Managing process design in a dynamic organisational context

Mohammad Moshiur Rahman Bhuiyan
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Managing Process Design in a Dynamic Organisational Context

A thesis submitted in fulfillment of the requirements for the award of the degree

Doctor of Philosophy (PhD)

from

UNIVERSITY OF WOLLONGONG

by

Mohammad Moshiur Rahman Bhuiyan

School of Computer Science & Software Engineering
May 2012
© Copyright 2012

by

Mohammad Moshiur Rahman Bhuiyan

All Rights Reserved
Dedicated to

My wife Luba,
My daughter Eleena,
My parents and
grandparents.
Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Mohammad Moshiur Rahman Bhuiyan
October 18, 2012
This dissertation proposes a methodology to manage business process design in a dynamic organizational context primarily by using $i^*$ modeling framework and Business Process Modeling Notation (BPMN). Agent-Oriented Conceptual Modeling (AOCM) notations such as the $i^*$ framework have gained considerable currency in the recent past. Such notations model organisational context and offer high-level social/anthropomorphic abstractions (such as goals, tasks, softgoals and dependencies) as modeling constructs. It has been argued that such notations help answer questions such as what goals exist, how key actors depend on each other and what alternatives must be considered. The technical focus of process modeling notations such as BPMN is especially suited for applications in the description, execution and simulation of business processes but is lacking in support for process redesign and improvement. These notations effectively provide a view of the responsibilities and required communications between classes of process participants, but do not provide a view of other social and intentional characteristics including the goals of participants and their inter-dependencies. We argue this gap can be minimised by using and correlating organisational models and process models in a complimentary fashion.

Business processes represent the operational capabilities of an organisation. In order to ensure process continuity, the effective management of risk becomes an area of key concern. We argue there is a need for supporting risk identification with the use of higher-level organisational models and business process models.

In this research we have conducted a detailed analysis of the concept that flexibility and combination of notations are required to facilitate the maintenance of the models. We have developed a methodology to support combined use of notations ($i^*$ and BPMN) for modeling business processes with a view to facilitate and support change at organisational and process models. We have also presented a methodology to integrate risks in process models through a set of intuitive metrics by extracting measures of actor criticality, and vulnerability from organisational models. This research has been validated through a detailed case study involving a major government agency and through an experiment conducted among participants from industry and academia.
First and foremost I offer my sincerest gratitude to my supervisor Professor Aditya K Ghose who has maintained his interest in my work and has supported me throughout my doctoral research with his knowledge and patience. Being a PhD candidate and a full time IT Consultant at the same time is quite challenging; I do thank my supervisor for allowing me the room to work in my own way. I am also thankful to my co-supervisor Dr. Aneesh Krishna for his advice in different phases of my work. I would like to thank my colleagues in Decision Systems Laboratory (DSL) especially George Koliadis and Zahidul Islam for their valuable comments, support, help and encouragement during the process of completing this thesis as well as during the period of my doctoral study. My parents deserve a special mention for their support and well wishes. I would not have considered doing a doctoral degree if my father Zahirul Haque Bhuiyan and my mother Hanufa Haque did not insist me to go for it. In any hurdles of my life they have always encouraged me, given me strength to proceed; words fail me to express my appreciation to my parents. My brothers Sabuj and Ujjal, and my one and only sister Soniya have always been proud of me, which has consistently acted as a major influential factor.

Special thanks go to my grandfather Bazlul Haque Bhuiyan who loves me the most among all his grandchildren. I would like to thank my uncles and aunties for their support and well wishes.

Finally I would like to thank my wife Luba Shabnam, without whom I could not have finished writing the thesis. She encouraged me, comforted me, set aside time for me when I needed the most. She did not mind when I stayed at home doing nothing but writing up papers and thesis days after days, weekends after weekends being an unsocial human being. Our daughter Eleena Shabnam too did her very best to keep me calm, many thanks to her.
List of Publications

A list of referred papers related to this work.


# Table of Contents

Abstract ................................................................. v
Acknowledgements ..................................................... vi
List of Publications .................................................... vii
Table of Contents ....................................................... ix

1 Introduction ........................................................... 1
  1.1 Research Problem and Motivation .............................. 1
  1.2 Research Aim and Objectives .................................. 5
  1.3 Research Scope and Strategy ................................... 5
  1.4 Research Contributions ......................................... 8
  1.5 Thesis Organisation ............................................. 9

2 Background ........................................................... 10
  2.1 Agent-Oriented Methodologies ................................. 10
    2.1.1 i* framework .............................................. 11
    2.1.2 Tropos ..................................................... 17
    2.1.3 Formal Tropos ............................................ 20
    2.1.4 AOR ......................................................... 24
    2.1.5 Prometheus ............................................... 26
    2.1.6 Gaia ......................................................... 27
    2.1.7 Comparison ................................................ 29
  2.2 Business Process Management ................................. 29
    2.2.1 Business Process Management Lifecycle ................. 31
    2.2.2 Business Process Modeling ............................... 34
    2.2.3 Business Process Modeling with BPMN .................. 38
    2.2.4 Business Process Modeling with UML Activity Diagram 40
  2.3 Risk and Risk Management ..................................... 43
  2.4 Non-functional Requirements .................................. 49
  2.5 Service-Oriented Architecture (SOA) ......................... 52
  2.6 Summary ........................................................ 54
3 Research Design 55
3.1 Research Methodology ........................................... 55
3.2 Feedback from Industry and Researchers: ....................... 60
3.3 Summary .......................................................... 63

4 A Combined Approach for Supporting the Business Process Model Lifecycle 64
4.1 Introduction ......................................................... 65
4.2 Background ......................................................... 66
  4.2.1 Agent-Oriented Conceptual Modeling (AOCM) with \( i^* \) .... 67
  4.2.2 Business Process Modeling with BPMN ..................... 70
4.3 Constrained Development Methodologies .................... 72
  4.3.1 Annotation and Propagation ..................................... 73
  4.3.2 Scope Projection .................................................. 75
  4.3.3 Consistency Evaluation ......................................... 75
  4.3.4 Constrained Development of a Business Process Model given a High-Level Conceptual Model .... 76
4.4 Constrained Development of a High-level Conceptual Model given a Business Process Model .... 80
  4.4.1 Reflecting Changes in a BPMN Model to an associated \( i^* \) Model .... 81
4.5 Summary .......................................................... 83

5 Correlating Business Process and Organisational Models to Manage Change 84
5.1 Introduction ......................................................... 85
  5.1.1 Organisational Modeling (with \( i^* \)) ....................... 88
  5.1.2 Business Process Modeling (with BPMN) .................. 89
5.2 Using Model Annotations To Assess Relationships between Business Process and Organisational Models .......................... 91
  5.2.1 Effect Annotation ................................................. 91
  5.2.2 Effect Accumulation ............................................. 93
  5.2.3 Fulfillment Conditions .......................................... 97
  5.2.4 Establishing Realization Relationships between Elements of both Organisation and Business Process Models .......... 98
  5.2.5 Assessing Realization between Organisational and Business Process Models ............................................. 100
5.3 Managing Process Portfolios with Architectural Models of an Organisation102
  5.3.1 Assessing the Impact of Process Change with Reference to an Organisational Model .... 104
5.4 Summary .......................................................... 106

6 Integration of Agent-Oriented Conceptual Models and UML Activity Diagrams Using Effect Annotations 108
6.1 Introduction ......................................................... 109
6.2 The $i^*$ Modeling Framework ............................................ 110
   6.2.1 Training System Strategic Dependency Model .................. 110
   6.2.2 Training System Strategic Rationale Model .................... 112
6.3 UML Activity Diagram .................................................... 115
6.4 Benefits of Mapping $i^*$ Model into UML Activity Diagram ...... 115
6.5 Methodology Supporting the Integration of $i^*$ and UML Activity Diagram .......................................................... 117
   6.5.1 Consistency Evaluation .............................................. 118
   6.5.2 Phase 1: Effect Annotations ....................................... 119
   6.5.3 Phase-2: Mapping Rules ............................................ 125
6.6 Reflecting Changes in an $i^*$ Model to an associated Activity Model .......................................................... 130
6.7 Summary .............................................................................. 131

7 Managing Business Process Risk Using Rich Organisational Models 132
   7.1 Introduction ...................................................................... 133
   7.2 Agent Oriented Conceptual Modeling ................................ 134
   7.3 Business Process Modeling with BPMN ................................ 137
   7.4 Identifying Risk within Organisational Models .................. 139
   7.4.1 Vulnerability ............................................................. 139
   7.4.2 Criticality ............................................................... 143
   7.5 Integrating Risks in Business Process Models .................... 146
   7.5.1 Treating Vulnerable Actors ......................................... 146
   7.5.2 Treating Critical Actors ............................................. 148
   7.6 Summary .............................................................................. 151

8 Risk Measure Propagation through Organisational Network 152
   8.1 Background ...................................................................... 153
   8.2 Risk Measurements at Organisational and Process Models .... 154
   8.2.1 Vulnerability ............................................................. 155
   8.2.2 Criticality ............................................................... 156
   8.3 Risk Measure Propagation Over Organisational Model ......... 158
   8.3.1 Vulnerability over Organisational Network .................... 159
   8.3.2 Criticality over Organisational Network ....................... 162
   8.4 Addressing Risk ............................................................ 166
   8.5 Summary .............................................................................. 169

