control different business functions delivery and process order of Auto Inc. These are called temporal constraints.

Additionally, I identified a location constraint in the ‘ERP process’. After receiving the purchase order request, the ERP process of Auto Inc. assigns the inventory date and location to deliver the order. The delivery location is a constraint for Auto Inc. Each obligation to this study is a ‘Constraint’ because such obligations must be satisfied by the partners during business transactions. The location constraints is named spatial constraint. This constraint is categorized as business perspective element as well.

The ‘Quality of Services’ attributes are important to Auto Inc. business scenario. It is a technical issue which has an impact on business operations. An example of a quality of services attribute is ‘processing time’ with regards to business process activities. This was essentially found in the part where the PIPs are described in Nguyen (2009, Section 2.2). The analysis concluded that quality of service attributes can influence issues such as ‘customer satisfaction’. A customer may cancel the purchase order due to dissatisfaction. In this thesis, I named the service quality attributes ‘quality of service constraints’.

In the order processing business process, activities are composed of other activities. It implies that the execution time of composed activities is greater than the other activities. These are named business activities. Additionally, the process contains activities which are non-decomposable. These are called the atomic activities. Moreover, I found some activities were defined in the process as alternatives to the core activities. And in some cases they were defined as a set. Thus, I called the alternative activities ‘Alternative Set’. For instance, in the pricing function, there are three alternative activities, aggregate prices for retailers, aggregate prices standard retailers, and aggregate prices individual retailers. These activities are categorised as functional perspective design-elements as these elements are related to operations. Another functional perspective element elicited is the ‘business function’ which is composed of process activities. I called the business functions transactional process fragment (TPF) because it is a suitable term from the technical perspective.

It was reported in Nguyen (2009) that Auto Inc. does not have a long-term logistics partner. The logistics process of Auto Inc. sends shipment requests to a list of shippers and negotiates with the one who responds first. I compared this workflow pattern with the standard control flow patterns defined by the workflow management coalition (WFMC), described by Russel et al. (2006). I found an exact match of this
4.3. **CASE 1: THE AUTO INC. SUPPLY CHAIN (NGUYEN, 2009)**

Workflow in a flow pattern called 'standard discriminator'. I added this workflow pattern. Since it defines the execution paths of process activities, it is a functional perspective element.

### 4.3.3 Identify Relationship between Design Elements

In this section I describe how relations between design-elements were discovered. Relation refers to the logical connection between elements. In the order processing business processes, the non-decomposable atomic activities were contained in composite activities. This implies that there is a *composition* relation between business and atomic activities. I observed that the same logical relation exists between business activities and alternative activities. Also, I found process activities are contained in transaction process fragments.

Since different constraints were found in this study (see Table 4.1), I defined ‘constraint’ as a composite element that contains all other constraints including time, spatial, and quality of service. Furthermore, as mentioned in the previous section, ‘Standard Discriminator’ defines execution paths of process activities; there is an association relation between ‘Standard Discriminator’ and activities.

These logical relations are presented graphically in Solúbtha BTM V1 (Appendix E, Page 285).

### 4.3.4 Identify Business Transaction Faults

This section describes how faults were identified in this study. The study results in a list of faults. See Table 4.2. It is worth noting that I do not provide a detailed description of faults in this chapter. Rather, I explain how they were found in my study. A comprehensive description of the faults is provided in Chapter 5.

<table>
<thead>
<tr>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Late Acknowledgment</td>
</tr>
<tr>
<td>Product Unavailability</td>
</tr>
<tr>
<td>Compliance Rule Violation</td>
</tr>
<tr>
<td>Uncertain Events</td>
</tr>
<tr>
<td>Late Shipment Notification</td>
</tr>
<tr>
<td>Credit Card Validation Fault</td>
</tr>
<tr>
<td>Message format fault</td>
</tr>
<tr>
<td>Data Fault</td>
</tr>
</tbody>
</table>

I discovered these faults in the order processing business process of Auto Inc. In the business process, an activity called ‘send acknowledgment’ is constrained over time. If an acknowledgment of purchase order is delayed or not sent to the requester it may result in the cancellation of the order. This fault is called ‘late
acknowledgment’. Each inbound purchase order request received is registered after inventory is checked. If the inventory department finds unavailability of products, it initiates an order to the respective supplier for replenishment of products. If the replenishment process is not successful, Auto Inc. cancels the purchase order. This fault is called ‘product unavailability’.

For selecting suppliers, Auto Inc. faces challenges in dealing with tax and compliance policies, which is typical in South East Asian countries (Nguyen, 2009). It was reported in the case that the compliance policies are important to business operations. The consequence of violating such policies may lead business transaction abortion. Therefore, violation of compliance policy is a critical fault.

Nguyen (2009) reported Auto Inc.’s concern with delay at border and bad weather which are uncertain events, however, they may happen. They are called uncertain because the time of occurrence of these events are not known in advance. These events may cause ‘delivery delay’ or ‘delay of sourcing raw materials’. These delays, especially the latter type, may lead to cancellation order.

The ‘late shipment notification fault’ was discovered in the ‘process shipment’ business function. In this function, ‘shipment notification’ is an activity constrained over time. It indicates that this activity should be carried out within a specified time. However, it was reported that the failure to satisfy this time constraint does not interrupt the progression of business operations. In other words, the business process continues running by tolerating this fault. This observation influenced the notion of ‘soft’ rule in this thesis.

‘Credit card validation fault’, ‘message format fault’, and ‘data fault’ were discovered in ‘order processing’ business process. This process contains an activity called ‘validate purchase order’. It validates customer credentials, in particular; ‘checks customer credits’; checks syntactic correctness of the purchase order; and checks data completeness. If these validations fail then the sales management system of Auto Inc. triggers exceptions. The data fault is triggered if purchase orders are missing required data; the credit card validation fault occurs if an insufficient balance was found in the customer account; and message format fault is triggered if syntactic incorrectness was found in the purchase orders.

4.4 Case 2: the AVERS OEM Supply Chain (Papazoglou et al., 2012)

The AVERS OEM supply chain case is an extension of the Auto Inc. supply chain by Papazoglou et al., (2012). In this case I analyse the secondary data of this case. The operational patterns of AVERS’s supply chain system is slightly different. For instance, in Auto Inc. the suppliers were chosen on the fly, while the business process is running, whereas in AVERS supply chain, the suppliers are trading partners and have a formal contracts with AVERS OEM. In addition, the company has long-term contracts with two logistics providers.
4.4. CASE 2: THE AVERS OEM SUPPLY CHAIN (PAPAZOGLOU ET AL., 2012)

4.4.1 Introduction

Like Auto Inc., AVERS (Advanced AutomotiVE paRts) OEM supply chain comprises three processes: source, make, and delivery (Papazoglou, 2012). In source process, the raw materials are procured from suppliers to manufacturing department of AVERS OEM. In make process, the manufacturing department of AVERS assembles the automotive parts and produces the finished products. Finally, in delivery process, the finished products are shipped to customers through a supply network include warehouse, distributor, and retailer.

Motor vehicle manufacturers require a wide number of automotive parts for producing the end products cars. Over time, the automotive supply chain has evolved from a model in which most parts were built by the primary manufacturer, to a highly-functionally segmented model in which Original Equipment Manufacturers (OEMs) provide finished parts to the assembly lines of car manufactures (Papazoglou, 2012). OEM refers to “a manufacturer of equipment that may be marketed by another manufacturer (Kidder, T., 1997), (IBM Dictionary). Today, the production of automotive parts is mostly outsourced rather than produced in-house. These parts are mainly sourced from OEM companies that are specialized in manufacturing automotive parts.

In the Automotive Supply Chain scenario shown in Figure 4.3, Auto Inc. sources the automotive parts from the AVERS OEM company. AVERS is a Tier 1 supplier for Auto Inc. The company has different departments including sales, logistics, and manufacturing. The AVERS OEM collaborates with various partners including suppliers, shippers, and warehouses. The suppliers are Tier 2 suppliers who supply components (e.g., valve) to AVERS. The components are used in manufacturing the finished parts which are then stored and inventoried in the warehouse. The shippers ship the ordered parts to Auto Inc. Furthermore, the bank, which is not a business partner but an intermediary, exchanges financial information with AVERS OEM and Auto Inc. Figure 4.3 depicts the actors involved and their relationships in the AVERS automotive supply chain case. In the figure, the legend shows two types of flow between the supply chain actors. The information flows are messages exchanged between the trading partners and the product flows refer to transportation of goods between partners.

The actors involved in the automotive supply chain case are briefly introduced in the following:

- **Automotive Incorporation (Auto Inc.)** – is a large automotive company which produces cars. It has a manufacturing plant for assembling cars. The company sources the automotive parts such as body parts, wheels, chairs, etc. from suppliers from different regions. AVERS OEM supplies fuel pumps to Auto Inc. The company distributes the cars to its different warehouses which are then shipped to the retailers.

- **AVERS** – is the central actor of this automotive supply chain. AVERS OEM produces various types of automotive parts such as fuel pumps, electrical components, steering, and braking systems etc.
• **Component Suppliers** – AVERS OEM needs different components, e.g., valve and Anti-Float springs for making fuel pumps. The company has several preferred suppliers who deliver these components. AVERS OEM sends replenishment requests to the component suppliers if the stock of components is found insufficient or the inventory level reaches a critical threshold. Replenishment is a critical activity for AVERS OEM for guaranteeing the *on time* deliver of goods ordered by Auto Inc. It is worth noting that AVERS OEM can switch to another supplier if the current supplier fails to replenish the components.

• **Warehouses** – AVERS OEM is partnered with two warehouses that provide storage and packaging services. The finished products are stored and inventoried in the Central Warehouse from where the products are distributed to the Subsidiary Warehouse. The products are collected from the subsidiary warehouse by the shippers and delivered to Auto Inc.

• **Shipper** – provides transportation service. AVERS OEM collaborates with two transportation service providers: Shipping Limited and Transportation Inc. They ship goods (fuel pumps) to Auto Inc. Shipping Limited is an alternative transportation service provider that replaces Transportation Inc. if Transportation Inc. fails to ship the ordered goods.

• **Financial Institution** – is the bank which cooperates with the AVERS OEM by providing credit information related to Auto Inc. Additionally, the bank relays the payment information to both AVERS OEM and Auto Inc.
Noticeably, a number of actors are excluded from this supply chain scenario. In particular, the Tier 3 suppliers are excluded from the downside of the chain and the distributor, wholesaler, and retailer are not included in the upstream of the chain.

4.4.1.1 Identify Design Elements

The study shows that the scenarios of Auto Inc. and AVERS OEM supply chain cases are somewhat similar except for that some internal operational activities of these organizations are different. In this study I found a very few design-elements.

Table 4.3: List of Design Element identified in the AVERS Supply Chain

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Party</td>
</tr>
<tr>
<td>Functional</td>
<td>XOR Choice</td>
</tr>
<tr>
<td></td>
<td>Sequence</td>
</tr>
<tr>
<td></td>
<td>AND Choice</td>
</tr>
<tr>
<td></td>
<td>Interleaved</td>
</tr>
<tr>
<td></td>
<td>AND Join</td>
</tr>
<tr>
<td></td>
<td>Exclusive Join</td>
</tr>
<tr>
<td></td>
<td>Multiple Instance without Synchronization</td>
</tr>
<tr>
<td></td>
<td>Control Flow Pattern</td>
</tr>
</tbody>
</table>

The AVERS OEM supply chain process consists of several processes that are grouped into five main processes: Material and Procurement, Order Processing and Production scheduling, Product Inventory Control, Product Shipment Administration, and Production Control. I studied them all except ‘Production Control’ process. I found each of these processes involves at least two parties. For instance, the ‘Order Processing and Production Scheduling’ process involves Auto Inc. which is the buyer and AVERS OEM which is the seller.

In this study, I investigated the workflows of AVERS OEM. I found several ‘control flow patterns’ (listed in Table 4.3) that are the functional perspective elements. No explicit definitions of these patterns were found, however, they were used in the processes as graphical notations. I studied the workflow patterns reported in Russel et al. (2006) to understand the semantics of these notations and compared them with the ones found in the business processes. I discovered nine control flow patterns in the business processes reported in the AVERS OEM. The XOR Choice was seen in the process activities that were split into two or more, however, the outgoing execution path was constrained over mutual exclusive logic. The AND Choice, AND Join, and Exclusive Join are similar types of design-elements found in the business processes. Sequence is the most common flow pattern found in each business process I investigated. Multiple
Instance without Synchronization flow pattern was found in ‘Raw Material Supplier’ process. Interleaved was observed in the ‘Payment function’. Table 4.3 lists these elements.

4.4.2 Identify Relationship Between Design Elements

In this section I describe how relations between the design-elements were discovered in this study. I discovered an element control flow pattern that is composed of interleaved, sequence, exclusive OR choice, exclusive AND choice, exclusive OR join, exclusive AND join, and Multiple Instances Without choice. ‘Party’ is the only new business perspective element found in this case. The relations discovered in this study are presented in the final version of the model ‘Solúbtha BTM’ (Figure 7.9, Chapter 7, Page 144).

4.4.3 Identify Business Transaction Faults

No new fault was found in this study. The faults credit card validation fault, late acknowledgment of the order, and product unavailability which were found in the Auto Inc. business case, were found here too.

4.5 Case 3: Walmart Changes Tactics to Meet International Tastes (Simchi-Levy, 2008)

This section presents the outcomes of the analysis of secondary data of WalMart business scenario. This was a specific-purpose study. The purpose of studying the business scenario was to identify the business related faults. In this study, I focused on analysing the ‘delivery’ process of Walmart. This case was taken from Simchi-Levy (2008). It was originally published in the Wall Street Journal (Friedland and Lee, 1997).

4.5.0.1 Introduction

In this section, I summarise the business scenario described by Simchi-Levy, (2008). Walmart Inc. is one of the fortune’s Global 500 companies and the largest retailer in the world. The company started its operation in two countries, Brazil and Argentina in South America with high expectation. Unfortunately, the expectations were not met during the period when this case was prepared. In fact, retail analysts forecasted $20 to $30 million losses in the fiscal period. One of Walmart’s executives conceded that the company was losing money, but said its performance met expectations. The head of international operations Mr. Martin said, “What counts is that we are finding great customer acceptance.”

Walmart has agreements with a number of retail stores in both Brazil and Argentina and has some partners such as Lojas Americanas which is a Brazilian partner. It distributes goods to these stores. It collaborates with a list of local suppliers of goods. Figure 4.4 shows the logistics scenario. The international logistics provider ships goods from the main distribution centre located in the United States (U.S.). Walmart
has partnerships with local shippers who ship goods from the local supplier’s warehouses to different retail centres.

![The Logistics Scenario of Wall-Mart](image)

**Figure 4.4: The Logistics Scenario of Wall-Mart.**

### 4.5.1 Identify Business Transaction Faults

Although expected many, only two business related faults were identified in my analysis of secondary data presented in this case report: *Delivery Delay* and *Disappearance of Goods*. I found in the report that in Brazil Walmart was experiencing *disappearance of goods* from the port during delivery of goods from the main distribution centre to the stores. This was causing *delivery delays*.

### 4.6 Business Partner Agreements

The main purpose of studying the business partner agreement was to identify the faults that lead to business transaction failure or abortion. This section describes how business transaction faults were identified in the agreements.

An agreement is a legally binding contract between business partners such as buyers, seller, and shipper (Governatory Milosevich, 2006). The business partners establish agreement through mutual understanding. An agreement contains the policies that the business partners must abide by during business operations.

### 4.6.1 Identify Business Transaction Fault

The analysis of secondary data of business partner agreements produced a list of faults which is presented in the Table 4.4. A description of these faults is provided in Chapter 5.

<table>
<thead>
<tr>
<th>Faults</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shipment Fault</strong></td>
<td>Several clauses related to shipment were found in agreements including location of shipment and method of shipment. The business transactions can be aborted if these clauses are not met during business operations. These are called ‘shipment fault’.</td>
</tr>
<tr>
<td><strong>Ordering Fault</strong></td>
<td>This fault was refined from a condition stated in ArceloMittal, (2013). The condition is this, the ‘purchase manager’ must sign purchase orders before sending it to the seller.</td>
</tr>
<tr>
<td><strong>Product Quality Fault</strong></td>
<td>I found each agreement contains clauses about the quality of products. The product quality faults were identified in these clauses.</td>
</tr>
<tr>
<td><strong>Breaching Confidentiality</strong></td>
<td>Confidentiality is a common clause found in all agreements. If any business secret is leaked to unauthorised individuals, then it is a breach of confidentiality.</td>
</tr>
<tr>
<td><strong>Uncertain Events</strong></td>
<td>This fault was found in ‘force majeure’ paragraphs in all agreements. It contains clauses of uncertain events including Acts of God, fire, floods etc.. mentioned in all agreements.</td>
</tr>
<tr>
<td><strong>Packaging Fault</strong></td>
<td>There are clauses found in agreements describing the consequences of not meeting agreed specification of packaging.</td>
</tr>
<tr>
<td><strong>Payment Delay</strong></td>
<td>This fault was found in the ‘invoice and payment’ clause of agreements.</td>
</tr>
</tbody>
</table>

### 4.7 Summary

The faults are an important finding in this research because faults help to develop failure prevention techniques. In this chapter, the secondary data of business cases were analyzed. A list of design elements were extracted. I explained how the design-elements were extracted. The relations between them were identified. The fault extraction processes were described. More specifically, I described how different types of faults were extracted from the case reports.
Chapter 5

Business Transaction Fault Models

A business transaction fault, from the technical perspective, is an exception or an error or an unexpected outcome. There are several types of business transaction faults. This chapter presents the fault models constructed by composing the faults elicited by analysing the secondary data of business cases and from the literature. In addition, the formal semantic of the faults are defined in this chapter.

5.1 Construction of Fault Models

This section summarizes how the fault models were constructed. To construct the models I analyzed various aspects of end-to-end business processes: message, protocol, application, data, and business. These aspects are related to the business transactions of end-to-end business processes. They can be put together in a simple business transaction scenario: A business transaction involves several applications that are hosted at different locations. These applications perform business transactions by satisfying business protocols and agreements. Since the applications are distributed, transactions are performed by exchanging messages (which contain data) via the Internet. At runtime, faults can occur with message, data, application, business protocol, and agreement.

For constructing the fault models I use a hybrid approach which combines top-down and bottom-up approaches. For instance, first the leaf level faults are added to the tree then the parent faults are added above the leaves. This is the bottom-up approach. However, in some cases leaf level faults are extended to another leaf level faults. This is the top-down approach. It is worth noting that I do not argue that the fault models are complete since my study is limited to the literature and secondary data of business cases and trading partner agreements discussed in the previous Chapter.

The faults are categorised into business related faults, business protocol related faults, message related faults, business object related faults, and application related faults. These are discussed in the following sections.
5.2 Business Related Fault Model

The business related fault model is the logical structure of the cause-and-effect relationship between the faults related to the business perspective of the end-to-end business processes. Figure 5.1 illustrates the business related fault model.

![Figure 5.1: The Causal Tree of Business Related Faults.](image)

The model is presented as a causal tree. The root fault is *violation of agreement* that consists of *product unavailability*, *uncertain events*, *policy Non-conformance*, and *product quality violation*.

Product unavailability refers to unavailability of goods. It was found in both the Auto Inc. and the AVERS OEM business cases. The uncertain events are the unpredictable events which are known but their exact time of occurrence is nearly impossible to predict. This fault was found in the business partner agreements.

The policy Non-conformance denotes a violation of policies. While performing business transactions, the business partners must comply with policies that were agreed on. For example, the *shipment fault* is a *Non-conformance* of policy that occurs when a shipper fails to comply with the agreed upon shipment policies.

The business related fault model is extended in Figure 5.2. In the extended model the policy Non-conformance fault is refined to *business policy violation*, *compliance policy violation*, and *security policy violation*. The refinement was done based on the study of trading partner agreements. The policy Non-conformance faults are related to business activities such as payment processing and order processing; the security policy violations are related to the security issues such as non-disclosure clauses; and the compliance policy violations are related to compliance issues such as legal constraints.
Figure 5.2: The Complete Causal Tree of Business Related Fault.
Table 5.1 summarizes these faults.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand Variability</td>
<td>Demand variability refers to the range of values for demand which is a variable based on the efforts of marketing or promotions, seasonality, holidays, special events, and extrinsic factors: the short-term impact of demand variability is inaccurate inventory deployment that causes product unavailability (Ramesh, 2009).</td>
</tr>
<tr>
<td>Product Unavailability</td>
<td>Demand uncertainty refers to the limited knowledge about what is going to sell or the inability to predict demand. The short-term impact of demand uncertainty is the same as the result of demand variability (Ramesh, V., 2009).</td>
</tr>
<tr>
<td>Ordering Fault</td>
<td>This fault was found in the DMS Contract. In the contract, there was a clause written as follows - the ‘purchase order’ must be prepared in accordance with the agreed specification. If the responsible business partner does not comply with this, it is considered a fault that leads to cancellation of the order. This leads to business transaction abortion.</td>
</tr>
</tbody>
</table>
Temporal Fault

The time related faults including late acknowledgment and delivery delay, were found in the Auto Inc. business case. Temporal faults were also found in the agreements, e.g., ArcelorMittal, (2013) and Hvitved et al., (2012). In this research, the time related faults are grouped and named temporal faults. Violating deadlines, e.g., delivery date, late acknowledgment, and delay of payment are the violations of temporal policies which cause business policy violation.

Business Policy Violation

Shipment Fault

The shipment faults were found in ArcelorMittal (2013). A shipment fault occurs if a shipment policy such as transportation mode is not satisfied.

Customer Credential Fault

This fault was found in several contracts including ArcelorMittal (2013), DMS Contract, Tenneco’s Contract (2012), and Tower International Contract. In addition, it was seen in both the Auto Inc. and the AVERS OEM supply chain cases. If a buyer’s credentials, in particular, the credit detail, could not be verified successfully by the seller’s sales and distribution system then the system raises the customer credential fault.
<table>
<thead>
<tr>
<th>Fault Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packaging Fault</strong></td>
<td>If a packaging specification agreed on by the trading partners is not satisfied, then it is a packaging fault, which leads to a business policy violation. This fault was elicited from the trading partner agreements. However, I found that an agreement, Tenneco’s agreement (2012), does not contain any packaging policy.</td>
</tr>
<tr>
<td><strong>Compliance Policy Violation</strong></td>
<td>The trading partners must comply with the regulations. The violation of regulations causes a compliance policy violation that leads to a violation of an agreement. Eventually, business transactions abort.</td>
</tr>
<tr>
<td><strong>Breaching Confidentiality</strong></td>
<td>It was extracted from the ArcelorMittal and the ITW Global Automotive agreements. If a business partner discloses any confidential information to any third party, who is not a business partner, then it is a security policy violation that causes agreement violation. This violation may lead business transaction abortion.</td>
</tr>
<tr>
<td><strong>Repudiation Fault</strong></td>
<td>If a business partner denies a valid and authentic object such as signature, then it is a repudiation fault. This fault was found in UMM (2003).</td>
</tr>
</tbody>
</table>
### 5.2. BUSINESS RELATED FAULT MODEL

<table>
<thead>
<tr>
<th>Fault Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security Policy Violation</strong></td>
<td>If a trading partner sends an unintelligible business document to another trading partner then it is an intelligibility constraint violation. This fault was found in UMM (2003).</td>
</tr>
<tr>
<td><strong>Intelligibility Constraint Violation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Tamper Resistance Constraint Violation</strong></td>
<td>If a business document sent by a trading partner is tampered (altered) then it is a tamper resistance constraints violation. This fault was found in UMM (2003).</td>
</tr>
<tr>
<td><strong>Damaged Goods</strong></td>
<td>In Tenneco’s Contract (2012), a clause was found stating that the consequence of damaged goods – “damaged goods will be returned to the seller and the sum of amount paid for the goods must be refund”. If a seller delivers damaged goods it is a product quality violation which causes agreement violation. This may lead to business transaction abortion.</td>
</tr>
<tr>
<td><strong>Product Quality Violation</strong></td>
<td>This fault was found in almost all the agreement studied in this thesis. If the quality of delivered goods does not match the specification agreed by the trading partners, the business partner (typically the buyer) can cancel the purchase order; consequently the business transaction aborts.</td>
</tr>
<tr>
<td>Fault Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Disappearance of Goods</td>
<td>I found this fault during the analysis of the Walmart business case. It can happen during the transportation of ordered goods. This event causes delivery delay which may lead the cancellation of the ‘purchase order’. This may result business transaction abortion.</td>
</tr>
<tr>
<td>Transportation Strikes</td>
<td>I found this fault in the ArcelorMittal agreement (2013). The transportation strike causes the violation of delivery deadline. The transportation strike results in cancellation of the purchase order that leads to the business transaction abortion.</td>
</tr>
<tr>
<td>Workforce Strike</td>
<td>This fault was identified in the ArcelorMittal agreement (2013). The workforce strike may result in business transaction abortion.</td>
</tr>
<tr>
<td>Uncertain Event</td>
<td>Acts of God</td>
</tr>
<tr>
<td></td>
<td>This fault was found in ArcelorMittal agreement (2013) and Tower International Contract. Some examples of ‘Acts of God’ are earthquake, Tsunami etc.. If this event occurs, it will abort the business transactions.</td>
</tr>
</tbody>
</table>
5.3 Business Protocol Related Fault Model

The term protocol denotes the code of correct conduct (FreeDictionary). In an end-to-end business process the partner applications must abide by the protocols for carrying out operations or interacting between them, while business transactions are running. Papazoglou and Kratz (2006) called them business protocols. Desai and Singh (2008) defined business protocol as a means of specifying the interactions between business partners. In Papazoglou et al. (2010), a business protocol is proposed as an approach for organizing the business functions. This research shares the same view. A business protocol structures and controls the operations performed by the activities contained in business processes.

While a business transaction is running, a business protocol related fault occurs when an incorrect sequence is detected during business transactions, if one or more protocol is not satisfied. This fault can result in business transaction failure or abortion. This section discusses the business protocol related faults and presents a model that is constructed by incorporating this type of fault.

The root causes of these faults are business function execution sequence faults and the messaging sequence faults. The causal tree of the business protocol related faults is shown in Figure 5.3.

![Figure 5.3: The Causal Tree of Business Protocol Related Fault.](image)

The business protocol related faults is summarized in table 5.2.
Table 5.2: Summary of the Business Protocol Related Faults

<table>
<thead>
<tr>
<th>Faults</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Function Execution Sequence Fault</td>
<td>A business function execution sequence fault happens if a ‘business function’ contained in a business process executes in an incorrect sequence. In the AVERS purchase order fulfillment process, the business partners agreed that the payment will be made after the goods are delivered. While the business transaction is running, if the payment function executes before the delivery function then it is treated as the business function execution sequence fault.</td>
</tr>
<tr>
<td>Messaging Sequence Fault</td>
<td>Another business protocol related fault is messaging sequence fault that occurs if a business partner sends a message out of sequence or a business partner receives an incorrect message. Since in electronic business transactions messages are the primary means of conversation between the business partners, and messages may contain the data that are inputs for a transaction, they must be sent in the correct sequence. The correctness of the sequence of a message refers to the sequences that were agreed upon by the business partners. If Auto Inc. in the purchase order fulfillment process receives order approval message before the order acceptance receipt message then it is a messaging sequence fault.</td>
</tr>
</tbody>
</table>

5.4 Business Object Related Fault Model

A business object is an electronic document. Technically speaking, business objects are the schema for purchase orders, invoices, etc. Several business objects can be instantiated while a business transaction is running. These business objects (BOs) may contain data. If a business object contains incorrect data or omits required data, then the business transaction produces inconsistent outcomes. The incorrect data or missing data are the business object related faults. They may cause business transaction abortion. For
instance, if a purchase order contains incorrect data or it is missing any data required for processing the order, then the business transaction can be canceled and consequently, it may abort. Figure 5.4 shows the business object related faults and the root causes of those.

The root causes of the business object related faults are: incorrect data and missing data. They are summarized in Table 5.3.

Table 5.3: Summary of the Business Object Related Faults

<table>
<thead>
<tr>
<th>Fault</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Data</td>
<td>I found this fault in the Auto Inc. (Nguyen, 2009) and the AVERS OEM (Papazoglou, 2012) business cases. If the body of a message contains incorrect data then the data fault occurs. For instance, if Auto Inc.’s purchase order message contains an incorrect data then the AVERS’s sales system replies with a fault message to Auto Inc.</td>
</tr>
<tr>
<td>Data Missing</td>
<td>This fault was found in the Auto Inc. and the AVERS OEM business cases as well. If data is missing in the body of a purchase order then the business transaction system generates the data fault message. For instance, if data is missing in Auto Inc.’s purchase order then the AVERS’s sales system replies with a fault message to Auto Inc.</td>
</tr>
</tbody>
</table>
5.5 Message Related Fault Model

The business transactions of a large-scale business process involve business partners, intermediaries, and service providers from different locations. The transactional activities are performed by the applications hosted on these partners’ sites. Since business transactions are performed collaboratively, these applications interact by exchanging messages. Therefore, messages are important constituents of business transactions.

The message related faults are the technical faults. I found three major faults of this type that affect business transactions: message format fault (Cohen, 2002), Response Fault (Edward, 2012), and Connection fault (Oracle, 2012). During the analysis of the business processes of the Auto Inc. and the AVERS OEM cases, I found that these faults may interrupt business transactions at runtime. Figure 5.5 presents the message related faults which I found in my study.

![Figure 5.5: The Causal Tree of Message Related Fault.](image)

The causes (shown in Figure 5.5) are described in Table 5.4.

<table>
<thead>
<tr>
<th>Faults</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message Format Fault</strong></td>
<td>When a business partner sends a message with an incompatible format to another business partner, the receiving system verifies the compatibility because it cannot READ electronic documents that are produced with an incompatible format. For instance, if Auto Inc. sends a message with incompatible format then the AVERS’s application server generates a fault message because the server is not able to parse the message due to the unknown format.</td>
</tr>
<tr>
<td><strong>Response fault</strong></td>
<td>If a data packet is lost or a host server does not respond during transportation of a message from one business partner to another, then the sending partner does not receive any response from the receiving end. This is a response fault.</td>
</tr>
</tbody>
</table>
5.6 Application Related Fault Model

The application related faults happen with software applications. This type of fault is well-understood because these faults have been studied extensively and reported in the literature.

Recently, enterprises have started using novel paradigms such as SOA and cloud computing for developing applications. The functional characteristics of these software applications are different from traditional ones. The advent of such applications promoted new type of application faults. I found three major faults of this type: Software Integrity Fault (Rosenberg, 2008), Processing Delay (Taher et al., 2012), and software unavailability (Papazoglou, 2003) (Taher et al., 2012). They may cause business transaction failure. Figure 5.6 shows the causal tree of the application related faults. These faults are summarized in Table 5.5.

| Connection fault | The connection fault occurs if the nodes (e.g., routers, web servers) of business partners fail to establish a connection between them. For instance, if AVERS OEM wants to send an advance shipment notification to Auto Inc., and if Auto Inc.’s logistics system fails to listen to the request sent by the AVERS’s messaging server, then it is a connection fault. |

**Figure 5.6: The Causal Tree Shows the Application Related Faults.**
5.7 Relationship between Faults

The business transaction faults are inter-related as well as intra-related. The intra-relation denotes connection between the faults that are member of the same parent. For instance, there is an intra-relation between demand variability and demand uncertainty. Also the packaging fault is internally related to the damaged product fault. The demand variability causes the demand uncertainty and the packaging fault causes the damaged product fault. The inter-related faults are the faults that are connected with each other but are the members of different parents in the fault tree. Two parent faults can be inter-related as well. The uncertain events transportation strike and workforce strike are inter-related to temporal fault. Figure 5.2 shows these relations. Additionally, there is a relation between software unavailability and processing delay. The software unavailability causes the processing delay. Figure 5.6 depicts is relation. There is a relation between business and application related faults. The faults processing delay and software unavailability can cause a temporal fault. Also, the business function execution sequence fault, messaging execution sequence fault, missing data, connection fault, and response fault can cause the temporal faults.
5.8 Formal Representation of Business Transaction Faults

The faults are formalized in this section. The formal definitions provide a precise and unambiguous interpretation of the faults and are used in automatic detection of faults through reasoning business transaction instance-models over fault models. The formal definitions of faults will underpin the business transaction systems in taking logical decisions based on the occurred faults to prevent business transaction abortion.

I use the propositional logic language for formalizing the faults. I choose this language because it is expressive, and the formalism is simple and decidable. The truth value of the statements formulated in the proposition logic can be checked in a simple way by a system (Klappenecker, A.). In this section, the business transaction faults are formulated as simple Boolean constraints.

The proposition logic language provides the logical connectives to formalize the statements. The alphabets of proposition logic comprises atomic sentences, atoms, or compound sentences that are composed of the atoms. The logical connectives of the propositional logic are given below,

- ¬ denotes Logical Not
- ∨ denotes Logical OR
- ∧ denotes Logical AND
- → denotes Implication
- ≡ denotes Equivalence
- = denotes Equal
- ⊕ denotes Mutual Exclusive

5.8.1 Formal Representation of BT Related Faults

For a business transaction ‘τ’, the business transaction faults may lead business transaction abortion ‘ζ’ unless they are recovered. Proposition 1 defines abortion as the result of a set of disjoint faults that are, business related fault ‘F_{Business}’, business protocol related fault ‘F_{BProtocol}’, message related fault ‘F_{Message}’, application related fault ‘F_{Application}’, and business object related fault ‘F_{BO}’.

Proposition 1. \( \forall \tau, (F_{Business} \lor F_{BProtocol} \lor F_{Message} \lor F_{Application} \lor F_{BO}) \rightarrow \zeta(\tau) \)

Proposition 2 defines business transaction abortion which is the consequence of the conjunction of the business related fault, the business protocol related fault, the business object related fault, the message related faults, application related fault, and business object related fault,

Proposition 2. \( \forall \tau, (F_{Business} \land F_{BProtocol} \land F_{Message} \land F_{Application} \land F_{BO}) \rightarrow \zeta(\tau) \)
A business transaction abortion is the Cartesian product which returns a set of business transaction faults that cause transaction abortion. Proposition 3 defines the Cartesian product of business transaction faults.

**Proposition 3.** \( \forall \tau, (F_{Business} \times F_{BProtocol} \times F_{Message} \times F_{Application} \times F_{BO}) \rightarrow \{F_{Business} \land F_{BProtocol} \} \lor \{F_{Business} \land F_{Message}\} \lor \{F_{Business} \land F_{Application}\} \lor \{F_{BProtocol} \land F_{BO}\} \lor \{F_{Message} \land F_{BO}\} \lor \{F_{Application} \land F_{BO}\} \rightarrow \zeta(\tau) \)

The formal definitions of these faults are provided in the following subsections.

### 5.8.2 Formal Representation of Business Related Faults

The violation of an agreement is the main business related fault denoted by \( \lambda(\Sigma) \) where ‘\( \lambda \)’ denotes violation and ‘\( \Sigma \)’ denotes agreement. It is defined in proposition 4 as a set of conjunctive faults include unavailability of resources ‘\( U_{Resource} \)’, violation of business policy ‘\( \lambda_{BPolicy} \)’, violation of compliance policy ‘\( \lambda_{CPolicy} \)’, violation of security policy ‘\( \lambda_{SPolicy} \)’, violation of product quality ‘\( \lambda_{PQuality} \)’, the uncertain events ‘\( E_{Uncertain} \)’.

**Proposition 4.** \( \forall \tau, (U_{Resource} \land \lambda_{BPolicy} \land \lambda_{CPolicy} \land \lambda_{SPolicy} \land \lambda_{PQuality} \land E_{Uncertain}) \rightarrow \lambda(\Sigma) \)

An agreement violation is the result of one or more business related faults. The formal definition of disjunctive business related fault is given in proposition 5.

**Proposition 5.** \( \forall \tau, (U_{Resource} \land \lambda_{BPolicy} \lor \lambda_{CPolicy} \lor \lambda_{SPolicy} \lor \lambda_{PQuality} \lor E_{Uncertain}) \rightarrow \lambda(\Sigma) \)

#### 5.8.2.1 Resource Unavailability

Resource unavailability ‘\( U_{Resource} \)’ is a set of disjunctive or conjunctive faults including demand variability ‘\( D_{Variability} \)’ and inventory forecasting inaccuracy ‘\( I_{InaccurateF} \)’. The Proposition 6 provides the formal definition of the resource unavailability fault.

**Proposition 6.** \((D_{Variability} \lor I_{InaccurateF}) \lor (D_{Variability} \land I_{InaccurateF}) \rightarrow U_{Resource} \)

Inventory forecasting inaccuracy can be the result of demand variability. Typically, the inventory is forecasted by analyzing the sales history of a company, which could go wrong for several reasons but mainly for the constant fluctuation of the customer demand. This can be written as follows,

**Proposition 7.** \( (D_{Variability} \rightarrow I_{InaccurateF}) \rightarrow U_{Resource} \)

#### 5.8.2.2 Business Policy Violation

Business policy violation \( \lambda_{BPolicy} \) is a set of disjoint faults including ordering fault ‘\( F_{Ordering} \)’, temporal fault ‘\( F_{Temporal} \)’, shipment fault ‘\( F_{Shipment} \)’, credential fault ‘\( F_{Credential} \)’, and packaging fault ‘\( F_{Packaging} \)’. It can formally be defined as follows,
Proposition 8. \((\mathcal{F}_{\text{Ordering}} \lor \mathcal{F}_{\text{Temporal}} \lor \mathcal{F}_{\text{Shipment}} \lor \mathcal{F}_{\text{Credential}} \lor \mathcal{F}_{\text{ProductQ}}) \rightarrow \lambda_{\text{BPolicy}}\)

A business policy violation is the outcome of a set of conjoined faults. Its formal definition is given below:

Proposition 9. \((\mathcal{F}_{\text{Ordering}} \land \mathcal{F}_{\text{Temporal}} \land \mathcal{F}_{\text{Shipment}} \land \mathcal{F}_{\text{Credential}} \land \mathcal{F}_{\text{ProductQ}}) \rightarrow \lambda_{\text{BPolicy}}\)

While a business transaction is running, more than one of these faults could occur for the same instance however, they may not occur at the same time. The Cartesian product of the business policy violation is defined in the proposition 10.

Proposition 10. \((\mathcal{F}_{\text{Ordering}} \times \mathcal{F}_{\text{Temporal}} \times \mathcal{F}_{\text{Shipment}} \times \mathcal{F}_{\text{Credential}} \times \mathcal{F}_{\text{Packaging}}) \rightarrow (\{\mathcal{F}_{\text{Ordering}} \land \mathcal{F}_{\text{Temporal}}\} \lor \{\mathcal{F}_{\text{Ordering}} \land \mathcal{F}_{\text{Shipment}}\} \lor \{\mathcal{F}_{\text{Ordering}} \land \mathcal{F}_{\text{Credential}}\} \lor \{\mathcal{F}_{\text{Ordering}} \land \mathcal{F}_{\text{Packaging}}\} \lor \{\mathcal{F}_{\text{Temporal}} \land \mathcal{F}_{\text{Shipment}}\} \lor \{\mathcal{F}_{\text{Temporal}} \land \mathcal{F}_{\text{Credential}}\} \lor \{\mathcal{F}_{\text{Temporal}} \land \mathcal{F}_{\text{Packaging}}\} \lor \{\mathcal{F}_{\text{Credential}} \land \mathcal{F}_{\text{Packaging}}\}) \rightarrow \lambda_{\text{BPolicy}}\)

5.8.2.3 Compliance Policy Violation

At runtime, one or more compliance regulations can be violated ‘\(\lambda_{\text{Regulation}}\)’, which results in the compliance policy violation ‘\(\lambda_{\text{CPolicy}}\)’.

Proposition 11. The violation of regulations causes compliance policy violation. It can formally be defined as follows,

\(\lambda_{\text{Regulation}} \rightarrow \lambda_{\text{CPolicy}}\)

5.8.2.4 Security Policy Violation

A security policy violation ‘\(\lambda_{\text{SPolicy}}\)’ is a set of conjunctive or disjunctive faults including confidentiality breaching fault ‘\(\mathcal{F}_{\text{CBreaching}}\)’, repudiation fault ‘\(\mathcal{F}_{\text{Repudiation}}\)’, intelligibility fault ‘\(\mathcal{F}_{\text{Intelligibility}}\)’, and a tamper resistance fault ‘\(\mathcal{F}_{\text{TamperR}}\)’. It can formally be written as follows,

Proposition 12. \((\mathcal{F}_{\text{CBreaching}} \lor \mathcal{F}_{\text{Repudiation}} \lor \mathcal{F}_{\text{Intelligibility}} \lor \mathcal{F}_{\text{TamperR}}) \lor (\mathcal{F}_{\text{CBreaching}} \land \mathcal{F}_{\text{Repudiation}} \land \mathcal{F}_{\text{Intelligibility}} \land \mathcal{F}_{\text{Temporal}}) \rightarrow \lambda_{\text{SPolicy}}\)

Proposition 13. A security policy violation is a Cartesian product, \((\mathcal{F}_{\text{CBreaching}} \times \mathcal{F}_{\text{Repudiation}} \times \mathcal{F}_{\text{Intelligibility}} \times \mathcal{F}_{\text{TamperR}}) \rightarrow (\{\mathcal{F}_{\text{CBreaching}} \land \mathcal{F}_{\text{Repudiation}}\} \lor \{\mathcal{F}_{\text{CBreaching}} \land \mathcal{F}_{\text{Intelligibility}}\} \lor \{\mathcal{F}_{\text{CBreaching}} \land \mathcal{F}_{\text{TamperR}}\} \lor \{\mathcal{F}_{\text{Repudiation}} \land \mathcal{F}_{\text{Intelligibility}}\} \lor \{\mathcal{F}_{\text{Repudiation}} \land \mathcal{F}_{\text{TamperR}}\} \lor \{\mathcal{F}_{\text{Intelligibility}} \land \mathcal{F}_{\text{TamperR}}\}) \rightarrow \lambda_{\text{SPolicy}}\)

5.8.2.5 Product Quality Violation

A product quality violation ‘\(\lambda_{\text{PQuality}}\)’ is the occurrence of a set of conjunctive or disjunctive faults including the Damaged Goods ‘\(\mathcal{F}_{\text{DGoods}}\)’ or Quality Mismatches ‘\(\mathcal{F}_{\text{QualM}}\)’ or both. The formal definition is given in the following:
Proposition 14. \((\mathcal{F}_{\text{Goods}} \lor \mathcal{F}_{\text{QualM}}) \lor (\mathcal{F}_{\text{Goods}} \land \mathcal{F}_{\text{QualM}}) \rightarrow \lambda_{\text{PQuality}}\)

5.8.2.6 Uncertain Events

Uncertain events result in business policy violations. The business policy fault assertions in this case are composed of uncertain events including disappearance of goods ‘\(\mathcal{F}_{\text{GDisapp}}\)’, workforce strike ‘\(\mathcal{F}_{\text{WorkST}}\)’, transportation of strike ‘\(\mathcal{F}_{\text{TransST}}\)’, and Act of God ‘\(\mathcal{F}_{\text{ActG}}\)’. The formal definition of the uncertain events is given below:

Proposition 15. \((\mathcal{F}_{\text{GDisapp}} \lor \mathcal{F}_{\text{WorkST}} \lor \mathcal{F}_{\text{TransST}} \lor \mathcal{F}_{\text{ActG}}) \rightarrow \lambda_{\text{BPolicy}} | (\mathcal{F}_{\text{GDisapp}}, \mathcal{F}_{\text{WorkST}}, \mathcal{F}_{\text{TransST}}, \mathcal{F}_{\text{ActG}}) \in \mathcal{E}_{\text{Uncertain}}\)

5.8.2.7 Compound Business Related Faults

A compound fault refers to a combination of two or more different types of faults. Each combination is a set of conjunctive faults. The compound faults cause a violation of an agreement that leads to a business transaction abortion. The possible compound faults are defined below:

Proposition 16. \(\forall \tau, (((\mathcal{U}_{\text{Resource}} \land \lambda_{\text{BPolicy}}) \rightarrow \lambda(\Sigma)) \rightarrow \zeta(\tau)), \text{if the ordered goods are not available and one or many business policies are violated, then the trading partner agreement is violated which results in the business transaction abortion.}\)

Proposition 17. \(\forall \tau, (((\mathcal{U}_{\text{Resource}} \land \lambda_{\text{CPolicy}}) \rightarrow \lambda(\Sigma)) \rightarrow \zeta(\tau)), \text{if the ordered goods are not available and one or many compliance policies are violated, then the trading partner agreement is violated which results in the business transaction abortion.}\)

Proposition 18. \(\forall \tau, (((\mathcal{U}_{\text{Resource}} \land \lambda_{\text{SPolicy}}) \rightarrow \lambda(\Sigma)) \rightarrow \zeta(\tau)), \text{if the ordered goods are not available and one or more security policies are violated, then the trading partner agreement is violated which results in the business transaction abortion.}\)

Proposition 19. \(\forall \tau, (((\lambda_{\text{BPolicy}} \land \lambda_{\text{SPolicy}} \land \lambda_{\text{CPolicy}} \land \lambda_{\text{PQuality}}) \rightarrow \lambda(\Sigma)) \rightarrow \zeta(\tau)), \text{the business policies, the compliance policies, the security policies, and the product quality violations result in the violation of an agreement which results the business transaction abortion.}\)

It is worth noting that the resource unavailability and the product quality violation may never occur in the same business transaction of a running process instance. If for a business transaction, the ordered goods are unavailable then the business transaction can be aborted. In such cases, the product quality violation can never happen.
5.8. FORMAL REPRESENTATION OF BUSINESS TRANSACTION FAULTS

5.8.3 Formal Representation of Business Protocol Related Faults

A business protocol related fault is a set of conjunctive or disjunctive faults including business function execution sequence fault \( F_{BFESeQ} \) and messaging sequence fault \( F_{MSeQ} \). The formal definition of business protocol related fault is given below:

**Proposition 20.** \( \forall \tau, ((F_{BFESeQ} \lor F_{MSeQ}) \lor (F_{BFESeQ} \land F_{MSeQ})) \rightarrow F_{BProtocol} \)

The business function execution sequence fault may result in a message sequence fault, which can be formally written as follows, \( F_{BFESeQ} \rightarrow F_{MSeQ} \).

5.8.4 Formal Representation of BO Related Faults

The business object related fault \( F_{BO} \) is a set of conjunctive or disjunctive faults including incorrect data \( F_{IncorrD} \) and missing data \( F_{MissingD} \). These two faults result in the data fault \( F_{Data} \) that cause a violation of an agreement which may force a business transaction abortion. The formal definition of the data fault is given below:

**Proposition 21.** \( \forall \tau, ((F_{IncorrD} \lor F_{MissingD}) \lor (F_{IncorrD} \land F_{MissingD})) \rightarrow F_{Data} \)

5.8.5 Formal Representation of Message Related Faults

A message related fault is a set of conjunctive or disjunctive faults including message format fault \( F_{MessageF} \), data connection fault \( F_{Connection} \), and response fault \( F_{Response} \). The formal expression of a message related fault is given below.

**Proposition 22.** \( \forall \tau, ((F_{MessageF} \lor F_{Connection} \lor F_{Response}) \lor (F_{MessageF} \land F_{Connection} \land F_{Response})) \rightarrow F_{Message} \)

A message related fault is a Cartesian product. Below is the formal definition of the Cartesian product of the Message Related Fault:

**Proposition 23.** \( \forall \tau, (F_{MessageF} \times F_{Connection} \times F_{Response}) \lor (F_{MessageF} \land F_{Connection} \land F_{Response}) \rightarrow F_{Message} \)

5.8.6 Formal Representation of Application Related Faults

For a running business process instance a business transaction can be aborted due to one or more application related faults including software integrity \( F_{SIntegrity} \), processing delay \( F_{ProcessingD} \), and unavailability of software \( F_{USoftware} \). The application related fault is formally defined below:

**Proposition 24.** \( \forall \tau, ((F_{SIntegrity} \lor F_{ProcessingD} \lor F_{USoftware}) \lor (F_{SIntegrity} \land F_{ProcessingD} \land F_{USoftware})) \rightarrow F_{Application} \)
The Cartesian product of the application related fault is defined in the following:

**Proposition 25.** \( \forall \tau, (\mathcal{F}_{\text{Integrity}} \times \mathcal{F}_{\text{ProcessingD}} \times \mathcal{F}_{\text{Software}}) \rightarrow (\{\mathcal{F}_{\text{Integrity}} \wedge \mathcal{F}_{\text{ProcessingD}} \} \vee \{\mathcal{F}_{\text{Integrity}} \wedge \mathcal{F}_{\text{Software}} \} \vee \{\mathcal{F}_{\text{ProcessingD}} \wedge \mathcal{F}_{\text{Software}} \}) \rightarrow \mathcal{F}_{\text{Application}} \)

Software unavailability can be caused by **Component Crash** or **Traffic Overload** or both such that,

\((\text{Components Cash} \lor \text{Traffic Overload}) \lor (\text{Components Cash} \land \text{Traffic Overload}) \rightarrow \mathcal{U}_{\text{Software}}\)

### 5.8.7 Fault Chain in Business Transaction

A fault may create a chain of faults during business transactions. In end-to-end business processes, the process spaces may contain activities that are dependent on other activities. In such a process space, a fault may create the **Ripple Effect** (Sue, 2001). The ‘Ripple Effect’ creates a chain of faults in a process space, which affect business transactions severely, in particular, it results in business transaction abortion if the business transaction systems fail to invoke any failure prevention mechanism. However, not every fault generates fault chain. Figure 5.7 shows the business transaction fault chain.

The fault chains can formally be defined as follows:

**Proposition 26.** \( \forall \tau, (\mathcal{U}_{\text{Uncertain}} \rightarrow \mathcal{U}_{\text{Resource}} ) \rightarrow \mathcal{F}_{\text{Temporal}} ) \rightarrow \lambda_{\text{BPolicy}} \rightarrow \lambda (\Sigma). \) For all business transactions, the occurrence of an uncertain event results in the unavailability of resources that results in a temporal fault that leads to a business policy violation. The outcome of business policy violation is the violation of agreement.

**Proposition 27.** \( \forall \tau, (\lambda_{\text{PQuality}} \rightarrow \mathcal{F}_{\text{Temporal}} ) \rightarrow \lambda_{\text{BPolicy}} \rightarrow \lambda (\Sigma). \) For all business transactions, product quality violation results in a temporal fault which effects a business policy violation that results in the violation of an agreement.

**Proposition 28.** \( \forall \tau, (\mathcal{F}_{\text{MessageF}} \rightarrow \mathcal{F}_{\text{Ordering}} ) \rightarrow \lambda_{\text{BPolicy}} \rightarrow \lambda (\Sigma). \) For all business transactions, the outcome of a message format fault is an ordering fault that results in a business policy violation that leads to the violation of an agreement.

**Proposition 29.** \( \forall \tau, (\mathcal{F}_{\text{ProcessingD}} \rightarrow \mathcal{F}_{\text{Temporal}} ) \rightarrow \lambda_{\text{BPolicy}} \rightarrow \lambda (\Sigma). \) For all business transactions, the outcome of delaying a request for processing results in the a temporal fault that leads to a business policy violation which results in a violation of agreement.

**Proposition 30.** \( \forall \tau, (\mathcal{U}_{\text{Software}} \rightarrow \mathcal{F}_{\text{ProcessingD}} ) \rightarrow \mathcal{F}_{\text{Temporal}} ) \rightarrow \lambda_{\text{BPolicy}} \rightarrow \lambda (\Sigma). \) For all business transactions, the unavailability of a software component or a software application results in a delay in processing a request placed by a business partner which results in a temporal fault that leads to a business policy violation which leads to the violation of an agreement.
5.9. SUMMARY

Figure 5.7: The Business Transaction Fault Chain.

Proposition 31. \( \forall \tau, (((F_{\text{Data}} \lor F_{\text{Connection}}) \rightarrow F_{\text{ProcessingD}}) \rightarrow \lambda_{\text{BPolicy}}) \rightarrow \lambda (\Sigma).) \). For all business transactions, a data fault or a connection fault results in processing delay which results in the temporal fault that results in a business policy violation that leads to an agreement violation.

Proposition 32. \( \forall \tau, (((F_{\text{Response}} \rightarrow F_{\text{ProcessingD}}) \rightarrow F_{\text{Temporal}}) \rightarrow \lambda_{\text{BPolicy}}) \rightarrow \lambda (\Sigma).) \). For all business transactions, if a response fault occurs then processing delay happens which leads to the temporal fault that results in a business policy violation which results in the violation of an agreement.

5.9 Summary

Fault is an important concept of this research. Understanding the nature of business transaction faults and their consequences are of paramount importance to define efficient failure prevention techniques. In this chapter, I discussed a wide variety of faults related to the end-to-end business processes. I explained how I constructed the fault models which are essentially the causal trees that consists of faults.
Also, in this chapter I formalized the fault models. I provided the formal semantics of each fault using propositional logic language. The formal definitions of the faults have two main advantages: (i) these will enable the business transaction systems to detect the faults automatically at business transaction runtime and (ii) these will help the decision support systems to select the best recovery method to prevent a failure upon the occurrence of these faults.

Additionally, I explained the 'Ripple Effect' of the faults. I provided a comprehensive description of how a fault can produce a chain of faults that can be propagated through the business process space. Also, I provided the formal definition of the fault chains.
Chapter 6

Flexible Business Transaction – Concepts and Characteristics

6.1 Introduction

Flexible behaviour of business transactions is one of the two core focuses of this research. The term flexible behaviour can be ambiguous to the readers, especially the semantics of ‘behaviour’ in the context of business transaction. Therefore, in this chapter, I provide a comprehensive overview of business transaction behaviour. I summarise different types of transaction behaviours documented in the literature.

In addition, I explain how flexible behaviour has been viewed in this research and the main drivers of highly flexible business transactions.

6.2 A Brief Overview of Business Transactions Behavior

Chrysanthis and Ramamritham (1994) view transaction behavior purely from the operational perspective. The authors pointed out that while performing operations, application components interact each other. The interactions are part of a business transaction and exhibit its behavior. According to Sun and Yang (2007), besides the operational states of a business transaction, the state of the flow (including control flow and message flow) between the transaction nodes exhibit business transaction behavior. They found the behavior of business partners influences business transaction behavior in a collaborative business environment. Zdravkovic and Ahlen (2004) view transaction behavior as the execution order (defined by control flow) of transactional activities. Additionally, according to the authors the conditions influence business transaction behavior. Conditions are the logical statements which are specified to constraining the enactment of transactional activities. Schuldt et al. (2002) define activity behavior as transaction behavior.

Summing up these views, the business transaction behaviour covers different aspects, including states of the operation, the message flow, the control flow, and the state of agreement between the trading partners. In this thesis, I specially focus on the state of agreement, satisfied and not satisfied, which represent consistent and inconsistent business transactions. Because, as mentioned several times in the previous chapters, agreement is the most critical entity of business transactions.
6.2.1 Types of Business Transaction Behavior

Two main types of business transaction behavior found in the literature are *atomic behavior* (Haerder and Reuter, 1983), (Papazoglou and Kratz, 2006), (Dayal et al., 1990), (Butler et al., 2005), (Leyman and Reuter, 2000), (Seth, et al., 1997), (Kuo et al., 1996) and relaxed atomic behavior. The latter is an extension of atomic behavior found in different works including Traiger et al., (1982), Bukhres et al., (1992), Hector and Salem, (1987), Chrysanthis and Krithi, (1990). This thesis proposes a new behavioural model *flexible behaviour*. The semantics of these behaviors are described below:

A. Atomic Behavior: Atomic behavior refers to the strict behavior of a business transaction. It has been defined in many ways in the literature, however, the foundation of these definitions is the same. A formal definition of atomic behavior is given below. The definition is taken from Leyman and Reuter (2000).

**Definition 1.** Let ‘*t*’ be a (sub) transaction and *t*<i>k</i> be the restriction of ‘*t*’ at sites *k* = 1, 2, ..., *n*. All restrictions of a (sub) transaction *t* at sites *k* = 1, 2, ..., *n* should be aborted if *t* aborts.

An abortion of ‘*t*’ forces the abort of each sub-transaction contained within the scope of ‘*t*’. Two Phase Commit (2PC) (Bernstein and Newcomer, 1987) is the protocol used for implementing the atomic behavior of transactions.

B. Relaxed Atomic Behavior: Relaxed atomic behavior is essentially partial flexible behavior. Various techniques, such as, compensation (Traiger et al., 1982), (Hector and Salem, 1987), (Leyman Reuter, 2000), forward recovery (Hector and Salem, 1987), and function replication (Kühn et al., 1995) have been proposed to design relaxed atomic behavior of transactions. The following examples depict the relaxed atomic behavior of transactions.

- **Relaxed atomic behavior using compensation function:** Compensation is a widely used approach for relaxing atomicity in business transactions. Leymann and Reuter define the compensation function as follows, if *T*<sub>1</sub>, ..., *T*<sub>n</sub> and *C*<sub>1</sub>, ..., *C*<sub>n</sub> are two sets of transactions in a business transaction, then *C*<sub>i</sub> is called the compensation function of *T*<sub>i</sub>, 1 ≤ *i* ≤ *n*. Each subtransaction ‘*T*<sub>i</sub>’ is associated with a compensation function which is invoked upon failure of a subtransaction. Specifically, if a subtransaction fails then the corresponding compensation function performs the logical undo of the completed transactions.

- **Relaxed atomic behavior using forward recovery:** Garcia-Molina and Salem (1987) proposed a technique called *savepoint* that is used in forward recovery of business transactions. From a technical point of view, a savepoint is a powerful technique. It denotes milestones of a transaction. When a failure occurs, the corresponding compensation function executes the business transaction to the
nearest savepoint in a reverse order and restarts the transaction from that savepoint. This prevents
restarting the entire transaction from the beginning.

C. Flexible Behavior: Strict atomicity can be relaxed to some extent by using compensation and forward
recovery. This is not sufficient for business transactions of a complex large-scale end-to-end business pro-
cess. This research introduces flexible behavior which is an extension of relaxed atomic behavior to achieve
greater flexibility in terms of business transaction behavior. The semantics of flexible behavior is explained
in the following,

Let ‘τ’ be the business transaction which includes the transactional activities α₁,...,αₙ within its
scope, the flexible behavior of business transaction denotes,

• business transaction aborts eventually, that is, the transaction abortion happens if and only if is not
  recovered eventually

• tolerance of faults

• operations are allowed to change their paths

• transaction nodes can be added when necessary

• operations are allowed to commit locally

A high-level flexible business transaction should demonstrates these behaviour at runtime.

6.3 Characteristics of Flexible Business Transaction

The definition of ‘flexible behaviour’ given in the previous section outlines five characteristics of flexible
business transactions. The list of characteristics are given below:

• it is able to tolerate faults (Gray and Reuter, 1993)
• it is failure-resilient (Nodine and Zdonik, 1992)
• it is non-deterministic (Papadimitriou and Yannakakis, 1993)
• the cohorts of business transaction can be performed autonomously by the applications hosted at
different sites of different trading partners (Dayal et al., 1990)
• it is extensible (Miscalef, 1987).

These characteristics are the basic building blocks of flexible business transactions. In the following
subsections, I provide a detailed description of these characteristics.
6.3.1 Fault-Tolerance

Fault-tolerance refers to the ability of a business transaction system to continue a business transaction upon occurrence of faults. It is a well-known concept that has been discussed a great deal in the literature. However, in the literature this characteristic has been viewed from the application and system level perspectives. For instance, Grey and Reuters explained how to develop a fault-tolerant database application. Little, (2002) highlights the role of the ACID principle in developing a fault-tolerant transaction system that involves multiple remote resources. Singh et al. (2009) propose the Zeno protocol for developing cloud-based byzantine fault-tolerant database applications (the term Byzantine fault stems from the Byzantine General’s Problem (Lamport et. al., 1982)). While studying the AVVERS OEM supply chain scenario, I discovered that business transactions of end-to-end business processes should be fault tolerant. It is a critical characteristic which drives flexible business transactions. This characteristic enables a business transaction system to continue transactions upon occurrence of one or more faults at runtime.

Example. I found some activities in the AVVERS purchase order fulfillment business process are less significant from logical point of view. I discovered that if certain faults happen with such activities, it does not affect the business transaction, and therefore, such faults can be ignored. For instance, if AVVERS OEM does not send a purchase order acknowledgment to Auto Inc. within the agreed time then it is a fault. This fault can be tolerated because it does not affect the subsequent business operations. The fault-tolerance property enables business transaction systems to ignore such faults.

6.3.2 Failure-Resilience

There are faults that results in business transaction failures. For instance, the occurrence of uncertain events such as a labor strike may result in a business transaction abortion. Failure-resilience refers to the ability of a business transaction system to recover business transactions from failure or abortion. This characteristic enables business transaction systems to repair failure instead of aborting business transactions. It is a critical characteristic for defining highly flexible business transactions.

6.3.3 Non-Determinism

Non-determinism denotes that the executions of a transactional activity are not predetermined, rather a set of paths is given for that activity, and a path is selected based on the condition(s). It has been used in other areas of the computer science domain. For instance, it has been used in building a non-deterministic Turing machine (Papadimitriou and Yannakakis, 1993), and in designing non-deterministic algorithms (Cormen et al., 2001). Also, it has been used in workflow based applications and transactional workflow (Kuo et al., 1996).

In the AVVERS OEM supply chain case, AVVERS OEM has collaborates with Transportation Inc. and
Shipping Limited. In the shipment transaction two execution paths are specified for the activity ‘Select Shipper’. The first path is ‘Select Transportation Inc.’ and the second one is ‘Select Shipping Limited’. At runtime, either of these two paths is selected mutually exclusively based on the transition conditions.

6.3.4 Operational Autonomy

Operational autonomy that denotes each partner application performs transactional activities independently, without being controlled by a central controller. A central controller reduces the performance of business transactions because the partner applications cannot perform actions without receiving instructions from the central controller. In addition, failure recovery can be delayed in a centralized system because the local transaction controllers are not allowed to make any logical decisions. Therefore, operational autonomy is an important characteristic. It enables building distributed business transaction systems without central controller. Consequently, local transactions becomes fully autonomous.

Considering the AVERS OEM supply chain case, the purchase order fulfillment business transaction involves the applications of the partners including Auto Inc., AVERS OEM, Suppliers, Transportation Inc., Shipping Limited, and the Bank. These applications perform the transactional activities contained in transaction modules such as ‘purchase order registration module’, ‘credit verification module’, ‘process shipment module’ etc... The operational autonomy in such a distributed transaction system enables the distributed control of transactional activities. Unlike a central controller, in distributed control, the local controllers are allowed to make decisions in terms of the local business transactions.

6.3.5 Extensible

Extensible denotes the ability of a business transaction system to extend its structure if it is required. This characteristic enables business transactions to span the core structure by plugging the new transaction functions or modules into it. An extensible business transaction is capable of dealing with various faults such as ‘resource unavailability’ and ‘shipment fault’ efficiently.

Service oriented businesses are usually customer–faced. In today’s competitive business environment, meeting customer demand is a dominant requirement (Simchi-Levi, 2008). Therefore, in some situations, the organization may need to extend existing collaboration at runtime by adding new partners. As currently business operations are performed electronically in most organizations, an extension of business collaboration at runtime needs technical support. Specifically, the running business transaction system should be extensible so that it can adopt evolved collaboration with a new partner dynamically at runtime.

Taking the AVERS OEM supply chain case into account, if the company predicts a potential failure in the shipment processing transaction module in particular, it cancels the shipment agreement with a shipper. If this failure happens then a business transaction abortion will result. The extensible business transaction
systems are able to prevent such abortions. The business transaction system will extend the structure of its partner network and include a new shipper.

6.4 Summary

This chapter provided a comprehensive description of the flexible behaviour of business transactions. Different types of business transaction behaviours were discussed. The fundamental characteristics of flexible business transactions were described. The characteristics described in this chapter are the basic building blocks of flexible business transactions. These characteristics enable building highly flexible business transaction systems.
7.1 Introduction

In this chapter, I describe how Solúbtha BTM was constructed. I constructed three versions of the model: the Initial Version, Solúbtha BTM V1, and Solúbtha BTM. The Initial Version is the foundation model. Solúbtha BTM V1 was built on top of the Initial Version. The Initial Version had to be modified to adapt comments made by the experts and the outcomes of the analysis of secondary data of Auto Inc. business case. The final version Solúbtha BTM was developed by adapting the outcomes of the AVERS OEM and Walmart case studies. Additionally, the results of trading partner agreements were adopted in this version.

This chapter presents the final version. The Initial Version and Solúbtha BTM V1 are provided in Appendix D and Appendix E. This chapter describes how various design-elements were logically assembled into a business transaction model.

The development of the model started with building the Initial Version. Then, an interim evaluation was performed and the model was modified to adopt the suggestions and opinions of experts. The next version is Solúbtha BTM V1 which was constructed by incorporating the elements identified through studying secondary data of the Auto Inc. business case. I constructed the final version by including the design-elements elicited by analysing secondary data of the AVERS OEM and Walmart business cases and from various trading partner agreements.

I used the Unified Modeling Language 2.0 (UML 2.0) (OMG, 2005) for presenting each version of Solúbtha BTM. The key reason behind using UML is that it is a semi-formal graphical language. It provides various formal notations to build models and enables one to analyse and evaluate them from a logical or syntactic point of view. A brief description of the UML 2.0 is provided in Appendix C.

7.2 Solúbtha BTM

Developing Solúbtha BTM was relatively more challenging than the previous versions. The key challenge was to define appropriate design-elements which enable users to specify the recovery logic for preventing failures or abortions of business transactions. Another challenge was to simplify the model by providing all necessary and sufficient design-elements to design business transaction instance-models from the business
and functional perspectives.

Solúbtha BTM was built on the top of Solúbtha BTM V1 to adopt the results produced through studying secondary data the business cases and trading partner agreements. Solúbtha BTM V1 was modified to adopt the new results. Appendix F (Page 295) describes the operations performed with design-elements during modification. New design-elements were elicited from AVS OEM supply chain and the ‘Walmart changes tactics to meet international taste’ business cases. A few design-elements were included which were found through studying trading partner agreements. The newly identified design-elements are described in Appendix G. In addition, I added operators, attributes, and a list of techniques for designing flexible business transactions.

Solúbtha BTM consists of six core packages: Agreement, Transaction Role, Transaction Process Fragment, Business Protocol, Transaction Recovery, and Business Transaction. Figure 7.1 shows the packages of Solúbtha BTM.

The root package of the model ‘Business Transaction’ has a logical relation with the other packages. The relation is defined using a rolename called dependency (the dotted arrow lines shown in the figure represent the dependency association). This dependency association indicates that the ‘Business Transaction’ package is dependent on these other packages. In addition, the Transaction Role and Business Protocol, and Transaction Process Fragment packages have dependency relations with the ‘Agreement’ package. The packages are detailed in the next sub-sections.

![Figure 7.1: The Core Packages of Solúbtha BTM](image_url)
This is a business perspective package. The main design-element contained in this package is ‘Agreement’ which is a method for establishing collaboration between business partners. It binds business partners through mutual understanding on obligations, prohibitions, penalties, etc., which control business operations. It is worth noting that these obligations and prohibitions are better known as terms and conditions or clauses. An agreement governs the external and internal operations of the trading partners processes. It implies that the business transactions are governed by the agreements as well. This is why in this thesis ‘Agreement’ is considered as a critical design-element of the Solúbtha BTM.

Technically speaking, an agreement is a script that contains clauses in plain text format. A formal approach is needed to specify agreement in a business transaction model because plain texts are unstructured data and cannot be parsed by the system. The design-element ‘Agreement’ of Solúbtha BTM provides a formal structure to insert the contents of agreements into the system as sets of inputs which then transformed into machine instructions and controls when business transaction instances are running. There are examples of XML-based technologies, e.g., WS-Agreement (OGF, 2007) developed by the Open Grid Forum ¹ and WSLA (Keller and Ludwig, 2003) developed by IBM. These focus on service contract which is essentially a subset of business transactions with some special syntax and vocabularies. In addition, some contract vendors such as eContract.be² offer electronic contracts, however, these are merely for negotiation between trading partners. Controlling operations in real-time is outside the scope of such technologies. The design-element ‘Agreement’ provided by Solúbtha BTM has an explicit relation with functional operations and, therefore it enables specifying agreement clauses inside the business functions as a logical structure. In consequence, at runtime, these operations are governed by the agreement.

I found in Auto Inc. and AVERS OEM that these companies do not provide logistics support to their buyers. Rather, they collaborate with third party logistics providers who ship ordered goods to the buyers. This means that they have separate agreements with the logistics providers. In other words, these companies have at least two different agreement scripts into their systems. one with their buyer and another with their logistics providers. This can be the case for each of their business partners. There is another approach which I found in the Walmart case. Walmart Inc. provides logistics service to their buyers. The company has own carrier, Walmart–Logistics³. In such cases, one agreement suffices to cover two business operations: purchase and delivery. Considering both scenarios, a reflexive association is defined using the aggregation relation. Figure 7.2 depicts the ‘Agreement’ and its parts.

The reflexive association denotes that an agreement can be composed of more than one agreement. The cardinality ‘1’ to ‘0..n’ associated with the aggregation relation indicates the number of instances an

¹https://www.ogf.org/ogf/doku.php
²https://www.e-contract.be/
‘Agreement’ allows in composition. If an agreement is composed of more than one agreement then it is called a composite agreement, otherwise it is a simple agreement.

Agreements can be short-term and also it can be long-term. For instance, Auto Inc. has short-term logistics providers who provide delivery services for a limited period of time. On the other hand AVERS OEM has two long-term logistics service providers. Considering both cases, two types of agreement: LTA (Long Term Agreement) and STA (Short Term Agreement) were incorporated into Solúbtha BTM. The model enables specifying these types as ‘data type’ of ‘Agreement’ class. The types are listed in an enumeration class called ‘AgType’ (shown in Figure 7.2). At design-time, since the value of the type attribute, the users can assign them.

The ‘Agreement’ consists of three parts: Period, Policy, and Party. The identifier of the ‘Agreement’ is a token that denotes a unique identity of an agreement. ‘String’ is a data type that allows assigning an alphanumeric (combination of letters and numbers) value to an identifier. The unique identifier of agreements is used as the reference key in transaction processing systems.

The agreement package contains three operators. The GOperator (Generic Operator) which contains a list of operators to specify the operations that are performed on agreements at runtime. Essentially, these operators enable customizing the agreement (e.g. adding clauses) while the business transaction is running. The agreement package is summarized in Table 7.1.
### Table 7.1: Summary of the Agreement Design-element of Solúbtha BTM

**Package: Agreement**

**Usage of the Agreement**

It will enable users to specify the parties that are involved in the agreement. Additionally, the users can specify the period and polices. The policies are the terms and conditions which are agreed on by the trading partners. The period is the lifetime of an agreement.

<table>
<thead>
<tr>
<th>Associated Concept</th>
<th>Description</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>An agreement should contain at least one policy but it can contain as many (must be a finite number) policies as users want. While investigating the trading partner agreements, I found that the agreements contain several terms and conditions. Therefore, the maximum cardinality is 1..*. This indicates that a transaction designer is allowed to specify many polices.</td>
<td>1 - 1..*</td>
</tr>
<tr>
<td>Party</td>
<td>The multiplicity indicates that there must be at least two parties involved in an agreement. It allows 'n’ number of partners in an agreement.</td>
<td>1 – 2..n</td>
</tr>
</tbody>
</table>

**GOperator**

GOperator refers to generic operators. These operators are not bound to any specific design-element of Solúbtha BTM. They can be shared by other elements. The GOperator enables specifying insertion, deletion, and modification operations.

<table>
<thead>
<tr>
<th>GOperator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>Is is used for inserting of new clauses into agreements.</td>
</tr>
<tr>
<td>Delete</td>
<td>It is used for removing clauses from agreements.</td>
</tr>
<tr>
<td>Modify</td>
<td>It is used for modifying agreements.</td>
</tr>
</tbody>
</table>
In some cases, users may use custom operators to define some specialized operations in their business transactions. The CustomGOperator was included in Solúbtha BTM to allow users to define custom operators. The notion of the user defined operators was adopted from the programming language C++ (Stroustrup, 1997).

The three parts of agreement are described in the next subsections.

### 7.2.1 Party

‘Party’ refers to trading partners. It is included in Solúbtha BTM to enable user to specify the information of the trading partners involved in an agreement. Figure 7.3 depicts the element ‘Party’. The Identifier is an attribute of ‘Party’. The data type String is for assigning alphanumeric values of Identifier. The trading partners are referenced in business transactions by the Identifier. Contact is the other attribute of the ‘Party’. The ConInfo is a data type used for assigning the values of the attribute Contact. It is a CompositeType attribute which has a set of members including Address, Phone, email, and Website. The CustomInfo attribute supports specifying custom information of business partners.

### 7.2.2 Period

‘Period’ refers to a range of time. An agreement is valid for a certain period. The ‘Period’ enables users to specify the range of time which is valid for an agreement. The ‘Period’ contains the attributes startDate and endDate. The data type Date allows one to assign the values of these attributes. Figure 7.2 depicts the ‘Period’.
7.2. SOLÚBTHA BTM

7.2.1.3 Policy

Policies are critical to business transactions (Papazoglou et al., 2010). Typically, in agreements the policies are written in the form of clauses using natural language. I explained previously in this section why agreements cannot be parsed into machine code. A formal data structure for these policies is a paramount requirement to automate the enforcement of agreement clauses on running business transactions. However, structuring clauses in a formal manner is non-trivial. I addressed this difficulty in Solúbtha BTM by providing the design-element ‘Policy’, which enables specifying the agreement clauses in a machine-readable format.

‘Policy’ is refined to three subtypes: BusinessPolicy, CompliancePolicy, and SecurityPolicy. Figure 7.4 depicts the ‘Policy’. The subtypes are related to the ‘Policy’ using the generalization relation because of the inheritance property between them. The subtypes are described below.

7.2.1.4 BusinessPolicy

The business policy is a plan, or a course of action that influences and causes decisions, and actions which concern an enterprise in a way that guarantees that the enterprise operates within specified guidelines and obeys regulations (Papazoglou et al., 2009). The business policies enable efficient management of an organization by defining the procedures and rules for its daily business operations (Peltier, 2004). In Solúbtha BTM, the ‘BusinessPolicy’ is a subtype of the element ‘Policy’. It enables specifying the clauses contained in agreements. In order to structure the clauses in a formal manner, I added an element called Business-Rules. The business clauses are structured in Solúbtha BTM as business rules. The data structure is defined as follows: the ‘Policy’ has subtype ‘BusinessPolicy’ which has a part ‘BusinessRules’.

In Figure 7.4, this is an aggregation relation between ‘BusinessPolicy’ and ‘BusinessRules’. Hay (1999) defines business rules as follows: business rules are derived from the business policies. Taking this definition into account, I added an association relation between ‘Policy’ and ‘BusinessRules’ with the rolename derivedFrom. This means the business rules are derived from the policies. ‘BusinessRule’ essentially enables specifying the business policies in business transaction instance-models. A business rule can be a hard or soft type. The notion of hard and soft rules is borrowed from the notion of hard and soft constraints (Domshlak et al., 2006). These types are described in Table 7.2.

Additionally, a business rule can be composite or simple. The composite rules are composed of business rules whereas the simple rules are non-decomposable.
7.2.1.5 Compliance Policies

In my study I found that the trading partner agreements contain a paragraph which consists of the clauses related to the compliance regulations. Below are examples of compliance related clauses which were found in the trading partner agreements.

- “Seller guarantees that the products will fully comply with all applicable federal, state and local laws, ordinances and regulations” (ITW Global Automotive Agreement).

- “Seller warrants and represents that the goods or services furnished pursuant to the purchase order will not be manufactured, sold, priced or provided in violation of any federal, state or local law from which liability might accrue to Purchaser, and the Seller shall indemnify and hold purchaser harmless for any damages, costs and expenses suffered or incurred by Purchaser as a result of any failure to comply with such laws” (ArcelorMittal).

- “Seller shall comply with all applicable laws, rules, regulations, orders, conventions, or standards enacted by the United States of America that regulate the manufacture, labeling, transportation, licensing, approval or certification of products or services, including but not limited to, those relating to environmental matters, data protection and privacy, wages, hours and conditions of employment, subcontractor selection, discrimination, and occupational health/safety, and each purchase order shall be deemed to incorporate by reference all the clauses required by the provisions of said laws, Purchase Orders, rules, regulations and ordinances” (Tower International Contract).

I added an element called CompliancePolicy to structure these clauses in a formal manner. Figure 7.4 shows the design-element ‘CompliancePolicy’. It is a subtype of ‘Policy’. The ‘CompliancePolicy’ is composed of ‘ComplianceRules’ that can be composite or atomic. It enables users to specify the clauses that are related to compliance issues. The clauses will be defined as compliance rules. The compliance rules must be satisfied during business transactions at runtime. These rules are typically the hard rules, however, the type of the rules always depend the compliance requirements and contexts.

Identifying the operators and variables for specifying the compliance policies of a business transaction was a challenge. From the study of the trading partner agreements, I discovered the operator called SegregationofDuty (SOD) which was included in Solúbtha BTM. Additionally, I added a design-element called CustomControlOperator that allows the users to define their own operators for specifying compliance rules in business transaction models. Furthermore, I added an element ControlVariable to enable users to specify
the variables that suit the target regulatory framework.

7.2.1.6 Security Policy

The agreements contain clauses related to data protection, the privacy act, and other security issues. I discovered a list of security parameters from the trading partner agreements. I adopted some security parameters from the UMM. The parameters I found in my study are ConfidentialityRequired, IntelligibilityRequired, TamperProofRequired, Non-RepudiationRequired, AuthorizationRequired, AuthenticationRequired, and IPRSecurityRequired. These parameters are included in Solúbtha BTM as *crispy constraints* with Boolean data type. Crispy constraints are essentially Boolean constraints. I added an element called *SecurityParameter* to enable users to specify these security parameters in their business transaction instance-models. The data structure of SecurityPolicy is defined as follows. The design-element ‘Policy’ has the part ‘SecurityPolicy’ which is composed of SecurityParameter.

Furthermore, the users need operators and variables to specify the business rules, the compliance rules, and the security parameters. Solúbtha BTM provides a list of built-in operators and *reserved identifiers* to specify these rules in business transactions. The reserved identifiers are essentially the vocabularies which are needed to specify the rules. These operators and variables are summarized in Table 7.2.

<table>
<thead>
<tr>
<th>Table 7.2: The Table Summarizes the Policy Construct of Solúbtha BTM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operators</strong></td>
</tr>
<tr>
<td><strong>AOperators</strong></td>
</tr>
<tr>
<td>AOperator denotes an assignment operator which enables assigning the values of the variables.</td>
</tr>
<tr>
<td><strong>Equal</strong></td>
</tr>
<tr>
<td>This operator enables specifying the value of a variable.</td>
</tr>
<tr>
<td><strong>ROperator</strong></td>
</tr>
<tr>
<td>ROperator denotes <em>Relational Operators</em> which enables defining the relations between two or more variables. Four relational operators rEqual, NotEq, lessThan, and greaterThan are added in Solúbtha model.</td>
</tr>
<tr>
<td><strong>rEqual</strong></td>
</tr>
<tr>
<td>It denotes the relational equality of two operands (<em>e.g.</em>, variables, literals, objects etc.).</td>
</tr>
<tr>
<td><strong>NotEq</strong></td>
</tr>
<tr>
<td>It denotes the relational inequality between two operands.</td>
</tr>
<tr>
<td>Operator</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>lessThan</td>
</tr>
<tr>
<td>greaterThan</td>
</tr>
<tr>
<td>LOperator</td>
</tr>
<tr>
<td>AND</td>
</tr>
<tr>
<td>OR</td>
</tr>
<tr>
<td>EOR</td>
</tr>
<tr>
<td>NOR</td>
</tr>
<tr>
<td>NOT</td>
</tr>
<tr>
<td>IFF</td>
</tr>
</tbody>
</table>

**Pattern Definition Identifiers**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIIdentifier</td>
<td>It refers to Temporal Reserved Identifier. The TRIIdentifier is used to specify the time related policies in business transaction models.</td>
</tr>
<tr>
<td>Within</td>
<td>It is used to specify durations of operations such as payment within ((time_1 \ldots time_2)).</td>
</tr>
<tr>
<td><strong>Before</strong></td>
<td>It is used to specify the temporal constraints that enforces an event or action to occur ‘before’ a specified time. It is used to specify the temporal policies.</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>After</strong></td>
<td>As opposed to ‘Before’, it is used to specify the time related statements where an operation occurs after a specified time.</td>
</tr>
<tr>
<td><strong>AtLeast</strong></td>
<td>It refers to the minimum number of times an operation must occur.</td>
</tr>
<tr>
<td><strong>AtMost</strong></td>
<td>As oppose to ‘AtLeast’, it refers to the maximum number of times to occurring an operation.</td>
</tr>
<tr>
<td><strong>AtMostOnce</strong></td>
<td>It refers to an operation must occur only once.</td>
</tr>
<tr>
<td><strong>AtLeastOnce</strong></td>
<td>It refers to the least number of times an operation must occur.</td>
</tr>
</tbody>
</table>

**CRIdentifier**

It denotes *Control Reserved Identifier*. It was added to specify compliance rules. An explicit connection between CRIdentifier and ComplianceRules was defined in the model (shown in Figure 7.4).

**SoD**

It refers to *Segregation of Duty*, it is a preventive control pattern found in the Elgamal *et al.*, (2011). It is an important mandate of Sarbanes Oxley. This identifier is used to specify the compliance policies regarding the duties of individuals who participate implicitly in business transactions.

**CustomContronIdentifier**

The CustomContronIdentifier was included in the model to enable the users to specify variables and compliance rules.

**RType**

It denotes *Rule Type*. It is a special *data type* used to specify the type of business, compliance, and security rules. The data type has two values *soft* and *hard*. 
### Soft
It denotes the weaker type rules. A violation of this type of rule associated with a sub-transaction does not affect other sub-transactions that are within the scope of a business transaction. More specifically, violation of a soft constraint can be tolerated. The Soft type is used to specify the soft types of business and compliance rules.

### Hard
Hard denotes strict rules. If hard rules failed to be satisfied, it may affect parts of or the entire business transactions. Essentially, it may lead to business transaction abortion if not repaired. The rule type ‘Hard’ enables to specifying hard types of business and compliance rules.

### UType
It refers to Unit Type, which is a unit of measurement. I added a list of unit types in Solúbtha BTM. They are: second, minute, day, month, year, percent, and CustomUType. These unit types are specified with business and compliance rules. The CustomUType allows specifying a user defined unit type.

### Variable

#### GVariable
GVariable refers to generic variables that are used in specifying the business, security, and compliance policies. Solúbtha BTM provides several GVariable.

#### NotificationRequired
It is used to specify the rules for notifying an event.

#### ASNRequired
It is used to specify the rules related to advanced shipment notification.

#### CertificateOfInsuranceRequired
It is used to specify the rules related to the certificate of insurance to ship a purchase order. The values of the attributes are Yes and No.

#### IsAvailabilityGuaranteeRequired
It is used to specify the rules related to guaranteeing of product availability.

#### IsPackagingGuaranteeQRequired
It is used to specify the rules related to guarantee the packaging quality.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMethod</td>
<td>It is used to specify the rules related to the method of payment.</td>
</tr>
<tr>
<td>PMeans</td>
<td>It is used to specify the rules related to the means of payment.</td>
</tr>
<tr>
<td>TMeans</td>
<td>It is used to specify the rules related to the transportation means.</td>
</tr>
<tr>
<td>PickLocation</td>
<td>It is used to specify the rules related to the location for picking-up the ordered goods.</td>
</tr>
<tr>
<td>SLocation</td>
<td>It is used to specify the rules related to the shipment locations.</td>
</tr>
<tr>
<td>SRoute</td>
<td>It is used to specify the rules related to the shipment routes.</td>
</tr>
<tr>
<td>DeliveredPQuality</td>
<td>It is used to specify the rules related to the qualities of the product that are delivered.</td>
</tr>
<tr>
<td>ExpectedPQuality</td>
<td>It is used to specify the rules related to product qualities expected by the buyers.</td>
</tr>
<tr>
<td>Penalty</td>
<td>It is used to specify the rules related to penalties.</td>
</tr>
<tr>
<td>FOB</td>
<td>It denotes <em>Free On Board</em>. It is used to specify the information of the parties who pay the shipment costs.</td>
</tr>
<tr>
<td>BillOfLading</td>
<td>It denotes Bill of Lading which is a document used in transporting goods. The BillOfLading operator is to specify an array of data items about the transportation of goods.</td>
</tr>
<tr>
<td>Invoice</td>
<td>It is used to specify the rules related to invoices.</td>
</tr>
<tr>
<td>CustomGVVariable</td>
<td>It is used to specify the user-defined variables.</td>
</tr>
<tr>
<td>DOCVariable</td>
<td>It refers to Document Object Constraint Variable. Solúbtha BTM provides two document object constraint variables: AcknowledgementRequired and AcknowledgementReceiptRequirement. These are used to specify the rules related to document objects. These variables have been borrowed from UMM (2006)</td>
</tr>
<tr>
<td>TCVариable</td>
<td>TCVариable refers to Time Constraint Variable that is used to specify the time related business rules. A list of time constraint variables is included in Solúbtha BTM.</td>
</tr>
</tbody>
</table>
7.2 SOLÚBTHA BTM

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResponseTime</td>
<td>It is used to specify the rules about time to respond to a request.</td>
</tr>
<tr>
<td>AcknowledgementTime</td>
<td>It is used to specify the rules of acknowledgment of the requests or document objects.</td>
</tr>
<tr>
<td>DeliveryTime</td>
<td>It is be used to specify the rules about delivery cycle time.</td>
</tr>
<tr>
<td>PaymentTime</td>
<td>It is used to specify the rules about the time to make payment against a purchase order.</td>
</tr>
<tr>
<td>MaxDelay</td>
<td>It is used to specify the rules about the maximum delay for the occurrence of business operations such as making payment and processing a request.</td>
</tr>
<tr>
<td>ASNTime</td>
<td>It is used to specify the rules about the time to send an advanced shipment notification.</td>
</tr>
<tr>
<td>Cooling-OffTime</td>
<td>It is used to specify the rules related to the time limit for the occurrence of an operation such as canceling a purchase order or making a payment.</td>
</tr>
<tr>
<td>OrderModificationTime</td>
<td>It is used to specify the rules related to the time to modify a purchase order.</td>
</tr>
<tr>
<td>CustomTimeConstraint</td>
<td>This variable allows users to define the context-specific rules related to temporal constraints in business transaction models.</td>
</tr>
</tbody>
</table>

7.2.2 Transaction Role

The TransactionRole refers to a specific role of a party in business transactions. The data structure is as follows. This element is refined from the design-element ‘Party’ that is contained in an ‘Agreement’. Figure 7.5 shows that the rolename has is used to refine the design-element ‘TransactionRole’ from ‘Party’. The association relationship has with cardinality ‘1..n’ (n geq 0) from ‘Party’ to ‘TransactionRole’, it determines that a party can have one or more roles in business transactions.