9 Supporting Business Modeling using Multiple Notations: A Case Study 170
   9.1 Introduction ...................................................................... 170
   9.2 Project Background ........................................................... 171
   9.3 Business Modeling Strategy ............................................. 172
   9.4 Modeling Approach And Methodology Used ....................... 177
   9.5 Discussion and Lessons Learnt ......................................... 184
   9.6 Conclusion ........................................................................ 187
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison among Agent Methodologies</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Annotation of Fullfillment Conditions to Respective Tasks/Dependancies</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>Associated Tasks and Effects for the Bond Package Process in Figure 5.9</td>
<td>104</td>
</tr>
<tr>
<td>6.1</td>
<td>Annotation of Tasks with Effects</td>
<td>121</td>
</tr>
<tr>
<td>6.2</td>
<td>Annotation of Dependencies with Fulfilment Conditions</td>
<td>122</td>
</tr>
<tr>
<td>6.3</td>
<td>Tasks that Realizes/ Requires the Fulfillment Conditions</td>
<td>123</td>
</tr>
<tr>
<td>6.4</td>
<td>Propagation of Fulfillment Conditions to Respective Tasks</td>
<td>124</td>
</tr>
<tr>
<td>7.1</td>
<td>Vulnerability Measurement of Actors at Organisational Model</td>
<td>141</td>
</tr>
<tr>
<td>7.2</td>
<td>Vulnerability Measurement at Process Level</td>
<td>142</td>
</tr>
<tr>
<td>7.3</td>
<td>Criticality Measurement of Actors at organisational Model</td>
<td>144</td>
</tr>
<tr>
<td>7.4</td>
<td>Criticality Measurement at Process Level</td>
<td>145</td>
</tr>
<tr>
<td>8.1</td>
<td>Vulnerability Measurement of Actors at Organisational Model</td>
<td>155</td>
</tr>
<tr>
<td>8.2</td>
<td>Vulnerability Measurement at Process Level</td>
<td>156</td>
</tr>
<tr>
<td>8.3</td>
<td>Criticality Measurement of Actors at organisational Model</td>
<td>157</td>
</tr>
<tr>
<td>8.4</td>
<td>Criticality Measurement at Process Level</td>
<td>158</td>
</tr>
<tr>
<td>8.5</td>
<td>Vulnerability Measurement at Organisational Network Level</td>
<td>162</td>
</tr>
<tr>
<td>8.6</td>
<td>Criticality Measurement at Organisational Network Level</td>
<td>166</td>
</tr>
<tr>
<td>8.7</td>
<td>Vulnerability Measurement of Actors at organisational Model</td>
<td>168</td>
</tr>
<tr>
<td>9.1</td>
<td>Comparison among Different Modeling Notations</td>
<td>176</td>
</tr>
<tr>
<td>9.2</td>
<td>Check List A</td>
<td>182</td>
</tr>
<tr>
<td>9.3</td>
<td>Check List B</td>
<td>183</td>
</tr>
<tr>
<td>10.1</td>
<td>Annotation of Fulfillment Conditions to Respective Tasks/Dependancies</td>
<td>191</td>
</tr>
<tr>
<td>10.2</td>
<td>Annotation of Effects to Respective Tasks/Dependencies</td>
<td>196</td>
</tr>
</tbody>
</table>
List of Figures

2.1 The Strategic Dependency Model of the Meeting Scheduling System . . 13
2.2 The Strategic Rationale Model of the Meeting Scheduling System . . 15
2.3 Business Process Life Cycle . . . . . . . . . . . . . . . . . . . . . . . . 32
2.4 A classified illustration of some business process modeling notations . . 37
2.5 Example of a Collaborative Patient Treatment Process in BPMN . . . 40
2.6 Example of a UML Activity Diagram . . . . . . . . . . . . . . . . . . . 41
2.7 Risk Management Phases . . . . . . . . . . . . . . . . . . . . . . . . . 44
3.1 Research Phases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
4.1 An i* Strategic Rationale (SR) Meeting Scheduling Model with a Rout-
tine Illustrated . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
4.2 A BPMN ‘Patient Treatment Process Model . . . . . . . . . . . . . . 71
4.3 BPMN Process Model derived using the Constrained Development Method-
ology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
4.4 An i* Patient Treatment Process . . . . . . . . . . . . . . . . . . . . . . 82
5.1 The ‘Business Process Lifecycle (BPLC) . . . . . . . . . . . . . . . . . 85
5.2 A simple i* model of a Transport organisation . . . . . . . . . . . . . 89
5.3 A BPMN ‘Package Routing ’process model . . . . . . . . . . . . . . 90
5.4 Pair-wise accumulation of two tasks in contiguous sequence . . . . . 94
5.5 Pair-wise accumulation to a task during an AND-join . . . . . . . . . 95
5.6 Pair-wise accumulation to a task during an XOR-join . . . . . . . . . 96
5.7 Pair-wise accumulation to a task during an OR-join . . . . . . . . . . 97
5.8 Enterprise modeling Framework . . . . . . . . . . . . . . . . . . . . . . 103
5.9 Assessing the impact of process change in the Package Routing example 105
6.1 A Strategic Dependency Model for Computer Based Training (CBT)
System . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Strategic Rationale model for CBT System (Describing Intentional Relationships that are Internal to Actors)</td>
<td>113</td>
</tr>
<tr>
<td>6.3</td>
<td>UML Activity Diagram Derived Using the Methodology</td>
<td>128</td>
</tr>
<tr>
<td>6.4</td>
<td>UML Activity Diagram Legend</td>
<td>129</td>
</tr>
<tr>
<td>7.1</td>
<td>Strategic Dependency Model of an Emergency Service Provider</td>
<td>136</td>
</tr>
<tr>
<td>7.2</td>
<td>BPMN model of the Emergency Service Provider</td>
<td>138</td>
</tr>
<tr>
<td>7.3</td>
<td>Business Process Model in BPMN</td>
<td>146</td>
</tr>
<tr>
<td>7.4</td>
<td>Extended Process Model Reflecting the Vulnerable Actor</td>
<td>147</td>
</tr>
<tr>
<td>7.5</td>
<td>Delegation of Dependencies among Actors</td>
<td>148</td>
</tr>
<tr>
<td>7.6</td>
<td>Business Process Model in BPMN</td>
<td>149</td>
</tr>
<tr>
<td>7.7</td>
<td>Extended Process Model Reflecting the Critical Actor</td>
<td>150</td>
</tr>
<tr>
<td>8.1</td>
<td>i* Organisational Model (SD) of a Home Telecare System</td>
<td>153</td>
</tr>
<tr>
<td>8.2</td>
<td>BPMN Model of Home Telecare System</td>
<td>154</td>
</tr>
<tr>
<td>8.3</td>
<td>Vulnerability Scenario 1</td>
<td>159</td>
</tr>
<tr>
<td>8.4</td>
<td>Vulnerability Scenario 2</td>
<td>160</td>
</tr>
<tr>
<td>8.5</td>
<td>Criticality Scenario 1</td>
<td>163</td>
</tr>
<tr>
<td>8.6</td>
<td>Criticality Scenario 2</td>
<td>164</td>
</tr>
<tr>
<td>9.1</td>
<td>Modeling Strategy</td>
<td>174</td>
</tr>
<tr>
<td>9.2</td>
<td>Requirements Capture Template</td>
<td>178</td>
</tr>
<tr>
<td>9.3</td>
<td>Sample SR Model of Demand Management</td>
<td>180</td>
</tr>
<tr>
<td>9.4</td>
<td>BPMN Model for Demand Management</td>
<td>182</td>
</tr>
<tr>
<td>10.1</td>
<td>Strategic Rationale Model for Computer Based Training System</td>
<td>190</td>
</tr>
<tr>
<td>10.2</td>
<td>Business Process Model in BPMN</td>
<td>195</td>
</tr>
<tr>
<td>10.3</td>
<td>Participants Organisation</td>
<td>199</td>
</tr>
<tr>
<td>10.4</td>
<td>Participants Experience</td>
<td>200</td>
</tr>
<tr>
<td>10.5</td>
<td>Participants Response on Methodology</td>
<td>205</td>
</tr>
</tbody>
</table>
In this chapter, a brief introduction of our research is provided. The first section starts with our motivation behind the research. We then illustrate our main contributions and organisation of the thesis.

1.1 Research Problem and Motivation

Requirement Engineering is one of the most important phases in software development lifecycle. Users typically know how to perform their day to day job operations. But they usually do not have an understanding of what systems should do or how would systems support their processes. Experienced consultants or analysts recognise ambiguous, incomplete or contradictory requirements. Requirement Engineering is the process of identifying the functions and real world goals of a software system [111]. It is divided into 5 stages:

- Elicitation of the requirements
- Analysis of the elicited requirements
- Management of the requirements
• Verifying requirements and

• Document requirements

Requirements capturing occurs in the early phase of software development. But these requirements specification should not be concerned only with the software specification, they should also relate organisational strategic dependencies, rationales with operational business processes describing the environment in which the system will function. The agent metaphor is powerful in modeling organisational contexts. Agent-Oriented Conceptual Modeling (AOCM) notations such as the $i^*$ [156] framework have gained considerable currency in the recent past. Such notations model organisational context and offer high-level social/anthropomorphic abstractions (such as goals, tasks, softgoals and dependencies) as modeling constructs. It has been argued that such notations help answer questions such as what goals exist, how key actors depend on each other and what alternatives must be considered [156] [157].

Many existing Business Process Modeling notations primarily focus on technical process aspects which is aimed at describing the sequence of activities, events and decisions that are made during process execution [92]. However, social and intentional components lack representation. The technical focus of these notations is especially suited for applications in the description, execution and simulation of business processes but is lacking in support for process redesign and improvement [154]. These notations effectively provide a view of the responsibilities and required communications between classes of process participants, but do not provide a view of other social and intentional characteristics including the goals of participants and their inter-dependencies. We argue this gap can be minimised by using and correlating organisational models
and operational process models in a complimentary fashion. For our work in this research, we choose Business Process Modeling Notation (BPMN) [145] and UML Activity Diagram [13] notations as technical notation to be used synergistically with the aforementioned teleological notation - i*.

Business processes represent the operational capabilities of an organisation. In order to ensure process continuity, the effective management of risk becomes an area of key concern. We argue there is a need for supporting risk identification with the use of higher-level organisational models and operational process models. This helps direct risk management attention to areas of critical importance within organisation models. Additionally, the information can be used to assess alternative organisational structures in domains where risk mitigation is crucial. At the process level, these measures can be used to help direct improvements to the robustness and failsafe capabilities of critical or vulnerable processes. We argue this approach will provide added benefits when used with other approaches to risk management during business process management.

One of the most fundamentally important decisions to be made during business process modeling is the choice of notation that will ultimately be used throughout a business process model’s lifecycle for description, communication, analysis and redesign. Many modeling notations have been developed in the past for this task, each with its own applied focus [12]. The choice of notation for modeling processes is however made to a specific notation that is deemed appropriate by the involved parties (e.g. analysts, participants, and/or senior management), given an evaluation of the benefits provided with its conceptual properties, at a specific point in time. Some of the common factors guiding the choice of notation may include personal preference for/ experience in a
1.1. Research Problem and Motivation

particular notation, or even the level/type of tool support available to the modelers. The persistent nature of business process models however, as a core artifact used for describing organisations leverages the importance of the decision when it is made for a particular specific notation. Furthermore, a preference obtained through evaluation at one point in time may vary in a future situation. Employee turnover, newly developed notations, added tool support and/or surfacing conceptual requirements might alter the liking towards and/or applicability of particular notations.