Each transaction role has an ‘Identifier’ which has the ‘String’ type value. The ‘Identifier’ can have the value: a name string or an alphanumeric string. It is a user-defined value so it depends on users. The design-element ‘TransactionRole’ is used to specify the attributes of transaction roles involved in business transactions.
7.2.3 Transaction Process Fragment

A transaction process fragment (TPF) is the smaller unit of an end-to-end business process. It can also be called as a composite activity, a sub-process or a business function. Figure 7.6 shows the UML representation of TPF. TPFs are the basic building blocks of end-to-end business processes. The processes are composed of sets of TPFs, which contain a set of ordered activities. The aggregation relation with cardinality between ‘TransactionProcessFragment’ and ‘Activity’ denotes that it can contain many activities. The activities perform the actual operations. The internal structure of TPFs is defined using the control flows. Solúbtha BTM provides an element called ‘ControlFlowPattern’ to define the internal structures. The association relationship between ‘Activity’ and ‘ControlFlowPattern’ determines the relation between these two elements. Solúbtha BTM provides a list of control flow patterns. These patterns were adopted from Russel et al., (2006). The description of the control flow patterns are summarized in Table 7.3.

An activity contained in a transaction process fragment can be composite or atomic. An atomic activity is non-decomposable, whereas the composite activity can be composed of one to many atomic activities. A reflexive aggregation relationship is defined in ‘Activity’. The composition relation provides the logical definition of composite activities in business transactions. The model enables specifying the type of attributes using ‘ATYPE’ whose values are: Vital or NonVital. The non-vital activities are weaker from a logical point of view. The failure of non-vital activity does not affect the execution of other running activities. The type of activity depends on users preference. More specifically, the user decides the type of their process activities.

While a business transaction is running, activities operate on the business objects. For instance, the ordering process fragment contains activities that manipulate the purchase order business object. Typically,
a business object is a business document that contains business data such as purchase order data. Solúbtha BTM provides an element called ‘BusinessObject’ to specify the business objects. It is worth noting that Solúbtha BTM allows the generic operators (shown in Figure 7.4) to specify the operation performed on business objects at runtime.

At runtime, the TPFs are governed by agreements, the operations that are performed within the scope of TPFs must comply with the agreement. The association between the design-element ‘Agreement’ and ‘TransactionProcessFragment’ is defined to establish an explicit correlation between these two design-elements. In fact, this association relation is the main logical connection between the business and functional perspectives. The logical connection enables defining the agreement inside TPF classes.

The model provides two basic operators commit and abort to specify system level operations of business transactions. The TOperator enumerated class (shown in Figure 7.6) is used to specify the commit and cancel operations. Table 7.3 summarizes the design-elements of TPF.
Figure 7.6: The Transaction Process Fragments of Solúbtha BTM.
### Table 7.3: The Table Summarizes the Transaction Process Fragment Construct of Solúbtha BTM

<table>
<thead>
<tr>
<th>TOperator</th>
<th>TOperator refers to Transaction Operator which is used to specify the system level operations: commit and abort.</th>
</tr>
</thead>
<tbody>
<tr>
<td>commit</td>
<td>It is used to define the commit operation of transaction processes fragments.</td>
</tr>
<tr>
<td>abort</td>
<td>It is used to define the abort operation of a business transaction.</td>
</tr>
</tbody>
</table>

**ControlFlowPattern**

The control flows are the execution paths for the activities contained in transaction process fragments. They are used to define the internal logical structure of transaction process fragments.

<table>
<thead>
<tr>
<th>NJoin</th>
<th>It stands for AND-Join which is a synchronizer. It enables specifying the synchronization (joining) of two or more activities. In a synchronization pattern, two or more inbound activities and an outbound activity are specified. At runtime, the outbound activity is started after the inbound activities have completed successfully.</th>
</tr>
</thead>
<tbody>
<tr>
<td>XJoin</td>
<td>It denotes Exclusive-Join flow pattern. Like NJoin, XJoin is a synchronizer of activities. The design style of these patterns is the same however, the runtime operational style are different. In XJoin, the outbound activity is triggered once any inbound activity is completed and the outcome has reached to synchronization point.</td>
</tr>
<tr>
<td>XChoice</td>
<td>It denotes the Exclusive-Choice flow pattern which is a splitter. It enables splitting an activity into two or more sub-activities. If an activity is triggered at runtime then control is allowed to move to any of the sub-activities.</td>
</tr>
<tr>
<td>NChoice</td>
<td>It denotes AND-Choice flow pattern. Like XChoice, it is a splitter. The operational style of NChoice is different from XChoice. In NChoice, if an activity is triggered then the control is allowed to move to two or more activities.</td>
</tr>
</tbody>
</table>
### Interleaved

It is an important control pattern. It enables specifying the non-monotonic structural pattern where an activity can be executed in any order. The non-monotonicity property relaxes the strict order of execution.

### SDiscriminator

It denotes Structured Discriminator. It consists of BlockingDiscriminator and CancelDiscriminator. To explain these two, consider a concurrent transaction composed of two activities running concurrently. If one of these activities has completed successfully and the result has arrived then the ‘CancelDiscriminator’ will cancel the other activity automatically. The Block discriminator allows instantiating the inbound paths of an activity multiple times.

### MIWSync

It denotes Multiple Instance Without Synchronization. This pattern is used to define the autonomous structural pattern where multiple instances of an activity run independently and that these instances do not require synchronization.

### Sequence

It is the simplest control flow pattern. It is used to define the serial structural pattern of activities. In a serial structure the activities execute sequentially at runtime.

### 7.2.4 Business Protocol

The ‘BusinessProtocol’ was included in the initial version of the model and renamed in the subsequent version. While describing Solúbtha BTM V1, the business protocol was explained briefly from the business perspective. In this version of the BTM, it is explained from the technical point of view. In this section, I explain the technical foundation of business protocol.

In a multi-layered structure of business transactions, there is a global layer which is visible to all the business partners who are involved in a business transaction, and there is a local layer, which is internal to the business partners. A trading partner can see only the global parts of business transactions and their states of the other trading partners. The BusinessProtocol is incorporated in Solúbtha BTM to enable users to define the structures of business transactions at the global layer. To be more specific, it defines the execution behavior of business transactions at the global layer. The behavior of business transaction at this layer refers to the order of the transaction process fragments and the order of interactions amongst the
trading partner systems.

Solutions such as Desai et al. (2011) have been proposed to define the order of the interactions between/among the trading partners. In addition, the RosettaNet PIPs are well-known and widely used industry standard protocols for defining the order of interactions. Several such solutions are available to define the business protocol for ordering the interactions between business partners. Solúbtha BTM does not provide any support to define the sequence of interactions. It enables defining the logical order of TPFs at the global layer. It is essentially a one-level abstraction of interactions.

The BusinessProtocol contains two composition patterns: 
precede and succeed. The association relation between BusinessProtocol and TransactionProcessFragment is refined using the rolename sequences. The cardinality ‘1’ and ‘*’ determines that a business protocol sequences many transaction process fragments. Figure 7.7 presents the business protocol.

The investigation of trading partner agreements reveals that the agreements contain clauses that determine the execution order of TPFs. For instance, in Tenneco’s Purchase Order Agreement (2012), the clause 4(C) says Goods and/or services purchased are subject to Buyer’s inspection and to approval or rejection despite prior payment. According the clause the payment has been made before the goods were delivered. Therefore, the delivery TPF succeeds the payment TPF. This indicates that agreements determines the business protocol. In Solúbtha BTM, I define an association relation between ‘Agreement’ and ‘BusinessProtocol’ with the rolename determines.
7.2.5 Transaction Failure Recovery Handler

In addition to ‘Agreement’, the transaction failure recovery handler is an important design-element provided by Solúbtha BTM. This enables defining the failure recovery policies which are essentially sets of logical statements. Each recovery policy represents a technique. The policies also represent the properties of flexible business transactions. I have included a policy-based recovery approach in Solúbtha BTM for preventing failures or abortions. I choose this because the policy based recovery is a flexible approach and enables to define context-specific recovery rules. Essentially, I realized the suitability of a policy-based recovery approach during the analysis of the secondary data of business partner agreements. I found that the agreement clauses define the faults and the consequences. Additionally, I observed that faults can happen with activities. Therefore, one failure prevention technique for all faults, e.g., compensation is certainly not an appropriate approach, particularly, in the case of business-related faults. The policy-based failure recovery approach would enable users to specify and execute particular recovery rules based on different factors such as the nature of fault and the context in which the faults occurred. Additionally, I developed several fault models (The fault models were described in Chapter 5) which can used as inputs for defining the recovery rules.

Figure 7.8 depicts the Transaction Failure Recovery Handler package. The main design-element in this package is ‘TransactionFailureRecoveryHandler’ that has a part called RecoveryPolicy. This contains Mutualize, Resume, Repair, Retry, Suspend, Postpone, Substitute, Ignore, Compensate, Wait. These are the recovery operators enable defining the recovery policies.

An association relationship between ‘Agreement’ and ‘RecoveryPolicy’ is defined in Solúbtha BTM. During my investigation, I found some clauses that state faults as well as the potential solutions to prevent them. These imply that agreement influences the recovery policies. Particularly, it can guide defining the appropriate recovery policies.

Many times faults occur with business activities such as when shipment faults occur with delivery activities. Therefore, I have defined a logical connection between TransactionFailureRecoveryHandler and TransactionProcessFragment to enable users to define the transaction recovery logic in the transaction process fragment classes. The design-elements of the transaction recovery handler package is briefly explained below:

- **Ignore**: It refers to ignoring faults that occur either with non-vital activities or with soft rules. During my study, I found that some business process activities, e.g., ‘purchase order acknowledgment’ can be less important. If a fault such as communication failure occurs with this type of activity, it can be ignored instead of repairing it. Repairing fault is computationally expensive. ‘Ignore’ is used to specify the logical conditions for ignoring a fault. It implements the fault-tolerant business transactions.

- **Suspend**: The business operations may need to be suspended temporarily. For instance, if a third
party logistics operator fails to deliver a purchase order due to some unexpected events, such as workforce strike, then an alternative provider is assigned temporarily. In such cases, the applications of the logistics provider are suspended temporarily or permanently and replaced by the applications of the alternative provider. This element is included in the model to enable users to specify the logical
condition for suspending the business operations temporarily.

- **Postpone**: An operation may need to be rescheduled in some situations. For instance, the delivery of a purchase order is rescheduled due to some faults (e.g., workforce strike or transportation strike). Postpone is used to specify the rescheduling operations of business transactions. This element complements other recovery elements in particular, mutualize.

- **Mutualize**: Violation of agreement is considered a serious fault in business transactions, which leads to transaction abortion. Aborting a business transaction due to a violation of an agreement may not be a viable option from the computational and the economic perspectives. This operator enables specifying the logical condition that allows the modification of an agreement to adapt some situations and prevent abortion of business transactions.

- **Repair**: It is the core element in the transaction failure recovery handler package. It may lead to a chain of system level operations such as add, modify, and delete. The users can use the global operators to design the repair operations. In addition, the repair operator may use one or more other recovery operators.

- **Retry**: This technique was found in several articles such as Crysanthesis and Ramamritham, (1992) and Papazoglous and Kratz, (2006). I adopted it in Solúbtha BTM V1. It is a widely used approach for repairing faults. This technique enables defining a logical statement that enforces a business transaction system to perform the Retry operation at runtime upon failure of business process activities. It has two main advantages in business transactions: (i) it prevents the transaction manager from broadcasting the abort operation to the cohorts (sub-transaction) until the ‘Retry’ operation fails and (ii) it delays the undo operation, which is computationally expensive. In short, Retry prevents a business transaction abortion upon failure of a process activity and business transaction systems do not invoke ‘undo’ operation until the retry has failed. This enhance the flexibility in business transactions.

- **Substitute**: I found this design-element in Mueller *et al.*, (1983). It has been adopted in many contexts including some transaction models (e.g. Bukhres *et al.*, 1992) where it has been used as the acceptability function. I included this design-element in my model because I found it a very useful technique for defining flexible behavior for business transactions. During the analysis of the secondary data of the Auto Inc. order processing scenario, I found that the Auto Inc. process allows substitution of a supplier, which indicates that a business transaction system can automate the substitution of an existing sub-transaction by another sub-transaction.

From the technical perspective, substitution refers to the replacement of a business activity by a functionally equivalent activity. The design-element ‘Alternatives Set’ of Solúbtha BTM V1 enables users to specify one or more substitute business activities. Solúbtha BTM V1 allows specifying
substitute activities as secondary activities, which, at runtime are invoked automatically only when the related primary activities are failed. This prevents the abortion of business transactions because substitute activities enable business transaction systems to continue processing transactions when sub-transactions fail. The execution of substitute activities are mutually exclusive, which means, if a set of substitute activities is specified in the business transaction model then at runtime only one out of the set is invoked upon failure of the associated primary activity.

- **Compensation:** This element was introduced in the ‘Saga’ transaction model (Garcia-Molina, 1987) and adopted in all advanced transaction models that are used today for modeling flexible business transactions. From the technical perspective, compensation refers to a process of undoing a business transaction from its savepoint to the initial state in reverse order. A savepoint refers to the last saved state of business transactions. Compensation is an effective technique for preventing transaction abortion. Therefore, it has a significant number of technological supports such as Windows Workflow Foundation from Microsoft and BPEL (OASIS, 2007). I adopted this element in Solúbtha BTM. Compensation enables users to specify the compensation activities with the process activities. Each process activity should be associated with a compensate activity that will be triggered at runtime only upon failure of a running activity and it will undo the effects of successfully completed activities from the semantic point of view in reverse order. A compensation activity must not be invoked unless the corresponding activity fails.

- **Resume:** It was introduced by Garcia-Molina (1987), however, it was called Redo in the Saga model. It refers to restarting a (sub-)transaction. Upon failure, the compensation functions of a business transaction systems revert the transaction to the initial state. However, they do not restart the transaction. I included ‘Resume’ in Solúbtha BTM to enable business transaction systems to restart business transactions upon completing the undoing operation. The Resume was essentially included in the model to complement the compensation technique. These compensation and resume operations should be defined sequentially.

- **Wait:** I found this element in Greenfield et al., (2003). It refers to the state of an activity in a transactional business process where it waits for completion of the execution of its precedent activity. This element was included in the model to enable users to specify wait operations.

### 7.2.6 The Complete Solúbtha BTM

In the previous sections I presented the parts of Solúbtha BTM. In this section, I bind the integral parts of the model logically and present the complete coarse-grained model. Figure 7.9 shows the Solúbtha BTM.

---

4 [http://microsoft.com](http://microsoft.com)
CHAPTER 7. DEVELOPMENT OF SOLÚBTHA BTM

Figure 7.9: The Complete Solúbtha Business Transaction Model.
7.3 Summary

In this chapter I presented the construction process of Solúbtha BTM. I presented and explained the final version of Solúbtha BTM. The Initial Version and Solúbtha BTM V1 are available in Appendix D and E. I provided descriptions of each element of the model. I described the logical relations between elements.

I have described how Solúbtha BTM enables designing business transactions by correlating business and functional perspective design-elements. Several logical association relations have been defined in Solúbtha BTM to make explicit connections between the business and functional perspectives. Without these logical connections, the transaction systems would not enable designing business perspective elements, particularly business, security, and compliance policies with the functional perspective elements such as ‘Activity’. In Solúbtha BTM, the association relation between ‘Agreement’ and TransactioProcessFragment’ enables the definition of all types of policies in the transaction process fragment classes. Also, the policies can be defined along with business activities so that each operation performed by the activity will be controlled by the policies.

Solúbtha BTM provide nine recovery elements: Resume, Ignore, Mutualize, Postpone, Suspend, Substitute, Repair, Compensate, and Wait and seven control flow patterns: NJoin, XJoin, MIWSync, NChoice, Interleaved, XChoice, and SDiscriminator. All these elements were included in the model to define the flexible behaviour of business transactions. The recovery elements were developed taking a wide variety of faults (described in Chapter 5) into account. These elements enable to define a highly fault-tolerant and failure-resilient business transaction models. In addition, the workflow patterns enables defining autonomous, non-deterministic, and extensible business transactions.

Combining all these features provided by Solúbtha BTM, it can be concluded that the model is capable of designing highly flexible business transactions by explicitly correlating the business and functional perspectives of end-to-end business processes.
Chapter 8

Formal Syntax of Solúbtha BTM

8.1 Introduction

In the previous chapter, Solúbtha BTM was presented using UML class diagrams. The class diagrams present models in a semi-formal manner which is less expressive (David, 2004). In order to provide the complete formal syntax of Solúbtha BTM, I formalize the model using set theory which is a mathematical formalism. This formalization is critical because, to verify the correctness of an instance-model at design-time, particularly, to verify whether the instance-model satisfies syntactic correctness criteria and other desirable properties, mathematical formalization of the model is necessary (Beek et al., 2007). A brief overview of Set Theory is provided in Section 8.2.1.

8.2 Formal Definition of Solúbtha BTM

This section presents the syntax of Solúbtha BTM using set theory which is a widely used approach for defining real-world objects in a formal manner.

8.2.1 Set Theory

Set theory is true study of infinity (Weiss, 2008). It is the foundation of mathematics used to define the mathematical concepts in terms of a primitive notion of a set membership (Kunnen, 1992). Xavier (2005) defines set as a collection of objects that are essentially the members of the set. A set can be empty and is called the empty set whereas a set with one element is called a singleton (Xavier, 2005).

Several theories such as ZFC set theory (Kunnen, 1992) and (Potter, 2004) have been developed to define sets mathematically. These are used to encode solutions (e.g., software). The mathematical codification enables verifying the solutions at design-time. Set theory is a language and is defined by Kunnen (1992) as follows,

Definition 2. A set theory is a language which uses the following notations,

- Variable \( v_0, v_1, v_2, \ldots \)
- Equality symbol \( = \)
8.2. FORMAL DEFINITION OF SOLÚBTHA BTM

• Membership symbol ∈
• Logical Connectives ¬, ∧, ∨, →, ←
• Quantifiers ∀, ∃
• Parenthesis (, )

8.2.2 Abstract Syntax of Solúbtha BTM

This section presents the formal definitions of the design-elements of Solúbtha BTM. As mentioned in the previous, the formalism used for defining the syntax of the model is Set Theory.

8.2.2.1 BusinessTransaction

BusinessTransaction is the central design-element of Solúbtha BTM. It subsumes five design-elements: TransactionProcessFragments, TransactionRoles, Agreement, BusinessProtocol, and TransactionRecovery-Handler. The BusinessTransaction can be defined as follows,

Definition 3. A BusinessTransaction \( \tau \) is a quintuple \( \tau := (\Gamma, \rho, \Sigma, \mathcal{P}, \eta) \)

Where,

• \( \Gamma \) denotes denotes TransactionProcessFragment
• \( \rho \) denotes TransactionRole
• \( \Sigma \) denotes Agreement
• \( \mathcal{P} \) denotes BusinessProtocol
• \( \eta \) denotes the TransactionRecoveryHandler.

8.2.2.2 Agreement

The ‘Agreement’ has three parts duration, party, and policy. It can be formally defined as follows:

Definition 4. The Agreement ‘\( \Sigma \)’ is a tuple \( \Sigma := (\varphi, \sigma, \Psi) \)

Where:

• \( \varphi \) denotes denotes Period
• \( \sigma \) denotes Party
• \( \Psi \) denotes Policy
The structure of ‘Agreement’ can be defined as follows,

\[ \Sigma = (\text{Identifier: String}, \text{Type: AgType}) \]

Where:

- \text{Identifier} is an item of Agreement with unique value of ‘String’ type which allows assigning the alphanumeric values.

- \text{Type} denotes the type of agreement whose value AgType = (LTA, STA) where LTA and STA denote Long Term Agreement and Short Term Agreement respectively.

\textbf{Definition 5.} An agreement can be simple or it can be composite. A composite agreement is the composition of two or more agreements. It can be written as follows, \( \Sigma := (\Sigma_1 \cup \Sigma_2 \ldots \cup \Sigma_n) \)

\subsection*{8.2.2.3 Period}

The period ‘\( \varphi \)’ is a valid duration of an agreement. It can be defined below,

\textbf{Definition 6.} A period ‘\( \varphi \)’ is a tuple \( \varphi := (\text{startDate : Date}, \text{endDate : Date}) \)

Where:

- \text{startDate} denotes the date of an agreement being operational.

- \text{endDate} denotes the date of an agreement ended.

\subsection*{8.2.2.4 Party}

‘Party’ refers to a business partner involved in an agreement. Parties are the active participants of business transactions.

\textbf{Definition 7.} The Party ‘\( \sigma \)’ is a tuple \( \sigma := (\text{Identifier : String}, \text{Contact : ConInfo}) \)

Where:

- Identifier is a data item that is used to specify the unique identity of a business partner. This identity are referenced at runtime by applications that perform business transaction throughout the lifetime of an agreement.

- Contact is a composite data item whose value \( \text{ConInfo} \) can be \( \text{ConInfo} = (\text{address, phone, email, website, CustomInfo}) \).
8.2. FORMAL DEFINITION OF SOLÜBTHA BTM

8.2.2.5 Policy

The formal syntax of ‘Policy’ can be written as follows:

**Definition 8.** Policy ‘Ψ’ is a tuple \( Ψ := (Ψ_{BP}, Ψ_{SP}, Ψ_{CP}) \)

Where:

- \( Ψ_{BP} \) denotes BusinessPolicy
- \( Ψ_{SP} \) denotes SecurityPolicy
- \( Ψ_{CP} \) denotes CompliancePolicy

8.2.2.6 BusinessPolicy

The BusinessPolicy ‘Ψ_{BP}’ is defined as follows,

**Definition 9.** BusinessPolicy ‘Ψ_{BP}’ is a set of business rules \( Ψ_{BP} := (bRule_1, ..., bRule_n) \)

Where:

- \( bRule \) denotes BusinessRule.

Business rules are essentially the constraints that are refined from the business policies. The formal structure of the BusinessRule is defined below:

**Definition 10.** A business rule ‘bRule’ is a tuple,

\[ bRule := (GVariable, TCAvariable, DOCvariable, AOperator, ROperator, TRIdenti f ier, UType, (Type : RType), LOperator, TRIdenti f ier) \]

Where:

- \( GVariable \) denotes global variables associate with business rules.
- \( TCAvariable \) denotes time constraint variables for specifying the time related rules with activities that are contained in the transaction process fragments.
- \( DOCvariable \) denotes document related variable for specifying the constraints related to business objects.
- \( AOperator \) denotes arithmetic operators for specifying the values of the variables such as ‘GVariable’, ‘TCVariable’, ‘DOCVariable’, and security parameters.
- \( ROperator \) denotes relational operators for specifying the logical relation between the operands.
- \( TRIdenti f ier \) denotes temporal reserved identifiers for specifying the temporal constraints.
• \( UType \) denotes unit type that refers to the smallest unit associates with the values of a variable.

• \( L\text{Operators} \) denotes logical operators for writing the business rules as the statements or the conditions.

• \( RType = (\text{Soft}, \text{Hard}) \) is an attribute for specifying the types of business rules.

Definition 11. A composite business rule \( \text{CbRule} \) is a tuple,

\[
\text{CbRule} := (\text{bRule}_1, (L\text{Operator}:\text{AND, OR}))
\]

Where:

• \( \text{CbRule} \) denotes composite business rules.

• \( \text{bRule}_1 \land \text{bRule}_2 \land ......, \land \text{bRule}_n \)

• \( \text{bRule}_1 \lor \text{bRule}_2 \lor ......, \lor \text{bRule}_n \)

The logical operators \( \text{AND} \) and \( \text{OR} \) are used to specify the composite rules. The simple business rules are the non-decomposable conditions.

8.2.2.7 CompliancePolicy

Compliance policies are the constraints that control business transactions from the regulatory perspective. A compliance policy can be defined as follows:

Definition 12. The CompliancePolicy is a set of compliance rules that can be written as follows,

\[
\Psi_{CP} := (\text{cRule}_1, ......, \text{cRule}_n)
\]

Where:

• \( \text{cRule} \) denotes ComplianceRule

Compliance rules are refined from the regulations contained in agreements. The design-element ComplianceRule is defined below,

Definition 13. The ComplianceRule is a tuple \( \text{cRule} := (\text{cVariable} , \text{RType} , \text{COperator} , \text{LOperator} , \text{ROperator} , \text{AOperator}) \)

Where:

• \( \text{cVariable} \) denotes control variable.

• \( \text{COperator} \) denotes control operator.

Definition 14. A composite ComplianceRule is a tuple, \( \text{CcRule} := (\text{cRule} , (L\text{Operator}:\text{AND} , \text{OR})) \)

Where:

• \( \text{CcRule} \) denotes composite ComplianceRule.
8.2. FORMAL DEFINITION OF SOLÚBTHA BTM

8.2.2.8 SecurityPolicy

Security policies are the security constraints which control business transactions from the security perspective. The SecurityPolicy is defined below,

Definition 15. The SecurityPolicy \( \Psi_{SP} \) is a set of security constraints, \( \Psi_{SP} := (sParameter_1, \ldots, sParameter_n) \)

Where:

- \( sParameter \) denotes SecurityParameter.

Each security parameter associates by a value. The concept security parameter is defined below:

Definition 16. Security parameter is a tuple, \( sParameter := (name, \text{type:Boolean}, RType) \)

Where:

- \( name \) denotes name of the parameter.

- \( type : \text{Boolean} \) denotes Boolean type values of the security parameters.

Solúbtha BTM provides a set of security parameters that are Boolean type whose values are \text{YES, NO}.

The security constraints can be soft or hard type.

8.2.2.9 TransactionProcessFragment

The transaction process fragment is composed of activity, business object, and control flow pattern. It can be formally defined as follows,

Definition 17. TransactionProcessFragment \( \Gamma \) is a tuple, \( \Gamma := (\alpha, \Pi, \beta) \)

Where:

- \( \alpha \) denotes Activity.

- \( \Pi \) denotes ControlFlowPattern.

- \( \beta \) denotes BusinessObject which is an optional element. The attribute optional is represented by \( \text{opt} \).

The data structure of TransactionProcessFragment can be defined as follows,

Definition 18. \( \Gamma := (\text{identifier: String}) \)

Where:
• *identifier* is an item denotes the unique identity of a transaction process fragment. The identifier is
‘String’ type allows alphanumeric value.

The integral parts of the TransactionProcessFragment are formally defined below,

**Definition 19.** The Activity ‘α’ is a tuple,

\[ \alpha := (\text{name: String}, \text{type: AType}) \]

Where:

• *name* is the name of activities.

• *type* is the type of activity; it has built-in values \( AType = (\text{Vital}, \text{NonVital}) \).

The Vital and NonVital activities are formally defined as follows.

**Definition 20.** If \( \alpha \) is an activity contained in \( \Gamma \) then \( \alpha \) is vital for \( \Gamma \) if and only if,

• for any execution, at any time, if the \( \alpha \) was committed, then the \( \Gamma \) was committed. It can be written as follows, \( AG(\text{commit } \alpha \Rightarrow \text{commit } \Gamma) \).

• for any execution, at any time, if the \( \alpha \) was aborted, then either the \( \alpha \) was repaired and eventually the \( \Gamma \) was committed or the \( \Gamma \) was aborted. It can be written as follows, \( AG(\text{abort } \alpha \Rightarrow \text{repair } \alpha \lor \text{abort } \Gamma) \).

**Definition 21.** As opposed to the vital activities, failure of the non-vital activities do not affect other activities that are contained in transaction process fragments. The formal definition of the NonVital activity is given below.

An activity \( \alpha \) contained in \( \Gamma \) is NonVital for \( \Gamma \) if and only if,

• for any execution, at any time, if \( \alpha \) was committed, then \( \Gamma \) was committed, \( AG(\text{commit } \alpha \Rightarrow \text{commit } \Gamma) \).

• for any execution, at any time, if \( \alpha \) was aborted, then the \( \Gamma \) was committed \( AG(\text{abort } \alpha \Rightarrow \text{commit } \Gamma) \).

A control flow pattern determines the execution patterns of business transactions. It is used to specify the sequence of activities. The control flow pattern is defined below.

**Definition 22.** The ControlFlowPattern ‘\( \Pi \)’ contains a set of flow patterns include \( \Pi_{NJ} \), \( \Pi_{XJ} \), \( \Pi_{XC} \), \( \Pi_{MC} \), \( \Pi_{Seq} \), \( \Pi^f \), and \( \Pi^{MWS} \).

The flow patterns are explained in the following:
8.2. FORMAL DEFINITION OF SOLÚBTHA BTM

- $\alpha_1 \land \ldots \land \alpha_n \leftarrow \Pi^{NJoin}$ is AND-Join pattern for joining two or more activities.

- $\alpha_1 \oplus \alpha_2 \oplus \ldots \oplus \alpha_n \leftarrow \Pi^{XJoin}$ is eXclusive OR-Join for merging a set of activities.

- $\alpha_1 \oplus \alpha_2 \oplus \ldots \oplus \alpha_n \leftarrow \Pi^{XChoice}$ is eXclusive OR-Choice for designing the selection of an activity from a given set of activities.

- $\alpha_1 \land \ldots \land \alpha_n \leftarrow \Pi^{NChoice}$ is AND-Choice for designing the selection of multiple activities from a given set of activities.

- $\text{seq}(\alpha_1, \ldots, \alpha_n) \leftarrow \Pi^{Seq}$ is sequence for designing the sequence of activities.

- $\alpha_1 \text{ interleave} (\alpha_2, \ldots, \alpha_n) \leftarrow \Pi^I$ is interleave for designing the interleaved routing of an activity within a transactional business process.

- $\Pi^{BD} \odot \Pi^{CD} \leftarrow \Pi^{SD}$

Where:

- $\Pi^{BD}$ denotes ‘Block Discriminator’.

- $\Pi^{CD}$ denotes ‘Cancel Discriminator’.

These discriminators can be defined as follows,

- $\text{trigger}(\alpha_i) \leftarrow \Pi^{BD}$ a block discriminator allows triggering the incoming path of an activity multiple times.

- $(\text{Cancel}(\alpha_{i+1}, \ldots, \alpha_{i+n})$ if $\alpha_i$ is completed $) \leftarrow \Pi^{CD}$ is cancel discriminator that is used for designing the cancellation of all subsequent activities upon completion of an activity.

- $(\alpha_i \rightarrow \neg \text{Sync}(\alpha_{instance1}, \ldots, \alpha_{instanceN})) \leftarrow \Pi^{MWS}$ is multiple instances without synchronization, a composition pattern that is used for designing the activity whose multiple instantiations do not synchronize.

**Definition 23.** The BusinessObject is a schema contains the business data in a structured manner. It can be defined as follows,

A *BusinessObject* is a tuple,

$$\beta := (\text{Identifier : String}, \chi)$$

Where:

- *Identifier* is a unique value.

- $\chi$ is an array of business data.
8.2.2.10 Business Protocol

The BusinessProtocol can be defined as below,

**Definition 24.** A BusinessProtocol is a tuple,

\[ \mathcal{P} := (ptID : String, bpSeq : TypeSeq) \]

Where:

‘TypeSeq’ is a special type whose values: (precede, succeed) have been enumerated in the ‘TypeSeq’ class. The values are described in the following:

- **precede** denotes preceding protocol that is used to design the preceding sequence of \( \Gamma \) such that, \( \Gamma_1 \) precede \( \Gamma_2 \) precede ..... precede \( \Gamma_n \).
- **succeed** denotes succeeding protocol that is used to design the succeeding sequence of \( \Gamma \) such that, \( \Gamma_1 \) succeed \( \Gamma_2 \) succeed ..... succeed \( \Gamma_n \).

8.2.2.11 TransactionFailureRecoveryHandler

The TransactionFailureRecoveryHandler is composed of the recovery policies which can be written as follows,

**Definition 25.** A TransactionFailureRecoveryHandler is a tuple,

\[ \eta := (\varepsilon) \]

Where:

- \( \varepsilon \) denotes the RecoveryPolicy.

**Definition 26.** The RecoveryPolicy ‘\( \varepsilon \)’ is a set of disjoint recovery operators include \( \varepsilon_{\text{Compensate}} \), \( \varepsilon_{\text{Resume}} \), \( \varepsilon_{\text{Repair}} \), \( \varepsilon_{\text{Retry}} \), \( \varepsilon_{\text{Substitute}} \), \( \varepsilon_{\text{Mutualize}} \), \( \varepsilon_{\text{Ignore}} \), \( \varepsilon_{\text{Postpone}} \), \( \varepsilon_{\text{Suspend}} \), and \( \varepsilon_{\text{Wait}} \).

The formal definition of these operators are given below,

- \( \forall \tau \), the compensate operator associates with the transaction process fragments such that, \( (\Gamma_i, \varepsilon_{\text{Compensate}}) \)
- \( \forall \tau \), the resume operator associates with transaction process fragments such that, \( (\Gamma_i, \varepsilon_{\text{Resume}}) \)
- \( \text{trigger}_{\text{Repair}} : \varepsilon_{\text{Repair}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{BProtocol}} \lor \mathcal{F}_{\text{Message}} \lor \mathcal{F}_{\text{BO}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the repair operation to business or application related faults.
- \( \text{trigger}_{\text{Substitute}} : \varepsilon_{\text{Substitute}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the substitute operation to business or application related faults. The substitute operation is triggered upon the occurrence of a fault.
8.2. FORMAL DEFINITION OF SOLÚBTHA BTM

• \( \text{trigger}_{\text{Mutualize}} : \epsilon^{\text{Mutualize}} \rightarrow \mathcal{F}_{\text{Business}} \) is a function which maps the mutualization operation to a certain business related fault.

• \( \text{trigger}_{\text{Ignore}} : \epsilon^{\text{Ignore}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the ignore operation to the business or application related faults that is caused by violation of a soft type rule.

• \( \text{trigger}_{\text{Retry}} : \epsilon^{\text{Retry}} \rightarrow \mathcal{F}_{\text{Message}} \) Message is a function which maps the retry operations to the message related faults. It is triggered if faults such as message delivery failure was occurred.

• \( \text{trigger}_{\text{Postpone}} : \epsilon^{\text{Postpone}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the postpone operation to the business or application related faults. This operation is triggered upon the occurrence of faults such as violation of agreement. Postpone is an auxiliary operator used with repair and mutualize operators.

• \( \text{trigger}_{\text{Suspend}} : \epsilon^{\text{Suspend}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the suspend operation to the business or application related faults. Like postpone, suspend is an auxiliary operator.

• \( \text{trigger}_{\text{Wait}} : \epsilon^{\text{Wait}} \rightarrow (\mathcal{F}_{\text{Business}} \lor \mathcal{F}_{\text{Application}}) \) is a function which maps the wait operation to the business or application related faults. The wait operation can be triggered upon the occurrence of a failure of an activity in a transaction process fragment. It is another auxiliary operator that supplements the Repair and Mutualize operator.

8.2.2.12 M-to-1 recovery function

Definition 27. \( \forall \tau, \) a transaction failure recovery can be an M-to1 recovery function that maps many operations to one fault. For a fault such as shipment fault, operators such as suspend, postponed, and mutualize operators can be used to specify the recovery operations to prevent the failure that can be caused due to that shipment fault.

8.2.2.13 Transaction Role

Each transaction involves a number of roles. The TransactionRole can be defined as follows,

Definition 28. TransactionRole \( \rho \) is a tuple,

\[ \rho := (\text{identifier: String}) \]

Where:

• Identifier is used for specifying the unique identity of a transaction role.
8.2.2.14 Business Transaction Model

The BusinessTransactionModel is composed of sets of design-elements. It can formally be defined as follows:

Definition 29. BusinessTransactionModel $\mathcal{M}$ is a tuple, $\mathcal{M} := (\Xi \subseteq \tau, \mathcal{R}, \mathcal{MUL})$

Where:

• $\Xi = ((\Gamma', \rho', \Sigma', \Theta', \eta') \subseteq \tau)$ is a set of elements which consist in a business transaction model.

• $\mathcal{MAP}: \mathcal{RType} \rightarrow \Xi$ is a function mapping the relationships to the elements. The mapping binds the design elements logically to construct the business transaction.

• $\mathcal{MUL}$ is the range of elements defined in (Vasilios et al., 2008) as $\mathcal{MUL}: [\text{min}_{\text{crd}}, \text{max}_{\text{crd}}]$

Where:

$(\text{min}_{\text{crd}}, \text{max}_{\text{crd}}) \in \mathbb{N}$ (the set of natural number) is the maximum and minimum multiplicities allowed for each member of the relationship.

8.2.3 Relationships between Design Elements

The relationships between design-elements were defined using the UML notations in the previous chapter. The relationship between design-elements can be classified into intra-relationship and inter-relationship.

The formal definitions of these relationships are provided below:

Definition 30. $\forall \mathcal{M}, \exists \mathcal{R}_{\text{intra}}(\Xi) ((e_1,.., e_n) \text{ RType } E) \text{ where } E \subseteq \Xi$, is intra-relationship between elements where all elements such as $e_1,.., e_n$ are contained in $E$ by means of a relationship type such as composition, aggregation, and association.

Definition 31. $\forall \mathcal{M}, \exists \mathcal{R}_{\text{inter}}(\Xi) (E_1 \text{ RType } E_2) \text{ where } E_1 \text{ and } E_2$, the elements are inter-related by means of a relationship type such as composition and aggregation.

The definition of the business transaction model shows that the construction of the business transaction model relies on the relationships among design elements. The correctness of relationship is important for designing a well-formed business transaction model. The intra and inter relationships between design elements can be defined using generalization, aggregation, composition, and association relationships. The definitions of the formal semantics of composition, aggregation, and association relationships can be found in (Andrikopoulos et al., 2008). They are summarized below.
• **Composition**: consider x and y are two elements. The composition $c : \forall y \exists! x : r (x, y) = r (x, y, c, \ldots)$; y belongs to exactly one composition relationship with x. Since composition denotes strong relationship type, the deletion of 'y' will imply the deletion of 'x' (cascading delete).

• **Aggregation**: The aggregation x and y are two elements. The composition $a : \forall x \exists x : r (x, y) = r (x, y, a, \ldots)$; y is allowed to more than aggregation relations with x. Aggregation is a weaker relationship type, the deletion of 'y' will imply the deletion of 'x' is and only if there are no other relationships of this type in which 'y' participates.

• **Association**: The association $S : \exists x \exists y : r (x, y) = r (x, y, S, \ldots)$; no further restrictions on the participation and the existence of 'y'.

From the classical object oriented point of view the generalization is inheritance which is a Object Oriented Design (OOD) technique that allows creating instances that inherit the properties of the parent construct (Meyer, B., 1997). The formal semantics of the generalization is given below:

• **generalization**: The aggregation $g : \forall x \exists! x' : r (x, y) = r (x, x', g, \ldots)$; $x'$ is a child element of 'x' which imply that if the parent element is deleted then the child element $x'$ is deleted but not the opposite. The deletion of $x'$ does not result deletion of 'x'.

The relations between the design-elements of Solúbtha BTM are defined in the following using these relations:

**Definition 32.** In Solúbtha BTM, the root construct BusinessTransaction has aggregation relation with TransactionRole, Agreement, BusinessProtocol, and TransactionFailureRecoveryHandler. This relation can be written as follows,

$$\forall \mathcal{R}_{\text{intra}} (\Sigma, \rho, P, \epsilon) \ a (\tau))$$

In a business transaction instance-model, the TransactionRole, BusinessProtocol, TransactionProcess-Fragment, and Agreement are defined as parts with multiplicities that essentially are the design constraints for the syntax of the instance-model. These constraints should be satisfied while designing the instance-model.

The following are the cardinality constraints that should be met in the instance-models. To define these constraints I used *alethic possibility* and *alethic necessary* (*Loss et al.*). modalities.

• $\forall M, M \not\subseteq [\exists \tau, \Sigma \mid \Sigma_i : i = 1..k(\neq \infty)],$ it is necessary that there exist at least one agreement and it is possible that the maximum cardinality can be spanned to ‘k’ number (must be finite) of agreements.

• $\forall M, M \not\subseteq [\exists \tau, \rho \mid \rho_i : i = 1..k(\neq \infty)],$ it is necessary that there exist at least two transaction roles and it is possible that the business transaction contains more than two roles.
• \( \forall M, M \cup L [\exists \tau, \epsilon_i \mid \epsilon_i : i = 1..k(\neq \infty)] \), it is **necessary** that there exist at least and at most one transaction failure recovery handler containing many recovery policies.

These design-elements has internal parts. The internal relationships between these elements are defined below.

**Definition 33.** An agreement subsumes party, period, and policy using aggregation relation.

\[ \forall M, R_{intra} ((\varphi, \Psi, \sigma) \alpha (\Sigma)) \]

The minimum and maximum cardinality of relations between these elements can be written as follows.

• \( \forall M, M \cup L [\exists \Sigma, \Psi_i \mid \Psi_i : i = 1..k(\neq \infty)] \), it is **necessary** that there exist at least one policy and it is possible that there exist ‘k’ number (must not be infinite) of policies.

• \( \forall M, M \cup L [\exists \tau, \varphi_i \mid \rho_i : i = 1] \), it is **necessary** that there exist at least and at most one period. ¹

• \( \forall M, M \cup L [\exists \Sigma, \sigma_i \mid \sigma_i : i = 1..k(\neq \infty)] \), it is **necessary** that there exist at least two parties and it is **possible** that an agreement subsumes more parties.

The ‘Agreement’ has inter-relationship with the BusinessProtocol and TransactionFailureRecovery-Handler. The relationship is defined using the association relationship that is refined using a rolename called **determines**.

• \( \forall M, R_{inter} (\Sigma S_{determines} \rho) \)

• \( \forall M, R_{inter} (\Sigma S_{determines} \epsilon) \)

Furthermore, the BusinessProtocol has the association relationship with the TransactionProcessFragment. The relation is refined using a rolename **sequences**. The formal definition of the relation is given below.

• \( \forall M, R_{inter} (\Sigma S_{sequences} \rho) \)

The ‘Party’ has an inter-relationship with the ‘TransactionRole’ using the association relation that is refined using the rolename **has**. This can be written as follows,

• \( \forall M, R_{inter} (\sigma S_{has} \rho) \)

The ‘Policy’ has an intra-relationships with the elements BusinessPolicy, SecurityPolicy, and CompliancePolicy. This relationships are defined below.