From their inception (i.e. commonly during the description of existing ‘as-is’ business processes), to their evolution (i.e. as part of the redesign process), and eventual decommissioning, business process models adapt to represent their designers interpretation of ‘what business process exists’, or ‘what business process is required to exist’ [138]. These interpretations are guided by many varying influences within the context of the modeling exercise. Varied stakeholder goals, policy implications, and/or operational constraints make it difficult to represent and trace such contextual details to the resultant process model.

The capability to change the representation of a given model for multiple applied circumstances is of great benefit in business process modeling, where multiple audiences (i.e. with either technical/non-technical backgrounds) or applications (i.e. possibly for description, re-design or execution) exist. A combination of notations could be of benefit in encouraging continued maintenance of the models themselves (i.e. given the cost of initial elicitation and design), to increase operational visibility via greater conceptual richness, or for analysis/application in specific improvement/re-design activities.
1.2 Research Aim and Objectives

The objective of this research is to facilitate process design in a dynamic organisational context by bridging the gap between organisational relationship (goals, dependencies, rational among actors) and operational processes by providing a process mapping and change management methodology. This is accomplished through a detailed literature review, development of a new methodology that supports integrated use of $i^*$ and BPMN models and testing the methodology in organisational context.

The outcomes of this research are as follows:

- Detailed analysis of the concept that flexibility and combination of notations is required in order to facilitate the maintenance of the models.
- Development of methods to facilitate and support the change and/or combined use of notations ($i^*$ and BPMN) for modeling business process.
- A methodology to integrate risks in process models through a set of intuitive metrics for extracting measures of actor criticality, and vulnerability from organisational models.

1.3 Research Scope and Strategy

The scope of this research includes:

- Facilitation of model maintenance.
- Facilitation of combined modeling notations and managing semantic and syntactic change.
• Measurement of risk in organisational and operational process models.

• Notations are restricted to i* modeling notation as part of Agent Oriented Conceptual Modeling/ Organisational Modeling and Business Process Modeling Notation (BMPN) as part of process modeling notations although it leverages the idea of applying the presented methodology to other process modeling notations such as UML Activity Diagram.

Following phases and activities were followed as part of our research strategy:

• **Literature review:** This stage involved the review and analysis of the related literature from printed sources (e.g. books, conference proceedings, journal papers, magazines etc.) as well as electronic sources (e.g. digital documents, online services). This process was carried out throughout the research project.

• **Research problem and scope:** The literature review was the starting point of this research and used as an input in defining the research problem and its scope at that point in time (i.e.2006). The focus of the research has been iteratively adjusted based on the better understanding of the area of study and achieved results.

• **Research question:** The research problem can be best expressed in terms of its underlying scientific question(s) that need to be answered. The potential research question was initially identified based on the review of existing literature published in public domain at that point in time (i.e.2006). The identified research question was tuned within the scope of this research and also referred throughout the research process.
1.3. Research Scope and Strategy

- **Methodology emergence:** This phase involved the development and continuous re-adjustment of the methodological components in small iterations during 2006 and 2011. Based on the review of existing knowledge and published literature, the components of the methodologies have emerged.

- **Initial testing:** This phase involved the high-level testing of the framework, which was conducted by the means of a case study to receive feedback from industry and then moving in the right direction. The results of the test cases have been used to further refine and improve the methodologies.

- **Empirical experiment:** The developed methodology has been presented to thirty four experts from industry and academia for review to identify their needs and experienced-based opinion via a two folded approach - case study based experiment and a questionnaire-based survey. The design of the methodology has been refined based on the feedback received from these experts.

- **Reporting:** The components of the methodologies that emerged were reported and published in the relevant peer-reviewed international conferences and scientific journals as part of the ongoing process of research communication during the development of this thesis between 2006 and 2011. From time to time, the components have also been presented to practitioners via presentations and focus groups for informal feedback and guidance. These methodologies have been continuously adjusted based on the feedback received by peers from both the research community and the IT industry.
1.4 Research Contributions and Application

The main contributions of this research are three fold.

- Firstly, we believe that the value of conceptual modeling in the $i^*$ framework lies in its use as a notation complementary to existing specification languages, i.e., the expressive power of $i^*$ complements that of existing notations. We examine how this might be done with BPMN and Unified Modeling Language (UML) [13]. Our aim, then, is to support the modeling of organisational contexts, intentions and rationale in $i^*$, while traditional specifications of functionality and design proceeds in BPMN and UML Activity Diagram. More generally, this research suggests how diagrammatic notations for modeling early phase requirements, organisational contexts and rationale can be used in a complementary manner with more traditional operational process specific notations.

- Secondly, we argue that the management of change throughout the business process model lifecycle can be more effectively supported by combining notations. In particular, we identify two potential sources of process change, one occurring within the organisational context and the other within the operational context. As such the focus in this thesis is on the co-evolution of operational (BPMN) and organisational ($i^*$) models. Our intent is to provide a way of propagating changes from one model to another one. We present constrained development methodologies capable of guiding an analyst when reflecting changes from an $i^*$ model to a BPMN model and vice-versa.
• Thirdly, we describe a novel approach to analysing risk, including business process risk, in an organisation context. In order to achieve this task, we propose an analysis of strategic dependencies between actors to measure and identify each actor’s vulnerability and criticality. Once actors’ vulnerability and criticality levels are identified and measured, business process models are reviewed to minimise risk in areas that require the most attention.

1.5 Thesis Organisation

The rest of the thesis is organised as follows. Chapter 2 presents a brief background of the major areas related to our research. These areas include Agent-oriented Conceptual Modeling techniques, Agent-oriented Methodologies, Business Process Lifecycle, Business Process Management Notation (BPMN), UML activity Diagram, risk, risk management etc. In Chapter 3, we present our methodology to integrate $i^*$ and BPMN for business process lifecycle management. In Chapter 4, we elaborate more on our methodology to present how we correlate business process and organisational models to manage change. In Chapter 5, we introduce our approach for integration of $i^*$ models and UML Activity Diagrams using effect annotations. In Chapter 6, we present our methodology for managing business process risk using rich organisational models. Chapter 7 describes risk measurement propagation through organisational network. We then illustrate the application of our constrained development methodology in a case study in chapter 8. In chapter 9 we present an experiment that we conducted with some domain experts in process modeling area. Finally, Chapter 10 draws conclusion.
Chapter 2

Background

In this chapter, a brief background of the major area related to our research is provided. The first section starts with the background of different Agent Oriented Conceptual Modeling techniques. We then present some background information about Business Process Management, Business Process Modeling Notations, UML Activity Diagram, Risk and Risk Management.

2.1 Agent-Oriented Methodologies

Recently, much work has been done to adopt agent-oriented perspective in requirement engineering. “The notion of agent in Requirements Engineering, however, is about agents in the world, most of which the software developer has no control over” [156]. The introduction of agent into requirement engineering supports requirements elicitation, exploration and analysis of software systems. Agent has characteristics such as intentionality, autonomy, and sociality. Therefore, agent orientation is currently seen as a software paradigm. Systems with characteristics such as autonomy, sociality, reactivity and proactivity, and communicative and cooperative abilities can provide greater functionality and higher quality when using agent orientation than
other software paradigm, such as object orientation. Agent models and languages can represent the computational behaviors in an abstract manner so that they can be realised in software programs [156] [73]. A number of agent-oriented methodologies have been proposed to model systems behavior. Modeling concepts, analysis techniques and tool supports are also available to use agent-oriented methodologies. In the subsequent sections, we will illustrate a survey of most popular agent-oriented methodologies.

2.1.1 \textit{i* framework}

Agent-Oriented Conceptual Modeling (AOCM) notations such as the \textit{i*} framework [156] have gained considerable currency in the recent past. Such notations model organisational context and offer high-level social/anthropomorphic abstractions (such as goals, tasks, softgoals and dependencies) as modeling constructs. It has been argued that such notations help answer questions such as what goals exist, how key actors depend on each other and what alternatives must be considered.

The central concept in \textit{i*} is that of intentional actor. Intentional properties of an agent such as goals, beliefs, abilities and commitments are used in modeling organisations [157]. The actor or agent construct is used to identify the intentional characteristics represented as dependencies involving goals to be achieved, tasks to be performed, resources to be furnished or softgoals (optimization objectives or preferences) to be satisficed. The \textit{i*} framework also supports the modeling of rationale by representing key internal intentional characteristics of actors/agents. The \textit{i*} framework consists of two modeling components [154]: Strategic Dependency (SD) Models and Strategic Rationale (SR) Models.
2.1. Agent-Oriented Methodologies

The SD and SR models are graphical representations that describe the world in a manner closer to the users perceptions. The SD model consists of a set of nodes and links. Each node represents an ‘actor’, and each link between the two actors indicates that one actor depends on the other for something in order that the former may attain some goal. The depending actor is known as dependee, while the actor depended upon is known as the dependor. The object around which the dependency relationship centres is called the dependum. The SD model represents the goals, task, resource, and softgoal dependencies between actors/agents.

As an example, consider a simplified version of the well-known Meeting Scheduler Scenario. This example will be used to illustrate the i* framework. The SD modeling process (refer to Figure 2.1 [156][84]) begins with identifying the actors involved with the meeting scheduling system and their mutual dependency relationship. The MeetingInitiator actor delegates much of the work of meeting scheduling to the MeetingScheduler. The MeetingScheduler no longer needs to be bothered with collecting availability information from Participants, or to obtain agreements about proposed dates from them. The MeetingScheduler also determines what are the acceptable dates, given the availability information. The MeetingInitiator does not care how the MeetingScheduler does this as long as the acceptable dates are found. This is reflected in the goal dependency of MeetingBeScheduled from MeetingInitiator to the MeetingScheduler actor. The MeetingScheduler expects the MeetingInitiator to enter the DateRange by following specific procedure. This is modeled via task dependency. Note, it is still the MeetingInitiator (not the MeetingScheduler) who has a stake in having Participants attend the meeting. The SD model models the meeting scheduling
process in terms of intentional relationships among agents, instead of the flow of entities among activities. This allows analysis of opportunity and vulnerability [156][84]. For example, the ability of a computer-based meeting scheduler to achieve the goal of MeetingBeScheduled represents an opportunity for the MeetingInitiator not to have to achieve this goal himself. On the other hand, the Meeting Initiator would become vulnerable to the failure of the MeetingScheduler in achieving this goal.