**Definition 34.** \( \forall M, R_{intra} ((\Psi_{SP}, \Psi_{CP}, \Psi_{BP}) \land \Psi) \).

¹Since a composite agreement contains two agreements and each agreement has one valid period, it has two periods.
The BusinessPolicy, SecurityPolicy, and CompliancePolicy have integral parts including BusinessRule, ComplianceRule, and SecurityParameter. The internal relationship between them can be written as follows:

- \( \forall M, R_{intra}(\Psi_{BP} \ \& \ bRule); \) the business rules are contained in the business policies. These relationships can be also written as follows,

\[ R_{intra}(bRule \ derivedFrom \Psi_{BP}) \]

The aggregation and derivedFrom is an inverse relation. Badder and Shattler (2001) defined this relationship as follows,

\[ M = (b,a) \in \Delta^I \times \Delta^I | (a,b) \in R^I \]

This inverse relation denotes that a business policy is the aggregation of business rules and business rules are derived from the business policy is equivalent.

- \( \forall M, R_{intra}(\Psi_{CP} \ \& \ cRule); \) or \( \forall M, R_{intra}(cRule \ derivedFrom \Psi_{CP}); \) A CompliancePolicy is the aggregation of ComplianceRules and the ComplianceRules are derived from the CompliancePolicy.

- \( \forall M, (\Psi_{SP} \ \& \ sParameter); \) The SecurityPolicy is the aggregation of the SecurityParameters.

The cardinalities between the BusinessPolicy, CompliancePolicy, and SecurityPolicy and BusinessRule, ComplianceRule, and SecurityParameter are given below:

- \( \forall M, M \in LL [\exists \Psi_{BP}, bRule_i | bRule : i = 1..k(\neq \infty)]; \) For all business policy, it is necessary that there exist at least one business rule and it is possible that there exist more than one business rule in a business policy.

- \( \forall M, M \in LL [\exists \Psi_{CP}, cRule_i | cRule : i = 1..k(\neq \infty)]; \) For all compliance policy, it is necessary that there exist at least one compliance rule and it is possible that there exist more than one compliance rule in compliance policy.

- \( \forall M, M \in LL [\exists \Psi_{SP}, sParameters_i | sParameters : i = 1..k(\neq \infty)]; \) For all security policy, it is necessary that there exist at least one security parameter and it is possible that there exist more than one security parameter in a security policy.

Furthermore, there exist a relationship between ‘BusinessTransaction’ and ‘TransactionProcessFragment’. The relationship is defined below.

**Definition 35.** The ‘TransactionProcessFragment’ is subsumed in BusinessTransaction using the composition relationship. This relationship can be defined as follows,

\[ \forall M , R_{intra}(\Gamma \ a \ \tau) \]
For all business transaction models and for exactly one business transaction, it is necessary that there exist at least one TransactionProcessFragments and it is possible that there exist ‘k’ (must be finite) TransactionProcessFragments. This can be written as follows,

\[ \forall M, \ MUL [\exists \tau, \Gamma | \Gamma : i = 1..k(\neq \infty)] \]

**Definition 36.** A TransactionProcessFragment subsumes the ControlFlowPattern and Activity using aggregation relation. It can be written as follows,

\[ \forall M, \ R_{\text{intra}} ((\Pi, \alpha) \ a \Gamma) \]

The cardinality constraint for TransactionProcessFragment and Activity is defined below,

\[ \forall M, \ MUL [\exists \alpha, \alpha_i | \alpha : i = 1..k(\neq \infty)], \text{for all business transaction model, it is necessary that there exist at least one activity and it is possible that there exist more than one activity in a TransactionProcessFragment.} \]

The ControlFlowPattern is inter-related with ‘Activity’ using association relation. This relationship is refined by the rolename ‘definesOrderOf’. Additionally, the ‘Activity’ has association relationship with the BusinessObject. This relationship is refined using the rolename ‘operatesOn’. These inter-relationships are defined below,

\[ \forall M, \ R_{\text{inter}} (\Pi S_{\text{composes}} \alpha) \]

\[ \forall M, \ R_{\text{inter}} (\Pi S_{\text{operatesOn}} O) \]

The TransactionRecoveryFailureHandler associates the TransactionProcessFragments. This relationship is defined in the following,

\[ \forall M, \ R_{\text{inter}} (\varepsilon S \Gamma) \]

### 8.2.4 Well-Formedness

This section provides the formal description of the well-formedness of Solúbtha BTM. A model must be well-formed to guarantee that its instance-models are logically consistent to the model. A well-formed model must satisfy a list of structural and behavioral properties. I define the well-formed properties of Solúbtha BTM that must be satisfied to ratify the instance-models of Solúbtha BTM are well-formed and therefore they are consistent.

**Definition 37.** The well-formed Solúbtha BTM is defined as follows,
Let $v_1, \ldots, v_n$ be a set of features, a business transaction model is well-formed if and only if $M \models v_n$.

The following properties must hold by the instance-model of Solúbtha BTM:

- $M$ is a directed acyclic graph (Leyman and Reuter, 2000). $\pi_1(\Xi_1) \neq \pi_2(\Xi_2)$ that is, the starting path of an element must not be an end path.
- $\forall M, M = \emptyset$ that is, a business transaction model must not be empty.
- $\forall M, M \models \Xi$, that is, for all business transaction model, a model entails the required design-elements for business transactions.
- $\forall M, (R_{\text{intra}}, R_{\text{inter}}) \subseteq \top$, that is, the the intra-relationships and inter-relationships between the elements must be true which means, they must be correctly defined.
- $\forall \Gamma \in M, \Gamma \neq \emptyset$.
- $\forall \Gamma, \text{IN}^{\text{TPF}} (\Gamma) \models 1 \land \text{OUT}^{\text{TPF}} (\Gamma) \models 1$, that is, each TransactionProcessFragment has one inbound path (connector) and one outbound path.
- $\forall \alpha, \text{IN}^{\text{TTP}} (\alpha) \models 1 \land \text{OUT}^{\text{TTP}} (\alpha) \models 1$, that is, each activity has one inbound and outbound path.
- $\forall \alpha, M \models \mathcal{P} : \alpha_1 \rightarrow \alpha_2 \exists \Pi (\alpha_1, \alpha_2)$, that is, two activities is connected using a ControlFlowPattern.
- $\forall \alpha, \in M, \Pi (\alpha_1, \alpha_2) \neq \emptyset$
- $\forall \Pi^{NJ}, \text{IN}^{\text{TPF}} (\Pi^{NJ}) \models 1 \land \text{OUT}^{\text{TPF}} (\Pi^{NJ}) \models 1$, that is, for AND-Join flow pattern, there is more than one inbound and outbound path.
- $\forall \Pi^{XJ}, \text{IN}^{\text{TPF}} (\Pi^{XJ}) \models 1 \land \text{OUT}^{\text{TPF}} (\Pi^{XJ}) \models 1$, that is, for exclusive-Join (XOR-Join) there is more than one inbound and outbound path.
- $\forall \Pi^{XC}, \text{IN}^{\text{TPF}} (\Pi^{XC}) \models 1 \land \text{OUT}^{\text{TPF}} (\Pi^{XC}) \models 1$, that is, for exclusive-Choice (XOR-Choice), there is one inbound path and more than one outbound paths, yet at runtime, only one path will be selected based on the predefined conditions.
- $\forall \Pi^{MC}, \text{IN}^{\text{TPF}} (\Pi^{MC}) \models 1 \land \text{OUT}^{\text{TPF}} (\Pi^{MC}) \models 1$, that is, for multiple-Choice there is one inbound path and more than one outbound paths, at runtime, more than one paths can be selected based on the predefined conditions.
- $\forall \Pi^{MWS}, \text{IN}^{\text{TPF}} (\Pi^{MWS}) \models 1 \land \text{OUT}^{\text{TPF}} (\Pi^{MWS}) \models 1$, that is, for all multiple instances without synchronization there is one inbound path and more than one outbound path.

An instance-model $M'$ of $M$ is not well-formed if one more of the above properties are not conformed (satisfied).
8.3 Summary

Since UML is not adequately expressive to define the complete formal syntax of Solúbtha BTM, in this section I formalized the model using set theory. In this chapter, I provided the formal definition of the logical structure of Solúbtha BTM and its elements. The relationships between the design-elements were defined. The relationships are of critical importance for a well-formed instance-models of Solúbtha BTM. In this chapter, I defined a list of well-formedness properties. An instance-model can be verified over the well-formedness properties.

The formal definition was provided to enable business transactions to verify the syntactic correctness of the instance-models which must conform to Solúbtha BTM.
Chapter 9

Formal Semantics of Solúbtha BTM

9.1 Introduction

The formal semantics define the behavior of a system by providing explicit meaning for the operations performed during business transactions (Slonneger and Kurtz, 1995). In this chapter, I provide the formal semantics of the design-elements of Solúbtha BTM. Additionally, I provide the formal interpretation of the behavioural properties that are written in natural language. Also, I present a list of algorithms which I develop in this thesis to enable the automation of the flexible behaviour of business transactions.

I investigated the most potential candidate property specification languages to formalize the behavioural properties of flexible business transactions. The candidate formal languages are, Linear Temporal Logic (LTL) (Pnueli, 1977), Formal Contract Language (FCL) (Governatory et al., 2006), and Computational Tree Logic (CTL) (Clarke et al., 1999). FCL is the family of deontic logic, the LTL is a member of temporal logic, and CTL is the family of Branching Temporal Logic. A comparative analysis of LTL, FCL, and CTL has been provided by Elgamaal et al., (2011) where the authors argue in favor of LTL and ruled out the possibilities of using FCL and CTL for formalizing compliance properties of business processes for the following reasons:

- LTL and CTL are not expressible for permission while fairness is not expressible using FCL and CTL.
- Existential properties are not expressible using FCL and LTL.
- LTL is more intuitive than CTL.
- CLT is relatively more complex than LTL from the verification point of view.

The existential quantifier is essential for formalizing behavioral properties. FCL and LTL do not have this quantifier. Therefore, CTL is the most suitable candidate for transforming the plain text expressions into formal expression.

Furthermore, I find Automata (Beek et al., 2007), Petri net (Resig and Rosenberg, 1998), and Process Algebra (Milner, 1999) are the potential candidates for formalizing the semantics of the design-elements of Solúbtha BTM. I give a brief introduction of these formalisms in Appendix I. I compare these formal...
languages over a set of features to identify the most suitable formalism for my model. Table 9.1 shows a high-level comparison of these formalisms.

<table>
<thead>
<tr>
<th>heightFormalism</th>
<th>Expressiveness</th>
<th>Concurrency</th>
<th>Temporal Specification</th>
<th>Property Specification</th>
<th>Tool Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automata</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Process Algebra</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Petri Net</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 9.1 shows that all three are powerful formal languages. They all are highly expressive, support concurrency, enable defining temporal properties, and have tool support. However, Petri Net and Process Algebra do not have any support for specifying behavioural properties stemming from the business perspective elements. The automata has the support from at least two property specification formal languages including CTL and LTL. Additionally, there is a tool called UPPAAL that can be used to design and verify the models that are developed using automata and it supports specifying the properties using both CTL and LTL. Therefore, I choose automata to formalize the semantics of the design-elements of Solúbtha BTM.

Automata has several variants. Time Extended Finite State Machine (TEFSM) is the one which supports specifying temporal properties. It is able to support defining the clock semantics of business transaction states. Also, it supports defining the nondeterministic state machines. Therefore, I use TEFSM in my formalization.

### 9.2 Mapping Solúbtha BTM Design-elements to TEFSM

This section describes mapping of Solúbtha BTM design-elements to TEFSM. To start with, I provide the definitions (taken from Baier Katoen, 2008) of some basic entities that are used in the next subsections.

- **State Space:** A state space in a Θ(TS) is a set of states, can be written as Υ = q₀,...,qₙ where Υ denotes the state space.

- **Predecessor:** Let q and q’ be two different states (q, q’) ∈ Q, and a ∈ Ω is an action. The predecessor can be defined as follows:

  \[
  \text{PRE}(Q, a) := q’ \xrightarrow{a} q
  \]

- **Successor:** Let q and q’ be two different states (q, q’) ∈ Q, and a ∈ Ω is an action. The successor can be defined as follows:

  \[
  \text{SUC}(Q, a) := q \xrightarrow{a} q’
  \]
9.2. MAPPING SOLÚBTHA BTM DESIGN-ELEMENTS TO TEFSM

- **Start State**: A start state in \( \Theta(TS) \) can be defined as below:
  \[
  \text{PRE}(q) := \emptyset
  \]

- **Terminal State**: A terminal state in \( \Theta(TS) \) can be defined as below:
  \[
  \text{POST}(q) := \emptyset
  \]

### 9.2.1 Business Transaction

The formal semantics of business transaction state machine is defined as follows, (taken from (Lallali et al., 2007)).

**Definition 38.** A machine \( \Phi \) is a tuple \( \Phi := (Q, \Omega, C, q_0, F, \Theta(TS), \text{Inv}) \)

Where:

- \( Q = q_0, \ldots, q_n \) is a finite set of states;
- \( \Omega = a, b, c, \ldots \) is the alphabet of the actions;
- \( V \) is the finite set of data variables;
- \( C \) is the finite set of clocks;
- \( q_0 \in Q \) is the finite set of clocks;
- \( F \subseteq Q \) Set of final states;
- \( \Theta(TS) \subseteq Q \times A \times 2^Q \) is transition relation where:
  \[
  A \text{ is set of transaction action } \Omega \times P(V) \times \phi(C) \times \mu(V) \times 2^C
  \]
  Where:
  - \( P(V) \times \phi(C) \) denotes the guarding conditions;
  - \( \mu(V) \) is the data variable update function;
  - \( 2^C \) denotes set of clocks to be reset.

- **Inv**: \( Q \mapsto \gamma(C) \) assigns a set of time invariants.

The set of all subsets of \( Q \) is the power set \( 2^Q \). The clocks are classified into global clock \( G_{\text{clock}} \) and local clock \( L_{\text{clock}} \). The global clock expresses the global time whereas the local clock expresses the local time. The values of these clocks can be compared to non-negative real numbers (Lillai et al., 2007). A transition \( \Theta(TS) \) of state machine is annotated by guard condition, actions, data variable update, and clock resets, which can be written as follows,
• $\Theta(TS) = q_i \rightarrow q_n$ where $I = (<\text{cond}, \text{a} \in \Omega, [\mu(V), R_c])$

The transition is presented in figure 9.1 using a state transition diagram.

![State Transition of Business Transaction State Machine](image)

**Figure 9.1: State transition of business transaction state machine.**

The figure shows the transition between state $q_n$ and $q_{n+1}$ is annotated using guard condition, action, data variable update function, and clock reset component. The guard conditions are Boolean conditions that hold a value either true or false. A transaction happens if and only if the condition holds. The following subsections describe the mapping of business transaction design-elements into TEFSM.

### 9.2.2 Transaction Process Fragment

The TransactionProcessFragment ($\Gamma_0 \ldots \Gamma_n \in M$) is mapped to state machine $(q_0, q_n) \in \Phi(\Gamma)$ such that,

$\Phi(\Gamma) = \{(q_0, q_n), \Omega, \nu, C, q_0, q_n, \Theta(TS), Inv\}$

This definition is illustrated using a scenario of the AVERS supply chain process that is composed of a set of transaction process fragments.

**Example.** The purchase order fulfillment business transaction of the AVERS supply chain scenario comprises a list of transaction process fragments such as purchase order registration and credit worthiness check. The state machine and the transition of purchase order transaction process fragment can be defined as follows,

$\Phi(\Gamma) = \{(q_0, q_1), \{\text{RegisterPurchaseOrder}, \emptyset, \emptyset, q_0, q_1, \Theta(TS), \emptyset\}\}$

$\Theta(TS) = (q_0, \text{RegisterPurchaseOrder}, <\text{POverification = true}>, \emptyset, \emptyset) q_1$)

Figure 9.2 shows the state machine and the transition in RegisterPurchaseOrder transaction process fragment.

![State Transition of Transaction Process Fragment](image)

**Figure 9.2: State Transition of Transaction Process Fragment.**

Figure 9.2 shows an example of conditional transition. The action RegisterPurchaseOrder is performed if and only if $<\text{POverification = true}>$ holds. The ‘(·)’ sign denotes the unused entity in a transition. The transition relation is a function that at runtime relies on various states contained within the transition space of a transaction process fragment.
The state space of the purchase order registration transition comprises a set of states include, ‘SubmitPurchaseOrder’, ‘ReceivePurchaseOrder’, ‘VerifyPurchaseOrder’, and ‘RegisterPurchaseOrder’. These states are contained within the space of the transition between $q_0$ and $q_1$. It can be written as follows,

$$\Upsilon = \{ \text{SubmitPurchaseOrder, ReceivePurchaseOrder, VerifyPurchaseOrder, RegisterPurchaseOrder} \}$$

The state machine produces the external and internal outcomes. The external outcomes are visible from outside. Committed, Aborted, Cancelled are three external outcomes that depend on internal outcomes include Sent, Received, Verified, and Registered etc.

9.2.3 Activity

The formal semantics of ‘Activity’ is similar to the semantics of transaction process fragments. Figure 9.3 shows the mapping of activity to state of business transaction state machine.

This graphical representation can be written formally as follows,

$$\Phi(\alpha) = \{ \{q_0, q_1\}, \{\text{SendPurchaseOrder}\}, \emptyset, \emptyset, q_0, q_1, \Theta(TS), \emptyset \}$$

Figure 9.3 shows that there is no guard condition specified in the transition between the activity $q_0$ and $q_1$. The state space of activity state transition contains only one activity $q_0$ which can be expressed as follows,

$$\Upsilon(\alpha) = \text{SendPurchaseOrder}$$

A state representing the activity is reachable to another state using the control flow patterns is provided in the next subsection.

9.3 Mapping Control Flow Patterns to TEFSM

The ControlFlowPattern is a crucial functional perspective element of business transactions. The execution behaviors of business transactions is designed (or defined) using control flows. In this section, I present the formal semantics of the control flow patterns.

9.3.1 Sequence

A sequential business transaction state machine is a series of countable infinite number of states that are connected via control connector without any split or loop. This control flow pattern represents the sequential
behavior of business transactions. In this behavioral model, the sequential connector connects the source and target states that represent the activities of a transactional business process. The formal representation of sequence control pattern is given below,

\[
\Phi_1(\alpha_1) = \{q_0, q_1\}, \alpha_1, V, C, q_0, q_1, \Theta(TS), Inv \quad \text{(I)}
\]

\[
\Phi_2(\alpha_2) = \{q_0, q_2\}, \alpha_2, V, C, q_0, q_2, \Theta(TS), Inv \quad \text{(II)}
\]

Equation I and II represent the formal definition of the source and target states. The sequence of these states can be written by combining these equations as follows,

\[
\Phi(\Pi_{seq}(\alpha_1, \alpha_2)) = \{q_0, q_1, q_2\}, \{\alpha_1, \alpha_2\}, V, C, q_0, q_2, \Theta(TS), Inv \}
\]

The sequential TEFSM is presented graphically in figure 9.4.

Figure 9.4: Formal Representation of Sequential Behavioral Model.

Example. Verify purchase order and Register purchase order are two activities in purchase order fulfillment business process of the AVERS supply chain scenario. These two activities are sequential in the process. The business transaction state machine of this sequential state model can be written as follows:

\[
\Phi_1(\alpha_1) = \{q_0, q_1\}, \{\text{VerifyPurchaseOrder}\}, \emptyset, \emptyset, q_0, q_1, \emptyset, \emptyset
\]

\[
\Phi_2(\alpha_2) = \{q_0, q_2\}, \{\text{RegisterPurchaseOrder}\}, \emptyset, \emptyset, q_0, q_2, \emptyset, \emptyset
\]

\[
\Phi(\Pi_{seq}(\alpha_1, \alpha_2)) = \{q_0, q_1, q_2\}, \{\text{VerifyPurchaseOrder, RegisterPurchaseOrder}\}, \emptyset, \emptyset, q_0, q_2, \emptyset, \emptyset
\]

9.3.2 XChoice

A business transaction state machine contains the XChoice control flow pattern which represents the non-deterministic automata where the subsequent state of operations is not determined in advance. Rather, it is a decision problem and the state machine decides the state based on the evaluation of given conditions. This control flow pattern guarantees flexibility at runtime in terms of execution of activities contained within the scope of a transactionalProcessFragment.

Unlike sequential pattern, the XChoice control pattern split a state into two or more states. The transition happens from the source state to either of a given set of target states depending on which of the states holds the given condition. This implies that the transition in XChoice is purely conditional. Precisely, the selection of outgoing state or path depends on the evaluation of the given condition.

The XChoice pattern contained in business transaction state machine is formally defined as follows,

\[
\Phi(\Pi^{XChoice}) = \{q_0, q_n\}, \Omega, V, C, q_0, q_n, \emptyset, \emptyset, \Theta(TS), Inv
\]
In this definition, $\varepsilon-\Theta(TS)$ denotes the ‘epsilon transition’ that determines the nondeterministic business transactions. Baier and Katoen describes the ‘epsilon transition’ as a type of transition that allows the system to change the states with no input. At runtime, the transition leads the system to any of its outgoing states. The graphical representation of XChoice state machine is depicted in figure 9.5.

![Figure 9.5: XChoice Automation.](image)

The state transition from $q_3$ to $q_5$ and $q_6$ is purely conditional as the control flows to any of these two states depending on which one of them holds the transition condition. Therefore, XChoice automation is nondeterministic.

**Example.** The inventory reservation of the purchase order fulfillment business process is nondeterministic and relies on transition conditions. The control flow to the next state depends on the conditions of the product availability. The transition conditions are:

- **Cond-1**: if product is available then schedule the ordered goods.
- **Cond-2**: if product is not available then replenish the ordered goods.

The inventory reservation automation is defined formally in the following considering these transition conditions,

$$\Phi(\Pi^{XChoice}) = \{ [q_3, q_3, q_5, q_6], \{\text{ReserveInventory, ReplenishOrderedGoods}\}, 0, 0, q_3, \{q_4\},$$

$$(\varepsilon-\Theta(TS_1), \varepsilon-\Theta(TS_2)) \emptyset \}$$

Where:

- $\varepsilon-\Theta(TS_1)[q_3] \{ \text{ProductAvailability} = \text{True}, \text{ReserveInventory}, 0, 0 \} = q_5$
- $\varepsilon-\Theta(TS_2)[q_3] \{ \text{ProductAvailability} = \text{False}, \text{ReplenishGoods}, 0, 0 \} = q_6$

These two transition conditions are mutually exclusive which can be written as follows,

$$\varepsilon-\Theta(TS_2)[q_3] \{(\text{ProductAvailability} = \text{True}) \lor (\text{ProductAvailability} = \text{False})) \rightarrow (\text{ReserveInventory} \oplus \text{ReplenishGoods}) \rightarrow (((\text{ReserveInventory} \lor \text{ReplenishGoods}) \land \neg(\text{ReserveInventory} \lor \text{ReplenishGoods}))$$

$\in \Omega), 0, 0 \}$$

The example of XChoice is shown in the figure 9.6.
9.3.3 NChoice

Unlike XChoice, NChoice allows more than one transitions which can be conjoined. However, the transition is conditional like the XChoice. At runtime, if two or more states are selected from a given set of states then the logical conjunction combines the transition conditions. This pattern promotes the parallelism at runtime and thereby ensures the flexible execution of business transactions. The automata of the NChoice pattern is presented below,

$$\Phi(\Pi^{NChoice}) = \{ [q_0, q_{n-1}, q_0, q_{n+1}], \Omega, V, C, q_0, \{q_{n+1}\}, \varepsilon-\Theta(TS), Inv \}$$

Where:

- \( \varepsilon-\Theta(TS_1)[q_0] \{ \text{cond}_1, \alpha_1, \emptyset, \emptyset \} = q_{n-1} \)
- \( \varepsilon-\Theta(TS_2)[q_0] \{ \text{cond}_2, \alpha_2, \emptyset, \emptyset \} = q_n \)
- \( \varepsilon-\Theta(TS_n)[q_0] \{ \text{cond}_n, \alpha_n, \emptyset, \emptyset \} = q_{n+1} \)
- \( \varepsilon-\Theta(TS_n) = \Theta(TS_1) \times \Theta(TS_2) \times \ldots \times \Theta(TS_n) \)
- \( \varepsilon-\Theta(TS_n)[q_0] \{ (\text{cond}_1 \land \text{cond}_2 \land \ldots \land \text{cond}_n), (\neg\text{cond}_1 \land \neg\text{cond}_2 \land \ldots \land \neg\text{cond}_n), (\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n), (\neg\alpha_1 \land \neg\alpha_2 \land \ldots \land \neg\alpha_n), \emptyset, \emptyset \} \)

Example. Shipment process fragment in purchase order fulfillment process of the AVERS supply chain case is a nondeterministic automata which is designed using NChoice pattern for some states in particular, choosing the transportation mode schema is designed using this flow pattern. At runtime, the business transaction state machine generates one or more transition systems based on the predefined Boolean conditions.

The following conditions are examples that an instance should satisfy:

- **Cond 1**: Buyer prefers railroad for shipping ordered goods.
- **Cond 2**: Buyer prefers Air Cargo for shipping ordered goods.
- **Cond 3**: Buyer does not prefers Sea Cargo for shipping ordered goods.

There is a possibility that a buyer may prefer more than one of these shipment methods such as if the buyers are from different continent then the buyer may prefer more than one transportation method.
9.3. MAPPING CONTROL FLOW PATTERNS TO TEFSM

Taking this conditions as a set of inputs, the business transaction state machine is presented in figure 9.7 and formally defined below,

- $\Phi(\alpha_1) = \{ q_5, q_6 \}, SelectAir, 0, 0, q_5, q_6, \varepsilon-\Theta(TS_1), 0 \}$
- $\Phi(\alpha_2) = \{ q_5, q_7 \}, SelectRailroad, 0, 0, q_5, q_7, \varepsilon-\Theta(TS_2), 0 \}$
- $\Phi(\alpha_3) = \{ q_5, q_8 \}, \neg(SelectSea), 0, 0, q_5, q_8, \varepsilon-\Theta(TS_3), 0 \}$

Where:

- $\varepsilon-\Theta(TS_1)[q_5] \{ (TransportationMode.Air = True), SelectAir, 0, 0 \} = q_6$
- $\varepsilon-\Theta(TS_1)[q_5] \{ (TransportationMode.Railroad = True), SelectSea, 0, 0 \} = q_7$
- $\varepsilon-\Theta(TS_1)[q_5] \{ (TransportationMode.Sea = False), SelectSea, 0, 0 \} = q_8$
- $\Phi(\Pi^{NChoice}) = \{ \{ q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11} \}, \{ q_5, q_6 \}, \{ q_5, q_7 \}, \{ q_5, q_8 \}, \{ q_6, q_9 \}, \{ q_7, q_{10} \}, \{ q_8, q_{11} \} \}.
  \{ SelectAir \land SelectRailroad \}, 0, 0, q_5, q_9, q_{10} \}$
  \{ (TransportationMode.Air = True),
  (TransportationMode.Railroad = True) \}, 0}$

![Figure 9.7: Example of NChoice Automata.](image)

9.3.4 XJoin

The state transitions generate many outgoing paths that can be synchronized using this control flow pattern. The XJoin refers to the exclusive OR (XOR) synchronization that allows only one incoming path
mutually exclusively out a set of paths. It complements the NChoice flow pattern. An NChoice without synchronization is prone to produce inconsistent outcomes.

**Example.** Referring the example given in section 9.4.3, if XJoin is used in business transaction automata then at runtime, the state machine will select only one path at runtime and the remaining incoming paths will be discarded automatically. Figure 9.8 depicts the XJoin control business transaction automata that is designed using XJoin.

![Figure 9.8: Example of XJoin Automata.](image)

At runtime, only one path is accepted and the remaining paths are discarded. The XJoin automata shown in figure 9.8 can be written formally as below:

\[
\Phi(\Pi^{XJoin}) = \{ \{ q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12} \}, \{ q_5, q_6 \}, \{ q_5, q_7 \}, \{ q_5, q_8 \}, \{ q_6, q_9 \}, \{ q_7, q_{10} \}, \{ q_8, q_{11} \}, \{ SelectAir \}, 0, 0, q_5 \} \}
\]

In this definition, only transportation mode 'Air' is selected which means only \(\text{TransportationMode.Air} = \text{True} \) transition condition holds at runtime.

### 9.3.5 NJoin

Like the XJoin, this control flow pattern complements the NChoice flow pattern. However, the NJoin flow pattern allows more than one incoming paths for synchronization. The state transition does not happen
unless all incoming paths arrive at NJoin synchronizer. This implies that the conditions defined with the states hold and the actions carried out successfully.

**Example.** The example given in section 9.4.3 is referenced to illustrate the NJoin control flow pattern. Figure 9.9 illustrates business transaction automata designed using NJoin control flow pattern.

The NJoin control flow pattern is shown in figure 9.9 is mapped to automata that can be written as follows,

\[
\Phi(\Pi^{NJoin}) = \{ \{ q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{5}, q_6, q_7, q_8, q_9, q_{10}, q_{11} \}, \{ \text{SelectAir} \land \text{SelectRailRoad} \land \text{SelectSea} \}, q_5, \{ q_6, q_7, q_8 \} \mid (\text{TransportationMode.Air} = True), (\text{TransportationMode.Railroad} = True), (\text{TransportationMode.Railroad} = True) \}\}

The formal definition of NJoin shows that the synchronizer waits to arrive all incoming paths. The next transition starts if and only if all incoming paths have arrived successfully.

### 9.3.6 Multiple Instances Without Synchronization

As opposed to XJoin and NJoin, this control flow pattern is used for designing business transaction automata where instances created form the transition system run in parallel without any synchronization required. The instances can be joined to other fragments of business transaction automata or to the terminate state. The automation of multiple instances without synchronization does not depend on conditions. The automata can be written as follows,

\[
\Phi(\Pi^{MWS}) = \{ \{ q_{11}, q_{12}, q_{13} \mid q_{11}, q_{12} \}, \{ q_{13} \}, \alpha_1, \alpha_2, \emptyset, q_{11}, \{ q_{12} \}, \{ q_{13} \}, \emptyset, \emptyset \}\}
Figure 9.10 shows an example of multiple instances without synchronization automata. The instances $q_{12}$ and $q_{13}$ are not synchronized and no further transition happened in this automation.

9.3.7 Interleave

Interleaving refers to execute transaction process fragments or activities in any order. This flow pattern promotes nondeterminism by making the execution order fully nondeterministic.

Consider $\Phi$ and $\Phi'$ are two state machines. The business transaction automata can be written as:

- $\Phi = \{ Q, \Omega, V, C, q_0, F, \varepsilon\cdot\Theta(TS), Inv \}$
- $\Phi' = \{ Q', \Omega', V', C', q'_0, F, \varepsilon\cdot\Theta(TS)', Inv' \}$

The $\Phi''$ is the new state machine produced by unifying these two state machines $\Phi$ and $\Phi'$. Interleaving is defined (taken from (Baier, 2008)) below:

$\Phi || | \Phi' = \{ Q \times Q', \Omega \times \Omega', V \times V', C \times C', q_0 \times q'_0, F \times F, \varepsilon\cdot\Theta(TS) \times \varepsilon\cdot\Theta(TS)', Inv \times Inv' \}$

where transition follows the rules below:

- $\varepsilon\cdot\Theta(TS) = (q_1 \overset{a}{\rightarrow} q'_1) \rightarrow < q_1, q_2 > < q'_1, q'_2 >$
- $\varepsilon\cdot\Theta(TS)' = (q_2 \overset{a}{\rightarrow} q'_2) \rightarrow < q_1, q_2 > < q'_1, q'_2 >$

**Example.** If three transaction process fragments shipment processing, invoice processing, and payment processing running in parallel then payment processing fragment can be executed in any order, which means it can occur before or after invoice processing or before or after the shipment processing.

9.3.8 Standard Discriminator

The Standard Discriminator can be defined as follows:
9.4. FORMALIZATION OF BEHAVIORAL PROPERTIES

\[ \Phi(\Pi^{SD}) = \{ \{ q_0, q_1, q_n, q_{n+1} \}, \{ q_0, q_n \}, \{ q_0, q_{n+1} \}, \{ \alpha_1, \alpha_2, \alpha_n \}, \emptyset, \emptyset, \emptyset, \emptyset, \{ q_1, q_n, q_{n+1} \}, \varepsilon - \Theta(TS), \emptyset \} \]

Example. This example is taken from the order processing scenario of Auto Inc. business case where suppliers are selected on the fly especially to replenish goods. The buyer sends purchase order to three different suppliers and starts negotiation with the one who responds to the purchase order first. The remaining suppliers are discarded. Standard discriminator control flow pattern designs automata that automates such business cases. Figure 9.11 depicts the standard discriminator automata.

This example mapped to the automata given below:

\[ \Phi(\Pi^{SD}) = \{ \{ q_0, q_1, q_2, q_3 \}, \{ q_0, q_1, q_2 \}, \{ q_0, q_3 \}, \{ SelectSupplier_1, SelectSupplier_2, SelectSupplier_3 \}, \emptyset, \emptyset, \emptyset, \emptyset, \{ PurchaseOrderAccepted = True \}, \emptyset \} \]

9.4 Formalization of Behavioral Properties

I provide the formal semantics of behavioral properties of the flexible business transactions. In addition, I define the operational semantics business transactions. Also, I describe the eventual failure atomicity which is the foundation of the flexible business transactions.

As mentioned in Section 9.1 I used CTL to formalize the behavioral properties of business transactions. A brief introduction of CTL is given in the next sub-section.

9.4.1 An Overview of Computational Tree Logic (CTL)

CTL is a branching-time logic that is expressive for the formulation of an important set of system properties (Baier and Katoen, 2008). According to Baier, a branching-time denotes that for each moment there may be several different possible futures. This means that CTL is a state based language that aids in defining the
nondeterministic temporal properties. Baier also describes that the semantics of branching-temporal logic are defined in terms of infinite, directed tree of states rather than an infinite sequence.

The two-stage syntax of CTL separates the state formulae from the path formulae. The state formulae are the assertions about the atomic propositions in the states and their branching structure whereas the latter expresses the temporal properties of paths. CTL is simpler than LTL in terms of path formulae. CTL path and state quantifiers, state formulae, and path formulae are listed below (Clarke et al., 2006),

- **Path Quantifier**
  - A (Textual), ∀ (Notational) denotes **For All**.
  - E (Textual), ∃ (Notational) denotes **There Exist**.

- **Temporal Quantifier**
  - X (Textual), ⚪ (Notational) denotes **Next**.
  - F (Textual), ◯ (Notational) denotes **Eventually**.
  - G (Textual), □ (Notational) denotes **Globally**.
  - U denotes **Until**.
  - R denotes **Release**.

- **Boolean Connectors**
  - ∨ denotes **Disjunction**.
  - ∧ denotes **Conjunction**.
  - ¬ denotes **Negation**.

- **State formulas**
  - P ∈ AP.
  - If f and g are state formulas then ¬f, f ∧ g, f ∨ g are state formulas.
  - If f is a state formula then ∃f, f ∧ g, ∀ f ∨ g are state formulas.

- P ∈ AP.

- Every state formula.

- If f is a state formula then ∃f, f ∧ g, ∀ f ∨ g are state formulas.
9.4. **FORMALIZATION OF BEHAVIORAL PROPERTIES**

9.4.2 **Theory of Flexible Behavior**

While developing the techniques for defining the properties of a flexible business transactions, I realize that it is not possible to achieve high flexibility without extending the state of the art transaction principle. In this thesis, I extend the atomicity principle and called the extension *eventual failure atomicity*. The formal description of the eventual failure atomicity is provided in this section.

**Theorem 1.** *For any business transaction* $\tau$ *the abortion of* $\tau$ *happens eventually such that,*

$$\forall \Gamma, \forall \alpha, \text{commit}(\tau) \rightarrow \Diamond \text{abort}(\tau) \text{ such that } \zeta(\tau) \leftarrow \zeta(\Gamma) \lor \zeta(\alpha) \lor \lambda(\Sigma)$$

**PROOF:** Let us consider a business transaction with two sub-transactions $= (T_1 (\Gamma_1), T_2 (\Gamma_2))$ where $\Gamma_1, \Gamma_2$ represent two sub-transactions. If $\Gamma_2$ fails or a constraint originates from agreement corresponding to $\Gamma_2$ is violated then the business transaction system that is built upon eventual atomicity will invoke failure prevention techniques and continue invoking techniques until the business transaction has recovered from the failure. A business transaction would fail eventually if and only if,

- $e^{\text{Retry}}(\Gamma) \rightarrow \mathcal{F}(\Gamma_i) \lor \mathcal{F} (\alpha_j)$, retry operation was not successful.
- $e^{\text{Repair}}(\zeta) \rightarrow \mathcal{F}(\Gamma_i) \lor \mathcal{F} (\alpha_j)$, repair operation was not successful.
- $e^{\text{Substitute}}(\Gamma) \rightarrow \mathcal{F}(\Gamma_i)$, substitution of transaction process fragment was not successful.
- $e^{\text{Substitute}}(\alpha) \rightarrow \mathcal{F}(\alpha_j)$, substitution of failed activity was unsuccessful.
- $e^{\text{Undo}}(\Gamma) \rightarrow \mathcal{F} (\text{Undo})$, undo attempt was not completed successfully completed.
- $e^{\text{Resume}}(\Gamma) \rightarrow \mathcal{F} (\text{Resume})$, resume attempt was not successfully completed.
- $e^{\text{Mutualize}}(\Sigma) \rightarrow \mathcal{F} (\text{Mutualize})$, mutualize of an agreement was unsuccessful.

I constructed a truth table based on eventual failure atomicity principle. The truth table below demonstrates the logical construction of the eventual failure atomicity.

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$\neg(\neg T_1), \neg(\neg T_2)$</th>
<th>$T_1 \land T_2$</th>
<th>Result($\tau$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committed</td>
<td>Committed</td>
<td>$\neg(\neg T_1), \neg(\neg T_2)$</td>
<td>Committed $\land$ Committed</td>
<td>Committed</td>
</tr>
<tr>
<td>Committed</td>
<td>$\neg$Committed</td>
<td>$\neg$Committed($T_2$)</td>
<td>Committed $\land$ Committed</td>
<td>Committed</td>
</tr>
<tr>
<td>$\neg$Committed</td>
<td>Committed</td>
<td>$\neg$Committed($T_1$)</td>
<td>Committed $\land$ Committed</td>
<td>Committed</td>
</tr>
<tr>
<td>$\neg$Committed</td>
<td>$\neg$Committed</td>
<td>$\neg$Committed($T_1), \neg$Committed($T_2$)</td>
<td>Committed $\land$ Committed</td>
<td>Committed</td>
</tr>
</tbody>
</table>

In the truth table, the complement of every negative outcome of transactions $T_1$ and $T_2$ leads to positive
state of business transactions eventually. A business transaction fails eventually if all recovery operations fail. Figure 9.12 depicts the eventual failure atomicity.

While a sub-transaction is running, it may generate one to many states. However, it produces one of the two outcomes that is either the sub-transaction is committed successfully or aborted. In business transactions, the state transition depends on these outcomes. In Figure 9.12, sub-transaction ‘T₁’ has completed successfully which leads to the transition from ‘T₁’ to sub-transaction ‘T₂’. The transition would not happen if T₁ was not completed successfully. This is the classical feature of business transaction however, the eventual failure atomicity extends this notion and this thesis developed techniques that implement this extended notion. Consider the business transaction shown in Figure 9.12, as soon as a fault (denoted by $\text{F}(T₂)$) is detected at runtime, the failure recovery handler (shown in figure 9.12) is invoked, the context is checked and the failure recovery policy $\varepsilon$ is initiated. The business transaction system issues the abortion instruction that forces abortion of completed sub-transaction $T₁$ eventually if and only if the recovery techniques are failed.

9.4.3 Preliminaries

In this section, I describe the basics that are important and relevant to business transaction behavior.
9.4. FORMALIZATION OF BEHAVIORAL PROPERTIES

A. Transaction Scope

Transaction scope refers to the boundary ‘[X, Y]’ where ‘X’ and ‘Y’ denote start and end point of the boundary respectively. A transaction scope can have one or more sub-scopes. For instance, the boundary of the AVERS purchase order fulfillment business transaction is,

\[ \text{Scope} = [\text{register purchase order transactional process fragment, deliver purchase order transaction process fragment}] \]

The starting point ‘register purchase order transactional process fragment’ and the endpoint ‘deliver purchase order transaction process fragment’ can have many sub-scopes. For instance, the sub-scope of ‘register purchase order transaction process fragment’ is,

\[ \text{Scope} = [\text{send purchase order, confirm purchase order registration}] \]

Figure 9.13 depicts the transaction scope and sub-scope.

![Figure 9.13: Transaction Scope and Sub-Scope.](image)

B. Commit

Commit is an instruction that a business transaction coordinator issues to persist the changes that was made in the database by performing certain operations during business transaction (Bernstein and Newcomer, 1997). This is the standard definition of commit. However, in this thesis, database persistency is not considered. Rather, commit here refers to a command to complete an activity or a transaction process fragment.
of a transactional business process. When all transaction process fragment are completed successfully, the business transaction is said to be committed.

C. Partial Commit

The successful commit of a transaction process fragments or an activities do not necessarily mean that a transaction process fragment has committed by satisfying the corresponding constraints. A transaction process fragment may have committed nevertheless one or more of its constraints may have not been satisfied. Such commits are called as partial commit. In addition, partial commit in this thesis is viewed as partial execution of transactional processes, which means the business transaction of a process has been completed however one or more activities of the process were not completed successfully.

D. Soft Rule

In transactional business processes, activities are driven by the rules such as business rules. This thesis introduces the Soft type rules in the context of business transaction. The formal definition of soft rule is given below,

**Definition 39.** Let us assume that $\alpha$ is a sub-transaction of $\Gamma$. A rule such as business rule ‘bRule’ is specified to constrain the execution of $\alpha$. This rule is called the soft rule for $\alpha$ if and only if,

- for any execution, at any time, if rule has been violated, then $\alpha$ commits in near future, this can be written as follows,

$$\lambda(Rule) \rightarrow \lozenge \text{committed } \alpha.$$

E. Hard Rule

Hard rule is the other type rule I introduced in this thesis. The formal definition of hard rule is given below,

**Definition 40.** Consider $\alpha$ is a sub-transaction of $\Gamma$ and a rule is specified to govern the execution of $\alpha$. This rule is called the hard rule for $\alpha$ if and only if,

- for any execution, at any time, if rule has been violated, then $\alpha$ aborts in near future, this can be written as follows,

$$\lambda(Rule) \rightarrow \lozenge \text{aborted } \alpha.$$

9.4.4 Transactional Properties and Behavioral Models

In this section, the fault-tolerant and failure-resilient behavioral properties are formally defined. These properties are defined using CTL. It is important noting that the fault propositions that were presented
in section 5.7 are embedded in the transactional properties. Additionally, this section presents the fault-tolerant and failure-resilient behavioural models that rely on these properties. In this section, I provide the algorithms that automate these behavioral models at runtime.

9.4.4.1 Fault Tolerant Business Transaction

The fault-tolerant behavioral model considers two scenarios: (i) fault-tolerant business transaction in case of violation of soft type business, compliance, and security rules, and (ii) fault-tolerant business transaction in case of the failure of a non-vital activity. The fault-tolerant behavioral models are presented in this section. These models represent the techniques of flexible business transactions. The transactional properties are formalized using CTL. The core property of fault tolerant business transactions is defined below,

Behavioral Property – 1: \((\forall \alpha, \forall \Gamma) \in M_T P\) commit\((M_T P)\) \(\rightarrow\) \(\Diamond\) committed\((M_T P)\) such that, \((bRule \lor cRule \lor sRule) \rightarrow \lambda(\Sigma), trigger^{ignore} \text{ if and only if } RType ((cRule, bRule, sRule) = \text{soft})\)

Where:

- \(\lambda(\Sigma) \leftarrow \lambda(\Psi_{BP}) \leftarrow \lambda(bRule)\), violation of a business rule leads to violation of the business policies which eventually results the violation of agreement.
- \(\lambda(\Sigma) \leftarrow \lambda(\Psi_{SP}) \leftarrow \lambda(sParameter)\), a violation of security constraints results in the violation of security policies which eventually results a violation of agreement.
- \(\lambda(\Sigma) \leftarrow \lambda(\Psi_{CP}) \leftarrow \lambda(cRule)\), a violation of compliance rules results in the violation of security policies which eventually leads to a violation of agreement.

If the value of RType of a rule is soft, the business transaction system ignores the violation of that rule.

If a transaction is committed with violation of agreement then by definition it is a partial commit of ‘\(\tau\)’ which turns into full commit of ‘\(\tau\)’ by ignoring the violations. The formal definition of ignore is provided below,

**Definition 41.** Ignore := continue\((\tau) \rightarrow \text{execute} \left(\bigcirc \alpha \bigcirc \Gamma \in \tau\right)\)

Two application level operators **continue** and **execute** are introduced to enable defining the ignore operation.

Behavioral Property – 2: Ignore\((\lambda(\Psi_{BP})(\alpha \lor \Gamma) \lor \lambda(\Psi_{SP})(\alpha \lor \Gamma) \lor \lambda(\Psi_{CP})(\alpha \lor \Gamma)) \rightarrow \text{execute} \left(\bigcirc \alpha, \bigcirc \Gamma \in \tau\right)\)

Figure 9.14 presents a fault tolerant behavioral model.
In figure 9.14, $q_0$ and $q_1$ represent the states of business transaction automata. While a business transaction is running, the transition from $q_0$ to $q_1$ must satisfy the transition conditions that can be rules or other constraints. The behavioral model in figure 9.14 shows that if the transaction failure recovery handler detects a fault, then the recovery handler checks the RType; if RType is soft type then the business transaction system tolerates the fault by ignoring the fault.

The behavioral model shown in figure 9.14 enables the business transaction systems to tolerate business, compliance, and security rule violation. Moreover, Solúbtha BTM supports tolerating faults regarding the activities contained in transaction process fragments that compose the end-to-end transactional business processes.

The algorithm 1 is developed to underpin the automation of fault-tolerant behavioral model of business transactions.

**Algorithm 1**: Automation of Fault Tolerant Behavioral Model based on Ignore Recovery Policy
Require: Transactional Business Process Instance $I_{TP} \in \mathcal{M}_{TP}$

Output: Violation of Soft rules is ignored.

procedure (Initialize)

\[
TS(q_0, q_n) \Rightarrow ((\lambda(bRule) \rightarrow \lambda(\Psi_{BP})) \lor (\lambda(cRule) \rightarrow \lambda(\Psi_{CP})) \lor (\lambda(SParameter) \rightarrow \lambda(\Psi_{SP}))) \Rightarrow Detected
\]

\{ Checking Rule Type \}

\[
\text{for each } \lambda(\Sigma)
\]

\[
\text{Check RType (bRule } \lor \text{ cRule } \lor \text{ SParameter)}
\]

\[
\text{if RType(bRule } \lor \text{ cRule } \lor \text{ SParameter) } = \text{ Soft}
\]

\[
\text{then trigger } \varepsilon^{\text{ignore}};
\]

\[
\text{initiate } (\bigcirc \alpha \lor \bigcirc \Gamma);
\]

Example: Fault Tolerant Behavioral Model based on Ignore Recovery Policy.

The $\mathcal{M}_{AVERS}$ is a purchase order fulfillment transactional business process that is modelled using Solútha BTM and $\Sigma_{AA}$ is an agreement between the AVERS OEM and the Auto Inc. The agreement contains the clauses that are agreed upon by these business partners. Each clause is structured as business rule that is essentially expressed as a condition that must be satisfied during business transaction. A business rule contained in $\Sigma_{AA}$ is given below.

- IF ‘Auto Inc.’ sends a purchase order to ‘AVERS’, ‘AVERS’ may reply order acknowledgment to Auto Inc.

The formal structure of this rule can be written as follows:

- $\text{bRule} = (\text{PurchaseOrder.acknowledgementRequired} = \text{Yes}) \land (\text{RType} = \text{soft})$.

A fault-tolerant behavioral model can be constructed based on this rule by employing behavioral property 1. The modal verb ‘may’ is interpreted as possibility which indicates that the provided rule is soft type. The activities correspond to this rule are send purchase order and reply purchase order that are contained in $\mathcal{M}_{AVERS}$ which can be written as,

- $(\text{send purchase order, reply purchase order}) \in \mathcal{M}_{AVERS}$.

The behavioral properties 1 and 2 are translated into business transaction rule (BTX rule) to implement the behavioral properties in business transaction as constraints that at runtime control operations within the scope of business transaction. The BTX rule 1 is defined based on the behavioral properties 1 and 2.
**BTX Rule 1:** IF (bRule.RType = Soft) ∧ IF ((replyPO.acknowledgementSent) = ¬ Successful(replyPO)) THEN TS (sendPO, replyPO) → Successful(replyPO) ∧ Initiate Check Customer Detail.

This rule determines the fault-tolerant behavior for example, it forces the business transaction system to continue transaction if the AVERS OEM does not send acknowledgment to Auto Inc.

The behavioral property 3 defines the fault-tolerant behavior with respect to the failure of non-vital activities.

**Behavioral Property – 3:** ∀αi, f(αi) → e\(^{\text{ignore}}\), commit(αi) → ♦(commit(τ)) such that, AType (αi) = NonVital; initiate (αi+1) → continue (τ).

Figure 9.15 depicts the fault-tolerant behavioral model where the failure of non-vital activities is tolerated.

---

**Figure 9.15:** Fault Tolerant Behavioral Model - II defined using Ignore Recovery Policy.
Upon detecting the failure of an activity during the transition from the state $q_0$ to $q_1$, the transaction failure recovery handler checks whether the failed activity is non-vital. If yes, then the system forces continuing business transactions by initiating the subsequent activity.

Algorithm 2 automates the fault tolerant behavioral model that is constructed to ignore the failure of a non-vital activity.

**Algorithm 2**: Automation of Fault Tolerant Behavioral Model based on Ignore Recovery Policy

**Require**: Transactional Business Process Instance $I_{TP} \in \mathcal{M}_{TP}$

**Output**: Failure of Non-Vital Activity is Ignored.

**procedure** (*Initialize*)

\[
TS(q_0, q_n) \xleftarrow{\text{Detected}} \{ \text{Checking Activity Type} \}
\]

\[
\forall \alpha \in \mathcal{F}(\alpha) \text{ Check } (\alpha.AType)
\]

\[
\text{if } \alpha.AType = \text{NonVital}
\]

\[
\text{then trigger } e^{\text{Ignore}}; \text{ initiate } (\bigcirc \alpha);
\]

**Example: Fault Tolerant Behavioral Model based on Ignore Recovery Policy**

The transactional process model $\mathcal{M}_{AVERS}$ is composed of activities. The *Send Payment Confirmation* is a non-vital activity contained in the $\mathcal{M}_{AVERS}$. This activity resides within the scope of ‘Payment Transaction Process Fragment’. When the AVERS OEM receives the payment from the Auto Inc., the company is supposed to send a confirmation to the Auto Inc. The following BTX rule is defined based on the behavioral properties 2 and 3 which determine the fault tolerant behavior of business transactions upon the failure of non-vital activity.

**BTX Rule 2**: IF ($\text{SendPaymentConfirmation}.AType = \text{NonVital}$) \land IF ($\text{commit(}\text{SendPaymentConfirmation}.AType) = \neg \text{Successful(}\text{SendPaymentConfirmation})$) THEN *Initiate ProcessDelivery*.

According to this rule, the business transaction is never interrupted upon the failure of *SendPaymentConfirmation* activity. The business transaction system can tolerate this fault since the failed activity is a non-vital one.
9.4.4.2 Failure-Resilient Business Transaction

This section provides formal definitions of the operators that are used in designing failure-resilient business transactions. In addition, this section formalizes the behavioral models of failure-resilient business transactions. I provide algorithms for automating the behavioural models. The formalization of recovery properties and algorithms are discussed in the following.

Substitute Recovery Policy

The formal definition of substitute recovery policy is given below,

**Definition 42.** Substitute of a service component, service provider, and resource provider can be written as, \[ \text{Substitute} (SC \lor SP \lor RP) := \text{add} (SC \lor SP \lor RP) \]

The following behavioral property is defined using the substitute recovery policy.

**Behavioral Property – 4:** \( \forall \Gamma_i \in \mathcal{M}_{TP}, \ \text{commit} (\Gamma_i) \rightarrow \Diamond \text{committed} (\mathcal{M}_{TP}) \) such that, \( (\Gamma_i = \{ \alpha_i \} \mid \text{AType} (\alpha_i) = \text{Vital}) \) \( (\text{Shipmem} \lor \text{U}SC \lor \text{U}RP) \rightarrow \text{trigger} \epsilon^{\text{Substitute}} \rightarrow \text{Substitute} (SC \lor SP \lor RP). \)

This property is translated into a BTX rules 3, 4, and 5. These rules are straightforward. They allow the continuation of business transactions upon the occurrence of faults. The BTX rule 3 shows the failure prevention of business transactions upon the occurrence of software component unavailability.

**BTX Rule 3:** \( \forall \tau (I_i \in \mathcal{M}_{TP}), \ \text{IF} \ \exists \text{U}SC \ \text{THEN} \text{Substitute} (SC) \text{ by } SC' \text{ such that } SC \sim SC'. \text{ SC and } SC' \text{ denote required software component and substituted software component respectively.} \)

Algorithm 3 is developed for automating the software component substitution recovery behavior shown in figure 9.16.

---

**Algorithm 3 : Automation of Software Component Substitution**

**Require:** Transactional Business Process Instance \( I_{TP} \in \mathcal{M}_{TP} \)

**Output:** Software Component is Substituted.

**procedure** *(Initialize)*
The business transaction rules below determines the substitution of resource (goods) provider. The new product supplier can replace the current resource provider due to the ‘unavailability of goods’ fault.

**BTX Rule 4:** \( \forall \tau (I_i) \in \mathcal{M}_{fp}, \text{ IF } \exists U_{rp} \text{ THEN } \text{Substitute}(RP) \) by other resource provider \( RP' \) such that \( RP \sim RP' \). \( RP \) and \( RP' \) denote resource provider and substituted resource provider respectively.

I defined transaction rule 5 for specifying the substitution of service provider in business transactions.

**BTX Rule 5:** \( \forall \tau (I_i) \in \mathcal{M}_{fp}, \text{ IF } \exists \text{Shipment} \text{ THEN } \text{Substitute}(SP) \) by another service provider \( SP' \) such that \( SP \sim SP' \). \( SP \) and \( SP' \) denote service provider and substituted service provider respectively.

It is an important note that the newly replaced resource provider or service provider must be capable of performing the desired functions.

Substitution of a resource provider or service provider may result in further actions that are mainly related to the software component. Some of these actions are listed below:

- adding new software component or
- refining existing software component by adding new function and/or
- remove one or more function from the component or
- modify the component by performing all these operations or
- Cartesian product of add, remove, and modify operations.

The BTX rule 6 shows the implication of the substitution of a service provider or a resource provider. The operator \( \text{refine} \) is introduced for writing this rule.

**BTX Rule – 6:** \( \exists \Gamma_i \in \mathcal{M}_{fp}, \text{ Substitute}(RP \lor SP) \rightarrow \text{add}(SC') \lor \text{refine}(SC) \rightarrow SC' \) such that,

- \( SC \rightarrow SC' \)
• \text{refine}(\text{SC}) \rightarrow (\text{add}(\text{Funct}) \lor \text{remove}(\text{Funct}) \lor \text{modify}(\text{Funct})) \lor (\text{add}(\text{Funct}) \land \text{remove}(\text{Funct}) \\
\land \text{modify}(\text{Funct})) \lor (\text{add}(\text{Funct}) \times \text{remove}(\text{Funct}) \times \text{modify}(\text{Funct}))

These operations are essentially situation specific. At runtime, in a transactional business process instance, if a resource provider or a service provider is substituted, then the current software component should be suspended for that instance. However, the other instances of this transactional business process can use this software component.

\textbf{Behavioral Property – 5:} \exists \Gamma_i \in \mathcal{M}_{TP}, \text{Substitute}(\text{SC}) \rightarrow \text{Suspend}(\text{SC}) \text{ for } I_i \in \mathcal{M}_{TP}, \text{if and only if} (\text{SC'}^i) \text{ is added in the business transaction system. } \mathcal{M}_{TP} \text{ denotes a business transaction model.}

When a new software component is added, the substitute recovery operator is used in tandem with the suspend operator that is used in specifying a transaction rule that enables the business transaction systems to suspend running software component for the running instance and thereafter the newly added component is invoked for automating business transactions for that instance.

The algorithm 4 is developed for the automation of substitute recovery behavioral model shown in figure 9.16 that prevents abortion of business transaction by performing joint operations substitute and suspend.

\textbf{Algorithm 4:} Automation of Failure Resilient Behavior based on Substitution and Suspension Recovery Policy

\textbf{Require:} Transactional Business Process Instance \( I_{TP} \in \mathcal{M}_{TP} \)

\textbf{Output:} Resource Provider is Substituted, new Software Component is Added, and running Software Component is Suspended.

\textbf{procedure} (\textit{Initialize})

\[ \text{TS}(q_0, q_0) \]

\[ U_{\text{Software}} \leftarrow \text{Detected} \]

\[ \text{do} \]

\[ \text{Search Resource Provider;} \]

\[ \text{Select } \text{RP}'; \]

\[ \text{Check } \text{RP}'; \]

\[ \text{while} (\text{RP}' \neq \text{Found} \land \text{RP} \neq \text{RP}'); \]

\[ \text{if} (\text{RP} == \text{RP}') \]

\[ \text{then add} (\text{RP}' \land \text{SC'}) \land \text{Suspend} (\text{SC}); \]

\[ \]

A similar algorithm is defined for automating substitution and suspension of running service component in the case of substituting service provider. Only the fault type and variables are different.
9.4. FORMALIZATION OF BEHAVIORAL PROPERTIES

Figure 9.16 represents the flexible behavioral model designed using substitute recovery policy. The model covers software component substitution, service provider substitution, and resource provider substitution.

Furthermore, while substituting a resource provider or a service component, the subsequent activity or transaction process fragment wait until the substitution operations is finished. This can formally be written as follows,

**Behavioral Property – 6:** \( \exists \Gamma_i \in \mathcal{M}_{rp}, \text{Substitute} (RP \lor SP \lor SC) \rightarrow \text{Wait} (\bigcap \alpha_i \lor \bigcap \Gamma_j) \) until substitution operations is completed.

In addition to waiting, the transaction process fragments that have already scheduled for the execution must be postponed and rescheduled after the substitution is completed.

**Behavioral Property – 7:** \( \Gamma_i \in \mathcal{M}_{rp}, \text{Substitute} (SP \lor RP \lor SC) \rightarrow \text{Postpone} (\Gamma_i) \) until repairing completed such that, \( \text{Substitute} (SP \lor RP \lor SC) \rightarrow \text{trigger} \epsilon^{Postpone} \) and \( \Gamma_i \lor \Gamma_j \) is re-scheduled.

In an exceptional case, if substitution of a software component or a resource provider takes time more than it was expected then it affects temporal constraints in the agreements; specifically it may cause the violation of time related business rules contained in agreement. If the violated rules are hard type then ‘Mutualize’ recovery policy is a suitable approach to prevent business transaction abortion. The Mutualize
operator is described in the next subsection.

**Behavioral Property – 8**: \( \forall (I_i) \in M_{TP} \), Substitute \((RP \lor SP \lor SC) \rightarrow \) Mutualize \((\Sigma)\) such that, \((\lambda (\Sigma) \mid E(TS) > A(TS)) \rightarrow \) trigger \( e^{Mutualize} (\Sigma)\) where, \( E(TS) \) and \( A(TS) \) represents expected and actual substitution time respectively.

**Example**: Failure-Resilient Behavioral Model based on Substitute Recovery Policy

The AVERS supply chain scenario involves the Transportation Inc. and the Shipping Limited and suppliers X and Y. The shipment and inventory reservation process fragments of \( M_{AVERS} \) involve these business partners. Furthermore, the Shipping Limited and the supplier Y are contingent partners. The Shipping Limited replaces the shipment service provider Transportation Inc. and the supplier X can be replaced by the supplier Y based on certain conditions.

Typically, substituting a service provider let’s say Shipping Limited by Transportation Inc. or substituting the supplier X by the supplier Y promotes the need to adapt change in technology specifically in transaction system. The adaptation may require substitution of the running software component or refining the component.

An opposite case could be the AVERS OEM would not substitute software component rather the company would prefer to perform transaction activities with the new providers manually to avoid technology related complexities. Performing a business transaction manually will certainly reduce the performance of the system.

The substitution operation should not be longer than the estimated time for recovering business transactions faults otherwise it might jeopardize the agreement. More specifically, it may increase the possibility of exceeding the estimated substitution operation time which leads to the violation of agreement.

**Repair Recovery Policy**

Solúbtha BTM offers a repair operator to specify the policies for handling the faults including software integrity fault, ordering fault, message format fault, data fault, business function execution sequence fault, messaging sequence fault, intelligible constraint violation, and temper resistance constraint violation.

Practically speaking, repairing a faulty transaction mostly happens at runtime, which very often needs manual intervention. For example, repairing faults such as software integrity fault cannot be automated. Therefore, Solúbtha BTM can automate most of the recovery actions but not all.

The repair operator is offered to specify the policies at design-time. These trigger the repair action automatically at runtime upon occurrence of faults. This behavioral property is defined below:

**Behavioral Property – 9**: \( \forall (I_i) \in M_{TP}, commit(\tau(I_i)) \rightarrow \) committed \((\tau(I_i))\) such that, \((\mathcal{F}_{Ordering} \lor \mathcal{F}_{BFESQ} \lor \mathcal{F}_{MSeQ} \lor \lambda_{Intelligibility} \lor \lambda_{Temporal} \lor \mathcal{F}_{MessageF} \lor \mathcal{F}_{Integrity} \lor \mathcal{F}_{Connection} \lor \mathcal{F}_{Data}) \rightarrow \) trigger \( e^{Repair} \rightarrow Repair(\tau(I_i))\).
A business transaction commits eventually after performing the repairing actions successfully. However, the actual time to repair a fault must be less than or equal to the expected repair time otherwise, it may lead to violation agreement. Like substitution recovery policy, the repairing policy is designed to trigger mutualization action upon the possibility of exceeding the actual time to repair fault.

**Behavioral Property – 10**: \( \forall \tau (I_i) \in \mathcal{MT}_F, \text{Repair}(\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \text{Mutualize}(\Sigma), \) such that, \( (\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \lambda (\Sigma) \mid \mathcal{E}(\text{TTR}) > \mathcal{A}(\text{TTR}) \rightarrow \text{trigger} \text{Mutualize}, \) where \( \mathcal{E}(\text{TTR}) \) and \( \mathcal{A}(\text{TTR}) \) denote expected time to repair (TTR) and actual time to repair respectively.

The key goal of flexible business transaction is to prevent abortion at any rate; therefore, once the business transaction system receives an alarm of the potential violation generated by a timer component then it triggers the mutualization action immediately. The behavioral properties 9 and 10 are transformed into transaction rules. At runtime, the repairing actions rely on these rules. They are as follows.

**BTX Rule - 7**: \( \forall \tau (I_i) \in \mathcal{MT}_F, \) if \( (\exists (\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow (\lambda (\Sigma)) \land (\text{commit} (\alpha_i \lor \Gamma_i) \rightarrow \neg \text{successful} (\alpha_i \lor \Gamma_i)) \) then \( \text{Repair}(\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \diamond \text{commit} (I_i). \)

The following business transaction rule is defined for the case where the actual repair time is more than the expected repair time.

**BTX Rule - 8**: \( \forall \tau (I_i) \in \mathcal{MT}_F, \) if \( \text{Repair}(\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \diamond \text{commit} (I_i) \) such that, \( (\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \lambda (\Sigma) \mid \mathcal{E}(\text{TTR}) > \mathcal{A}(\text{TTR}). \)

The repairing transaction rules may not associate specific activity because the faults do not happen with activity. For instance, business function execution sequence faults occur with controls flows of transactional business processes. During business transaction if these faults occurred they should be repaired to prevent abortion.

Furthermore, while repairing a fault the subsequent activities and transaction process fragments must wait until the repairing actions has been carried out.

**Behavioral Property – 11**: \( \forall \tau (I_i) \in \mathcal{MT}_F, \text{Repair}(\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \rightarrow \text{Wait} (\bigcirc \alpha_i \lor \bigcirc \lambda_i) \) until repairing is completed such that,\( ((\mathcal{I}_\text{Ordering} \lor \mathcal{I}_\text{BFESeQ} \lor \mathcal{I}_\text{MSeQ} \lor \lambda \text{Intelligibility} \lor \lambda \text{Temporal} \lor \mathcal{I}_\text{MessageF} \lor \mathcal{I}_\text{SIntegrity} \lor \mathcal{I}_\text{Connection} \lor \mathcal{I}_\text{Data}) \) trigger \text{Wait}. \)

The repair action triggers the postpone recovery policy that changes the schedule of the execution of the
transaction process fragments and activities.

**Behavioral Property – 12:** $\forall (I_i) \in M_T . \text{Repair}(F_{\text{Ordering}} \lor F_{\text{BFESq}} \lor F_{\text{MSeq}} \lor \lambda_{\text{Intelligibility}} \lor \lambda_{\text{Temporal}} \lor F_{\text{MessageF}} \lor F_{\text{SIntegrity}} \lor F_{\text{Connection}} \lor F_{\text{Data}}) \rightarrow \text{Postpone}(\bigcirc \alpha_i \lor \bigcirc \lambda_i)$ until repairing is completed such that, $(F_{\text{Ordering}} \lor F_{\text{BFESq}} \lor F_{\text{MSeq}} \lor \lambda_{\text{Intelligibility}} \lor \lambda_{\text{Temporal}} \lor F_{\text{MessageF}} \lor F_{\text{SIntegrity}} \lor F_{\text{Connection}} \lor F_{\text{Data}}) \rightarrow \text{trigger} \ e^{\text{Postpone}}$.

Figure 9.17 shows the failure-resilience behavioral model that relies on repair and mutualize policies (Mutualize recover policy is presented in subsection C).
I developed the algorithm 5 to automate the failure-resilient behavioral model.

Algorithm 5: Automation of Failure Resilient Behavior based on Mutualization, Postponement and Wait Recovery Policy

Require: Transactional Business Process Instance $I_{TP} \in \mathcal{M}_{TP}$

Output: Agreement Mutualized, Activities, TPFs Postponed and Waited.

procedure (Initialize)
Example: Failure-Resilient Behavioral Model based on Repair Recovery Policy

Let us assume that the transaction of the purchase order fulfillment business process is designed using Solúbtha BTM. Also, assume that the customer Auto Inc. has sent a purchase order that was not approved (signed) by the Auto Inc.’s purchase order manager. Upon receiving the purchase order, the AVERS’s sales and distribution system which is a part of business transaction system checks the purchase order against the agreement between the Auto Inc. and the AVERS OEM. The agreement contains the following clause,

- **Clause:** Auto Inc.’s purchase order manager who is responsible to approve the purchase order must approve each purchase order sent by the Auto Inc.

While verifying the purchase order, the sales and distribution system throws an exception message “the purchase order has not been approved by the purchase manager” – that is, a violation of agreement has been caused by the ordering fault.

The AVERS sales and distribution sends the fault log attaching a time constraint for repairing the purchase order to the Auto Inc. Upon receiving the report, the Auto Inc. repairs the purchase order by adding a legitimate approval method such as a signature that is intelligible and resend the purchase order. The subsequent activities such as check customer account wait until the repairing of purchase order is done. If the repairing action takes longer time than the expected (which may never happen in practice, still it is possible that Auto Inc. may take long time) time and if the AVERS OEM predicts that this additional time taken by the Auto Inc. may cause violation agreement such as not meeting the delivery deadline, then the business transaction system trigger ‘Mutualize’ recovery action.

**Mutualize Recovery Policy**

Mutualize is one of the most important operators incorporated in Solúbtha BTM. The effectiveness of this operator was shown in the two previous sections. Mutualize is used to prevent the business transaction abortion when substitution and repair recovery actions take more time than expected. Essentially, mutualiz-
ation is expected to refrain the business transaction systems from the failures that happen due to agreement violation.

Mutualization triggers renegotiation between the business partners who are engaged in business transactions. The formal definition of mutualize is given below.

**Definition 43.** I defined the Mutualize as follows,

\[
\text{Mutualize} \equiv ((\text{insert } (\Psi_{SP} \lor \Psi_{BP} \lor \Psi_{CP}) \lor \text{insert } (\Psi_{SP} \land \Psi_{BP} \land \Psi_{CP})) \lor \text{insert } (\Psi_{SP} \land \Psi_{CP}) \lor \text{insert } (\Psi_{BP} \land \Psi_{SP})) \lor \text{remove } (\Psi_{SP} \lor \Psi_{BP} \lor \Psi_{CP}) \lor \text{remove } (\Psi_{SP} \land \Psi_{BP} \land \Psi_{CP}) \lor \text{modify } (\Psi_{SP} \land \Psi_{BP} \land \Psi_{CP}) \lor \text{modify } (\Psi_{SP} \land \Psi_{BP}) \lor \text{modify } (\Psi_{SP} \land \Psi_{CP}) \lor \text{modify } (\Psi_{BP} \land \Psi_{SP}))
\]

Mutualization of an agreement produces a newer version of the agreement,

\[
\text{Mutualize } (\Sigma) \rightarrow \Sigma'.
\]

The purpose of mutualization is to prevent the possible or occurred violation of agreement e.g. violation of agreed delivery cycle time proactively. Renegotiation may take place upon occurrence of temporal fault, uncertain events and processing delay that cause violation of agreement. However, renegotiation can happen upon occurrence of any fault that leads to violation of agreement or promote the possibilities of agreement violation. The ‘Mutualize’ operator is used to define mutualization recovery policies. The behavioral property – 13 represents mutualization recovery policy that is defined using the ‘Mutualize’ operator.

**Behavioral Property – 13:** \(\forall \tau (I_i) \in M_{TP}, \text{ committed}(\tau) \rightarrow \Diamond (\text{commit } (\tau)) \text{ such that } (E_{Uncertainty} \lor F_{Temporal} \lor D_{Processing}) \rightarrow \lambda (\Sigma) \rightarrow \text{trigger } \varepsilon^{\text{Mutualize}} (\Sigma).
\]

Mutualization of agreement is a time-intensive action. It takes time to complete which could be minutes to days. Thus, all successive scheduled transaction process fragments or activities must be postponed until the mutualization operation is completed successfully.

**Behavioral Property – 14:** \(\forall \tau (I_i) \in M_{TP}, \text{Mutualize}(\Sigma) \rightarrow \text{Postponed } (\Gamma_i \lor \alpha_i) \text{ such that, Mutualize}(\Sigma) \rightarrow \text{trigger } \varepsilon^{\text{Postpone}} (\Gamma_i \lor \alpha_i) \text{ until mutualization is completed successfully.}
\]

Mutualization must take finite amount of time which can be written as, \(\text{mutualization}(t) \neq \infty\), where 't' denotes time.

Furthermore, upon occurrence of a fault in an activity or transaction process fragment, the subsequent activity from the last savepoint has to wait until the mutualization between transaction parties (business partners) is finished.

**Behavioral Property – 15:** \(\forall \tau (I_i) \in M_{TP}, \text{Mutualize}(\Sigma) \rightarrow \text{Waited } (\alpha_i \lor \Gamma_i) \text{ such that, Mutualize}(\Sigma) \rightarrow \varepsilon^{\text{Wait}} (\alpha_i \lor \Gamma_i) \text{ until the mutualization is finished successfully. Mutualization must take finite amount of time which can be written as, Mutualize}(t) \neq \infty.
\]
The behavioral properties 13, 14, and 15 are translated into the transaction rules below that essentially controls business transaction systems at runtime.

**BTX Rule 9**: \( \forall \tau(I) \in \mathcal{M}_T \), IF \( \exists (\text{EUncertainty} \land \lambda_{\text{Temporal}} \land D_{\text{Processing}}) \) \& IF \( (\text{EUncertainty} \lor \lambda_{\text{Temporal}} \lor D_{\text{Processing}}) \Rightarrow \lambda(\Sigma) \) \& IF \( \text{commit}(\alpha_i \lor \Gamma_i) \rightarrow \neg \text{Successful}(\alpha_i \lor \Gamma_i) \) THEN Mutualize(\Sigma) such that Postpone(\Gamma_i \lor \alpha_i) \& Wait(\alpha_i \lor \Gamma_i) until mutualization is finished.

The failure-resilient behavioral model based on mutualize recovery protocol is presented graphically in figure 9.18.

![Failure Resilient Behavioral Model based on Mutualize Recovery Policy](image)

Figure 9.18: Failure Resilient Behavioral Model based on Mutualize Recovery Policy.

Algorithm 6 is designed to support the automation of mutualize recovery policy. The algorithm is presented below.

---

**Algorithm 6**: Automation of Failure Resilient Behavior based on Mutualize Recovery Policy

**Require**: Transactional Business Process Instance \( I_T \in \mathcal{M}_T \)

**Output**: Agreement is Mutualized.

**procedure** (*Initialize*)
9.4. FORMALIZATION OF BEHAVIORAL PROPERTIES

\[
\begin{align*}
\text{TS}(q_0, q_n) ; \\
(\mathcal{E}_{\text{Uncertainty}} \| \lambda_{\text{Temporal}} \| \mathcal{D}_{\text{Processing}}) & \leftarrow \text{Detected} \\
\text{if } (\mathcal{E}_{\text{Uncertainty}} \| \lambda_{\text{Temporal}} \| \mathcal{D}_{\text{Processing}} \rightarrow \lambda(\Sigma)) \\
\text{then } \\
\varepsilon_{\text{Mutualize}} (\Sigma) & \& \text{Postpone}(\bigcirc \alpha_{i} \lor \bigcirc \Gamma) \\
& \& \text{Wait}(\bigcirc \alpha_{i} \land \bigcirc \Gamma); \\
\end{align*}
\]

Example: Failure-Resilient Behavioral Model based on Mutualize Recovery Policy

While the transactional business process instance is running, the business transaction system of the AVERS OEM may trigger ‘Mutualize’ recovery actions in many occasions. Assuming that the labor strike happened in the AVERS manufacturing plant and if it takes long time to resolve this uncertain event, the business transaction system confirms the possible violation of delivery deadline agreed between the Auto Inc. and the AVERS OEM. Consequently, the business transaction system triggers the ‘Mutualize’ recovery action proactively. The same action can be triggered upon the occurrence of other events such as transportation strike.

Retry Policy

Since business transactions are distributed, the business transaction system integrates several applications hosted at different sites. The distributed applications perform business transactions over the messages via the Internet. In such architectures, the connection fault may happen while business transactions are running. This fault may cause loss of packet such as loss of purchase order, advanced shipment notification, etc. Practically speaking, the connection fault may happen frequently however the communication may reestablish shortly after the fault happens. In some cases, it may reestablish in a few milliseconds.

The retry action is performed when a business transaction system throws an exception message connection error, the message cannot be delivered to the recipient. The ‘Retry’ resends the packet containing payloads to the host. It is formally defined as follows,

Definition 44. \( \text{Retry} := \text{op(task)} | \text{retryTaskLimit} = N. \)

Retrying operation is constrained by \( \text{retryTaskLimit} \) that limits number of times ‘N’ the operation should be retried and cannot exceed that limit.

Behavioral Property – 16: \( \forall \tau (l_i) \in \mathcal{M}_{FP}, \text{commit} (\alpha_i) \rightarrow \text{committed} (\alpha_i) \) such that, \( \mathcal{F}_{Connection} \rightarrow (\text{Retry} = \text{op(task)} | \text{retryTaskLimit} = N). \)

In a distributed business transaction environment, retry can be performed upon the occurrence of software component unavailability. The software component can be unavailable due to the overloaded traffic or the application server is temporarily unavailable. In such situations, retry is a very effective policy.

It is evident that the retry operation can be performed multiple times which essentially promotes the
risk of producing unexpected results. In order to keep the operation safe by preventing any unwanted consequence, the idempotency (Goodearl, 1991) approach was adopted in Solúbtha BTM.

**Corollary 1:** \( \forall (I_i) \in M_{TP}, \text{Retry} \) is an idempotent operation such that, \( op(op(task)) = op(task) \).

The behavioral property 16 is transformed into the business transaction rule presented below,

**BTX Rule 10:** \( \forall (I_i) \in M_{TP}, \text{if } F_{\text{Connection}} \land \text{if (commit } (\alpha_i) \rightarrow \neg \text{Successful}(\alpha_i) \text{) then Retry } (\alpha_i) \).

Figure 9.19 presents the behavioral model where the retry recovery policy is used for preventing business transaction abortion.

![Figure 9.19: Failure Resilient Behavioral Model based on Retry Policy.](image)

Algorithm 7 presented below is developed to automate the retry policy for recovering the transaction fault.

---

**Algorithm 7**: Automation of Failure Resilient Behavior based on Retry Policy

**Require**: Transactional Business Process Instance \( I_{TP} \in M_{TP} \)

**Output**: An Operation is Retried.

**procedure** *(Initialize)*
9.4. FORMALIZATION OF BEHAVIORAL PROPERTIES

\[TS(q_0, q_n) ;
(F_{Connection}) \leftarrow \text{Detected};
\]
\[
\begin{cases}
\text{if } (F_{Connection} \rightarrow \neg \text{committed } (\alpha_i)) \\
\text{then} \\
\text{while } (\text{retryTaskLimit} \leq N) \\
\text{Retry}(\alpha_i);
\end{cases}
\]

**Example:** Failure-Resilient Behavioral Model based on Retry Policy

While the purchase order fulfillment transactional business process is running, the AVERS OEM tries to send an advanced shipment notification (ASN) to confirm the date of delivering the ordered goods to the destination. When the logistics system of the AVERS OEM performs *send ASN operation*, it receives the connection error message. The system then retries the same operations until it exceeds the maximum time to retry.

**Compensation Recovery Policy**

This section provides the formal definition of the compensate operator, the formal description of compensation behavioral model, and the algorithms for automating the compensation behavior at runtime.

Compensation triggers the execution of the transactional business process in an inverse sequence from the point of failure. It undoes the business transaction from the semantic point of view. The formal definition of compensation is provided below,

**Definition 45.** Compensation can be defined as,

\[\text{Compensation} := \text{execute } (B_{\neg \text{seq}} (M_{\neg FP}) \rightarrow (S_{\text{Undo}} (\Gamma_n, \Gamma_{n-1}, ..., \Gamma_0)) \mid \Gamma = \alpha_i, \text{ where } i = 1...n)\]

where $B_{\neg \text{seq}}$ denotes the backward sequence of transaction process fragments.

**Behavioral Property – 17:** \(\forall \tau (I_i) \in M_{\neg FP}, \text{commit } (\tau(\alpha_i)) \rightarrow \Diamond \text{ aborted } (\tau(\alpha_i)) \land \text{ such that, } \neg \text{committed } (\tau(\alpha_i)) \rightarrow \text{trigger } \epsilon^{\text{Compensation}} (\tau)\).

Compensation is a useful mechanism however if a transaction process fragment fails completely and the fragment cannot be recovered from the failure then compensating end-to-end business transaction is impractical. In order to prevent abortion of specific transaction process fragment, this research adopts partial compensation approach. The partial compensation limits the compensation operation. The operation does not affect other transaction process fragments within a transactional business process. Solúbtha BTM enables designing the partial compensation. The partial compensation is influenced from the concept

---

1Semantic undoing of business transaction means the activities within the scope of transaction process fragments are semantically undone. An important note is that upon occurrence of a failure, compensation is triggered after all other recovery policies have been applied. In other words, this is the last option for a business transaction system to recover transactions from failures. The behavioral property below represents the compensation recovery policy defined using compensate operator.
compensation sphere (Leymann and Reuter, 2000). The partial compensation is formally presented in the following behavioral property.

**Behavioral Property – 18:** \( \forall \tau (I_i) \in M_{TP}, \text{commit}(\Gamma) \rightarrow \Diamond \text{aborted}(\Gamma) \land \text{Compensated}(\Gamma) \) such that,

\[
(\neg \text{committed}(\Gamma) \land \neg \text{Repaired}(\Gamma) \land \neg \text{Substituted}(\Gamma) \land \neg \text{Mutualized}(\Sigma)) \rightarrow \text{trigger}_{\text{PartialCompensation}(\Gamma)}.
\]

Full compensation and partial compensation are controlled at runtime by business transaction rules 11 and 12 that are the translation of behavioral property - 18.

**BTX Rule 11:** \( \forall \tau (I_i) \in M_{TP}, \text{IF commit}(\tau) \rightarrow \Diamond \text{aborted}(\tau) \text{ THEN } \text{Compensate}(M_{TP}). \)

The full compensation rule given in the above forces the business transaction system to revert business transactions to initial state. Conversely, the partial compensation rule drives the execution of a business transaction system to a certain state in a reverse order.

**BTX Rule 12:** \( \forall \tau (I_i) \in M_{TP}, \text{IF commit}(\Gamma) \rightarrow \neg \text{committed}(\Gamma) \text{ THEN } \text{Compensate}(\Gamma). \)

Figure 9.20 presents the compensate and partial compensate behavioral model of business transaction.
Figure 9.20: Failure-Resilient Behavioral Model based on Compensation Recovery Policy.
Figure 9.20 shows that partial compensation undoes $q_1$ only whereas full compensation undoes the transaction states in reverse order starting from $q_3$ to $q_0$.

Algorithm 8 is developed to automate the compensation behavioral model.

Algorithm 8 : Automation of Failure Resilient Behavior based on Compensation Recovery Policy

Require: Transactional Business Process Instance $I_{TP} \in M_{TP}$

Output: Transaction is Compensated.

procedure (Initialize)

$$TS(q_0, q_n) :$$

$$(complete \text{ } abortion \lor partial \text{ } abortion) \leftarrow Detected;$$

$${\textbf{if}} (commit (\Gamma) \rightarrow aborted (\Gamma))$$

$${\textbf{then}}$$

$${\textbf{while}} (i \neq \alpha_0)$$

$$\{$$

execute $B_{seq}(\alpha_i);$$

$$\}$$

$${\textbf{if}} (commit (\tau) \rightarrow aborted (\tau))$$

$${\textbf{then}}$$

$${\textbf{while}} (i \neq \alpha_0)$$

$$\{$$

execute $B_{seq}(M_{TP});$$

$$\}$$

Resume Recovery Policy

Two cases can be considered in the context of resume recovery policy: (i) business transaction of a transactional business process instance is redone entirely – this is called global redo and (ii) sub-transaction(s) of one or more parts (transaction process fragment) of that instance are redone – this is called partial redo (Haerder and Reuter, 1985).

From the case studies, I concluded that the partial redo is more appropriate than the global redo for business transactions in some situations. The reason is quite straightforward. The business transactions are controlled by agreement, and therefore upon the occurrence of a failure, it is not possible to redo the entire the business transaction immediately after completing the compensation, business partners may need to renegotiate the agreement, which will consume time. Considering this scenario, I adopted partial redo
with savepoint technique in Solútha BTM.

An important design aspect of the resume operation is that it should be designed in combination with compensation. The resume must succeed the compensation action. The resume recovery policy is defined below:

**Definition 46.** Resume can be defined as,

\[
\text{Resume} := \text{execute } (B_{\text{Seq}}(\mathcal{M}_{TP} \cup \text{Checkpoint})) \land (F_{\text{Seq}}(\mathcal{M}_{TP}.\text{FromCheckPoint})) \text{ such that,}
\]

\[
B_{\text{Seq}} \rightarrow (S_{\text{Undo}}(\Gamma_n, \Gamma_{n-1}, \ldots, \Gamma_0) \mid \Gamma = \alpha_i, \text{ where } i = 1..n)
\]

\[
F_{\text{Seq}} \rightarrow \text{Redo}(\Gamma_1, \Gamma_2, \ldots, \Gamma_n)
\]

Where:

- \(B_{\text{Seq}}\) denotes the backward sequence of transaction process fragments.
- \(F_{\text{Seq}}\) denotes the forward sequence of transaction process fragments.
- \(F_{\text{Seq}}\) must be succeeded by \(B_{\text{Seq}}\).

When a transaction process fragment aborts, the business transaction system issues compensation recovery operation that reverts the transactional business process until (U) the last savepoint from where the business transaction resumes. The savepoint can be found from the business transaction logs which contain the history (record) of the business transaction. The property below is defined using the resume operator for preventing business transaction abortions.

**Behavioral Property – 19:** \(\forall \tau(I) \in \mathcal{M}_{TP} \text{, if commit } (\Gamma) \rightarrow \Diamond \text{aborted}(\Gamma) \land \text{Compensated(\mathcal{M}_{TP} \cup \text{Checkpoint}))} \)

\(\land \text{Resumed(\mathcal{M}_{TP}.\text{FromCheckPoint})}\) such that, \(\text{aborted } \Gamma \land \neg \text{Repaired}(\Gamma) \land \neg \text{Substituted}(\Gamma) \land \neg \text{Mutualized}(\Sigma)\).

The business transaction rule defined below controls the resume recovery behavioral model of business transaction at runtime.

**BTX Rule 13:** \(\forall \tau(I) \in \mathcal{M}_{TP} \text{, IF commit } (\Gamma) \rightarrow \text{aborted } (\Gamma) \text{ THEN trigger } \epsilon_{\text{Resume}}.\)

Figure 9.21 presents resume recovery behavioral model.
Figure 9.21: Failure-Resilient Behavioral Model based on Resume Recovery Policy.
In figure 9.21, \( q_2 \) is aborted and upon abortion, the compensation executes the business transaction in reverse order until \( q_1 \) which is the last savepoint from where the redoing starts. Algorithm 9 presents the automation of resume recovery behavioral model of business transactions.

**Algorithm 9** : Automation of Failure Resilient Behavior based on Resume Recovery Policy

**Require**: Transactional Business Process Instance \( I_{TP} \in M_{TP} \)

**Output**: Transaction is Resumed.

**procedure** \((\text{Initialize})\)

\[
\begin{align*}
\text{TS}(q_0, q_n) ; \\
\text{(aborted}(\lambda) & \leftarrow \text{Detected}; \\
\text{if}(\text{commit}(\Gamma) \rightarrow \text{aborted}(\Gamma)) \\
& \quad \text{then} \\
& \quad \text{while}(i \neq \alpha, \text{CheckPoint}) \\
& \quad \quad \{ \\
& \quad \quad \text{execute } B_{seq}(M_{TP} \cup (\text{CheckPoint})); \\
& \quad \quad \text{execute } F_{seq}(M_{TP}); \\
& \quad \quad \}
\end{align*}
\]

**9.4.5 Summary**

The formal semantics of the functional perspective elements of Solúbtha BTM were provided in this chapter. The semantics were defined using time extended finite state automata. The design-elements were translated into TEFSM. The formalization enables Solúbtha BTM to support designing verifiable business transaction instance-model.