Figure 2.1: The Strategic Dependency Model of the Meeting Scheduling System

An SR model (see Figure 2.2) represents the internal intentional characteristics of each actor/agent via task decomposition links and means-end links. The task decomposition links provide details on the tasks and the (hierarchically decomposed) sub-tasks to be performed by each actor/agent while the means-end links relate goals to the
resources or tasks required to achieve them. The SR model also provides constructs to model alternate ways to accomplish goals by asking why, how and how else questions. For example, the Participant has internal task to ParticipateInMeeting. This task can be performed by subtasks AttendMeeting and ArrangeMeeting (these are related to the parent task via task decomposition links). For MeetingInitiator, the goal of MeetingBeScheduled is an internal goal. This goal can be met (represented via task decomposition links): obtain availability dates from Participants, finding a suitable date (and time) slot, proposing a meeting date, and obtaining agreement from the participants. In the case of Participants, the internal tasks FindAgreeableDateUsingScheduler and FindAgreeableDateByTalkingToInitiator are alternative means to achieve the goal Agreeable(Meeting, Date). How the alternatives contribute to softgoals is also represented. The SR model thus provides a way of modeling stakeholder interest, how they might be met, and the stakeholders’ evaluation of various alternatives with respect to their interests.
2.1. Agent-Oriented Methodologies

In a goal-dependency, the depender depends on the dependee to bring about a certain state in the world. The dependee is given the freedom to choose how to achieve it. In a task-dependency, the depender depends on the dependee to carry out an activity. Task and goal-dependencies may often appear interchangeable. One way to understand the distinction is to view goals as more coarse-grained, abstract entities and tasks as more fine-grained, specific entities (while recognizing that goals can always be reformulated as tasks and vice versa)[156].

Another dimension to this distinction is the relative autonomy of the dependee in deciding how a goal is achieved, while in a task the depender and dependee must coordinate in a far more tightly-coupled fashion. In a resource-dependency, one actor (the depender) depends on the other (the dependee) for the availability of a resource. In each of the above kinds of dependencies, the depender becomes vulnerable in situations where
the dependee fails to achieve a goal, perform a task or make a resource available. In a softgoal-dependency, a dependee depends on the dependee to perform certain goals or task that would enhance the performance. The notion of a softgoal derives from the Non-Functional Requirements (NFR) framework [25][26][27] and is commonly used to represent optimization objectives, preferences or specifications of desirable (but not necessarily essential) states of affairs.

In order to have a better understanding of the requirements the analyst may ask some strategic questions such as what are the drivers behind scheduling meeting ahead of time, why is it important for the meeting initiator to collect availability from the participant, why is it important to have a software based solution rather than having a manual one, who in the organisation will be benefited out of this solution, will electronic notifications via the system serve the purpose, how or why is the solution fits the broader organisational objectives etc [156].

To date majority of the requirement modeling languages do not provide the capability to capture the information mentioned above. The reason behind this is, they put more emphasis on ‘what’ rather than ‘why’. In [156], the author argues it is important to ask these ‘why’ questions in order to build a successful information system that has the capabilities to meet the stakeholders requirements in a better way as well as facilitating integration exercises with other systems easily. With the concept of SD and SR models $\text{i}^*$ has the capability to capture both ‘what’ and ‘why’ questions.
2.1.2 Tropos

Information systems exist in a rapidly changing organisational context. However the traditional or object oriented systems development methodologies were inspired by the technical solutions or programming concepts rather than organisational contextual view. This has naturally guided to a semantic gap between the systems and its operational context. Tropos framework which provides an agent-oriented software engineering methodology was proposed to reduce this gap [19, 120]. Tropos is motivated and adopted by requirements driven software development approach exploiting goal analysis and actor dependencies analysis techniques. The foundation of this framework is $i^*$ modelling framework. Tropos adopts the idea of modeling what and how together with why. In [19] and [120] the authors argue $i^*$ methodology was quite fit for the purpose of early phase requirements modelling and analysis in organisational setting, but in order to capture the other phases in systems development it needs to be extended and revised. Therefore, Tropos covers concepts of NFR by Chung [25, 26], KAOS [85] in addition to $i^*$ proposed by Eric Yu.

In Tropos through all the phases, systems are viewed as a social structure whose components are actors having goals to fulfill, tasks to do, resources to deliver. It also deals with the adoption of a transformational approach, i.e. we perform refinement steps, inside one phase or between phases, using a set of transformation operators. The approach is based on requirements-driven software development approach, exploiting goal analysis and actor analysis techniques [16, 18, 109].

The notion of agents, goal, task, resource and social dependencies are used along all the phases. This ensures the key elements and dependencies describing the organisational
setting can be used to rationalize and inspire each solution design and implementation alternatives. Each artifact can be retraced back to analysis performed during requirement phases. In this methodology system behaviour is easy to understand as there is a direct and natural correspondence between requirement analysis and implementation code.

The four phases of the Tropos framework are early requirements, late requirements, architectural design and detailed design [19]. In early requirements phase the analyst tries to understand the problem domain by analysing the organisational settings. The product of this phase is organisational models with relevant actors, their goals, tasks and dependencies with each other. In this phase Strategic Dependency models are used to capture actors, their goals and interdependencies whereas Strategic Rationale models are used to determine how the goals can be fulfilled through the contribution of other actors generally via a means-end analysis [19].

In late requirements phase the system is detailed within it operational environments along with relevant functions and qualities which means SD and SR models are revised further in the following manner:

- An actor is included in the SD model to represent the software system.

- Modify the SR model by conducting a means-end analysis considering the new system actor.

- Decompose the system actor into sub-actors if required and make adjustments in SD and SR models accordingly [19].
2.1. Agent-Oriented Methodologies

The third phase, architectural design captures the architectural settings of the systems in terms of other systems, subsystems, interconnections through models, data, controls and other dependencies. This phase refines the SD and SR models further in the following manner:

- An NFR diagram is developed to represent the selection and design rationale.
- New system actors and dependencies are introduced if required. Decomposing the existing actors into sub-actors and sub-dependencies are also done when necessary.
- Actors are assigned to agents and roles to fulfill actor’s goals [19].

In detailed design phase the behaviour of each architectural components are defined further in order to facilitate implementation. Following activities are covered in this phase:

- Class Diagram is produced based on SD and SR models.
- Sequence and Collaboration models are developed in order to capture inter-actor dynamics.
- State-based diagrams are produced in order to capture both intra-actor and inter-actor dynamics.

Tropos also suggests techniques for implementation from a detailed design phase. Some ways of implementation are using TAOM4E (Tool for Agent Oriented visual Modeling for the Eclipse platform) [103], BDI (Beliefs-Desires-Intentions) [15] agent architecture,
applying JACK Intelligent Agents [24] which is an agent-oriented development environment that extends Java with BDI agent model.

In Tropos methodology, the transformational approach is adopted for early and late requirements and partially for the architectural design. It starts with a limited list of Tropos conceptual elements such as actors, goals, softgoals etc and then iteratively adds details, dependencies where each step corresponds to the introduction/deletion of elements/relationships in the model. This approach provides systematic description of the processes, guidelines to the analyst, allows process analysis and provides a sound basis for describing and evaluating requirement gathering and design strategies. Overall, this methodology views software systems from five complementary perspectives which are social, intentional, communicational, process-oriented and objective-oriented perspective.

2.1.3 Formal Tropos

The aim of Formal Tropos (FT) is to facilitate effective integration of formal methods in Tropos software development process. As part of the Tropos project FT supports the Tropos methodology by introducing formal methods with a range of analysis tools [48]. Formal Tropos provides a specification language that offers the primitive concepts of early requirements specification of $i^*$ (actors, goals, strategic dependencies among actors), but supplements them with a rich temporal specification language inspired by the KAOS project [48]. Rather than representing only the ‘structural’ aspects presented in the $i^*$ model, FT provides the representation of the ‘semantic’ aspects of the model. This makes a range of additional analyses possible. Analyst using FT has the capability of defining circumstances under which a dependency among actors may
arise, as well as defining the fulfilment conditions for a dependency to be satisfied.

Formal methods have the capability to provide powerful specifications and early debugging techniques, hence they are accepted in the industry and have been applied to verify software systems. They have become an inherent part of software standards in some industries [14]. However, formal methods are usually applied in late or advanced phases of software development projects. Some of the drawbacks of applying formal methods during late phase include,

- systems size and scope become too large to handle
- late detection of issues and bugs [48][47].

FT provides a framework whereby an analyst is capable of adapting results from requirements engineering and formal methods communities to facilitate accurate analysis and modelling of early requirements. It provides a uniform approach by having one specification language across all phases.

FT is supported by a tool called T-Tool. This tool is based on a popular symbolic model checker NuSMV [28]. Using the T-Tool an analyst can transform an FT specification automatically into an Intermediate Language specification which can then be used by linking FT with different verification engines. Once the Intermediate Language representation is transformed into NuSMV (this transformation is automated), the analyst can then perform different kinds of formal analysis which includes animation of the specifications, checking consistency, verifying properties etc.

A specification in Formal Tropos consists of a sequence of declarations of entities, actors, goals, tasks, resources, dependencies, and global properties. In FT declarations for entities, actors, goals, resources, task, softgoals and dependencies are structured
in two layers. The outer layer declares their attributes, and is in a sense similar to a class declaration. The inner layer expresses constraints on the instances, and thus implicitly determines their evolution [47].

In [47] FT grammar was specified which is:
The outer layer

specification := (entity | actor | int-element | dependency | global-properties)

entity := Entity name [attributes] [creation-properties] [invar-properties]

actor := Actor name [attributes] [creation-properties] [invar-properties]

int-element := type name mode Actor name [attributes] [creation-properties] [invar-properties] [fulfill-properties]

dependency := type Dependency name mode Depender name Dependee name [attributes] [creation-properties] [invar-properties] [fulfill-properties]

type := (Goal | Softgoal | Task | Resource)

mode := Mode (achieve | maintain | achieve&maintain | avoid)

Attributes

attributes := Attribute attribute

attribute := facets name : sort

facets := [constant ]...

sort := name | integer | Boolean...

The Inner Layer

creation-properties := Creation creation-property

creation-property := property-category event-category temporal-formula

invar-properties := Invariant invar-property

invar-property := property-category temporal-formula

fulfill-properties := Fulfillment fulfill-property

fulfill-property := property-category event-category temporal-formula

property-category := [constraint | assertion | possibility ]

event-category := trigger | condition | definition

Global properties

global-properties := Global global-property

global-property := property-category temporal-formula

FT has been applied in several projects and analysts have found it useful during early development phases [49]. Model checking relied mostly on design-inspired specification languages in order to use them during early phase requirements elicitation. FT is a
novel approach in the sense that it provides the capability for the analysts to perform model checking during early requirements analysis and modelling without relying on design-inspired specification languages. We plan to utilise some concepts of FT specifically the concept of fulfilment conditions in our constrained development methodology proposed later in the thesis.