The Chapter presented and described the design-elements such as recovery operators and control flows for defining extensible, nondeterministic, fault-tolerant, and failure-resilient behavioral models of business transactions. The formal definitions of these elements were provided. The formal semantics of extensible, nondeterministic, fault-tolerant, and failure-resilient behavioral models were presented using the formal languages, TEFSM and CTL. Also, in this chapter, I defined several behavioral properties using CTL. Furthermore, I developed a list of algorithms to automate the flexible behaviors of business transactions.
Chapter 10

Validation and Evaluation

10.1 Introduction

This chapter discusses the validation of Solúbtha BTM. The purpose of validation is to demonstrate that Solúbtha BTM meets the research objectives. As mentioned in Section 3.3.4, ‘demonstration is an approach suggested by many design-science researchers to validate the design artifact. In this research the design artifact is Solúbtha BTM. The researchers suggested several methods which can be used for demonstration, that they are, experimentation, simulation, case study, proof or any other applicable methods. I used simulation and formal methods to validate the model against four criteria. I choose these methods because both are formal approaches and since Solúbtha BTM has been built on the formal methods, these approaches are the most appropriate ones for validating Solúbtha BTM and also to make the validation rigorous.

Considering the criteria of validating the Solúbtha BTM, I explained the reasons for choosing these criteria in Chapter 3. I recall these criteria again in this chapter.

- **Correctness**: It proves the correctness of Solúbtha. This criterion is chosen for validating the syntactic correctness and the semantics of the model. It was chosen because correctness is a critical criteria for a model that is used for designing instance-models. Solúbtha BTM is used for designing the business transaction instance-model. Therefore, the correctness of the logical structure of the model and the semantics of design-elements is of critical importance.

- **Flexibility**: It demonstrates that the behavioural flexibility of business transactions is ensured at runtime by Solúbtha BTM. This criterion was chosen to validate one of the two research objectives: highly flexible behavior of business transactions at runtime. More specifically, this criterion proves that the techniques provided by Solúbtha BTM guarantee highly flexible business transactions. Also, to show that the techniques enable business transaction systems to prevent business transaction failures or abortions. It is a critical criterion for validating business transactions.

- **Correlation**: It demonstrates the ability of Solúbtha BTM to correlate explicitly the business and functional perspectives of end-to-end business processes. This was chosen to validate Solúbtha’s capability to provide explicit logical connection between the design-elements from business and functional perspectives. This is another critical criterion because it validates the other objective of this
10.2 Validation Setting

This section describes the business and technical settings in which Solúbtha BTM is tested.
10.2.1 Business Setting

This section describes the Rhea Automotive business scenario. Additionally, it presents the trading partner agreement that is used in the validation.

10.2.1.1 Overview of the Rhea Automotive Business Scenario

The main business partner in this scenario is Rhea Automotive Limited, which is an automotive original equipment manufacturing company, established in 1988 and based in Iran. The company used to manufacture motorcycle parts, however the company added polymer products to its product lines from 1995. When this case was prepared the company employed 200 people. The company has ISO9002 and QS9000
10.2. VALIDATION SETTING

Rhea Automotive Ltd. has three main suppliers who supply the raw materials and semi-finished products. In addition, it collaborates with companies from different countries including Taiwan, Korea, China, and India. The company has no collaboration with any fixed shipper. It has a sort-term agreement (STA) with a shipper. The business scenario has been captured in Figure 10.2.

![Figure 10.2: The Rhea Automotive Business Scenario.](image)

10.2.1.2 Agreements

This section presents the agreement used in the validation process. In Chapter 3 (Section 3.3.4), I provided a detailed explanation of the criteria for selecting this agreement. I recalled these below:

- The agreement is about automotive trading partner agreements.
- It is a well-written agreement.

**Agreement**

- **Clause 1:** “Rhea Automotive Ltd.” may send order acknowledgment to “Retailer”.
• **Clause 2:** “Rhea Automotive Ltd.” must reply with acceptance notification within 7 days from the day the order is placed.

• **Clause 3:** Retailer should send correct purchase order.

• **Clause 4:** Rhea Automotive Ltd. must deliver the correct goods to Retailer otherwise Retailer have the right to cancel the purchase order.

• **Clause 5:** Rhea Automotive Ltd. should issue the Invoice to the Retailer on delivery of the purchase order.

• **Clause 6:** Retailer makes payment IF AND ONLY IF Rhea Automotive Ltd. has delivered the product/goods in full to the Retailer.

• **Clause 7:** The Retailer should pay Rhea Automotive Ltd within 15 days from the day the purchase order has been delivered.

• **Clause 8:** Rhea Automotive Ltd. should deliver the product within 90 days from the day the order placed.

• **Clause 9:** ‘Xport Limited’ must pick-up the goods from the agreed location.

• **Clause 10:** ‘Xport Limited’ must deliver the goods to the agreed point of delivery.

• **Clause 11:** ‘Xport Limited’ must deliver the goods within the agreed period.

• **Clause 12:** Rhea Automotive Ltd. reserves the right to hire a new carrier if Xport Limited delays the shipment.

In Chapter 3 (Section 3.3.3.4, Page 71), I explained the interpretation of modal verbs such as should, may, must *etc.*. In the agreement, Tenneco’s Purchase Order Agreement (2012), I found the use of ‘may’ in clauses such as in 3(a), which I interpret as both permission and possibility. I interpret it from linguistic perspective and also from the operational standpoint. Both are necessary for the logical constructions of the behavioural properties of business transactions. In the case of ‘may’ used in a clause I interpreted it as Rhea Automotive Ltd is allowed to send an acknowledgment and it is a possible action which can be performed or not by the ‘Sales and Distribution’ system of the company.

**10.2.1.3 Purchase Order Management Process**

The workflow of the purchase order management process is described by Manzouri *et al.*, (2000). I slightly modify the workflow but does not change the core workflow, because, I change the labels of process activ-
10.2. VALIDATION SETTING

ities, the departments etc. which are essentially minor changes. The modification does affect the core business process and the chain of operations.

The process comprises sales and distribution, finance, and logistics departments. The process involves business partners and an intermediary. The Retailer and Rhea Automotive Ltd. are the main business partners; the supplier are a secondary business partner of the Rhea Automotive Ltd.; bank is an intermediary; and Xport Limited and Warehouses (subsidiary and central warehouses) are the logistics and inventory service providers respectively.

The ‘purchase order management process comprises a set of ordered activities. The process starts with ‘Submit Purchase Order’ activity by the retailer who is the buyer for Rhea Automotive Ltd. The sales and distribution department of the Rhea Automotive Ltd. receives the order and carries out the registration process. Then, the Customer Relationship Management (CRM) application ‘checks customer details’ and ‘verifies the credit worthiness’ of the customer. For customer credit worthiness, a request is sent by the CRM application to the ‘bank’.

The bank authority checks the customer’s credit status and replies to Rhea Automotive Ltd. Next, the ‘price of ordered parts’ is calculated. The control then flows to the inventory department of Rhea Automotive Ltd. The inventory reservation process’ is carried out by the inventory department. In this process, the inventory is checked against the number of fuel pumps ordered and then the purchase order is registered.

The production process is triggered if the required number of parts are unavailable. In the production process, the availability of raw materials is checked. If the required amount of raw materials is not available for producing the parts then an order is placed for new raw materials with the supplier. The supplier processes the order and delivers the raw materials to Rhea Automotive Ltd. The raw materials are transformed into finished goods and then moving the finished goods to product storage.

The raw material production process runs in parallel with the delivery process carried out by the logistics management department. The logistics system sends a ‘shipment request’ to Xport Limited which prepares the shipment and sends the delivery (shipment) information to the logistics system of Rhea Automobile Ltd., which then sends ‘Advanced Shipment Notification (ASN)’ to the retailer.

While the shipper is processing shipment information, the finance department – in co-operation with sales and distribution department – processes the payment. As soon as the sales system has completed the ‘Calculate Total Cost’ activity, the control flows to ‘Process Payment’ activity that is performed by the Rhea Automotive Ltd.’s financial system. The invoice is prepared and is sent to the Retailer who then verifies the invoice and sends the verification result to Rhea Automotive Ltd. If the order is ready to be dispatched, the shipper picks up the goods from ‘Product Storage’ and delivers them to the agreed destination. The retailer receives the goods and triggers a request to the bank to execute the payment. The bank executes the
payment request placed by the retailer and sends the payment confirmation to both the Rhea Automotive Ltd. and the Retailer.

10.2.2 Technological Setting

This section describes the technological setting of the validation process. I used ARIS EXPRESS for designing the Rhea POM business transaction model.¹ A brief introduction to ARIS EXPRESS is provided in this section. For simulating the Rhea POM business transaction model I used the UPPAAL model checker. The model checker is briefly described in this section.

10.2.2.1 ARIS Express

ARIS Express is an integrated tool developed by the Software AG ². It is used for modeling business processes, organizational charts, data models, process landscapes etc. ARIS Express integrates business process modeling notation (BPMN) (OMG, 2011) and Event Process Chain (EPC) (Scheer, A-W., 2002) for modeling business processes and event-driven systems.

10.2.2.2 UPPAAL

UPPAAL is an integrated tool for modeling the behavior of systems in terms of states and transitions between states, simulation, and verification of a timed automata (Alur and Dill, 1994). UPPAAL enables the description of abstract state machines with synchronization. It employs a dense-time model that uses the clock variable which evaluates to a real number. The clocks progress with synchronization. UPPAAL provides bounded discrete variables such as read and write. In addition, it offers C-like syntax for describing the guarding conditions and the updates in transitions. The language supports describing the communicating state machine via common (shared) variables.

In UPPAAL, a system is designed as a network of several time automata in parallel. The state of the system is defined by the locations of all automata, clock values, and values of the discrete variables. Interested readers are referred to Behrmann et al., (2006), for a detailed introduction to the model checker.

UPPAAL offers a query language for specifying properties that are checked while a transaction is running. The query language is called Timed Computation Tree Logic (TCTL) which is a combination of CTL and clock constraints. It is a CTL-like language and offers CTL quantifiers. Thus, it is possible to specify the CTL properties defined in Chapter 8 using the UPPAAL property specification language. The language comprises state formulae and path formulae. The state formulae describe the individual states, whereas the path formulae quantify the paths or traces of the model (Behrmann et al., 2004). The state formulae are the

¹http://ariscommunity.com/arism-express
²http://softwareag.com
expressions evaluated for a state without looking at the behavior of the model. These expressions are essen-
tially the transaction conditions attached to transaction states. It is worth mentioning that the UPPAAL
does not allow nested path formulae. Behrmann et al. classified path formulae under three properties:
reachability, safety, and liveliness. The properties are introduced in the following:

- **Reachability Properties**: These are the simplest form of the properties. These properties check
whether a given state formula can be satisfied by any reachable state.

- **Safety Properties**: These confirm that something bad will never happen. The safety properties ensure
the business transaction will not abort.

- **Liveness Properties**: These guarantee that something will happen eventually. A business transaction
will eventually be aborted or committed.

### 10.2.2.3 System Environment

This section describes the system specification used in performing the validation. The specification is
provided in the following:

- **Operating System**: Microsoft Windows 7 Professional
- **Processor Architecture**: 64-bit
- **Processor**: Intel (R) Core (TM) i5-3340M CPU @ 2.70 GHz
- **Main Memory**: 16.0 GB

### 10.3 Designing Rhea POM Business Transaction Model

This section discusses the first step shown in Figure 10.1. The Rhea POM business transaction model is
designed in this section. The model is designed in two steps. In the first step, the POM business transaction
model is defined and in the second step the corresponding rules such as business rules, security rules, and
compliance rules are specified.

#### 10.3.1 The POM Business Transaction Model

The Rhea POM scenario was described in section 10.2.1. This section discusses how Solúbtha BTM is used
to specify the design-elements of the POM business transaction model. ARIS Express is used to design the
POM business transaction model.

The scope of the POM business transaction involves several parties including the Retailer, Rhea Automotive Ltd, Bank, Product Storage, Xport Limited, and Supplier 1. These parties collaborate to perform
transaction activities. The collaboration produces an ‘agreement’ between the parties. The agreement contains the common goals and obligations of or between the parties. The transaction roles are derived from the parties.

The POM business transaction model is composed of transaction process fragments (TPFs) that contain activities and control flows between these activities. In addition, it engages several transaction roles that are involved in the Rhea POM business transaction. Solútha BTM provides elements (refer to Chapter 7 for detailed information on Solútha BTM elements) for defining the TPFs, activities, control flows, and transaction roles. Figure 10.3 depicts business transaction model.

The POM business transaction is composed of five TPFs. Each TPF has a specific objective that must be achieved by executing the TPFs successfully. The TPFs are briefly described below:

- **Registration TPF**: The purpose of this transaction fragment is to register the purchase order placed by the Retailer. This fragment is composed of the ‘submit purchase order’, ‘receive purchase order’, ‘purchase order registration’, AND ‘credit worthiness check’ activities. Rhea Automotive Ltd., Bank, and the retailer, perform these activities collaboratively.

- **Inventory Reservation TPF**: This TPF reserves inventory for the order placed by the retailer. This fragment is composed of ‘process inventory reservation’ and ‘confirm reservation’. The ‘Process Inventory’ can be further decomposed into ‘check availability’ and ‘reserve inventory’. This fragment is performed by the sales and distribution department of Rhea Automotive Ltd.

- **Manufacturing TPF**: The ‘Manufacturing TPF’ is relatively more complex as it involves several partners. Rhea Automotive Ltd. collaborates with Supplier 1 for producing the new motor cycle parts.
This collaboration is operational at runtime depending on the situation such as replenishment of goods required due to shortage of Rhea’s inventory. Since ‘Manufacturing TPF’ is large, the transaction scope is decomposed into sub-scopes, they are, ‘Production of Goods Process’, ‘Transform Raw Materials into Finished Products’, and ‘Move Finished Goods to Warehouse’. In addition, it involves the activities Receive Order’, ‘Process Order’, and ‘Deliver Order’ of the Supplier 1 process. The decomposition results in a fine-grained manufacturing transaction process fragment. The granularity eases the complexities of business transaction management at runtime.

• **Invoicing TPF**: The purpose of this TPF is to prepare the invoice. The scope of invoicing TPF includes ‘Calculate Price’ and ‘Calculate Total Price’, and ‘Prepare Invoice’ activities which are included in the pricing transaction scope. Rhea Automotive Ltd and the Retailer perform these activities collaboratively.

• **Delivery TPF**: The purpose of the delivery transaction fragment is delivering the ordered products successfully to the buyer. The scope of the delivery transaction includes ‘Process Goods Delivery’ and ‘Deliver Goods to Destination’ activities. Rhea Automotive Ltd. performs the delivery transaction in collaboration with Xport Limited and the Retailer.

• **Payment TPF**: The purpose of this TPF is to execute the payment successfully against the invoice. The scope of the payment transaction fragment includes ‘process payment’ and ‘execute payment’. In the payment transaction, Rhea Automotive Ltd sends an invoice to the Retailer who triggers a payment execution request to the Bank.

Figure 10.4 shows the POM business transaction model composed of transaction process fragments. The control flow patterns: Sequence, NChoice, and XChoice, are used in the POM transaction model. The ‘NChoice pattern’ splits the outgoing path from register purchase order into ‘Calculate Price’ and ‘Process Inventory Reservation’. The ‘XChoice pattern’ splits the outgoing path of the Process Inventory Reservation activity. The remaining activities are composed using the ‘Sequence’ control follow pattern.
In the POM business transaction, some activities are simple while many of them are composite. ‘Register Purchase Order’, ‘Check Credit Card Worthiness’, ‘Execute Payment’, ‘Prepare Invoice’, ‘Process Inventory Registration’, and ‘Process Goods Delivery’, are composite activities whereas ‘Submit Purchase Order’, ‘Receive Purchase Order’, and ‘Calculate Price’ are simple activities.

The type of activity is an attribute. Solútha BTM provides two enumerated types of process activity. It is worth mentioning that in some cases the type of an activity is determined by the agreement. For instance, in the agreement used for validation, the first clause “Rhea Automotive Ltd. may send an acknowledgment of the purchase order” implies that ‘send PO acknowledgment’ is a possible action. Technically, if this action is not performed, the business transaction system does not issue abort command. This implies that the attribute type of ‘send PO acknowledgment’ activity as ‘non-vital’.

Additionally, the business protocol of the POM business transaction model is determined by the agreement. Clause 8 of the agreement determines the delivery and payment protocol of the POM business transaction model. This clause is, “the Retailer should make the payment to Rhea Automotive Ltd. within 15 days from the day the purchase order has been delivered”. This indicates that the payment TPF happens after the occurrence of the delivery TPF. The business protocol of the POM business transaction model is specified using the ‘succeed’ identifier. At runtime, the messages corresponding to these two TPFs will conform to this protocol. The listing below shows the business protocol of the POM business transaction models.

```
BusinessTransaction = “Rhea POM Business Transaction Model”
<BusinessProtocol>
    ptID = ‘‘bpIDentifier00001’’
    <Registration_TransactionProcessFragment>
    <Precede>
        <Inventory_TransactionProcessFragment>
        ptID = ‘‘bpIDentifier00002’’
        <Inventory_TransactionProcessFragment>
        <Precede>
        <Manufacturing_TransactionProcessFragment>
        ptID = ‘‘bpIDentifier00003’’
        <Inventory_TransactionProcessFragment>
        <Precede>
        <Delivery_TransactionProcessFragment>
        ptID = ‘‘bpIDentifier00004’’
        <Inventory_TransactionProcessFragment>
```
10.3.2 Specifying Agreement

I used the agreement package of Solúbtha BTM to specify the policies. The policy construct of the agreement package is used to specify the business policies are sets of rules. The specifications of these rules are presented in this section. The specification of these rules have been presented as schema using XML-like syntax.

TransactionProcessFragment = “Registration”

Document name = “Purchase Order”

BusinessPolicy:

bRule:

DocVariable: DocVariable1
AcknowledgementRequired = “Yes” RType = “soft”
DocVariable: DocVariable2
VerificationRequired = “Yes” RType = “hard”
GVariable: GVariable1
AcceptanceNotificationRequired = “Yes”
NotificationTime = 10 days
RType = “hard”

SecurityPolicy:

SecurityParameter: Parameter1
ConfidentialityRequired = “Yes”
RType = “hard”

Listing 2: Code Snippet of Business Rules for Registration TPF.
This code snippet is the specification of business and security rule that have been mapped from the agreement using the policy construct of Solúbtha BTM. The listing below presents the business rules of the invoice transaction process fragment.

TransactionProcessFragment = “Invoice”

Business Policy:
  bRule:
    DocVariable: DocVariable1
    CheckInvoice = “Yes” RType = “hard”

**Listing 3: Code Snippet of Business Rules for Invoice TPF.**

Listing 4 and 5 show the business rule specifications of the payment and delivery transaction process fragments.

TransactionProcessFragment = “Payment”

Business Policy:
  bRule:
    TCVariable: TCVariable1
    PaymentTime = 10 days

**Listing 4: Code Snippet of Business Rules for Payment TPF.**

TransactionProcessFragment = “Delivery”

Business Policy:
  bRule:
    TCVariable: TCVariable1
    DeliveryTime = 20 days
    GVariable: GVariable1
    PickUpLocation = “”
    DeliveryLocation = “”

**Listing 5: Code Snippet of Business Rules for Delivery TPF.**
10.4 POM Business Transaction Model Transformation

The POM business transaction model is simulated in this section. In order to simulate the model particularly, to simulate the behaviors of business transactions, the model needs to be transformed into machine-readable format. To this end, the POM model is transformed into executable codes using the UPPAAL language. The transformation occurs in two steps: firstly, the model is transformed into TEFSM, and secondly the transaction properties in particular, the business rules derived from agreement, are specified using CTL.

10.4.1 POM Automata

This section presents the POM business transaction automata which represents the business transaction system model. This model provides the semantics of the POM business transaction model. The automata is the machine-readable (formal) representation of the model presented in Figure 10.4.

The POM automata consists of five automata: registration template, inventory reservation template, invoicing template, payment template, and delivery template. These represent the transaction process fragments described in Section 10.3.1. These automata are parameterized automata that are presented graphically in this section. Behrman et al.,(2004) defines an automation as a graph which has locations (nodes) and edges.

The global integer variables, clocks, synchronization channels, and constants are defined in the global declaration section. Furthermore, variables and channels can be local to automata. The local declaration section facilitates specifying the variables, channels etc., of the corresponding automata.

The automation of transaction process fragments results in four basic states which are Inactive, Active, Committed, and Aborted. Also, the automation produces one or more intermediary states, ignored, resumed, waiting, repaired, substituted, mutualized, and compensated.

10.4.1.1 Registration Automata

The registration automata is composed of ‘sendPO’, ‘ReceivePO’, ‘SendAcknowledgement’, ‘VerifyPO’, ‘acceptPO’, ‘SendAcceptanceConfirmation’, and ‘CheckCreditWorthiness’. In addition to these, it contains the recovery states: ‘Ignoring’, ‘RepairingPO’, ‘Mutualizing’, ‘RepairingAckCW’, and ‘ResumingChkCW’. These states are executed if any fault is detected.

Each state in the registration automata is a commit type which indicates that no state can be skipped. The transition happens if and only if the active state is completed. It is worth mentioning that the three other types are initial, urgent, and regular.

The paths (edges) are annotated by specifying conditions using either guard or update. A guard is an expression that is specified using variables and clocks of the model to indicate when the transition is enabled, whereas an update is an expression which is evaluated as soon as the path is triggered (Guillermo
Figure 10.5: Registration Automata.

In the registration transaction, the SendPO state is activated as soon as the begin state is activated. The subsequent state, ReceivePO, will be inactive unless SendPO is committed (Completed). The commit of ReceivePO follows the next state. The transition from SendAcknowledgement to VerifyPO is enabled if the guard condition $!sendAck$ is evaluated to true, otherwise ignoring the state starts. The guard conditions as well as update conditions, are specified as Boolean conditions which are either true or false.

After committing the VerifyPO state, if the outgoing path condition $!veriSuccessful$ is evaluated to false, then RepairingPO is triggered and after successfully repairing, the acceptPO is activated. The temporal constraint corresponding to acceptPO is expressed as the invariant ‘$x <= 10$’ (refer to Listing 1 section 10.3.2 where this invariant has been specified as NotificationTime = 10 days) which means the total time to complete the production of goods should be evaluated to true, otherwise it will be treated as a fault and the Mutualize recovery operation will be activated. Once the mutualization is completed, the subsequent state CheckCreditWorthiness is activated. The commit of CheckCreditWorthiness leads the control to the final state, RegisterPO. The RepairChkCW is triggered otherwise (i.e., not committed).

If mutualization is not successful, then the path ‘!MutualizeAG == false’ is triggered and control moves to the ‘Compensating’ operation. The Compensating operation reverts the control from the last saved point.
to the initial state. Then the resume operation is triggered. With this, the transaction restarts from its initial states.

It is evident that Solúbtha BTM provides a series of failure prevention techniques, that from the technical point of view, do not let the running instance abort.

### 10.4.1.2 Inventory Template

The inventory automata is composed of CheckInventory PrepareProdOfGoods, ProcureRawMaterials, TransformRawMaterialToProducts, MoveProductsToPStorage, and ConfirmInventory. The automata contains a recovery state ‘MutualizeAg’. Figure 10.6 shows the inventory automata.

After the successful commit of the registration automata, the automation of inventory starts. It starts with committing the CheckInventory state. Then the transition condition ChkAvailability is evaluated. If it is true then ReserverInventory is activated and committed, otherwise, the PrepareProdOfGoods state triggers. The ProcureRawMaterial state is triggered when the evaluation of the expression !RMaterialAvailable is false, otherwise, TransformRawMaterialstoProduced is activated, followed by the commit of MoveProductsToPStorage.

![Inventory Automata](image)

**Figure 10.6: Inventory Automata.**

If MoveProductsToPStorage takes more than 15 (days), the mutualizing operation is triggered automatically. The guard condition !MutualizeAG is evaluated. If it is ‘true’ then control moves to the ‘TransformRawMaterialstoProduced’ state. If it is ‘false’, the consequence is always the same, the compensation
triggers and after compensation is committed successfully the resume operation starts.

### 10.4.1.3 Invoice Automata

The invoice automata is composed of ‘CalculateTPrice’, ‘PrepareInvoice’, ‘SendInvoice’, ‘ReceiveInvoice’, ‘VerifyInvoice’, and ‘StoreInvoice’ states. For preventing, the connection fault and incorrect data fault two recovery states have been incorporated in the automata. These states are ‘RepairInvoice’, ‘RepairConnection’, and ‘Retry’. Figure 10.7 shows the invoice template.

Invoice automation begins with activating ‘CalculatePrice’ which is committed as soon as the purchase order registration fragment is committed. Then, at the CalculateTPrice state, the total price for the order is calculated. The subsequent transaction states ‘PrepareInvoice’, ‘SendInvoice’, ‘ReceiveInvoice’, ‘VerifyInvoice’ and ‘StoreInvoice’ are committed sequentially.

![Invoice Automata Diagram](image)

**Figure 10.7: Invoice Automata.**

The ‘Retry’ state is triggered if the connection has failed. The path from ‘SendInvoice’ to the ‘Retry’
state is annotated with !connectionFailed == true. If the evaluation is true then the Retry state is activated. The retry state is executed a maximum of three times, expressed as the invariant Retrycount <= 3. When the outgoing path from Retry is activated, the !retryAttemptSuccessful is evaluated. If it is true, ReceiveInvoice state is activated, otherwise, the RepairConnection state is activated. Here, the system needs manual intervention to repair the connection because the connection cannot be repaired by the system.

If the invoice verification commits successfully, then ‘StoreInvoice’ is activated and committed. The ‘RepairInvoice’ activity remains inactive until the invoice verification state results in false. This could be because of ‘incorrect data’ or ‘missing required data’.

10.4.1.4 Delivery Automata

The delivery automata comprises a SendShipmentDetail, receiveShipmentDetail, CalculateShipmentCost, SendShipmentInvoice, SendASN states, and ProcessShipment. The paths are annotated using update expressions. Figure 10.8 shows the delivery automata.

![Delivery Automata](image)

The ‘SendShipmentDetail’ is triggered as soon as the delivery transaction begins. The subsequent states ‘receiveShipmentDetail’, ‘CalculateShipmentCost’, ‘SendShipmentInvoice’, ‘SendASN’, and ‘ProcessShipment’ are committed sequentially. The ‘ProcessShipment’ state should be completed within 20 days (refer to the Listing 5 section 9.3.2 where this invariant has been specified as DeliveryTime = 20 days). The expressions !ShipmentDelayed == true and !ShipmentDelayed == false are the outgoing paths of the ProcessShipment state. The invariant isShipmenDelayed is evaluated. If the evaluation results is ‘false’ then ShipGoodsToDestination state is triggered. Otherwise another shipper is invoked. This technique is called the ‘substitute’ provided by Solúbtha BTM.
10.4.1.5 Payment Automata

Payment is the last transaction process fragment of the the Rhea Automotive scenario. The payment automata is composed of SendPaymentexecutionR, ExecutePayment, and sendPaymentConfirmation states. The mutualize operation is performed at ‘Mutualizing’ state upon the occurrence of fault ‘payment delay’ which is expressed as a Boolean condition.

The payment transaction begins at sendPaymentConfirmation state, followed by the ExecutePayment state. The invariant $B\leq 10$ is a timing constraint for making a payment. The next transition depends on this condition. As soon as the path is activated, the expression PayPeriodExceeded is evaluated. If it is true then Mutualizing is triggered otherwise SendPaymentConfirmation will be activated and committed. Figure 10.9 shows the payment automata.

Furthermore, if mutualization is committed successfully, that is, if the payment period is extended, which is confirmed at runtime by evaluating the expression !MutualizationAgSuc == true, then the ExecutePayment is triggered and committed, otherwise a penalty will be imposed. The invariant $B\leq 10$ represents the time constraint for completing this state (referred to in Listing 4 in Section 10.3.2 where this invariant has been specified as PaymentTime = 10 days). Penalty is expressed as a condition '!Penalize == true'. The outgoing path of Mutualizing is evaluated by combining these two expressions into a single expression shown in Figure 10.9. If the conditions are evaluated to true then the ExecutePayment is activated and committed.
10.4.1.6 Compensation and Resume

For each transaction process fragment, there must be at least one compensation and one resume state. If any state of a transaction process fragment fails to be completed successfully, then compensation is activated. Once the compensation is completed successfully, resume is started. The figure below depicts the compensation and resume template.

![Compensation and Resume Template](image)

Figure 10.10: Compensation and Resume Template.

10.4.2 Transaction Property Specification

This section describes the underlying properties of the automata presented in the previous sections. The POM business transaction model relies on these properties to guarantee fault-tolerant and failure-resilient characteristics. In Chapter 9, several transaction properties have been defined to design flexible business transaction which exhibits these characteristics. Some of these properties have been used in the validation to tackle a subset of faults. The table below shows the properties that have been implemented to prevent failures in the POM business transaction.

| Property | Behavioral Property – 1 : (\forall \alpha, \forall \Gamma) \in M_{TP} \text{ commit}(M_{TP}) \rightarrow \square (\text{committed}(M_{TP})) \text{ such that, } (b\text{Rule} \lor c\text{Rule} \lor s\text{Rule}) \rightarrow \lambda (\Sigma), \text{ trigger } \epsilon^{\text{Ignore}} \text{ if and only if } \text{RType } ((c\text{Rule}, b\text{Rule}, s\text{Rule}) = \text{soft}). |
10.4. POM BUSINESS TRANSACTION MODEL TRANSFORMATION

Behavioral Property – 3: \( \forall \alpha_i, F(\alpha_i) \rightarrow \psi_{\text{Ignore}}, \text{commit}(\alpha_i) \rightarrow \) 
\( \Diamond(\text{commit}(\tau)) \) such that, \( \text{AType}(\alpha_i) = \text{NonVital}; \) \( \text{initiate}(\alpha_{i+1}) \rightarrow \) 
\( \text{continue}(\tau) \).

Substitute Behavioral Property – 4: \( \forall \Gamma_i \in \mathcal{MTP}, \text{commit}(\Gamma_i) \rightarrow \)  
\( \Diamond \text{committed}(\mathcal{MTP}) \) such that, \( (\Gamma_i = \{\alpha_i\} | \text{AType}(\alpha_i) = \text{Vital}) \), \( (F_{\text{Ship}} \lor U_{\text{SC}} \lor U_{\text{RP}}) \rightarrow \text{trigger} \psi_{\text{Substitute}} \rightarrow \text{substitute}(\mathcal{SC} \lor \mathcal{SP} \lor \mathcal{RP}) \).

Repair Behavioral Property – 9: \( \forall \tau(I_i) \in \mathcal{MTP}, \text{commit}(\tau(I_i)) \rightarrow \)  
\( \Diamond \text{committed}(\tau(I_i)) \) such that, \( (F_{\text{Ordering}} \lor F_{\text{BFSeQ}} \lor F_{\text{MSeQ}}) \land \text{Intelligibility} \lor \lambda_{\text{Temporal}} \lor F_{\text{MessageF}} \lor F_{\text{SIntegrity}} \lor F_{\text{Connection}} \lor F_{\text{Data}} \rightarrow \text{trigger} \psi_{\text{Repair}} \rightarrow \text{Repair}(\tau(I_i)) \).

Retry Behavioral Property – 16: \( \forall \tau(I_i) \in \mathcal{MTP}, \text{commit}(\alpha_i) \rightarrow \)  
\( \Diamond \text{committed}(\alpha_i) \) such that, \( F_{\text{Connection}} \rightarrow (\text{retry} = \text{op}(\text{task}) | \text{retryTaskLimit} = N) \).

Mutualize Behavioral Property – 13: \( \forall \tau(I_i) \in \mathcal{MTP}, \text{commit}(\tau(I_i)) \rightarrow \)  
\( \Diamond (\text{commit}(\tau(I_i))) \) such that, \( (E_{\text{Uncertainty}} \lor F_{\text{Temporal}} \lor D_{\text{Processing}}) \rightarrow \lambda(\Sigma) \rightarrow \text{trigger} \psi_{\text{Mutualize}}(\Sigma) \).

Compensate Behavioral Property – 17: \( \forall \tau(I_i) \in \mathcal{MTP}, \text{commit}(\tau(I_i)) \rightarrow \)  
\( \Diamond \text{aborted}(\tau(I_i)) \) such that, \( \neg \text{committed}(\tau(I_i)) \rightarrow \text{trigger} \psi_{\text{Compensation}}(\tau(I_i)) \).

Resume Behavioral Property – 19: \( \forall \tau(I_i) \in \mathcal{MTP}, \text{if Commit}(\Gamma) \rightarrow \)  
\( \Diamond \text{aborted}(\Gamma) \land \text{Compensated}(\mathcal{MTP}.\cup(\text{CheckPoint})) \land \text{Resumed}(\mathcal{MTP}.\cup(\text{CheckPoint})) \) such that, \( \text{aborted}(\Gamma) \land \neg \text{Repaired}(\Gamma) \land \neg \text{Substituted}(\Gamma) \land \neg \text{Mutualized}(\lambda) \).

Postpone Behavioral Property – 7: \( \Gamma_i \in \mathcal{MTP}, \text{substitute}(\mathcal{SP} \lor \mathcal{RP} \lor \mathcal{SC}) \rightarrow \)  
\( \text{Postponed}(\Gamma_i) \) until \text{repairing completed} such that, \( \text{substitute}(\mathcal{SP} \lor \mathcal{RP} \lor \mathcal{SC}) \rightarrow \text{trigger} \psi_{\text{Postpone}} \land \Gamma_i \text{ is re-scheduled.} \)
CHAPTER 10. VALIDATION AND EVALUATION

<table>
<thead>
<tr>
<th>Wait</th>
<th>Behavioral Property – 6: $\exists \Gamma_i \in M_{TP}$, Substitute $(RP \lor SP \lor SC) \rightarrow$ Wait $(\bigcirc \alpha_i \land \bigcirc \Gamma_i)$ until substitution operations is completed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Behavioral Property – 11: $\forall \tau (I_i) \in M_{TP}$, Repair $((F_{Ordering} \lor F_{BFESeq}) \lor F_{MSeQ} \lor \lambda_{Intelligibility} \lor F_{Temporal} \lor F_{MessageF} \lor F_{SIntegrity} \lor F_{Connection} \lor F_{Data}) \rightarrow$ Wait $(\bigcirc (\alpha_i \land \lambda_i))$ until repairing is completed such that, $((F_{Ordering} \lor F_{BFESeq} \lor F_{MSeQ} \lor \lambda_{Intelligibility} \lor F_{Temporal} \lor F_{MessageF} \lor F_{SIntegrity} \lor F_{Connection} \lor F_{Data})$ trigger $\epsilon_{Wait}$.</td>
</tr>
<tr>
<td>Suspend</td>
<td>Behavioral Property – 5: $\exists \Gamma_i \in M_{TP}$, Substitute $(SC) \rightarrow$ Suspend $(SC)$ for $I_i \in M_{TP}$, if and only if $(SC')$ is added in the business transaction system. MTP denotes a transactional business process that represents a business transaction model as well.</td>
</tr>
</tbody>
</table>

These properties ensure the safety and liveliness properties of business transactions. The liveliness property represents the event failure atomicity which is the fundamental principle of Solúbtha BTM. The safety property ensures that for any fault or failure that occurs at runtime, the potential abortions can be prevented. The properties in the table are defined using CTL, however, as UPPAAL supports TCTL, I transformed the CTL properties into the TCTL. The properties are encapsulated with the POM business transaction automata.

10.4.3 Evaluation

This section discusses the results of the evaluation of the model. In this section the POM business transaction model is simulated. In Section 10.1, I explained the different criteria for evaluating Solúbtha BTM. In this section, I evaluate whether Solúbtha BTM can satisfies these criteria. The discussion is based on the outcomes of the simulation except for the correctness, the correctness of the model has already been demonstrated using formal approaches in Chapter 8 and 9.

10.4.3.1 Correctness

In Chapter 8, the formal definitions of the design-elements were provided using ’Set Theory’, which is a formalism for defining and proving the syntactic correctness of a model or system. Solúbtha BTM was defined by composing these elements into a single structure. An important criterion for guaranteeing correctness of a model is its well-formedness. Well-formedness refers to the syntactic integrity of the model.
Section 8.2.4, I provided the formal definition of the structural integrity of Solúbtha BTM. I defined a set of structural constraints for the model. The instance-models must conform to these constraints. Additionally, the semantics of the design-elements were defined in Chapter 9. I used automata to define the formal meanings of the design-elements. I also used CTL to define the recovery policies which are the critical elements for flexible business transactions. The theory of flexibility is defined and proved using a formal method in Chapter 9.

The formal definitions and proofs demonstrate the correctness of Solúbtha BTM and also guarantee that the instance-models that are designed by conforming to the syntactic constraints are correct.

10.4.3.2 Correlation-ability

Solúbtha BTM was constructed by incorporating the business and functional perspective elements. The business perspective elements, in particular, ‘Agreement’ were logically connected explicitly to the functional perspective design-elements, ‘Transaction Process Fragments’ Transaction RecoveryHandler’ and ‘Business Protocol’. The logical structure of Solúbtha BTM exhibits a strong correlation between the business and functional constructs.

The model is used to design the Rhea POM business transaction model. Section 10.3 demonstrates the use of Solúbtha BTM in designing this business transaction instance-model. For this particular business scenario, Solúbtha BTM provided all necessary design-elements.

Considering the simulation outcomes of POM business transactions, the attribute NonVital of transaction activity, in particular, the SendAcknowledgement is determined by the clause of the agreement between the ‘Retailer’ and the ‘Rhea Automotive Ltd’. Additionally, the Retailer and Rhea Automotive Ltd agree upon the temporal policies corresponding to the operations such as ‘Send Oder Acceptance’. The Rhea Automotive Ltd. promises to send a notification to the Retailer within 10 days. Such policies are sourced from the agreement. Solúbtha BTM provides the keywords and operators to define them in the model as a business transaction rules. Solúbtha BTM enables to encapsulating the rules in TPFs as well as the activities.

The business protocol of TPFs is determined by the agreement. In the agreement, the payment happens on delivery of the motor cycle parts to the Retailer. Solúbtha BTM provides the business protocol operator, precede, to define the operational flow of the payment and delivery TPFs. In the POM business transaction model, the precede operator is used by using the UPPAAAL flow operator. Consequently, while simulating the payment TPF was activated after the delivery of goods was completed successfully. It shows that Solúbtha BTM enables defining the correlation between the business and functional perspectives. In consequence, the business transaction is controlled by the business perspective element ‘agreement’. Additionally, the recovery policies defined in the Rhea Automotive POM business transaction are in some cases,
and particularly, in the case of dealing with business related faults, influenced by the agreement. Solúbtha BTM provides parameters and operators to define them formally in the POM transaction model.

A correlation between the business perspective design-element ‘agreement’ and functional perspective elements TPFs, activities, business protocol, and recovery policies has been demonstrated by simulating the POM business transaction model. It also demonstrates that Solúbtha is able to provide necessary and sufficient design-elements for the establishment of such explicit correlation effectively and efficiently for the POM scenario.

10.4.3.3 Applicability

In this research, applicability has been viewed as the ability of the model to provide all required design-elements to design the business transactions of a particular business scenario. As I said in Chapter 3, I do not argue that the model is generic, although I strongly believe that the model is sufficiently abstract to enable designing business transactions for different scenarios. However, the model needs to be validated in different scenarios to establish this claim. In this section, I evaluate whether the model is applicable for the purchase order management scenario.

Solúbtha BTM has been developed to design the complex end-to-end business processes. The model has been constructed by studying three business cases. While studying the cases, the business processes were analyzed critically, understood, and realized in the business transaction model. Furthermore, each business entity, including agreement and policies that are related explicitly or implicitly to the business processes, was investigated separately. The design-elements which compose Solúbtha BTM are essentially the outcomes of the analysis. This is the first evidence that the model contains elements to design business transactions of end-to-end business processes. This has been validated in two ways. First, the Rhea POM business transaction instance-model has been designed in Section 10.3. The business scenario comprises source, make, and deliver processes. It involves multiple parties include the Rhea Automotive Ltd, the retailer, and Xport Limited. There is an agreement which essentially controls the business operations. Solúbtha BTM was used in the POM business transaction model. The results show that the model is applicable to the purchase order management business scenario because it provides the necessary design-elements. It is worth noting that the model does not provide elements for designing messages because it is out of the scope of the model. Second, the simulation demonstrates the successful execution of the model, specifically, no incompleteness for this scenario was observed during simulation. This proves the applicability of the model for such scenarios. Yet, as mentioned in Chapter 3 there are some instances that are specific to particular organization, which may not be specified or defined using Solúbtha BTM.
10.4.3.4 Flexibility

This feature was partly validated using automata in Chapter 9. This section discusses the simulation outcomes. Flexibility of Solúbtha BTM is evaluated over the characteristics discussed in Chapter 6. The evaluation outcomes are discussed in the following subsections.

**non-determinism**

: Solúbtha BTM provides six control flow patterns, NJoin, XJoin, NChoice, Interleaved, MIWSync and XChoice for designing non-deterministic transaction fragments. The syntax and semantics of these patterns were defined formally in Chapter 8 and 9. In addition, in Chapter 9, the non-deterministic business operations were exemplified using the AVERS OEM business scenario. In this chapter, the non-determinism properties have been used in the POM business transaction instance-model (See Section 10.3). It has been demonstrated that the model enables designing the non-deterministic transactions that enhance operational flexibility. Runtime flexibility was observed during the simulation of POM business transaction.

**Fault Tolerance and Failure-Resilience**

I present the simulation of the Rhea POM business transaction model in Figure 10.11. The Rhea Automotive POM business transaction begins with *Registration Transaction Fragment*. This begins with ‘Submit Purchase Order’ followed by the state ‘ReceivePO’. ‘SendAcknowledgement’ follows ‘ReceivePO’. The simulation shows that this activity did not commit which is a fault. However, the transaction continued. It activates the subsequent activity. The transaction system ignores this because ‘SendAcknowledgement’ was specified as a NonVital type activity. This demonstrates the fault-tolerant nature of business transactions.

The next state of ‘registration transaction fragment’ is ‘VerifyPO’. This state verifies the *Purchase Order* business object. Due to failure in the verification control moves to the ‘RepairingPO’ state which repairs the purchase order. While the fault is being repaired, the business transaction system *postpones* the commit of the subsequent states. These states must ‘Wait Until’ repairing of the fault is completed. Although, in the registration fragment, only the ‘Repair’ property is used, postpone and wait are the other underlying fault recovery design-elements encoded in the ‘Registration Transaction Fragment’. The purchase order is repaired successfully and the verification is committed, this prevents failure of the registration transaction.

The next fragment is the Inventory TPF, which is completed without any fault. However, the messaging fault occurred during the invoice transaction. While committing the ‘SendInvoice’ activity, the connection failure occurs. This activates the ‘Retry’ state to re-execute the failed state. The invariant ‘Retrycount=3’ is specified for this state. Because of this invariant the business transaction system allows execution of ‘SendInvoice’ three times. If all retries fail then the ‘RepairConnection’ triggers. This state requires *manual operation*. In addition, a new clock is set for this activity. The connection is repaired and ‘sendInvoice’ is
tried and committed successfully. This prevents the failure of the payment transaction.

Also, two other faults are seen in the simulation of the payment and delivery transactions. In the payment transaction, it takes more than 10 days to complete the ‘ExecutePayment’ state. This is a serious violation of the agreement. Business transactions fail due to the occurrence of this fault. However, in this case, MutualizeAg is triggered to set a new date for the payment, which prevents the failure of the payment transaction fragment and hence prevents business transaction abortion.
Figure 10.11: Simulation of Business Transaction of Rhea Automotive PMO Business Case.
In the delivery transaction fragment, the ‘ProcessShipment’ state could not be completed successfully within 20 days. According to the agreement, Rhea Automoive Ltd. reserves the right to cancel the shipper. When the company cancels the shipper, it activates the ‘InvokeNewShipper’ and carries out the shipment transaction. Invoking a new shipper refers to substituting a new shipper for the current shipper to prevent failure of the delivery transaction fragment.

Compensation and resume are triggered when all other recovery operations fail. If any TPF of a POM business transaction fails and cannot be recovered, then the compensation state activates and it reverts the transactions from the last savepoint to the initial state. A different simulation is designed for demonstrating the compensation and resume operations. Figure 10.12 shows the outcome of the compensation and resume simulation.

![The Simulation of Compensation and Resume](image)

The simulation reveals that a business transaction system which relies on Solúbtha BTM does not allow business transaction abortion until all recovery policies are executed. The abortion happens if and only if all recovery policies fail which can happen in a real-world scenario. Yet with all these recovery policies provided by Solúbtha BTM, I strongly believe that the transaction abortion can be prevented.
Furthermore, the recovery elements provided by Solúbtha BTM enable users to design cooperative recovery operations in the business transaction model. For instance, if repairing of a fault fails then the mutualize operation starts, which prevent business transaction abortion.

**Operational Autonomy and Extensible**

The real-world implementation of Solúbtha BTM relies on the SOA paradigm (which is the one context considered for this research) that prescribes distributed and loosely coupled implementation of transaction process fragments, so that the fragments can run on different sites autonomously. Yet they collaborate with each other through messages. The simulation demonstrates that no transaction process fragment in the POM model is controlled by any central controller. They are executed independently. Also, SOA enables extending the model by adding new fragments dynamically at runtime.

### 10.5 Comparative Analysis

I perform a comparative analysis to demonstrate the shortcomings of the state of the art which are addressed by Solúbtha BTM. In addition, the comparison justifies the novelty of Solúbtha BTM. The comparison is done in two phases: first Solúbtha BTM is compared with existing business transaction models and technologies against business related design-elements, and second the model is compared with existing models’ techniques for defining the flexible behaviour of business transactions. I studied existing models and technologies in Chapter 2. I extracted the features, specifically the design-elements and techniques, provided by the models and technologies (See Table 2.5 in Chapter 2). Since the models and technologies (shown in Tables 10.2 and 10.3) are currently being used to design business transactions, I compare my model with those to show the differences and to demonstrate the novelty of my model. I list the design-elements of Solúbtha BTM in the tables (the columns in Table 10.2) and then compare each of them to check whether any model provides the same or equivalent elements. The equivalence means functionally equivalence. If an element exists then I put ‘+’ sign in the cell of the corresponding model or technology. Otherwise I put ‘-’. I compare the techniques in Table 10.3 using the same approach.

#### 10.5.1 Comparison with Business Related Design Elements

This section presents a comparison of Solúbtha BTM with exiting models that I studied in Chapter 2. Table 10.2 presents the comparison of business related design-elements provided by business transaction models and technologies including Solúbtha BTM. The ‘+’ denotes ‘Yes’ meaning, the design-element exists in the model and the ‘-’ denotes ‘No’ if a model does not provide the element.
Table 10.2: Comparison between Existing Business Models and Technologies and Solábtha BTM

<table>
<thead>
<tr>
<th>Business Transaction Models and Technologies</th>
<th>Business Related Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested (Moss, 1981)</td>
<td>-</td>
</tr>
<tr>
<td>Sagas (Garcia-Molina and Salem, 1987)</td>
<td>-</td>
</tr>
<tr>
<td>Cooperative (Nodine and Zdonik, 1992)</td>
<td>-</td>
</tr>
<tr>
<td>Split-Join (Pu et al., 1988)</td>
<td>-</td>
</tr>
<tr>
<td>S (Veijalainen and Eliassen, 1991)</td>
<td>-</td>
</tr>
<tr>
<td>ConTract (Reuter and Wächter, 1991)</td>
<td>-</td>
</tr>
<tr>
<td>TMLA (Dayal et al., 1991)</td>
<td>-</td>
</tr>
<tr>
<td>WIDE (Ceri et al., 1997)</td>
<td>-</td>
</tr>
<tr>
<td>BAT (Papazoglou and Kratz, 2006)</td>
<td>+</td>
</tr>
<tr>
<td>Tx-QoS (Wang, 2011)</td>
<td>-</td>
</tr>
<tr>
<td>Asset (Biliris et al., 1994)</td>
<td>-</td>
</tr>
<tr>
<td>Cheetah (Pardon and Alonso, 2000)</td>
<td>-</td>
</tr>
<tr>
<td>JAS (Oracle Inc., 2006)</td>
<td>-</td>
</tr>
<tr>
<td>WS-TX (OASIS, 2009)</td>
<td>-</td>
</tr>
<tr>
<td>WS-CAF (Bunting et al., 2003)</td>
<td>-</td>
</tr>
<tr>
<td>BTP (OASIS, 2004)</td>
<td>-</td>
</tr>
<tr>
<td>UMM (UMM, 2006)</td>
<td>-</td>
</tr>
</tbody>
</table>
10.5. COMPARATIVE ANALYSIS

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>-</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPMN (OMG, 2010)</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BPEL (OASIS, 2007)</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Solúbtha</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

10.5.2 Comparison with Techniques

This section presents a comparison of techniques provided by existing business transaction models and technologies and Solúbtha BTM. The results of comparison are set out in Table 10.3.

The table shows that most of the existing business transaction models provide basic support to define the properties for flexible business transactions. Mostly, they provide compensation and resume operators. Very few provide retry, wait, and substitute operators. Solúbtha BTM provides more techniques than the existing business transaction models and technologies. Therefore, I conclude that Solúbtha BTM model is richer than existing models and technologies for designing highly flexible business transactions.
|               | Flexible | Resisute | Split-Join | Coalition | WIDE | TML A | BPEL | MDM | CHeetah | ASSL | T-X-QoS | WIDE | BPEL | T-KAM | CTMLA | WS-CAF | BTP | EAI | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | \mbox{Mark}\! | Solútha |
|---------------|----------|----------|------------|------------|------|-------|------|-----|----------|------|--------|------|------|-------|-------|-------|-----|----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| BPEL          | +        | -        | -          | -          | +    | +     | +    | +   | -        | -    | +      | +    | +    | +     | +     | +     | +   | -  | +             | -              | +              | -              | +              | -              | +              |
| EAI           | -        | +        | -          | -          | -    | +     | +    | +   | -        | -    | +      | +    | +    | +     | +     | +     | +   | -  | -             | +              | -              | +              | -              | +              | -              |
| WS-CAF        | -        | -        | +          | -          | -    | -     | +    | +   | -        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| BTP           | -        | -        | -          | +          | -    | -     | +    | +   | -        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| WS-TX         | -        | -        | -          | -          | -    | -     | -    | -   | +        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| ASSL          | -        | -        | -          | -          | -    | -     | +    | +   | -        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| Asset         | +        | -        | -          | -          | -    | -     | +    | +   | -        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| TL-GOS        | -        | -        | -          | -          | +    | +     | -    | -   | +        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| +             | -        | -        | -          | -          | +    | +     | -    | -   | +        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| -             | -        | -        | -          | -          | +    | +     | -    | -   | +        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
| BPEL          | -        | -        | -          | -          | +    | +     | -    | -   | +        | -    | -      | -    | +    | -     | -     | +     | +   | -  | -             | -              | -              | -              | +              | -              | -              |
10.6 Summary

Solúbtha BTM was validated and evaluated in this chapter. The model was validated against four criteria: correctness, flexibility, applicability, and ability to correlate the business and functional perspectives of end-to-end business process. The validation process consists of five phases shown in Figure 10.1.

Solúbtha BTM was used in the Rhea Automotive business case. In the first step, I designed the Rhea business transaction model using Solúbtha BTM. The model provided all necessary elements to design the business transactions. In addition, it was found that the model is able to provide techniques to define the fault-tolerant, failure-resilient, Non-deterministic, extensible, and operational autonomy properties of business transactions. In the next step, the POM business transaction model was simulated using the UPPAAL model checker. The simulation demonstrated how faults in the purchase order management business transaction were repaired efficiently and therefore, that failures or abortion business transactions are prevented.

The model was evaluated against a set of criteria. The simulation results were discussed against these criteria. The Solúbtha model was compared with existing business transaction models and technologies to justify the novelty of the model. In addition, it was compared over the techniques for defining the flexible business transactions.

Although the model was validated using rigorous approaches, in particular, the formal language, a few issues must be addressed in future work. First, the model should be validated in different and real-world business scenarios. A prototypical implementation of a business transaction could be a better approach for evaluating the efficiency and effectiveness of the model. In addition, the agreement used in this validation contains a very limited number of clauses which would not be the case in real-world scenarios. An implementation of business transactions with a real-world agreement would certainly be a better approach for evaluating the model.
Chapter 11

Conclusions

11.1 Introduction

In this chapter I summarise my research. I discuss the significance of the main findings of the research; outline the contributions; and suggest future extensions of this research.

The chapter begins by summarizing the research methodology. Then, I revisit the research objectives to verify whether or not the objectives were met. I summarise the research contributions. This chapter also highlights some limitations of this research. A list of potential extensions is suggested and the dissertation ends with a few remarks on the works that has been done in the past few years.

11.2 Usage of the Model

The research presented in this dissertation produced Solúbtha BTM. The purpose of the model is to enable users to design business transactions of end-to-end business processes. The model has been developed to cater for the automation of end-to-end purchase order management business processes in service-oriented environments. Thus, I positioned the model in the SOA layers (Papzoglou, 2008). In SOA layers, the business transaction model resides between the choreography layer and the business domain layer.

Model Users

The target users of this model are business analysts. Essentially, this is one of the main reasons I developed a model instead of a language. A language would require coding experience which might be a barrier against business analysts. Additionally, SBA developers and the application service providers could use the model to build business transactions of end-to-end business processes.

How to use the Solúbtha BTM?

At this stage, there is no graphical tool for using Solúbtha BTM. Therefore, implementing business transaction instance-model using Solúbtha BTM would be a code-intensive job. Also, a user would need to understand the formal foundation of the model in order to implement the transaction execution and recovery logic.
11.3 Summary of the Research Approach

I plan to develop a graphical tool. This tool would provide graphical notations for designing business transactions. Also it would hide the formal methods from end-users. Since the primary users are from the business domain, the graphical tool would be suitable. A business analyst could use it without having any expertise in formal method. The two main inputs would require for modeling business transactions are: an agreement and a business specification where the business operations are described precisely. Typically, the graphical notations are easier to use than hard-coding the transaction logic using XML or other high-level languages (example, C++, Java).

I used design science research methodology to conduct this research. The research method provides guidelines for conducting research in a rigorous manner. I used various methods to build the design-artifact which in this thesis is Solúbtha BTM. These methods guided every phase of the research, from literature review through to validation of the model. This research was aimed at creating a bridge between the business and functional perspectives of end-to-end business processes. The key purpose of integrating these two perspectives is to ensure that business transactions are governed from the business perspective. Since in real-world scenarios business operations are controlled by business domain entities such as ‘Agreement’ this integration is of critical importance.

I started the research by reviewing the literature. The literature review helped to position the research to meet the objectives. By reviewing the literature, the research problems were well-understood. It was found in the literature, that the existing transaction models and technologies mostly focus on the functional perspective of end-to-end business processes. The review pointed out an important shortcoming of the state-of-the-art, that is, the current business transaction models do not support correlating the business perspective with the functional perspective. Therefore, the existing models are not able to support integrating the business and functional perspectives.

This research aimed to developed a business transaction model which provides the logical connection between the business and the functional perspective of end-to-end business processes. To this end, Solúbtha business transaction model was developed by studying the secondary data of business cases. The secondary data were analyzed to extract the design-elements of the model. Three versions of the model were developed in this thesis. The Initial Version of the model was built by composing the design-elements elicited from the literature. The Initial Version is the foundation model of this research. An interim evaluation was conducted between the Initial Version and Solúbtha BTM V1 using expert opinion. Solúbtha BTM V1 was developed from the Initial Version. After the interim evaluation, the initial version of Solúbtha BTM was edited based on the suggestions made by the experts and the outcomes of the analysis of the secondary of Auto Inc. business case. This customization resulted in Solúbtha BTM V1.
I studied the secondary data of another two more business cases. The second case is about the automotive purchase order management of AVERS OEM and the third case is about logistics management. Logistics management was studied specifically for finding delivery related faults. During the study I found that business partner agreement is the main business perspective element that largely influences business operations. I studied several agreements in this thesis. The findings of the business cases and agreements were adopted by customizing Solúbtha BTM V1. The customization produced the final version of the model Solúbtha BTM.

Solúbtha BTM was constructed using UML 2.0. Since UML is a semi-formal language, the model was formalized using more expressive formalisms. Three different formalisms were used in defining the formal syntax and semantics of the model. The formalization provides the formal foundation of the model and enables business transaction systems to verify the instance-model at design-time. In addition, formalization provided the mathematical rigor of the model.

Solúbtha BTM was validated in the fourth phase. The validation environment was built using two different tools: the ARIS Express for designing the business transaction model and the UPPAAL model checker for simulating the behaviour of business transactions. First, I designed the purchase order management business transaction model. I used Solúbtha BTM in the Rhea purchase order management scenario to design the instance-model. Then, I transformed the model into automata. The model was simulated using the UPPAAL model checker. Finally, I evaluated the results by conducting a comparative study.

11.4 Meeting the Research Objective

Three objectives were set in the beginning of the thesis. The section describes how the objectives are met in the research.

# Objective 1: Develop a business transaction model that provides the elements to design business transactions by correlating explicitly the business and functional perspectives of end-to-end business processes.

Solúbtha BTM is the main contribution of this thesis. The model was developed by composing design-elements that were extracted from the literature, business cases, and trading partner agreements. The model provides elements to design business transactions of end-to-end business processes. The design-elements cover the business perspective and the functional perspective of end-to-end business processes. The business perspective elements enables specifying the real-world business related entities such as agreement and business policies. The functional perspective elements are used to specifying function related entities such as activity and transaction process fragment. The model provides the logical connections between the main business perspective element, ‘Agreement’, and the functional perspective elements, ‘TransactionProcess-Fragment’, ‘BusinessProtocol’, and ‘TransactionRecoveryHandler’. Using these connections, a correlation between the elements from these two perspectives can be defined. The logical connection enables defining
the business perspective elements along with the functional perspective elements. Additionally, Solúbtha BTM offers different types of operators and attributes. In short, Solúbtha BTM is able to provide elements to design the business transactions of purchase order management business processes by correlating the business and functional perspectives.

**Objective 2:** Develop a business transaction model that offers the techniques to design the flexible behavior of business transactions.

Solúbtha BTM provides several techniques to define the flexible behavior of business transactions. The model provides seven flow patterns to define the non-deterministic, extensible, and operational autonomy properties of business transactions. In addition, the model provides several operators to define the fault-tolerant and failure-resilience properties of business transactions. The model enables defining the policy based recovery properties of business transactions. The policy based recovery approach enables defining failure prevention techniques for a number of faults (discussed in Chapter 5). Essentially, it enables defining fault-specific recovery technique. Also, the model enables defining failure prevention techniques by combining one or more recovery operators. The model provides the logical operators AND, OR etc. to define conjunctive recovery policies.

**Objective 3:** Provide support to verify the instance models at design-time.

The UML representation of Solúbtha BTM is a semi-formal expression. Therefore, I encoded the model using automata and computational tree logic. The formal foundation of the model enables business transaction systems to verify the instance-models. Verification enables detecting the incorrectness of the model and facilitates re-engineering the model before it is deployed.

### 11.5 Summary of the Contribution

**Business transaction model.** Solúbtha BTM is the main contribution of this thesis. The model is a complete package for business transactions of end-to-end business processes. The model provides design elements, techniques, different types of operators, attributes, and logical operators to define the relationships between design-elements that compose business transaction models. Solúbtha BTM contains a rich set of design elements that cover both business and functional perspectives of the business transactions of end-to-end business processes. The elements are connected in the model by logical relationships that were defined using logical connectors such as composition, aggregation, generalization, and association. These enable structuring the elements into a business transaction model. The logical coupling between business and functional perspective elements force business transactions to be governed at runtime from the business perspectives. The model offers recovery operators to define the techniques for provisioning flexible
behavior of business transactions. These techniques implement flexible business transactions which are autonomous, non-deterministic, failure-resilient, fault-tolerant, and extensible.

Furthermore, the model offers application operators logical operators, arithmetic operators, relational operators, and logical operators to design business transactions. In addition, the model offers different types of attributes such as time related attributes, business document related attribute, and security attributes. These are used to define the structure of transaction process fragments and activities. Since Solúbtha BTM provides a list of built-in operators and attributes, the users can directly use these to defining business transaction instance-models.

**Transaction principle.** I developed a novel theory of flexible business transactions called *eventual failure atomicity* by extending the classical atomicity principle. Solúbtha BTM relies on this principle to facilitate the implementation of the fault-tolerant and failure-resilient characteristics of business transactions. It enables business transaction systems to prevent abortion. When a fault occurs, the business transaction system invokes recovery operation(s). A business transaction system does not issue an *abort* instruction until all possible failure prevention techniques have been executed. Consequently, business transactions become highly flexible.

**Fault models.** The research provides fault models that were constructed by incorporating the faults which were discovered by studying business cases. The fault models were presented as tree-like structures that enables traversing the tree and finding root causes of business transaction abortion. The root causes are the leaf node faults in the tree. These models were used as inputs to reasoning the operations at runtime and detecting faults when they occur. The fault models are very effective in finding the root causes of business transaction failures. Also, the faults enable invoking fault-specific recovery operations to prevent business transaction failure or abortion.

These fault models can be research interests for other researchers in academia allowing them to discover new mechanisms for preventing business transaction abortions. Researchers can now use and extend these fault models. For industry practitioners, these faults models are like a supervised dataset which they can use to build failure-resilient and fault-tolerant business transaction applications. Therefore, the use of the fault models goes beyond the context of business transactions. These models can be used in any fault-tolerant or failure-resilient business applications.

**Flexible characteristics.** This research defined five characteristics: fault-tolerance, failure-resilience, operational autonomy, extensible, and non-determinism, which are the basic building blocks of flexible business transaction. Solúbtha BTM has been built on these properties. Fault-tolerance enables business transaction systems to tolerate faults and allows business transactions to continue. If it is not possible to tolerate a fault then it results in a failure. The failure-resiliency enables business transaction systems to recover business transactions from failure. Solúbtha BTM enables business transaction systems to allow the applications to
11.6 Limitations of the Work

Innovations such as a product, model, theory, and idea may suffer from one or more limitations. This section explains the limitations of this research.

11.6.1 Limitations to the Research Method

Limitation of the Case Studies. This research used secondary cases for data collection. Although the main target was to study industrial cases to investigate business processes and to discuss them with the business experts of the companies, this was not possible due to some constraints. I then studied business cases that are publicly available. The S-Cube consortium prepared two of these case studies and made them accessible to everyone. The third case was found in Schimchy-Levy’s book (2007). Although the cases were well-written, these are the secondary cases. This poses various threats to the model such as threats related to applicability of the operators to define the flexible behaviour of real-world business scenario.

Limitation of the Agreement Study. I found that agreement is the central component of the business perspective. I studied several agreements independently of business cases to elicit more granular design
elements. I collected agreements from the Web. This is yet again a secondary data collection approach. Usually the trading partner agreements are confidential and are not disclosed – which could be a reason that agreements are not publicly available. This poses the risk of incompleteness in the model because real-world agreements may need some special keywords which would be specific to a context.

**Limitation of the Observation.** Since it was not possible to access any company, it was not possible to observe real-time business transactions. In particular, fault tolerant and failure-resilient characteristics of real-time business transactions could not be observed. Therefore, I used Solúbtha BTM and the Rhea POM business transaction model and simulated business transactions on paper to investigate the behavioral aspects of the business transactions of these cases. Since I designed the simulated business transaction model, it is likely that the model is different from the business transaction models of real-world business scenarios. Thus, the outcomes produced through observing the paper based simulation could be biased.

**Limitation of the Compliance Study.** Compliance is about ensuring that business processes, operations and practices are in accordance with a prescribed and/or agreed on set of norms (Sajia et al., 2007). Compliance is an important design element that was identified in every trading partner agreement studied in this research. I found that the clauses in the compliance paragraph provide references to legal frameworks such as Basel or the State law. I attempted to study the compliance issue separately. I partly studied Basel I, II, III and the Irish Statutory Law, and Irish Commercial Law. I found that specialized knowledge is required to interpret these laws because these are not straightforward statements. Additionally, each legal framework contains a large volume of laws. Interpretation of such a large volume of legal statements is time consuming task. In order to provide formal definition of these law, further investigation is necessary.

**Limitation of the Validation.** Although simulation is an efficient and widely used approach for validating technologies, one can argue that a prototypical implementation would demonstrate the real-world use and the bottleneck of the model. The business case where Solúbtha BTM has been used for validation is secondary, which pose threats such as threat to completeness of the model. The sustainability in terms of the failure resiliency and fault-tolerance of Solúbtha BTM has been validated, however, the scalability of the model is beyond the scope of this study.

**11.6.2 Limitations to the Research Product**

**Limitation of the Model.** Today, businesses cross borders, they are dynamic in nature and thus business requirements vary enormously and also they evolve over time. Although the Solúbtha BTM was built by studying three business cases and real-world business entities such as the trading partner agreements and compliance, I do not claim that the model is complete. Since it was not possible to study all compliance issues, the model may not be adequately supportive of all the required compliance rules in business transactions.
Additionally, the recovery policies depend on agreements. Therefore, an agreement that contains highly restrictive clauses may promote complexity in defining appropriate recovery policies for flexible business transactions. Specifically, if the clauses of agreement determines *hard rules* then it is difficult to define the flexible properties.

**Limitation of the Formalization.** Solúbtha BTM was formalized in this thesis. The formal syntax and semantics of the model were provided. However, there is a scope to improve the formal model.

### 11.7 Further Work

From this project there are research opportunities for future work.

#### 11.7.0.1 Feasibility and Scalability Study

Solúbtha BTM must be validated in other business scenarios. I strongly recommend conducting a rigorous test in highly complex business cases. Furthermore, while Solúbtha BTM was constructed by studying secondary data of three large-scale business cases, the model was validated in a small-scale business scenario which was also taken from the literature. In other words, no primary study was conducted in this research. Therefore, it is necessary to conduct a further research with the Solúbtha BTM in a large-scale business processes. This would validate the scalability of the model. Although simulation is a very useful method for validating a model, software, method, or theory, however, it not a real implementation of a system. An ideal case would be a prototypical implementation of the entire model. A prototype would be suitable for conducting feasibility and scalability studies.

#### 11.7.0.2 Extension of Solúbtha BTM

As already mentioned, today’s dynamic business environment promotes new requirements in terms of new design elements to design business transactions and techniques for preventing business transaction abortions. A potential research area is to enrich the business transaction model by adopting more elements and techniques.

#### 11.7.0.3 A Language for Designing Business Transaction

The aim of this research is to develop a technology agnostic business transaction model which means the model should be implemented without using any platform specific language. However, Solúbtha BTM incorporated a few elements such as standard discriminator which could be foreign to existing languages, especially to many declarative languages. I suggest a declarative language for designing the business transactions because it would be more human readable than an imperative language (*e.g.* Java) or functional programming language (*e.g.*, Python). A potential and interesting research direction would be to design
and develop a new declarative language or to extend the most suitable candidate language for designing business transactions.

11.7.0.4 Integration with SOA Technologies

Technological support is needed to implement Solúbtha in a SOA based runtime environment. A component which enables a seamless integration of Solúbtha BTM with the World Wide Web Consortium (W3C)\(^1\) recommended standard technologies such as ‘WS-Coordination’, and ‘WS-BusinessActivity’ will be very efficient and effective in managing the business transactions of complex SBAs. This will vigorously change the current business transaction management landscape as well.

11.7.0.5 Tool Support for Business Transaction

Lastly, building a tool for designing, implementing, deploying, and managing business transactions could be a potential future research topic. The main concern of this topic would be building a user-friendly tool for designing the business transaction models, simulating the business transactions for verifying the correctness of the model, implementing and deploying the model, and managing the business transactions at runtime. As mentioned in the previous section, I have used ARIS Express for designing the POM business transaction model. I found that such tools do not provide a few critical elements such as business protocol to design business transactions. The tool would be built by adapting Solúbtha BTM. Such a tool would be very useful for industry.

11.8 Closing Remarks

Lately, business transactions have drawn research interests in the area of service oriented computing. Perhaps it is one of the most complex research topics in the service oriented domain\(^2\). The main reasons for complexity are, (i) it is distributed over a large number of nodes; (ii) it may access tens of databases; and (iii) since the business transaction systems are SBAs, often composed of third party software, it poses a huge challenge regarding business transation management and software availability. For an the effective design of business transactions, an efficient model is required. Solúbtha BTM is an effort to address this requirement.

\(^{1}\)www.w3c.org

\(^{2}\)This is a comment made by one of the reviewers of a high-ranked conference where the core result of this research was submitted.


ArcelorMittal (2013). ArcelorMittal Tubular Products - Purchase Order.


PriceWaterhouseCoppers (2007). Guide to Key Performance Indicators: communicating the measures that matters’.


263
Vandrager, F. A First Introduction to UPPAAL. Institute for Computing and Information Sciences, Radboud University Nijmegen, Heijendaalseweg 135, 6525 AJ Nijmegen, The Netherlands.


Appendix

Appendix A. Fundamental Concepts

Many advanced properties of business transaction models and technologies are originated from the ACID (Atomicity, Consistency, Isolation, Durability) properties especially from the atomicity, isolation, and consistency properties. Some models e.g. Nested Transaction Model (Moss, 1981) and also technologies e.g. WS-TX (OASIS, 2009a) still rely on atomicity property. Therefore, I provide a detailed description of these properties in this section. This may help readers to understand the fundamental principles of business transaction models and technologies. In addition, concurrency is another important concept of business transaction discussed in this section.

Appendix A1. ACID

Atomicity

The atomicity guarantees the successful execution of all (sub-) transactions (aka Cohort) that are contained in a transaction scope. Atomicity does not allow the failure of any transaction within the transaction scope because it relies on the all or nothing principle that means, either executing all transactions or none.

Atomicity is strongly desired in cases such as software or hardware crashes or other unexpected or undesired behaviors of transactions. Any failure of a transaction is undesirable for the users. In the case of a failure, transaction systems are not allowed to stay in an intermediate state. Therefore, the failure of a transaction results in transaction abortion. Upon transaction abortion, the cohorts that were completed before the failure occurred are forced to fail and the cohorts which have not started yet will never start – this is the atomicity principle.

Consistency

The Consistency refers to the correctness of the state of the database that a committed transaction produces (Chrysanthis and Ramanritham, 1990). Consistency has become a crucial transaction property with the advent of the concurrency concept. Concurrent transactions give rise to several anomalies that may occur while a transaction is running, in particular lost updates and dirty reads (Traiger et al., 1982). Consistency resists these anomalies and ensures correctness (Gray and Reuter, 1989) (Bernstein, A. P., 1997). Con-
sistency guarantees that transactions produce results by complying with business rules that are constraints associated with its manipulated state (Leymann and Roller, 2000).

**Isolation**

Events within a transaction must be hidden from the other transactions running concurrently (Härder Reuter, 1983). A program running under transaction protection must behave exactly as it would in single-user mode; it is not an entirely independent requirement; rather it must be achieved to guarantee consistent input data, which is a prerequisite for consistent output (Gray and Reuter, 1989). If a transaction processing system contains a collection of sub-transactions that have to be executed concurrently by sharing the resources (e.g., database), the isolation property ensures that the sub-transactions are executed separately; otherwise a transaction will produce an inconsistent outcome. Isolation is realized by forcing the *serializability*. Serialization in transactions is not observable from outside because in the transaction processing system the internal details are hidden.

**Durability**

The durability property ensures data persistency. When a transaction completes execution, all of its updates are stored on a container such as disk storage that will survive the failure of the transaction processing system (Bernstein, A. P., 1997). Durability guarantees the user that the outcomes are retrievable. Thus, durability is an important transaction property.

**Appendix A2. Concurrency**

Concurrency denotes simultaneous occurrence (transaction) of multiple processes, functions, or threads. Unlike sequential systems, concurrent systems carry out a collection of processes simultaneously on multiple processors or multi-cored single processor. For instance, in a sequential system execution of processes P, Q, and R execute on single-cored processor at time \( t_1, t_2 \) and \( t_3 \) respectively. Since the execution of one process makes the succeeding process wait until it finishes execution, the system is not able to operate these three processes in parallel. Whereas, in a concurrent system, processes P, Q, and R execute on multi-cored processor at time \( t \); the process scheduler or dispatcher of concurrent system assigns core (independent processing unit) for each process that represents one (sub-) transaction. Figure 1 shows sequential and concurrent execution.

Concurrency is the cornerstone of parallel computing. Concurrency comes in different variants, particularly, *concurrent execution* and *concurrent access* to shared resources. Concurrent execution means multiple processes or functions running simultaneously to carry out operations on different processors. Computing paradigms such as parallel computing, grid computing, multiprogramming and cluster computing are built...
on the concurrent execution model. Concurrent access to shared resources – the other variant – denotes more than one process or function or thread access to shared memory, databases, and applications.

Appendix B. Interim Evaluation

In this section, the I report the discussion with the experts and their opinions on the initial version of Solúbtha BTM. The experts gave their opinions on the issues: simplex or complex, Complete or Incomplete, Over Modeled or Under Modeled, Degree of Correlation between Business and Functional Perspective, and Correctness of Relationships between Elements. A brief description of these issues has been provided in the research methodology chapter. It is important to note that the initial version of the model was sent to the participants before the discussions. The expert opinions on these issues are provided in the sections below.

Simplex or Complex

The first participant found the model moderately complex as it incorporated a number of design elements. According to the participant, the model is complex due to the complex relationships between the design elements. The users of the model might have difficulties in designing business transactions using this version of the model. In response, the I argued that the design elements were included considering the requirements of a business transaction; essentially, these elements are necessary for designing a business transaction by correlating the business and functional perspectives. Then, in response, the expert suggested simplifying the model by applying merging or abstraction mechanisms.

The second participant found that some of the logical connections between elements were complex too, especially the dependencies between the elements. The dependency between Activity and TransactionActivityHandler is an example of complex dependency. The participant argued that the transaction designers might have difficulties in designing the dependency relationships between these elements.

The third participant found that the functional perspective package was relatively more complex than the business perspective. The expert strongly argued that the elements (those listed in the Table 1 below) would increase the design complexity for the designer, specifically, mapping the conditions of FaultHandler,
EventHandler, and CompensationHandler in TransactionEventHandler would be problematic. Besides, the expert strongly opined that these elements would create confusion for the transaction designer or developer because these elements are functionally equivalent; for instance, EventHandler and TransactionEventHandler are functionally the same.

Table 1: List of Excessive Design-elements

<table>
<thead>
<tr>
<th>Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Event Handler</td>
</tr>
<tr>
<td>Transaction Failure Handler</td>
</tr>
<tr>
<td>Fault Handler</td>
</tr>
<tr>
<td>Event Handler</td>
</tr>
<tr>
<td>Compensation Handler</td>
</tr>
</tbody>
</table>

Considering the suggestion made by the first participant, I studied the model and found that several merges were possible because either one of two elements could serve the same purpose. In a word, they are functionally similar. For instance, ProcessPerformance can be used to specify the TransactionProcessPerformance because they share the same performance metric. Thus, I merged transaction ProcessPerformance and ProcessPerformance into ProcessPerformance. Additionally, the concepts role and transaction role are the same. Both represent a role such as buyer or seller. These two elements were merged into Role. Moreover, there is no difference between BusinessFunction and ProcessFragment (Eberle et al., 2009). For instance, payment function of a order management process is a payment fragment. These two elements were merged into ProcessFragment.

The third participant pointed out which of the elements could be merged. He suggested FaultHandler and TransactionFaultHandler could be merged because TransactionEventHandler is not required as EventHandler handles events which are practically the transaction events. Similarly, FaultHandler is adequate to specify the handling of transaction failures. I performed these merging operations.

**Complete or Incomplete**

All three participants gave the same opinion regarding the completeness of the model. According to them, it is difficult to justify the completeness as completeness depends on whether the business transaction model has the required design elements to design business transactions of almost any large-scale business processes of the target domain. It is not easy to evaluate the completeness by only studying the model. The participants are suggested a suitable option for evaluating completeness is designing a business transaction of a business scenario. In any case, the first and third participants made a few opinions on this issue.

According to the first participant, the model contains a number of design elements covering both the business and functional perspectives. He opined that the model contains the necessary and sufficient ele-
ments for designing business perspective of business transactions.

The third participant shared the same opinion as the first one on the business perspective package of the model. He commented more on the functional aspect of the model. The expert thought that the functional package contains a sufficient number of elements. However, I was suggested investigating the business process concept to find if any design-element were missing.

The suggestions from the experts motivated me to conduct case studies. I analysed secondary data of three business cases to determine if the model could be extended.

**Over Modeled or Under Modeled**

Over modeled denotes that the model contains one or more elements which may not be used in practice whereas under modeled denotes one or more elements are missing from the model.

The second participant had no opinion on any specific part of the model, however, the participant made the generic comment that this version of the model is over modeled.

Both the first and third participants concluded that this version of the model was over modeled. The first participant pointed out that the element StrategicConstraints is a redundant element because it is functionally and semantically the same as ‘Policy’. The participant cited the definition of policy given in JRA (2008), *a policy is a set of rules which are essentially the strategic constraints which must be followed by the trading partner*. Thus there are no differences between the concept strategic constraint and policies. The expert also opined the same for the element ContractualConstraints. It is redundant because Agreement is same as contractual constraint. This implies that ContractualConstraints can be merged into Agreement. Moreover, the participant identified a list of elements that are not required. For instance, TransactionActivityClassifier, TransactionActivityHandler, TransactionActivityCharacterizer, TransactionContextHandler, and TransactionWorkflowActivityHandler will be of no use in designing business transactions.

The third participant shared the same opinion as the second participant on redundant elements as well as excessive elements. The expert identified that the transaction structure can be defined using the element BusinessProcess. This implies that the element TransactionStructureStyle is an excessive element in the model. Moreover, while looking at the business perspective package, the participant pointed out that the elements TransactionConstraints, BusinessArea and ProcessArea may not be used in practice. The expert opined that while it is no harm if the model contains these elements, however, the unused elements will promote the unnecessary design complexity.

I removed StrategicConstraints. I agreed with the second participant that it is an redundant element. I also agreed with suggestion made about excessive elements after reviewing Ouyang *et al.* (2005). I understood that no extra element is needed to classify transaction activity because activity can be typed
using the attribute \textit{type}. Furthermore, transaction activity classification, characterization, activity handling, and workflow handling can be done by the process manager which is a software component (Leymann and Reuter, 2000).

I also removed TransactionStructureStyle, business area, process area, and transaction constraints. I agreed with the third participant that BusinessProcess is adequate for specifying the transaction structure. The transaction structure can be flat or nested. The atomic activities in a business process are flat which means that if a transaction scope contains only the non-decomposable activities, the transaction structure is flat. Conversely, if a transaction scope contains nested activity, the structure is nested. The BusinessProcess defines these structures. Therefore, no extra element is required.

\textbf{Degree of Correlation between the Business and Functional Perspective}

This is an important issue as one of the main objectives of this dissertation is to establish a correlation between business functional perspective of business transaction.

The second participant thinks that she lacks sufficient level of expertise on the business aspects of transactions. However, based on her understanding, she commented that there is a composition relation between the functional perspective element TransactionConstraints and business perspective elements Agreement. Thus, the participant believes that there exists a correlation between business and functional perspective.

The first and third participants also stated that their expertise is not sufficient to measure the degree of correlation. Nevertheless, the third participant opined that I could connect the business perspective element Agreement to business transaction. This would increase the degree of correlation between these two perspectives.

I did not perform any modification in the model on this issue. I decided to conduct the case study first.

\textbf{Syntactic Correctness of Relationships between Elements}

The UML representation of Solúbtha BTM proved quite effective during the interim evaluation. It helped the experts to understand the logical structure of the model. The UML class diagram depicted the structure in a semi-formal manner and thus, the experts were able to assess the correctness of the model.

The first participant commented that the relation between the concepts Policy and Compliance is logically incorrect. She argued that compliance is a subtype of policies and therefore compliance should be composed into policy.

The second participant concluded that the composition relation between the key performance indicator and agreement is too strong from the logical point of view. The expert stated that the composition relation between these elements is correct, nevertheless, from the technical perspective the composition relation develops a strong relationship between the elements which may pose difficulties at runtime. She suggested
replacing the composition connector by an aggregation connector to avoiding any unwanted consequences at runtime–unless there is a strong reason to use a composition connector.

The third participant brought one important point to the my attention. The expert opined that the association between the concepts Protocol and BusinessProcess could be semantically rich by using a rolename that refers to a name for specializing association.

I modified the composition relations to aggregation relations. I agreed that the composition connector between Agreement and KeyPerformanceIndicator should be changed to aggregation. This will release the strictness in terms of relationships between these elements. For example, in a composition relationship, if KeyPerformanceIndicator is deleted from the model then Agreement is deleted automatically, whereas, in an aggregation relationship, deletion of KeyPerformanceIndicator does not result in the deletion of Agreement. In aggregation, a parent class can be an empty set. Table 2 shows the modification of connectors between the elements.

Table 2: Actions on Relationships between Design Elements and Status

<table>
<thead>
<tr>
<th>Relation (E1, E2)</th>
<th>Connector</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Agreement, KeyPerformanceIndicator)</td>
<td>Composition</td>
<td>Modify</td>
</tr>
<tr>
<td>(Compliance, Policy)</td>
<td>Composition</td>
<td>Modify</td>
</tr>
<tr>
<td>(Policy, BusinessRules)</td>
<td>Composition</td>
<td>Modify</td>
</tr>
</tbody>
</table>

Table 3 shows the operations and corresponding operations that were performed during interim evaluation. The left and right column show the design element and the operations simultaneously. Merge action denotes merging an element into another element. Delete action refers to removing an element from the model.

Appendix C. Unified Modeling Language 2.0

The Unified Modeling Language (UML) is developed by Grady et al. (1999). The developers were employee of then Rational Software currently IBM\(^3\). It was adopted by OMG in 1997 and was accepted by the International Organization for Standardization (ISO)\(^4\) in 2000. UML is a general-purpose visual language for specifying, constructing, and documenting the artifacts of systems (OMG, 2005). It is used to understand, design, browse, configure, maintain, and control information about a system (Booch et al., 1991).

UML is a widely used language for capturing structure, behavior, and functionalities of a software system (Noran, 2000). It contains nine different types of diagrams. Noran (2000) provides a detailed

\(^3\)www.ibm.com
\(^4\)www.iso.org/iso/home.html
Table 3: The List of Design-elements and Actions

<table>
<thead>
<tr>
<th>Design Elements</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Event Handler</td>
<td>Merged</td>
</tr>
<tr>
<td>Business Collaboration</td>
<td>Merged</td>
</tr>
<tr>
<td>Transaction Activity Handler</td>
<td>Merged</td>
</tr>
<tr>
<td>Transaction Activity Classifier</td>
<td>Deleted</td>
</tr>
<tr>
<td>Transaction Activity Characterizer</td>
<td>Deleted</td>
</tr>
<tr>
<td>Transaction Performance</td>
<td>Merged</td>
</tr>
<tr>
<td>Transaction Workflow Activity Handler</td>
<td>Deleted</td>
</tr>
<tr>
<td>Transaction Performance Handler</td>
<td>Deleted</td>
</tr>
<tr>
<td>Strategic Constraints</td>
<td>Merged</td>
</tr>
<tr>
<td>Business Area</td>
<td>Deleted</td>
</tr>
<tr>
<td>Process Area</td>
<td>Deleted</td>
</tr>
<tr>
<td>Transaction Roles</td>
<td>Merged</td>
</tr>
<tr>
<td>Contractual Constraints</td>
<td>Merged</td>
</tr>
<tr>
<td>Transaction context handler</td>
<td>Deleted</td>
</tr>
<tr>
<td>Business Function</td>
<td>Merged</td>
</tr>
<tr>
<td>Transaction Structure Style</td>
<td>Deleted</td>
</tr>
</tbody>
</table>

description of these diagrams.

- **Class Diagram** consists of classes and relationships. It is used for capturing the structure of a (software) system.

- **Object Diagram** is an instantiation of a class diagram for capturing a specific context.

- **Statechart Diagram** is used for describing the states of a system. These states capture the lifecycle of objects or systems.

- **Activity Diagram** is used for describing the activities and actions of a software system.

- **Sequence Diagram** is used for capturing messages that are exchanged between objects.

- **Collaboration Diagram** is similar to sequence diagram however, it is used for capturing complex interactions and relations between collaborating objects.

- **Use Case Diagram** is used for capturing parts of a system’s functionality.

- **Component Diagram** is used for structuring components in a software system.

- **Deployment Diagram** is used for demonstrating hardware within a software system.
UML has extension mechanisms which include stereotypes, tagged value, and constraints expressed in Object Constraint Language (OCL) to extend UML to serve specific purposes. UML was developed for building software systems, nevertheless, the class diagram serves various purposes such as describing models. The semantics of UML notations that are used in constructing Solúbtha BTM are given below. These semantics are taken from Grady et al. (Grady et al. 1999).

- This notation denotes **class** used to capture the **elements** of Solúbtha BTM.

- This notation denotes **generalization** used to define inheritance and polymorphism relations between the elements of Solúbtha BTM.

- This notation denotes **association** used to define association relation between the elements of Solúbtha BTM.

- This notation denotes aggregation. Aggregation is a special type of association. It represents part-whole relationship between the elements of Solúbtha BTM.

- This notation denotes composition. Composition is another special type of association however, it is a stronger form of association which represents whole-part relationship. It is used to define the composition relationship between the elements of Solúbtha BTM.

- It denotes dependency. Dependency notation represents the dependency relation between the elements of Solúbtha BTM.

The UML class diagram facilitates specifying the multiplicity of a set. It indicates an allowable number of instances in a set in a data structure. Booch et al. (2000) define multiplicity as a specification of the range of allowable cardinality values.
Appendix D. The Initial Version of Solúbtha BTM

In the Initial Version of Solúbtha BTM, my main focus was to construct the model by incorporating the design-elements that cover the business and functional perspectives of end-to-end purchase order management processes. I constructed this version by including the design-elements which were elicited from the literature. Additionally, some novel elements are added in this version.