2.1.4 AOR

Agent-Object-Relationship (AOR) modeling was proposed as an agent oriented extension of ER diagram and UML style class diagram [133, 134]. AOR conceptual modeling technique is based on several ontological principles including Entity Relationship modeling and actor-responding diagram language. AOR makes an ontological difference between active and passive entities which is between agents and ordinary objects. Thus it facilitates the representation of deeper dynamic and deontic aspects of organisations and organisational information systems by taking into accounts the events and actions related to the business processes [134]. AOR modeling argues the semantics of business transactions can be captured effectively and efficiently if the specific agents related with the events and actions are represented in the organisational information systems in addition to passive business objects.

In AOR modeling an entity is regarded as either an agent, an event, an action, a claim or an ordinary object; an organisation is considered as a complex institutional agent that defines the rights and duties of its internal agents that can act on behalf of it [133, 134]. These internal agents can be involved in interactions with external agents. Agents are the only entity that can communicate, act, perceive, make commitments and satisfy claims. The objects on the other hand can not communicate, act, perceive
anything to make commitments or satisfy claims. Similar types of agents and objects share a number of attributes that represent their properties and characteristics.

In AOR basic models are divided into two categories - external and internal models. In an external AOR model, the view of an external observer who is observing the agents and their interactions in the problem domain is considered. The external AOR model has a focus, which is either an agent, or a group of agents, for which an analyst develops a state and behavior model. An external AOR model can consist of one or more agent diagram, interaction frame diagram, interaction sequence diagram, interaction pattern diagram. In an internal model the internal view of a particular agent to be modeled is taken into consideration. An internal AOR model consists of one or more reactions frame diagram, reaction sequence diagram and reaction pattern diagram.

External and internal models both jointly help the analyst to build a system. In the analysis phase an external AOR model including one or more focus agents is drawn as domain model. In the design phase, each of the focus agent’s external model is transformed into an internal AOR model according to the agent’s perspective. Then the internal AOR model of each focus agent is refined into an implementation model for the target language such as SQL or Java [142]. ER and UML use relational and object oriented database technology to support the design of object oriented information system. The AOR modeling language on the other hand is based on AOR metamodel to support the high-level design of agent oriented information system.

AOR modeling has several strengths; it has a rich set of basic ontological concepts which allows more semantics of a domain. It also unifies many of the fundamental
domain modeling concepts found in enterprise modeling approaches [134]. AOR allows integrating state and behavior in one diagram which provides added benefits to the developer. It also applies the important concept of reaction rules for behaviour modeling.

2.1.5 Prometheus

Prometheus [114, 115, 116] methodology supports the development of intelligent agent systems using BDI platforms. It defines a generic modeling language to any MAS architecture and implementation environment. Prometheus offers a specific process for specifying, designing, and implementing intelligent agent systems.

Prometheus distinguishes itself from other methodologies in the following areas [116]:

- Supports the development of intelligent agents which use goals, beliefs, plans, and events. By contrast, many other methodologies treat agents as “simple software processes that interact with each other to meet an overall system goal” [33].

- Provides “start-to-end” support (from specification to detailed design and implementation) and a detailed process, along with design artifacts constructed and steps for deriving artifacts.

- Provides hierarchical structuring mechanisms which allow design to be performed at multiple levels of abstraction. Such mechanisms are crucial to the practicality of the methodology on large designs.

- Uses an iterative process over software engineering phases rather than a linear “waterfall” model.
• Provides (automatable) cross checking of design artifacts.

Prometheus methodology consists of three phases: system specification, architectural design and detailed design. The system specification phase focuses on modeling the basic functionalities of the system, as well as inputs (percepts), outputs (actions) and any important shared data sources [114, 115, 116]. The architectural design phase consists of modeling agents, the interrelationships and interactions between agents by using the outputs from the previous phase. The detailed design phase focuses on building the internal structure of each agent and its capabilities within the overall system. Prometheus methodology is supported by the Jack Development Environment (JDE) and the Prometheus Design Tool (PDT) that can be used for the artifacts developed in detailed design phase.

2.1.6 Gaia

Gaia methodology supports the analysis and design of agent-based systems. Gaia can be applied to a wide range of multi-agent systems, and it supports “both the levels of the individual agent structure and the agent society in the multi-agent systems” [108]. In Gaia methodology, multi-agent systems are viewed as computational organisations consisting of various interacting roles. The key concept in Gaia is roles, which have associated with them responsibilities, permissions, activities and protocols. Roles can interact with each other in certain institutionalised ways, which are defined in the protocols of the respective roles [108].

Gaia is appropriate for large scale systems that hold the following characteristics [108]:

• Agents are coarse-grained computational systems, each making use of significant
computational resources.

- It is assumed that the goal is to obtain a system that maximises some global quality measures which may be sub-optimal from the point of view of the system components. Gaia is not intended for systems that admit the possibility of true conflict.

- Agents are heterogeneous. As a result agents may be implemented using different programming languages, architecture platforms and techniques. There is no assumption about the delivery platform in the methodology.

- The organisation structure of the system is static, in that inter-agent relationships do not change at run-time.

- The abilities of agents and the services they provide are static, in that they do not change at run-time.

- The overall system contains a comparatively small number of different agent types (less than 100)\[108\].

The Gaia design process consists of three models [108]:

- Agent model: The agent model identifies the agent types that will make up the system, and the agent instances that will be instantiated from these types.

- Service model: The service model identifies the main services that are required to realise the agent’s role.

- Acquaintance model: The acquaintance model documents the lines of communication between the different agents [108].
There are three steps in Gaia design process. The first step is to map roles into agent types and to create the right number of agent instances of each type. The second step is to determine the services model needed to fulfill a role in one or several agents. The last step is to create the acquaintance model for the representation of communication between the different agents [108].

2.1.7 Comparison

The following table (Table 2.1) summarises the main characteristics of those methodologies presented above.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Gaia</th>
<th>Prometheus</th>
<th>AOR</th>
<th>Tropos</th>
<th>i*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Elicitation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Coding &amp;</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Comparison among Agent Methodologies

2.2 Business Process Management

Business Process Management (BPM) is a re-emerging field of study, with determined focus from both industry [129] [59] and academia [138] [39] alike. BPM promotes that a clear understanding through the explicit modeling of the processes underlying an organisation is required to support effective organisational management/ improvement practices [59] [125]. The importance of process-orientation, as opposed to functional
or departmental orientation is evident in the holistic view it provides of organisational operations, as described in the many antecedents to the field [58] [37] [129]. That is, the process perspective supports improvement initiatives by enabling a holistic impact assessment in terms of operational changes, rather than the traditional approach of piecemeal improvement in a specific functional area to the detriment of other functional areas, or even to some extent, the overall operational effectiveness of the organisation. From the perspective on information systems developers, the semi-formal representation of organisational context that business process models provide allow for rapid/traceable development and deployment of systems that can effectively support the operations of an enterprise. Added benefits arise when the models can be translated to ‘process code’ [44] [113] that can be understood by ‘process-aware information systems’ [39], effectively accelerating the change process.

In order to provide scope for a common understanding on business processes, we provide a definition based on the meanings in [129], [58], [37] and [36]. A Business Process is a set of dynamically coordinated activities, controlled by a number of socially dependent participants, aimed towards the achievement of a specific operational objective. For example, take a process within a transport organisation aimed towards “the timely delivery of packages to customers”. The coordination of activities classified under “package procurement”, “tracking and scanning”, “routing”, and “delivery fulfillment” may be required, however their coordination will be dynamically tailored on a case-by-case basis. The dynamic nature of a business process exists due to the autonomous nature of the participant[s] that control activity execution. Given a case, say “routing a particular package to a sorting destination”, the participants have the discretion to
take the most suited course of action given current circumstances, including their own (e.g. deciding on costly air freight, as opposed to ground freight, due to their time constraints). Any action chosen is however governed by the fulfillment of dependencies between participants. For example, “package routing by sorting operations” may not begin until “clearance is provided by a regulatory body”.

### 2.2.1 Business Process Management Lifecycle

Business processes evolve throughout their lifecycle of management, brought on by the constant nature of change in today’s business environment [152]. This evolution can be characterized by a number of distinct phases [129] [138]. Figure 2.3 illustrates the major phases of the business process lifecycle, the areas of core focus for each phase (i.e. organisational, conceptual, and/or technical) through Conception, Production, Operation, Cessation, and the feedback mechanisms that aim to maintain effective and efficient operation. For greater clarity, these phases are briefly described below.

**Discovery.** Recognition of a process is required (e.g. communication, improvement, compliance), and elicitation, analysis and modeling is conducted to accurately identify important characteristics of the business process in consideration. Process participants, line management and/or executive staff may be queried regarding their understanding of the process, and models are used to illustrate / validate interpretations with varied stakeholders. The model may be further refined / elaborated upon when new information / inconsistencies arise.

**Design.** Two distinct situations that cause the need for design to be initiated - (1) clean sheet process design, and (2) process re-design or improvement. Design requires
that goals and aims be clearly expressed, available resources / capabilities are identified, and a business process designed given operational / structural constraints to most optimally achieve the goals of involved stakeholders. Social context (including dependencies between participants) is also of importance during design and should be taken into consideration [76]. Yu [155], states the requirement for models that can aid in reasoning about how the design requirements (i.e. enterprise motivations and constraints) can be effectively realized by alternate prescribed configurations of roles, responsibilities, actions and interactions. These configurations must then be able to be communicated for review and possibly deployment.

**Deployment.** A new or improved design has been developed and needs to be communicated to all people that are enrolled in enacting and monitoring the design. This may include people responsible for automating tasks using IT infrastructure. Resources
that have been identified as available need to be acquired and made available for use by the participants. All parties require detailed and concise specification of their responsibilities, with access to the overall context, motivations and importance of their assignments.

**Enactment.** The process is enacted and the design must be understood and followed to meet current aims and objectives. The availability of policy, procedure, and performance information during the operation of the process to all participants in the process (i.e. or parts thereof) is required for guidance (i.e. both managerial and participant).

**Monitoring.** The execution of the process is examined in order to apply controls that ensure the current design is being adhered to. This type of analysis is primarily compliance driven. Reactive controls are applied when participants in the process do not follow the prescribed activities as per the design. Rich models that can describe the technical requirements of the process including the procedures, policies, and performance measures are required for successful enactment.