The Initial Version of the model is composed of 39 design-elements. 16 of the 39 are compound element and the remaining are atomic elements. The compound elements are those that contain other elements whereas the atomic elements are the non-decomposable ones. These elements are described in Table 4. The Table also explains the reasons for including them in the model. The left column of the table shows the name of the elements and the right one shows causes for adding them in the model.

Table 4: The Design-elements and causes for including them in the Initial Version of Solúbtha BTM

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCollaboration</td>
<td>It is a business perspective compound element. Since many business transaction properties rely on collaboration between trading partners, BusinessCollaboration is an important element. It enables users specifying business messages that are exchanged between two trading partners, their content, and their precise sequence and timing (Papazoglou, 2003). It was introduced in UMM and later adopted in the business process specification schema (BPSS) (eBXML BPSS, 2001) and in the business transaction model proposed by Papazoglou and Kratz (2006).</td>
</tr>
<tr>
<td>BusinessArea</td>
<td>It is a business perspective element. It was included to specify the areas of business such as logistics, manufacturing etc. within the scopes of business transactions.</td>
</tr>
<tr>
<td>ProcessArea</td>
<td>It is a business perspective element. It was incorporated to specify the areas of processes within the scopes of business transactions.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Compliance</td>
<td>This element was introduced in the Business-Aware Transaction Model (Papazoglou and Kratz, 2006). Also, it was found in trading partner agreements and in the compliance management framework proposed by Elgammal et al. (2011). It is a business perspective element. The element was included to specify policies and different types of constraints.</td>
</tr>
<tr>
<td>StrategicConstraints</td>
<td>It is a business perspective element. This was included to specify the constraints related to organizational strategies.</td>
</tr>
<tr>
<td>ContractualConstraints</td>
<td>It is a business perspective element. This was included to specify the constraints related to contracts between trading partners.</td>
</tr>
<tr>
<td>OperationalConstraints</td>
<td>It is a business perspective element. This was included to specify the constraints related to business operations.</td>
</tr>
<tr>
<td>Policy</td>
<td>It is a business perspective element. This was included to specify business rules in business transaction models.</td>
</tr>
<tr>
<td>BusinessRules</td>
<td>It is a business perspective element. The element was included in the model to specify business related constraints.</td>
</tr>
<tr>
<td>Agreement</td>
<td>It is a business perspective element. This was included in the model to specify the clauses of an agreement between the trading partners.</td>
</tr>
<tr>
<td>KeyPerformanceIndicator</td>
<td>It denotes the factors by which the development, performance or position of the business of the companies can be measured effectively (PriceWaterhouseCoppers, 2007). KeyPerformanceIndicator was included to specify performance metrics of the business processes that are scoped in business transactions.</td>
</tr>
<tr>
<td>BusinessFunction</td>
<td>A business function is defined as a description of the well-defined and commonly accepted critical business principle that transforms business values and causes state changes to transaction participants, transform an unpaid order to a paid order (Papzoglou and Kratz, 2006). A business transaction may consist of several business functions. This element was introduced Business-Aware Transaction Model. It is a functional perspective element. It was adopted in Solóbtha BTM to enable users to specify business functions.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BusinessProcess</td>
<td>It was found in literature e.g. (OASIS, 2005), (OMG, 2008), (OASIS, 2004) etc.. It is the key functional perspective element. It was included in Solúbtha BTM to define business processes within the scope of business transactions. A business transaction may involve several business processes within its scope.</td>
</tr>
<tr>
<td>BusinessObject</td>
<td>It is a functional perspective element. It was included to specify the business objects such as purchase order and invoice.</td>
</tr>
<tr>
<td>ReferentialPrimitive</td>
<td>It is a functional perspective element. This element has been taken from (Papazoglou and Kratz, 2006). It enables to specify the reference functions such as payment function refers to order function.</td>
</tr>
<tr>
<td>BusinessPrimitive</td>
<td>It is a functional perspective element. It was included to specify various business functions such as means of payment.</td>
</tr>
<tr>
<td>ProcessFragment</td>
<td>It is a functional perspective element. It was found in (Eberle et al., 2009). It was included in the model to enable users to specify different units of business processes within the scope of business transactions.</td>
</tr>
<tr>
<td>Actors</td>
<td>It is a functional perspective element. It was included to specify the actors involved in business transactions.</td>
</tr>
<tr>
<td>Roles</td>
<td>Roles are refined from actors. This is a functional perspective element which enables specifying the roles involve in business transactions.</td>
</tr>
<tr>
<td>Activity</td>
<td>It is a functional perspective element. It enables to specify activities that compose process fragments.</td>
</tr>
<tr>
<td>FaultHandler</td>
<td>It is a functional perspective element. It is used for specifying the conditions for handling faults upon their occurrence.</td>
</tr>
<tr>
<td>EventHandler</td>
<td>It is a functional perspective element. It is used for specifying conditions to handle events.</td>
</tr>
<tr>
<td>CompensationHandler</td>
<td>It is a functional perspective element. This element specifies conditions for undoing the completed sub-transactions upon occurrence a fault such as payment canceled.</td>
</tr>
<tr>
<td><strong>TransactionFailureHandler</strong></td>
<td>It is a functional perspective element. This element enables specifying conditions to prevent failures of business transactions at runtime.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TransactionConstraints</strong></td>
<td>It is a functional perspective element. It was included in the model to specify the clauses of agreement as transaction constraints. Also, it enables to specify the compliance constraints as transaction constraints.</td>
</tr>
<tr>
<td><strong>TransactionContextHandler</strong></td>
<td>This element was found in J2EE Transaction Context, Java Transaction API, and Microsoft Transaction Server. It is a functional perspective element. It has been adopted to define business transaction scopes. A business transaction scope denotes the boundary of a business transaction. Additionally, it enables to specify techniques for handling contexts such as extending the transaction scope automatically at runtime.</td>
</tr>
<tr>
<td><strong>TransactionWorkflowHandler</strong></td>
<td>It is a functional perspective element. A business transaction scope includes a set of sub-transactions that should be sequenced correctly to produce the outcome expected by a user. The TransactionWorkflowHandler was included in Solúbtha BTM to specify sequences of these sub-transactions.</td>
</tr>
<tr>
<td><strong>TransactionStructureStyle</strong></td>
<td>It is a functional perspective element. Business transactions have different structures such as flat or nested. This element was included in Solúbtha BTM to specify these structures.</td>
</tr>
<tr>
<td><strong>TransactionActivityClassifier</strong></td>
<td>It is a functional perspective element. It enables to specify the types of transaction. For instance, alternative activity is a type of activity.</td>
</tr>
<tr>
<td><strong>TransactionActivityCharacteriser</strong></td>
<td>It is a functional perspective element. This element was included to characterising the transaction activities. For instance, vitality is a characteristic of transaction activities.</td>
</tr>
<tr>
<td><strong>TransactionPerformanceHandler</strong></td>
<td>It is a functional perspective element. This element was included to specify the techniques that enables business transaction systems to perform consistently at runtime.</td>
</tr>
<tr>
<td><strong>TransactionEventHandler</strong></td>
<td>It is a functional perspective element. It is refined from EventHandler. It was included in the model to enable users to specify techniques for handling events at runtime such as fault occurred, activity failed etc..</td>
</tr>
<tr>
<td><strong>TransactionRole</strong></td>
<td>It is a functional perspective element. It is refined from the element Role. It was included to specify the roles involved directly or indirectly in business transactions.</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TransactionActivityHandler</strong></td>
<td>It is a functional perspective element. The element was included to classify and characterise transactional activities.</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>It is a business perspective element. It was adopted from Business-Aware Transaction Model (Papazoglou and Kratz, 2006). It is a business perspective element. Protocols are constraints that control business messages exchanged between the applications of involved trading partners.</td>
</tr>
<tr>
<td><strong>ProcessPerformance</strong></td>
<td>It is a functional perspective element. It enables specifying the metrics for measuring performance of process fragments.</td>
</tr>
<tr>
<td><strong>TransactionPerformance</strong></td>
<td>It enables specifying the metrics for measuring performance of transactions. For instance, transaction processing time is such a metric.</td>
</tr>
<tr>
<td><strong>ContingencyTransaction</strong></td>
<td>Contingency transactions are alternative process fragments or business functions or activities. This element enables specifying these alternative elements in business transactions. For example, a shipment process fragment Y is an alternative of shipment process fragment X.</td>
</tr>
<tr>
<td><strong>Business Transaction</strong></td>
<td>This is the central element of Solúbtha business transaction model. It is the root element of Solúbtha BTM. All design elements are encapsulated within it.</td>
</tr>
</tbody>
</table>

**UML Representation of the Initial Version of Solúbtha BTM**

The initial version of Solúbtha BTM consists of two packages: *functional perspective* and *business perspective*. Figure 2 (Page 287) shows the Initial Version of Solúbtha BTM. These packages contain the modeling elements. Central to these elements is BusinessTransaction composed of TransactionConstraints, TransactionEventHandler, TransactionWorkflowHandler, BusinessFunction, TransactionPerformanceHandler, Transactioncontexthandler, TransactionRole, BusinessProcess, TransactionFailureHandler, and TransactionActivityHandler.
TransactionConstraints is composed of two main business perspective elements: Compliance and Agreement. Agreement is composed of KeyPerformanceIndicator that contains two parts TransactionPerformance and ProcessPerformance. Compliance has four parts including OperationalConstraints Policy, StrategicConstraints, and ContractualConstraints. Policy is a set of business rules that controls executions of business transactions at runtime. The OperationalConstraints and StrategicConstraints are atomic elements. Operational constraints denote constraints that control executions of business transaction cohorts (sub-transactions). Strategic constraints are rules that represent organizational strategies which should be satisfied during business transactions. ContractualConstraints is composed of BusinessCollaboration that contains BusinessArea.

A business area is composed of an atomic element called ProcessAreas. A process area denotes a sub-process or activity. ProcessArea is the specialized element of BusinessArea. It inherits the properties of BusinessArea.

The Initial Version of Solúbtha BTM contains six functional perspective concepts that are atomic: TransactionPerformanceHandler, TransactionContexthandler, TransactionEventHandler, TransactionRole, TransactionStructureStyle and TransactionWorkFlowHandler. The remaining are compound elements. BusinessFunction is a compound element that is composed of ReferentialPrimitive, BusinessPrimitive, and BusinessObject. Papazoglou and Kratz define business object as a business document upon which business activities are operated. BusinessObject is used for specifying business documents that are exchanged between the business partners. BusinessPrimitives and ReferentialPrimitives are attributes used to describe business functions.

BusinessProcess is another compound element which is composed of ProcessFragment that contains Actors, Activity, CompensationHandler, EventHandler, and FaultHandler. The association relation has between Actors and Roles indicates that an actor can have different roles in a business transaction. The element TransactionRole is refined from Roles.
Figure 2: The Initial Version of Solúbtha Business Transaction Model.
There is a dependency relation between FaultHandler, EventHandler, and CompensationHandler and TransactionFailureHandler. The TransactionFailureHandler contain a functional perspective element called ContingencyTransaction.

The other compound element is TransactionActivityHandler composed of TransactionActivityClassifier and TransactionActivityCharacterizer. It is dependent on the element Activity.

It is important to note that flexible behavior was not the focus of the initial version of Solúbtha BTM however, the elements TransactionFailureHandler, EventHandler, FaultHandler, CompensationHandler, and ContingencyTransaction can be used to specify the properties of flexible business transactions. ContingencyTransaction enables specifying alternative sub-transactions and CompensationHandler enables specifying operations that undoes the effect of completed sub-transactions upon failure of a sub-transaction. FaultHandler is for specifying operations that deal with faults.

**Appendix E. Solúbtha BTM V1**

I constructed Solúbtha BTM V1 by addressing the opinions of the experts during the interim evaluation. Table 5 shows the operations that I carried out on the Initial Version of the model based on the outcomes of the interim evaluation.

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Event Handler</td>
<td>Merge</td>
</tr>
<tr>
<td>Business Collaboration</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction Activity Handler</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction Activity Classifier</td>
<td>Delete</td>
</tr>
<tr>
<td>Transaction Activity Characteriser</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction Performance</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction Workflow Activity Handler</td>
<td>Delete</td>
</tr>
<tr>
<td>Transaction Performance Handler</td>
<td>Merge</td>
</tr>
<tr>
<td>Strategic Constraints</td>
<td>Merge</td>
</tr>
<tr>
<td>Business Area</td>
<td>Merge</td>
</tr>
<tr>
<td>Process Area</td>
<td>Merge</td>
</tr>
</tbody>
</table>
Customization of the Initial Version of Solúbtha BTM

In this section, I describe the customization of the Initial Version of Solúbtha BTM. I describe the causes of adding, renaming, merging, and joining design-elements contained in Solúbtha BTM V1. I added the design-elements that I found through studying the secondary data of the Auto Inc. supply chain case.

Table 6 shows the list of elements added the model and the causes for adding the elements. The left hand side of the table shows the elements, the middle column shows the Operations, and the right hand side column shows the causes for adding them in the model.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Roles</td>
<td>Merge</td>
</tr>
<tr>
<td>Contractual Constraints</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction context handler</td>
<td>Delete</td>
</tr>
<tr>
<td>Business Function</td>
<td>Merge</td>
</tr>
<tr>
<td>Transaction Structure Style</td>
<td>Delete</td>
</tr>
<tr>
<td>Design-elements</td>
<td>Operation</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Constraints</td>
<td>Add</td>
</tr>
<tr>
<td>TemporalConstraint</td>
<td>Add</td>
</tr>
<tr>
<td>SpatialConstraint</td>
<td>Add</td>
</tr>
<tr>
<td>QoSConstraint</td>
<td>Add</td>
</tr>
<tr>
<td>LanguagePrimitive</td>
<td>Add</td>
</tr>
<tr>
<td>BusinessActivity</td>
<td>Add</td>
</tr>
</tbody>
</table>
As opposed to business activities, I found some activities that are non-decomposable. These activities are short-lived which means, the transaction completion time of these activities is much shorter than the business activities. Like business activities, this is another important component of business processes.

I found a number of elements in order fulfillment business process of the Auto Inc. supply chain are optional. They are called alternative set in this research. As already described in chapter 4, at runtime, from the alternative set, one one or more activity are chosen based on the flow pattern.

I proposed this element in the model. It is a specialization of process fragment. This will be used to specify the transactional properties of business processes.

Table 7 shows the list of elements that were renamed. Additionally, the table explains the causes of rename operation.

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Operation</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ProcessPerformance</em></td>
<td>Rename</td>
<td>This element was renamed to process performance metric. The term was borrowed from (Branimir et al., 2009) I found <em>process performance metric</em> is a standard term and understandable by the non-technical specialists such as process analysts.</td>
</tr>
<tr>
<td><em>Agreement</em></td>
<td>Rename</td>
<td>The purpose of this element is to specify the quality of services with business transaction models. Agreement is a generic element that has a broader scope. I renamed this concept to SLA which is a specialized term which is easily understandable to service provider and service client.</td>
</tr>
</tbody>
</table>

285
Protocol

The element Protocol was renamed to Business Protocol. I understood that term Protocol may confuse users because protocol refers to both technical protocol and business protocol. Since protocol in this research is meant business protocol, I used the term business protocol.

TransactionFailureHandler

I renamed the element Transaction Failure Handler to Exception Handler. The term was borrowed from BPEL (OASIS, 2007). The term exception can be more understandable to the developers, designer, and practitioners.

Several *merge* and *join* operations were performed. Table 8 explains the why design-elements were merged and joined.

**Table 8: The list of design-elements that were merged and joined and causes of these operations**

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Operation</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compliance</td>
<td>Merge</td>
<td>While studying secondary data of Auto Inc. case, I found that the company is concerned of the compliance rules that are essentially constraints. The compliance rules can be specified using the new design-element constraint. To simplify the model, I merged the element Compliance with the Constraint.</td>
</tr>
<tr>
<td>BusinessCollaboration</td>
<td>Merge</td>
<td>According to Papazoglou and Kratz (2006), SLA represents business collaborations which are established through negotiation. An SLA contains the obligations between them. I merged the element business collaboration into SLA.</td>
</tr>
<tr>
<td>EventHandler</td>
<td></td>
<td>I moved these concepts from the BusinessProcess and joined them with the ExceptionHandler. These concepts assist in preventing failures that are caused by faults. Therefore, logically they should be joined to exception handler.</td>
</tr>
<tr>
<td>CompensationHandler</td>
<td>Join</td>
<td></td>
</tr>
</tbody>
</table>
Description of the Newly Identified Design-elements of Solúbtha BTM V1

In this section, I describe the newly added design-elements of Solúbtha BTM V1. Table 9 presents the design-elements and descriptions.

Table 9: Description of design elements identified through studying secondary data of Auto Inc. Automotive Supply Chain Scenario.

<table>
<thead>
<tr>
<th>Design Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>Constraints are the conditions that should be satisfied while a business transaction is running. Constraints can be pre-conditions, post-conditions, rules, and so on.</td>
</tr>
<tr>
<td>SpatialConstraints</td>
<td>Spatial constraint denotes the location constraint which should be satisfied during the business transactions.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TemporalConstraints</strong></td>
<td>Temporal constraint denotes time related constraints which should be satisfied while a business transaction is running.</td>
</tr>
<tr>
<td><strong>QoSConstraints</strong></td>
<td>QoSConstraint refers to the obligations in terms of qualities of the services that should be provided to the customers.</td>
</tr>
<tr>
<td><strong>BusinessActivity</strong></td>
<td>BusinessActivity denotes long-running activities.</td>
</tr>
<tr>
<td><strong>Atomic Activity</strong></td>
<td>AtomicActivity denotes non-decomposable short-running activities.</td>
</tr>
<tr>
<td><strong>AlternativeSet</strong></td>
<td>AlternativeSet denotes a set of alternative activities associate with an activity. Alternative activities are invoked at runtime.</td>
</tr>
<tr>
<td><strong>TransactionProcessFragment</strong></td>
<td>TransactionProcessFragment refers to a functional space of a business transaction. It is a compound element composed of Language Primitives.</td>
</tr>
<tr>
<td><strong>LanguagePrimitive</strong></td>
<td>LanguagePrimitives are the operatives that characterize the transaction process fragments. SystemLevelPrimitive, ReferentialPrimitive, and ApplicationLevelPrimitive are three different types of language primitives. The ApplicationLevelPrimitive is specialized into OperationalPrimitive and ContractualPrimitive.</td>
</tr>
</tbody>
</table>

**UML representation of Solúbtha BTM V1**

Solúbtha BTM V1 is made up of eight different packages: Transaction, KPI, Service Level Agreement, Constraints, Business Protocol Definition, Business Process Definition, Activity Classification, and Language Primitive. Service Level Agreement, KPI, Constraint, and Business Protocol, are the business perspective packages and the remaining are the functional perspective packages. Central to these packages is the Transaction package. Figure 3 presents Solúbtha BTM V1.

Transaction package contains two design-elements BusinessTransaction and TransactionalProcessFragment. Service Level Agreement and Business Protocol definition packages contain the design-elements ServiceLevelAgreement and BusinessProtocol respectively. The elements KPI and ProcessPerformanceMetrics are contained in the KPI package. Activity classifier package contains five design elements: Activity, BusinessActivity, AtomicActivity, AlternativesSet, and BusinessObject.
The package Constraints contains TemporalConstraint, SpatialConstraint, OperationalConstraint, Policy, and QoSConstraint. The elements Roles, ProcessFragment, BusinessProcess, ExceptionHandler, CompensationHandler, EventHandler, and FaultHandler are contained in the Business Process Definition package. The Language Primitive package comprises six elements: LanguagePrimitive, SystemLevelPrimitive, ReferentialPrimitive, ApplicationLevel Primitive, ContractualPrimitive, and OperationalPrimitive. There are four system level primitives Resume, Cancel, Commit, and Retry and three operational primitives are Payment, Goods, and Delivery.

Business transactions make up the core of the transaction model, and may as such incorporate a combination of transactional and non-transactional process fragments that are coordinated through business protocols.
Figure 3: Solução Business Transaction Model V1.
A TransactionalProcessFragments is a compound design-element that contains LanguagePrimitives. The aggregation connector establishes a logical relation between these two. The cardinality 1..* indicates that a transaction process fragment contains at least one language primitive but may contain many.

The TransactionalProcessFragments embodies one or more activities that can be either of the two activity types: vital or non-vital. The association relation between TransactionalProcessFragments and Activity is specialized using a rolename contains which denotes that the activities are contained in the transaction process fragment. The multiplicity 1 to 1..* indicates that one transaction process fragment contains one to many activities. One of these activities must be vital - it is the logical restriction for the relationships between process fragments and activities. The model contains three types of activity BusinessActivity, AtomicActivity, and AlternativeSet. An activity can be any of these types but not all. The BusinessActivity is a compound design element consisting of one to many activities. The AlternativeSet is a compound design element containing a group of alternative atomic activities. The relationship between the alternatives set and atomic activity is constrained by a condition which is an alternatives set and can contain only non-vital atomic activities. Since which an alternative set will be executed at run-time is not pre-defined alternative activities cannot be vital.

Business activities operate on BusinessObject that is associated with the items such as purchase orders, catalogues (documents that describe products and service content to purchasing organizations), inventory reports, ship notices, bids and proposals. Such objects may associate the agreement and contracts or bids. This allows business transactions to interchange everything from product information and pricing proposals to financial and legal statements. The rolename ‘operates on’ of the association link between activities and business activities indicates that the activities operate on business objects.

Furthermore, the ExceptionHandler concept contains activities. It is composed of CompensationHandler, EventHandler, and FaultHandler. The logical relation between ExceptionHandler and Activity enables the transaction designer to specify a special type of activity that will be performed in case a running activity fails.

The SystemLevelPrimitives including resume, cancel, commit and retry characterize the transaction process fragments. The ReferentialPrimitives are language primitives that correlate an activity with other activities using control or data flow, for example, payment refers to an associated purchase order. ApplicationLevelPrimitive comprises ContractualPrimitives and OperationalPrimitives. The mutual agreements between the trading partners are specified using ContractualPrimitives. OperationalPrimitives is used to specify the conditions that enforce the business partner commitments at runtime. The operational primitives Payment, Goods, and Delivery enable specifying the obligations related to payment, goods, and delivery transactions.

An ServiceLevelAgreement is an aggregation of a coherent set of explicitly stated policies and KPIs.
Policy is a compound element that is an aggregation of one or more Constraints that has four specialized design-elements SpatialConstraints, TemporalConstraints, OperationalConstraints, and QoSConstraints. These elements enable specifying constraints such as location, time, sequence of operations etc. Furthermore, policies are set of BusinessRules. Another design-element contained in ServiceLevelAgreement is KPI which enables specifying the ProcessPerformanceMetrics.

The relationship between BusinessProtocol and ServiceLevelAgreement is refined in this model using a rolename called Governs. The BusinessProtocol is used for specifying the sequence of a process fragment. While a business transaction is running, the SLA governs the sequences of transaction process fragments.

The business protocol is logically connected with process fragments which are composed into the business transaction. This logical connection results in a correlation between the business perspective elements and the functional perspective elements.

**Appendix F. Customization of Solúbtha BTM V1**

Several operations were performed with the design-elements of Solúbtha BTM V1. This section describes the customization of Solúbtha BTM V1, which resulted in the final version Solúbtha BTM. I explain the reasons for performing the operations. Table 10 below shows a list of design-elements added and it also shows the causes for adding them in the model. The leftmost column shows the name of design-elements, the middle one shows the operations performed, and the rightmost column shows the causes for performing the operations.

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Operation</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessPolicy</td>
<td>Add</td>
<td>While studying the trading partner agreements (I discussed the agree-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ments in Chapter 4 ), I found that agreements contain paragraphs called</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compliance and confidentiality that contain clauses. Therefore, I ad-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ded two design-elements called CompliancePolicy and SecurityPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in Solúbtha BTM to enable users to specify the corresponding clauses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Agreements contain other clauses as well. I added an element Business-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Policy which enables users to specify the clauses other than compli-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ance and confidential.</td>
</tr>
<tr>
<td>CompliancePolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityPolicy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Design-elements and cause of actions
RecoveryPolicy | Add | For guaranteeing flexibility, I proposed policy based recovery in Solúbtha BTM. I added this design-element to enable users to specify the failure recovery statements or conditions.

Control Flow Pattern | Add | I identified nine flow patterns through studying the purchase order management process of the AVERS OEM. The Control-FlowPattern was added to enable users to specify the require flow patterns in their business transaction models.

Some elements of Solúbtha BTM V1 were renamed. Table 11 shows the list of renamed elements and explains the causes for carrying out the operation.

**Table 11: Design-elements and causes of actions**

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Operation</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExceptionHandler</td>
<td>Rename</td>
<td>This design-element was renamed to TransactionRecoveryHandler. Since exceptions in business transactions are failures and exception handlers deal with the failures, I found TransactionRecoveryHandler is more appropriate.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Rename</td>
<td>I found that the design-element ‘Constraints’ is semantically and functionally similar to the element Policy. However, ‘Policy’ is a more business-oriented term. Since ‘Constraints’ is a business perspective design-element, I renamed ‘Constraints’ to ‘Policy’. Additionally, the parts of ‘Constraints’, QoSConstraints, SpatialConstraints, and TemporalConstraints are abstracted to the design-element ‘BusinessPolicy’.</td>
</tr>
</tbody>
</table>
During my study of secondary data of business cases, I found that the ServiceLevelAgreement is a specialized element that was added to enable specifying service level agreements that contain obligations related to the services provided by the service providers. However, since service level agreement specifically focuses on service related businesses, it may not be possible to define other business related obligations. Also, I found that agreement is a generic concept and it is used in the real-world business scenario. Therefore, I renamed the element ServiceLevelAgreement to ‘Agreement’.

Role is a generic concept. A business partner in a business transaction plays one or more roles. For example, in a purchase order transaction, the seller may play a role of seller as well as shipper. I mapped this concept to the context of business transactions and renamed it to TransactionRole which not only increase the specificity but also fits to the functional aspect of business transactions.

In Addition, some elements were merged, joined, and deleted. Table 12 shows the list of design-elements and causes of merging, joining, and deleting operations.
<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Operation</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>FaultHandler</td>
<td></td>
<td>These elements were merged into TransactionRecoveryHandler which enables specifying techniques for fault recovery, event handling, and invoking compensation function on occurrence of failures. I merged these three elements into one to simplify Solúbtha BTM.</td>
</tr>
<tr>
<td>EventHandler</td>
<td>Merge</td>
<td>By analysing the secondary data of the trading partner agreements I found that the constraints: QoS constraints, spatial constraints, and temporal constraints are essentially business related clauses in agreements. I added the design-element BusinessPolicy that enables specifying these clauses. Thus, the design-elements including QoS Constraint, Spatial Constraint, and Temporal Constraint are not needed for the model. Therefore, I merged these different types of constraints into ‘BusinessPolicy’.</td>
</tr>
<tr>
<td>CompensationHandler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QoSConstraints</td>
<td>Merge</td>
<td></td>
</tr>
<tr>
<td>SpatialConstraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TemporalConstraint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusinessActivity</td>
<td></td>
<td>The study of the secondary data of Auto Inc. and AVERS OEM business processes unfolded some important aspects. I discovered that no specialized element is required to specify business activities and atomic activities. The logical relation composition is sufficient to define business activities. Therefore, to simplify the model I merged business activities and atomic activities into single element called Activity. Additionally, ‘AlternativeSet’ is another design-element not needed for designing alternative activities in a business transaction because, ‘Activity’ enables users to specify any type of activities. In practice, ‘AlternativeSet’ can be specified using flow pattern, no specialized element is required. Therefore, I merged this element into ‘Activity’.</td>
</tr>
<tr>
<td>AtomicActivity</td>
<td>Merge</td>
<td></td>
</tr>
</tbody>
</table>
This element was deleted along with its parts SystemLevelPrimitive, ContractualPrimitive ReferentialPrimitive, and OperationalPrimitive. I replaced ‘LanguagePrimitive’ with a set of operators.

Resume was a part of the LanguagePrimitive. Since this element was deleted, Resume was joined to ‘TransactionRecoveryHandler’ where it fits the best because resume is recovery operation.

This element is used for measuring the performance of a business. However, this concept was not found in the ‘Agreements’ which I investigated in this research. Therefore, I eliminated this concept from the model to avoid unnecessary complexity.

The element ‘TransactionProcessFragment’ enables users to specify process fragments in a business transaction model. No additional element in particular ‘Process’ is required. Therefore, I deleted this element from Solúbtha BTM.

**Appendix G. Description of the Newly Identified Design-elements of Solúbtha BTM**

In this section, I describe the newly added design elements of Solúbtha BTM V1. Table 13 describes these elements.

**Table 13: Description of the Newly Added Design-elements**

<table>
<thead>
<tr>
<th>Design-elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompliancePolicy</td>
<td>It refers to a set of compliance rules that must be satisfied during business transactions. This element enables to specify the compliance rules in business transactions.</td>
</tr>
<tr>
<td>SecurityPolicy</td>
<td>Security Policy is a set of security constraints. The security constraints can be specified in business transactions using this element.</td>
</tr>
<tr>
<td><strong>BusinessPolicy</strong></td>
<td>It is composed of business rules which should be satisfied while a business transaction is running. It enables specifying business clauses as rules which are contained in agreements.</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RecoveryPolicy</strong></td>
<td>It refers to the logical conditions that are essentially the recovery properties of business transactions.</td>
</tr>
<tr>
<td><strong>ControlFlowPattern</strong></td>
<td>It refers to the execution paths of the activities and the transaction process fragments which compose the end-to-end business transactions using the control flow patterns.</td>
</tr>
</tbody>
</table>

## Appendix H. Automata, Petri Net, and Process Algebra

### I. Automata:

Automata theory is a very useful and one of the earliest mathematical techniques for modeling system behavior (Baier and Katoen, 2008). It is a well-known model underlying formal system specification describes the system behavior. An automation (aka finite automation and finite state machine) is an abstract model of a digital computer that has the mechanism to read input that is a string over a given alphabet (Xavier, 2005). There are two types of automation: Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata (NFA). The main difference between these two different types of automations is the execution patterns. The deterministic finite automata have one exiting transition path vector (directed edge) which means the execution pattern is purely sequential whereas the non-deterministic finite automata may have many exiting paths, therefore, the execution sequence in NFA automation can be parallel. The concurrency is the major advantage of NFA. The NFA allows defining the parallel state machines for a set of inputs, which is an essential feature for designing parallel business transaction state machines.

The variants of automata include I/O automata (Kaynar, 2006), timed automata (Alur and Dill, 1994), and teamed automata (Beek *et al.*, 2003). The TEFSM (Lallali *et al.*, 2007) is another variant of automata.

There is a list of model checking tools that support verifying the automata include SPIN model checker (Holzman, 2003) and UPPAAL (Behrmann G., 2004). These model checking tools support defining the properties that are defined using formal language such as Linear Temporal Logic (LTL) and Timed Computational Tree Logic (TCTL) (Baier and Koetln, 2008).

### II. Petri net:

297
It is a widely used formalism for modeling concurrent systems. A Petri net can be thought of as a transition system where instead of a transition occurring from a single global state, an occurrence of an event imagined to affect only the conditions in its neighborhood (Winskel and Neilsen, 1993). Petri nets are graphical and mathematical modeling tool to many systems such as describing and studying information processing systems that are concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic (Murata, 1989). They can be used as a graphical tool similar to block diagram and flowchart. However, Petri nets provide a set of powerful notations to define mathematical equations such as state equations. This is why Petri net formalism enables identifying the basic concepts of concurrent systems both conceptually and mathematically. There is a tool support called BPEL2PN is a tool that transforms the BPEL processes into the Petri Net model.

III. Process Algebra:

The term process algebra encompasses a collection of theories that support mathematically rigorous (in)equational reasoning about systems consisting of concurrent, interacting process (Cleaveland and Smolka, 1999). This formalism has three main characteristics (Cleaveland and Smolka, 1999):

- A language, or algebra, is defined for describing systems.
- A behavioural equivalence is introduced that is intended to relate systems whose behaviour is indistinguishable to an external behaviour.
- Equational rules, or axioms, are developed, which permit proofs of equivalence between systems to be conducted in a syntax-driven manner.

It is a well-known formalism for automatic verification of the functional and non-functional properties of a system. In addition, the process algebra is an efficient formalism to compare the behavior of two different objects. The \( \mu \)-calculus is a well-known process algebra.

There is a tool called BPEL2EC (Rouached, 2007) for automatic verification of the process algebra models. The CADP (Fernandez et al., 1996) is another toolbox that supports automatic verification of the model based on process algebra.
List of Acronyms

**ACID**  Atomicity Consistency Isolation Durability

**ASSET**  A System for Supporting Extended Transaction

**BAT**  Business Aware Transaction

**B2B**  Business-to-Business

**BPEL**  Business Process Execution Language

**BPMN**  Business Process Modeling Notation

**BPSS**  Business Process Specification Schema

**BT**  Business Transaction

**BTM**  Business Transaction Model

**BTP**  Business Transaction Protocol

**BTTP**  Business Transactions Technical Committee

**CKD**  Completely Knocked-Down

**CTL**  Computational Tree Logic

**DFA**  Deterministic Finite Automata

**ECA**  Event-Condition-Action

**FCL**  Formal Contract Language

**HLS**  High Level Service

**IT**  Information Technology

**JAS**  Java Activity Service

**KPI**  Key Performance Indicator

**LTL**  Linear Temporal Logic
LRA  Long Running Activity
LTA  Long Term Agreement
NFA  Non Deterministic Finite Automata
OCL  Object Constraint Language
OEM  Original Equipment Manufacturer
OMG  Object Management Group
QoS  Quality of Service
SBA  Service Based Application
SCOR Supply Chain Operation Reference
SKU  Stock-Keeping Unit
SLA  Service Level Agreement
SOA  Service Oriented Architecture
SOC  Service Oriented Computing
STA  Short Term Agreement
TEFSM Time Extended Finite State Machine
TPF  Transaction Process Fragment
TMLA Transaction Model for Long-running Activities
Tx-QoS Transactional Quality of Services
UML  Unified Modeling Language
UMM  UN/CEFACT Modeling Methodology
UoW  Unit of Work
W3C  World Wide Web Consortium
WS  Web Service
WS-AT  Web Service Atomic Transaction
WS-AT  Web Service Business Activity
**WS-CAF**  Web-Service Composite Application Framework

**WS-CF**  Web-Service Coordination Framework

**WS-TX**  Web-Service Transaction

**WS-TXM**  Web-Service Transaction Management

**XML**  eXtensible Mark-up Language
Glossary

**Abortion**  Abortion is a state of business transaction which indicates the transaction has been terminated.

**Abort**  Abort is a system level operator that will facilitate specifying the abort protocol for transaction process fragments contained in a business transaction.

**Application Related Fault**  The application related faults happen in software applications. This type of faults is well understood by now as these fault were studied extensively and reported in literature.

**Atomicity**  Atomicity is a property that guarantees the successful execution of all (sub-) transactions (aka Cohort) that are contained in a transaction scope. The atomicity property does not allow the failure of any transaction within the transaction scope because it relies on the all or nothing principle that means, either executing all transactions or none.

**Atomic Activity**  Atomic activities are a type of activity in business transactions. They are short-lived which means the transaction completion time of these activities is much shorter than the business activities.

**Atomic Behavior**  Atomic behavior is a type of behavior that refers to the strict behavior of business transaction.

**Axiology**  Axiology is a branch of philosophy that studies judgments about the value.

**Business Activity**  Business Activities are types of activities that are composite and long-running activities.

**Business Area**  Business area is a business perspective design element which denotes the specific business processes that are scoped in a business transaction.

**Business Collaboration**  Business collaboration is a business perspective design element. It specifies all the business messages that are exchanged between two trading partners, their content, and their precise sequence and timing.

**Business Function**  Business function is defined as a description of the well-defined and commonly acceptable critical business principle e.g. delivery of goods that transforms business values and causes state changes to transaction participants e.g., transform an unpaid order to paid order.
**Business Object** A business object is a business document upon which the business activities are operated.

**Business Process** A business process is a sequence of activities. A business process contains process fragment that contains a list of design elements that include activity, actors, roles, and event.

**Business Protocol** Business protocols sequence transaction activities. They define the execution behavior of business transactions.

**Business Transaction** A business transaction is a series of transaction fragments that are distributed across a network of business partners and performed collaboratively in a flexible manner by accomplishing the commitments agreed upon by the partners.

**Business Related Fault** Faults that are related business entities such as agreement, policies etc. Examples of business related faults include agreement violation, policy violation.

**Business Protocol Related Fault** Faults that are related to business protocol. An example of business protocol related faults is business function execution sequence fault.

**Business Object Related Fault** Faults that are related to business objects. Incorrect data and missing data are business object related faults.

**Commit** Commit is a system level operator that will facilitate designing the commit protocol for transaction processes fragments. This operator can also be used to define the global commit which means the commit operator can be used to specify the conditions to commit all transaction process fragments.

**Compensation** Compensation is a failure recovery operator. It undoes the effect of successfully completed activities from the semantic point of view in a reverse order.

**Compliance Policy** Compliance policy is a set of compliance rules that must be satisfied during the business transactions. This concept is offered to specify the compliance rules in business transaction.

**Constraints** Constraints are the conditions that should be satisfied while a business transaction is running. Constraints can be preconditions, post-conditions, rules, and so on. This element will facilitate specifying various conditions for guaranteeing the operational consistency or integrity of business transactions.

**Contingency Transaction** Contingency transaction is a function perspective design element. It refers to alternative activities.

**Consistency** Consistency is a property of transactions. It refers to the correctness of the state of the database that a committed transaction produces.
Concurrency  Concurrency denotes simultaneous occurrence (transaction) of multiple processes, functions, or threads. Unlike a sequential system, a concurrent system carries out a collection of processes simultaneously on multiple processors or multi-cored single processor.

Durability  Durability is a transaction property that ensures data persistency.

Epistemology  Epistemology is concerned with the nature and forms of knowledge. It is concerned with how knowledge can be produced, acquired, and communicated.

eBusiness  eBusiness stands for Electronic Business which is a form of business where business activities are performed electronically.

Failure Resilience  Failure-resilience refers to the ability of a business transaction to recover from failures. There are faults which cannot be tolerated rather they lead to business transaction abortion. If such a fault occurs and causes business transaction failure, the business transaction system must be able to recover the business transaction from the failure to avoid abortion.

Fault Tolerance  Fault-tolerance refers to the ability of a business transaction system to continue a business transaction in the event of failures of one or more transactional activities.

Flexible Business Transaction  A business transaction is flexible if it is able to tolerate faults, it is failure-resilient, the business transaction is non-deterministic, the cohorts of business transaction can be performed autonomously by the applications hosted at different sites of different trading partners, and the business transaction is extensible.

Isolation  Isolation refers to a situation where a program running under transaction protection must behave exactly as it would in single-user mode; it is not an entirely independent requirement; rather it must be achieved to guarantee consistent input data, which is a prerequisite for consistent output.

Ignore  It is a operator to specify the logical conditions for ignoring a fault, this operator guarantees the fault tolerant business transactions.

KPI  Key performance indicator (KPI) denotes the factors by reference to which the development, performance or position of the business of the companies can be measured effectively.

Language Primitive  Language primitive is a function perspective design element that characterizes the transaction process fragments.

Message Related Fault  Faults that are related to messages exchanged between business partners during business transactions.
Mutualize Mutualize is a failure recovery operator which is used to mutualize an agreement between business partners.

Objectivism Objectivism is an ontological position which represents the objective reality that explains what really exist.

Ontology Ontology is a research philosophy which deals with the existence of a reality (e.g. an object), the classification of reality, and properties of a reality

Party Party is the business perspective design element of Solúbtha BTM which denotes the business partner engaged in business collaboration.

Period Period is a business perspective design element of Solúbtha BTM which is a part of another concept Agreement. It denotes a valid time interval of agreement.

Process Performance Process performance is a business perspective design element that is used to specify the performance of business processes.

Postpone Postpone is a recovery operator that is used to postpone a transaction activity.

Positivism Positivism is an epistemological position. It is a position that holds the goal of knowledge is simply to describe the phenomena that are experienced.

Quality of Service Quality of Service is a business perspective design element which is used to specify the quality attributes of a service such as software service which is a part of business transaction.

Recovery Policy The testing of a hardware component in isolation, equivalent to a unit test in software terminology

Referential Primitive Referential primitive is a functional perspective design element. Referential primitives are language primitives that correlate an activity with other activities using control or data flow, for example, payment refers to an associated purchase order.

Relaxed Atomic Behavior Relaxed atomic behavior is essentially the partial flexible behavior. The purpose of relaxing atomicity is preventing the abortion of business transaction.

Repair Repair is a recovery operator that is used to repair a business transaction fault at runtime.

Resume Resume is a recovery operator that is used to restart a business transaction.

Retry Retry is a is a recovery operator that is used to retry an operation after it was failed.

SBA An SBA is obtained by composing various services in order to satisfy the desired functionalities. SBAs orchestrate the services (also known as web services) that are essentially software components
containing the methods that perform the operations and produce outcome(s). The development of SBAs relies on the service oriented computing (SOC) paradigm.

**Security Policy** A Security Policy is a business perspective element. It is a set of security constraints.

**SLA** SLA refers to an agreement between a service provider and service client. It contains promises and obligations agreed between a service provider and client.

**SOA** Service oriented architecture is an architectural paradigm for organizing the software components contained in the SBAs.

**SOC** Service oriented computing is a computing paradigm. SOC utilizes services as the fundamental elements for developing SBAs.

**Spatial Constraints** Spatial constraint is a business perspective design element. It denotes the location constraint which should be satisfied during the business transactions. This element assists transaction designers in specifying the location constraints in particular, the delivery location of ordered goods.

**Substitution** Substitution is a recovery operator. It is used to replace an activity upon failure of that activity.

**Suspend** Suspend is a recovery operator that is used to suspend a running activity.

**Temporal Constraint** Temporal constraint is a business perspective element. It is a time related condition to restrict operations.

**Transaction Context Handler** Transaction context handler is a functional perspective design element for specifying the scope of the business transaction. A business transaction scope denotes the boundary of a business transaction.

**Transaction Event Handler** Transaction event handler is a functional perspective design element. It is used for specifying the techniques for handling events at runtime such as fault occurred, activity failed etc.

**Transaction Failure Handler** Transaction failure handler is a compound design-element. It enables specifying the transaction failure prevention techniques using compensation handler, fault handler, contingency transaction, and event handler.

**Transaction Process Fragment** Transaction process fragment is a variant of process fragment. It refers to a functional area of a business transaction. A payment process fragment is an example of transaction process fragment. This concept facilitates specifying the transaction process fragments within the scope of business transactions. Transaction process fragment is a compound element composed of Language Primitives.
**Transaction Role**  Transaction Role is a business perspective design element. It denotes an active participant of business transaction.

**Transaction Structure Style**  Business transaction has a structure which is either flat or nested. Transaction Structure Style is a business perspective element for specifying the mode of business transaction structure such as flat or nested.

**Transaction Workflow Handler**  A business transaction scope includes a set of sub-transactions that should be sequenced correctly to produce outcome expected by a user. Transaction Workflow Handler is a business perspective element. Using this element a transaction developer will be able to specify the sequence of these sub-transactions. It will enable a transaction manager to modify the sequence (if required) at runtime.

**Web Service**  Web Service is a service such as software service available on the web.

**Wait**  Wait is a recovery operator. It is used for making an activity wait until the previous activity is finished.
Publications