**Analysis.** This entails the proactive analysis of operational information aimed towards improving the process given the feedback obtained from monitoring the process or its environment of operation. During monitoring, design deficiencies may be identified to exist that may be improved upon given an analysis of the causes introducing particular deficiencies. Some feedback may have indicated possible improvement due to some change in the process’ environment of operation. This may include the availability of new resources or modification to some constraints on the process (e.g. regulatory, operational, or structural). Analysis may also result in identifying outsourcing opportunities, whereby the process is redeployed within a partner organisation.
Disposal. Problem Being Addressed: The process is no longer required by the enterprise. Involved participants must be informed of the disposal. Resources must be identified and reallocated to other areas of operation. Information collected during the operation of the process within the organisation may still be of value now or in the future. Therefore, it is important that the process information (i.e. including models) be archived for possible future use or communication.

2.2.2 Business Process Modeling

Business process modeling aims to conceptualize current or desired business processes in a common (preferably graphical) language/notation that can be communicated to all involved participants [59]. These include business users (i.e. analysts, partners, managers and even first line workers etc.) and technical users (i.e. systems/network architects, programmers and support staff etc.). Business process models also form the basis of Business Process Management Systems (BPMS). Thus, it is of benefit if a business process model can be mapped to an executable process language (e.g. XPDL [44]) that is understood, and can be executed by, a BPMS.

The notations used for business process modeling, have been categorized in many works, based on their conceptual features and areas of application. An understanding as to the scope and focus of BPM, can be obtained by surveying these existing classifications. Firstly, [12] categorizes some common business process notations into four groups based on their specific “concept representation and use”. I/O Flow - focusing on resource production, modification and consumption between activities, that is useful for reasoning about the properties and logistical aspects of physical objects (e.g. IDEF0); Workflow - focusing on temporally ordered sequence of activity execution, useful for
introducing strict order in production like processes (IDEF3, UML Activity Diagrams, and other Petri Net based notations); *Agent-oriented* - focusing on the interaction of participants and their ability to control and monitor process execution, that helps in facilitating their coordination/communication (UML Collaboration Diagrams, Role Activity Diagrams); *State-flow* - focusing on the transitional effect of activities on the state of the current environment, that effectively illustrates progress towards the attainment of desired goal states (UML State Transition Diagrams).

A similar categorization is provided in Kavakli et. al. [77], where business process modeling notations are classified into Activity-Oriented approaches that “describe a process as an ordered set of activities”, with decomposition, synchronization, and information flow represented (e.g. IDEF0, DFD, Workflows, and the F3 process model). *Agent-Oriented (or Role-Oriented)* - whereby “the focus is on the entity that performs a process element” (e.g. Role Interaction Nets, Role Activity Diagrams, i* and the ORDIT approach). *Product-Oriented* - with a focus towards the results of activity completion, i.e. the products and transformations made on them, whereby “each product entity has a defined sequence of states and triggers that cause state transformations”.

A criterion for classification based on enterprise modeling is also discussed in another of their related works [92], whereby three specific views are used to describe enterprise operations. *Technical View* - which includes the processes, information flows, data and resources of an organisation. *Social View* - where policies, structures and work roles are represented. *Teleological View* - that captures the intentional reasoning behind organisational activities and their assignment to various organisational agents.

Katzenstein et. al. [76] further categorizes notations where the criteria is the richness
in their representation of “social context”. This is of high importance when reasoning about social dependencies and alternative organisational structures. These include: 

*Traditional Systems Analysis Methodologies* - focused on capturing activities and information or resource flow, with little to no social context represented (DFD, Flow Charts, IDEF0); *Coordination Models* - focused towards the specification of the conditions and states must be met at each stage in process execution, however their focus on co-ordination omits much of the social aspects inherent within process (Role Interaction Nets, Role Activity Diagrams, Petri Net based notations); *Socio-technical/Qualitative Systems Models* - focused specifically on the social and intentional representation of business process including goals, rich information, causal and structural relationships (ETHICS, Soft Systems Methodology, Multiview, i*).

Finally, Yu [155] makes the distinction between models suited for the description of business processes, and those more suited for supporting their design. As outlined, *Work Description Models* - focus on describing alternate work arrangements, whereby the ‘salient’ features of technical work configurations are highlighted (e.g. IDEF0, Action-Workflow, and the i* Strategic Dependency diagram). These models tend to descriptively represent technical aspects of the process design including designed flows of control between activities. *Work Design Models* - focus more on supporting the design of work arrangements by facilitating the systematic exploration of possibilities, their analysis, and the construction of most optimum solutions (e.g. gIBIS, KAOS, and the i* Strategic Rationale diagram). These types of models tend to represent the intentional aspects of a process including inter-agent dependencies, motivations and alternative possibilities for meeting operational requirements.
The BPM classifications outlined share many commonalities that can be traced to our aforementioned definition. In effect, each category represents a specific view of certain business process related aspects. These aspects are represented in Figure 2.4, in relation to the views of enterprise operations, and the notations that support their representation.

As observed in Figure 2.4, many notations focus on specific aspects, with limited relation/traceability to other important aspects. Business process modeling requires an integrated view to support the development of rich models that provide an enhanced ability to conceptualize, communicate and understand business processes in support
their effective management.

A small number of notations, supported with associated development methodologies provide an integrated and holistic view (e.g. KAOS, EKD) however their use is limited to their specific underlying models and methods. Certain applications may require the analysis capabilities of a particular notation (e.g. conflict identification/resolution procedures available in KAOS [20]). Certain views may be more suited to particular audiences (e.g. Role Activity Diagrams for business analysts and UML activity diagrams for systems developers). Notational flexibility is required in order to facilitate the modular use of views/aspects when required, given certain contexts.

### 2.2.3 Business Process Modeling with BPMN

Many existing BPM notations primarily focus on technical process aspects including the flow of activity execution/information and/or resource usage/consumption [92]. This perspective is aimed at describing the sequence of activities, events and decisions that are made during process execution, however social and intentional components lack representation. The technical focus of these notations is especially suited for applications in the description, execution and simulation of business processes but is lacking in support for process redesign and improvement [155].

One such notation is the Business Process Modeling Notation (BPMN), developed by the Business Process Management Initiative (BPMI.org). BPMN can be seen as primarily a technically-oriented notation that is augmented with an ability to assign activity execution control to entities (e.g. roles) within an organisation with ‘swimlanes’ [145]. This effectively provides a view of the responsibilities and required communications between classes of process participants, but does not provide a view of other
2.2. Business Process Management

social and intentional characteristics including the goals of participants and their interdependencies.

Since it’s initial publication [145], BPMN has been accepted by the greater BPM community, due to its expressiveness and ability to map directly to executable process languages including XPDL [44] and BPEL [36]. The wide uptake of the notation by most business process modeling tool vendors is also a sign of its longevity. Some practitioners have hailed BPMN as supplying a rich representation that allows Business Process Management Systems (BPMS) the ability to control the required interactions with humans and 3rd party applications [107]. Processes are represented in BPMN using flow nodes: events, activities, and decisions; connecting objects: control flow links, and message flow links; and swimlanes: pools, and lanes within pools.

We take Figure 2.5, a collaborative medical treatment process, as an example to illustrate BPMN. The process requires the interaction of three high-level process participants - the patient, receptionist and doctor. Collaboration between participants on the model is represented by message flow links between activities within pools. Responsibility within the Medical Centre is delegated to two roles - a receptionist, and a doctor. Responsibility assignment within a pool is represented using lanes (i.e. pool divisions). Each pool within a process model represents a single process. Processes are initiated by a start event, represented as a circle at the beginning of each pool. Control flow links between activities, decisions and events represent the controlled progression through each process. Finally, the process is completed with an end event, or bold circle towards the end of a process.
Our constrained development methodology primarily consists of two modeling notations; $\iota^*$ as organisational modeling notation and BMPN as process modeling notation.

### 2.2.4 Business Process Modeling with UML Activity Diagram

An activity diagram is an uncomplicated and perceptive illustration that depicts the actions, parallel activities and any possible alternative ways through the workflow. Activity diagrams defined in the Unified Modeling Language [137] are consequential from various methods to pictorially express sequence of activities or sub-activities and
conditions taken within a process. Activity Diagrams explain the operational flow from an initiating point to the terminating point specifying many decision paths that exist in the development of proceedings contained in the activity. They are also used to explain states where parallel processing may occur in carrying out of some activities.

The design of an activity diagram may demonstrate the organisational stage or the

![Figure 2.6: Example of a UML Activity Diagram](image-url)
system stage. The most important exercise of using activity diagrams are in designing the operational progression that defines the sequences of operations and the realization of operation. In a system development process, design is important as the lack of good design of processes will lead to non-maintainable, non-reusable system having obscure functionality [86]. Activity diagrams are useful and important for modeling the dynamic aspects of a system for several reasons; it describes the internal actions of an operation graphically, helps to recognize activities whose accountability belongs to another place, illustrates activities that can occur in parallel, allows the detection of common functionality within a system and can construct executable systems through forward and reverse engineering. Figure 2.6 illustrates an example of a UML Activity Diagram.

Graphically an activity diagram is an anthology of vertices and arcs which generally contains activity states, action states, transitions and objects. Activity states are non-atomic as they can be interrupted and usually they may take some time to be accomplished. But action states are atomic, their work is non-interrupted. Action states can not be decomposed. Transitions depicts the path initiated from one action or activity state and passed to next action or activity state as the action or activity of a state is completed. Transitions are represented as a simple directed line in the activity diagram [13].

We will illustrate how our constrained development methodology can be applied to UML Activity Diagram later in the thesis.
2.3 Risk and Risk Management

The notion of risk has been extensively discussed in the literature. In general, risk is anything that threatens the successful attainment of organisational goals. From the project management perspective ‘risk’ is described as a problem that has not yet happened but which could cause some loss or threaten the success of the project if it did [146]. Project Management Institute defines risk as “an uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective” [121]. In classic decision making theory, risk is defined as “Reflecting variation in the distribution of possible outcome, their likelihoods, and their subjective values” [95]. A general definition of risk describes risk as an event that has some probability of happening, and if occurs, will result in some loss [72].

There are many ways of calculating measurement of risk. Mostly, risk is measured in terms of consequence and likelihood where,

\[ \text{Risk} = \text{Consequence} \times \text{Likelihood} \]

Risk management is the systematic application of management policies, practices, and procedures to the task of identifying, analyzing, assessing treating and monitoring risk [71]. Risk management is also defined as a disciplined approach used to recognise, estimate, and reduce or eliminate the likelihood of an unfavorable context from the expected outcome of certain activities [66]. It is a decision making process which takes relevant risk assessments related to potential hazard into considerations to develop optimal decisions to manage the risks. Risk management processes involve establishing the context; acknowledge and identify risks; evaluate and prioritize risks; select
appropriate risk management strategies and implement the plan and monitor and update the risk management program [7]. Many organisations are aware of the risks that may have impact on their processes. They generally have good intentions to develop plans to mitigate identified risks, but end up producing minimal plans or no plans at all. As a result the effects of a disaster or disruption of business may be catastrophic and may never be overcome [100]. Preparing for a negative event allows reducing the impact by allowing to recover quickly by reducing the overall damage to the organisation. The Australian and New Zealand standard highlights the risk management is a “multifaceted process” and involves seven key elements [7]. Figure 2.7 illustrates these
2.3. Risk and Risk Management

...elements.

Establishing Contexts. In this phase the process of establishing the context for organisation is determined in terms of strategic contexts, the organisational context, the risk management context, the development criteria and deciding the structure.

Identify Risks. Risk is identified by analyzing “what can happen” and “how can it happen”.

Analyze Risks. The objective of this phase is to separate the minor acceptable risks from the major risks and to provide data to assist in evaluation and treatment of risk. This phase is completed by two major roles, first determine the likelihood of risks and then determine the consequences.

Evaluate Risk. In this phase the found risks are compared against the criteria, risks are prioritized and risks control policies are determined.

Treat Risks. In this phase risk treatment options are identified, evaluated and selected. Preparation and implementation of treatment plans are also part of this phase.

Monitoring and Review. This is a continuous process. Monitoring the risks and their response are very important along with the process of evaluating the risks and the factors that have changed. Information needs to be collected in this phase. This can be done via self assessment, physical inspection, checking action plans, audit and reassessment of risk profile.

Communicate and Consult. This is also a continuous process that connects all the phases to include communication and consultation activities.

Several risk-handling / risk-treatment strategies are found in the literature. The techniques are acceptance, avoidance, risk transfer and impact mitigation [6] [119]. But...
the fact is not all the risk handling strategies can be applied to all types of risk. For example, risk mitigation and avoidance are applied to reduce the probability of a given error, where risk acceptance and transfer consider more magnitude of the error consequences. The four risk-treatment options are discussed below.

**Acceptance.** Risk acceptance is appropriate where the remaining risk levels are insufficient to justify potential treatment options or where it is not possible or is not financially viable to treat the risk. Practically it is not possible to eliminate all the risk due to some constraints such as time, resources etc. Enactment of risk contingency plan might be required in these circumstances.

**Avoidance.** In a situation where the risk is unacceptable and the risk controlling techniques are not viable it is better not to proceed with an activity or component that could generate risk. This helps to eliminate the likelihood of a risk before it occurs.

**Impact mitigation.** The aim of this risk-handling strategy is to reduce the probability of risk and the impact that the occurrence of the risk may bear. This technique may be appropriate for a high consequence/low likelihood threat such as total computer system failure. More attention is given on the contingency plans, evaluation plans and business resumption plans.

**Transfer Risk.** This is to shift the risk and its consequences from one party to another. In this technique responsibility for treating risk is transferred to other organisations that are better able to manage risk, such as outsourcing the activities such as computer services and security.

A number of research studies have investigated the issue of relative importance of various risks in software development projects and have attempted to deal with them in
various ways. A knowledge based risk management approach for a utility business service model was projected in [23] [155] which proposed an intelligent software early warning system based on fuzzy logic using an integrated set of software metrics. A formal model was introduced to assess risks and the duration of software projects based on a few objective indicators that can be measured early in the process [110]. In all the cases described here, individual risks factors are assessed directly based on software metrics. Overall risks associated with the entire process or system is not obtained.

Schmitt introduced a means of designing a business process that matches the business goals of its underlying business model by detailing the risks inherent to a business value proposition, and identifying process pieces that would satisfy to those goals [102]. Although a requirement approach based on $i^*$ was proposed for capturing the needs and constraints, it does not manage business process risks using organisational models.

A study was conducted to identify risks factors in implementing traditional MIS projects [130]. The study describes the risk factors associated with enterprise-wide/ERP projects and identifies the risk factors which are unique to those projects. It was outlined in the literature how risk analysis and risk mitigation instruments can be used for the purpose of business and process modeling [97]. The approach suggests a stepwise development of process models from business models. However, it lacks the integration of organisational goals and identification of risks in both organisational and process models.

Risk has been considered from a project management perspective although it is an
inherent property of every business process [105]. A risk-aware process modeling technique was proposed along with four interrelated model types which together can accommodate risk related requirements [105]. Thus, this technique provided a detailed overview of integrating risks in business process models. In addition an overview of risks associated with BPM projects along the phases of the BPM lifecycle was also provided [104]. The authors consider that the majority of risks identified lie in mismatch with the methods employed within the various phases of the process lifecycle, a lack of clarity who is responsible for the individual phases or their results and a mismatch of process design, automation and evaluation objects. We believe that risk can be better viewed by using a combined notation proposed in [79]. This way we can identify and deal with risks from higher level organisational model to the actual business processes. In [160], the authors examine the definition of the risk generated by business process from a holistic perspective of business process management where they considered business process risks comprise risk from operational environment of business process, risk from the main part of business process, and risk from the object of business process. In this work the authors also analysed the four milestones of risk (i.e. risk identification, risk evaluation, risk control and risk management effect analysis) mentioned in [89] from business process perspective. There is no denying of the fact that in order to achieve the objective of an organisation, it is highly critical that continuous, effective and efficient performances of its business processes are maintained. In [68], the authors argue that the improvement from an economical viewpoint is mainly performed by the domain of business process management, whereas the consideration of risks and continuous execution of business processes is considered separately by risk management
and business continuity management. This separation is highly likely to lead to inefficiencies as decisions can be contradictory. The authors hence, introduced a vision of risk-aware business process management that is capable of providing information for economic as well as for security discipline. In [69], the authors then illustrated a reference model which comprises extensions of existing modeling languages in order to meet our simulation-based risk-evaluation needs. According to [29], existing process modeling notations do not include a complete notation for documenting how processes can fail, in other words artefacts to represent risk are limited. The authors in [26] developed a conceptual framework for extending standard business process metamodels to include complete information that is helpful for managing and quantifying operational risk in business process. The authors provided an artefact, a set of risk extensions to BPMN modeling notation. We argue these extensions have the capability to provide the analyst/ risk manager to provide a better representation of risk in process models, hence taking mitigation actions should they rate high in vulnerability and criticality measurements. In our thesis, we will provide a methodology to manage business process risk using rich organisational models.

2.4 Non-functional Requirements

Requirements consist of functional requirements and non-functional requirements [27]. Functional requirements define the capabilities that the system should provide. Functional requirements concern the data input and output; they can be easily quantified and evaluated. Non-functional requirements (NFRs) are constraints on the functions
offered by the system such as performance constraints, timing constraints, etc. Non-functional requirements do not describe any of the system’s input, processing, or output.

Based on the definition provided in [135] a non-functional requirement is “a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, software design constraints, and software quality attributes”. According to Charette [21], NFRs deal with those aspects of the system that are not easily quantifiable; therefore, they are not conclusively testable and cannot be an absolute precondition for the correct functioning of the system. In [80] Kotonya argues that NFRs are very important because they define the overall qualities or attributes of the resulting system as well as placing restrictions on the product being developed and specifying external constraints that the product must meet. He also suggests that functional requirements might need to be sacrificed to meet the constraints of NFRs.

In [94], Malan classifies non-functional requirements into two categories: qualities and constraints. Qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system. Constraints are not subject to negotiation and, unlike qualities, are (theoretically) off-limits during design trade-offs. Both qualities and constraints have five characteristics, SMART requirements, which are Specific, Measurable, Attainable, Realizable, Traceable [94].

Chung et al [27] argues treatments of non-functional requirements can be classified into two categories: quantitative and qualitative ones. Most product-oriented approaches
use quantitative metrics for measuring the satisfying degree to a non-functional require-
ment; therefore product-oriented approaches are quantitative while process-oriented
approaches are qualitative based on the ideas adopting from qualitative reasoning [5].
Chung also suggests that these two approaches are best regarded as “complementary,
both contributing to a comprehensive framework for requirement engineering”. In [27],
Chung et al proposes a novel framework called NFR Framework for representing and
analyzing non-functional requirements. The aim of this NFR framework is to put
non-functional requirements foremost in the developer’s mind. It adopts softgoal con-
cept to represent non-functional requirements. Softgoal represents a goal that has no
clear-cut definition and/or criteria as to whether it is satisfied or not [27]. There are
some influence or interdependency relationships among softgoals. This relationship is
used further to form a softgoal interdependencies graphs(SIGs) to record the develop-
ers consideration of softgoals. Furthermore, softgoal interdependencies can be used to
illuminate design decisions which relate implementation objects to their design coun-
terparts, and design objects to their requirements counterparts.
In [27], Chung argues most conventional approaches to system design are driven by
functional requirements and majority of developers focus only on the functionality of
the system, while the how to achieve those functionality might be ignored. Chung et.
al. proposes eight steps in the design process that relate to non-functional requirements:

- “Acquiring or accessing knowledge about the particular domain and the system
  which is being developed, functional requirements for the particular system, and
  particular kinds of NFRs and associated development techniques
- identifying particular NFRs for the domain

- decomposing NFRs

- identifying “operationalizations” (possible design alternatives for meeting NFRs in the target system)

- dealing with ambiguities, trade offs, priorities, and interdependencies among NFRs and operationalizations

- supporting decisions with design rationale

- evaluating the impact of decisions” [27].

In business process management and modeling, non-functional requirements are also important factors; they are usually called Quality of Services (QoS). Quality of service (QoS) [99] is a combination of several qualities or properties of a service: such as capability, performance, reliability, integrity, security etc.

### 2.5 Service-Oriented Architecture (SOA)

A Service-Oriented Architecture (SOA) is an information technology approach or strategy in which applications make use of services available on the network such as the World Wide Web [117]. A SOA is a form of distributed systems architecture, essentially a collection of services. Different from traditional application architecture, which links all the business logic together to compose a single application, the fundamental component of SOA is a service and services can be composed in specific ways to build applications [149]. The foundation of SOA is made of basic services, their descriptions and basic operations, such as publication, discovery, selection and binding [117].
SOA provides consistent interoperability and reuses existing services where possible. Implementing a SOA can involve developing applications that use services and making applications available as services. The W3C Web Services Architecture Working Group defines SOA as “a form of distributed systems architecture that is typically characterized by the following properties [141]:

- Logical view: The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.

- Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves.

- Description orientation: A service is described by machine-processable meta data.

- Granularity: Services tend to use a small number of operations with relatively large and complex messages.

- Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.

- Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces.”

A SOA consists of three layer: (from the bottom to the top of pyramid) basic services layer, composite services layer and managed services layer. The service composition layer of SOA includes necessary roles and functionality for combination of
multiple basic services by Service aggregators. Composite services may be regarded as
basic service for further service compositions, and they can also be used as applica-
tions by service clients. The specifications and/or code developed by service aggrega-
tors allow composite services to perform functions, such as coordination, monitoring,
conformance and quality of service (QoS) composition. The services management layer
of SOA is to manage critical applications/solutions and specific markets and provide
support for open service marketplaces [117].

2.6 Summary

In this chapter, we have provided a brief background of the major areas related to
our work. These areas include Agent Oriented methodologies especially $i^*$, Tropos
and Formal Tropos (FT), Business Process Management, Business Process Modeling
Notations (BPMN), UML Activity Diagram, Risk and Risk Management, Service-
Oriented Architecture (SOA). In the next chapter, we provide a brief background of
the research approach of our work.
Chapter 3

Research Design

This chapter illustrates the research design for this thesis work. It discusses the research methodology we adopted for this research followed by the previous chapters where we discussed the research questions and the objectives of our research with a literature review of the relevant areas.

3.1 Research Methodology

A research methodology explains how a researcher should go about finding out the veracity of his or her beliefs [53]. The selection of a research methodology and its research instruments depends upon a number of factors, such as the nature of the research problem, the availability of resources and research traditions that are local to that institute or organization [11].

Different research methods could have been used to conduct this research. However, as can be seen from the nature of this research (e.g. development of a new framework), it is very clear that a qualitative constructive research approach would be the most suitable method of investigation and development. Therefore, the features of different
possible qualitative research methodological approaches and their usability in the con-
text are reviewed, compared and described below.

The objective of this research is to facilitate process design in a dynamic organisational
context by bridging the gap between organisational relationship (goals, dependencies,
rational among actors) and operational processes by providing a process mapping and
change management methodology.

This dissertation proposes a methodology to manage business process design in a dy-
namic organizational context primarily by using i* modeling framework and Business
Process Modeling Notation (BPMN). Agent-Oriented Conceptual Modeling (AOCM)
notations such as the i* framework have gained considerable currency in the recent
past. Such notations model organisational context and offer high-level social/anthropo-
omorphic abstractions (such as goals, tasks, softgoals and dependencies) as modeling
constructs. It has been argued that such notations help answer questions such as what
goals exist, how key actors depend on each other and what alternatives must be con-
sidered.

The technical focus of process modeling notations such as BPMN is especially suited
for applications in the description, execution and simulation of business processes but
is lacking in support for process redesign and improvement. These notations effectively
provide a view of the responsibilities and required communications between classes of
process participants, but do not provide a view of other social and intentional char-
acteristics including the goals of participants and their inter-dependencies. We argue
this gap can be minimised by using and correlating organisational models and process
models in a complimentary fashion.
Business processes represent the operational capabilities of an organisation. In order to ensure process continuity, the effective management of risk becomes an area of key concern. We argue there is a need for supporting risk identification with the use of higher-level organisational models and business process models.

In this research we have conducted a detailed analysis of the concept that flexibility and combination of notations is required in order to facilitate the maintenance of the models. We have developed a methodology to support combined use of notations (i* and BPMN) for modeling business process with a view to facilitate and support change at organisational and business process model. We have also presented a methodology to integrate risks in process models through a set of intuitive metrics for extracting measures of actor criticality, and vulnerability from organisational models. This research has been validated through a detailed case study involving a major government agency and through an experiment conducted among participants from industry and academia.

The nature of the research objective dominated the selection of the appropriate research methodology. This research has been conducted by the iterative application of build, review and adjusts research activities. The constructive research process [74], from the pragmatic philosophy of science, has been used to outline the constructed research methodology, which is called here qualitative empirical study. The qualitative empirical study in the constructive research process [70] permits the development and evaluation of novel artefacts presented in this thesis. The constructive research process may result in only a prototype or plan instead of a complete product as the
research outcome, which is suggested [70] as an acceptable outcome of the constructive research process. This research is more concerned with qualitative data and not mainly quantitative data. However, this research, during the data analysis, maps some of the qualitative data (i.e. empirical investigation survey questionnaire feedback and framework empirical assessment questionnaire feedback) to quantitative data for the purpose of readability, visibility and understanding. The assembled or combined qualitative research techniques for this research has the strength and ability to contribute to the evolution of a new theory (methodologies presented in this work) as new ideas emerge over a period of time [40].

Following phases and activities (3.1) were followed in order to develop the constrained development methodology and risk integration methodology presented in this work and its review and empirical evaluation.

- **Literature review:** This stage involved the review and analysis of the related
literature from printed sources (e.g. books, conference proceedings, journal papers, magazines etc.) as well as electronic sources (e.g. digital documents, online services). This process was carried out throughout the research project.

- **Research problem and scope:** The literature review was the starting point of this research and used as an input in defining the research problem and its scope at that point in time (i.e. 2006). The focus of the research has been iteratively adjusted based on the better understanding of the area of study and achieved results.

- **Research question:** The research problem can be best expressed in terms of its underlying scientific question(s) that need to be answered. The potential research question was initially identified based on the review of existing literature published in public domain at that point in time (i.e. 2006). The identified research question was tuned within the scope of this research and also referred throughout the research process.

- **Methodology emergence:** This phase involved the development and continuous re-adjustment of the methodological components in small iterations during 2006 and 2011. Based on the review of existing knowledge and published literature, the components of the methodologies have emerged.

- **Initial testing:** This phase involved the high-level testing of the framework, which was conducted by the means of a case study to receive feedback from industry and then moving in the right direction. The results of the test cases have been used to further refine and improve the methodologies.
• **Empirical experiment**: The developed methodology has been presented to thirty four experts from industry and academia for review to identify their needs and experienced-based opinion via a two folded approach - case study based experiment and a questionnaire-based survey. The design of the methodology has been refined based on the feedback received from these experts.

• **Reporting**: The components of the methodologies that emerged were reported and published in the relevant peer-reviewed international conferences and scientific journals as part of the ongoing process of research communication during the development of this thesis between 2006 and 2011. From time to time, the components have also been presented to practitioners via presentations and focus groups for informal feedback and guidance. These methodologies have been continuously adjusted based on the feedback received by peers from both the research community and the IT industry.

### 3.2 Feedback from Industry and Researchers:

Case Study:

We developed a case study based on a large state government body in Australia where we applied our methodology to put in practice. The organisation structure is a complex array of directorates and business units with varying needs. It required an enterprise software solution, which can accommodate its strict security requirements while supporting standardised and decentralised processes for time tracking, project management, resource management, financial management and reporting.

Given the organisational size and complexity of the department it is quite normal to
have a varied and large range of business requirements models. The large scope of the business units leads to greater complexity. It was decided that the constrained development methods presented in this thesis will be used in order to facilitate the maintenance of the models in lieu of changes in the context of their usage over the course of their lifecycle. For initial requirements engineering exercise $i^*$ organisational modeling technique was used. These models represented the scope, organisational actors/roles and their dependencies and intentional rationale. We then mapped the $i^*$ organisational models into operational BPMN models and vice versa (when required) using our constrained development methodology. Our constrained development methodology was were adjusted based on the feedback received.

**Empirical Experiment:**

We conducted an experiment with thirty four participants from industry and academia on the constrained development methodology. The purpose of this experiment is to further demonstrate that this methodology was tested and assessed by the experts in organisational and process modeling area. This experiment provided us an insight opportunity to see the actual perception and effectiveness of the constrained development methodology.

The evaluation approach taken in this experiment is twofold. One part of this study is based on a case study where the participants are asked to develop a business process model given a high level conceptual model and vice versa. The case study is an empirical research method useful in realistic environments where strict control over variables is not possible. The other part of this study is based on questionnaires. The structure of this study preserves the opportunity for rigorous support of an argument.
in the absence of formal controls. We use the experiment methodology mentioned in [151].

Questionnaires: We asked the participants to provide their feedback on the methodology. The questions were:

BQ1. How do you rate the overall quality of the methodology?

1 = Failed to meet my expectations
2 = Neutral
3 = Met my expectation
4 = Strongly met my expectation
5 = Exceeded my expectation

BQ2. What are your most important ‘take away’ ideas from the methodology?

BQ3. Do you think this methodology is ready to use at organisational and process level?

BQ4. What are the most positive sides of this methodology?

BQ5. What are the most negative sides of this methodology?

BQ6. Do you think the $i^*$ and BPMN models will be consistent with each other if you use this methodology? Please provide details.

BQ7. Do you have any other comments on the methodology?

Participants: 34, Invitation sent to take part in the experiment via email and phone. All the interested participants were included in the experiment.

Participant Type: Consultants, Managers, Designers, Architects, Research Scientist, Software Developers

Participants Countries: Australia, China, Thailand, UK, Germany, India, Pakistan,
Bangladesh. There was no explicit need for involving practitioners from overseas - the experiment was held in Sydney and it is a multicultural city where people from different countries work in the industries. The proposed integration model is independent of the cultural background of the business process consultants.

*Participants Organisation Type:* Public Sector (8), Consulting (11), University (15)

Feedback, from the industry, researchers, and journal paper reviews and conferences, had been used to build-review-readjust the components of the methodology and also to set the directions for future work. The final, reviewed and improved version of the methodology is reported in this thesis.

### 3.3 Summary

In this chapter, we have provided a brief background of the research approach of our work. This includes describing all the phases we followed. In the next chapter, we introduce our constrained development methodology to support business process model lifecycle.
Chapter 4

A Combined Approach for Supporting the Business Process Model Lifecycle

Business processes evolve throughout their lifecycle of change. Business Process Modeling (BPM) notations such as BPMN are used to effectively conceptualize and communicate important process characteristics to relevant stakeholders. Agent-oriented conceptual modeling notations, such as i*, effectively capture and communicate organisational context. In this chapter we argue that the management of change throughout the business process model lifecycle can be more effectively supported by combining notations. In particular, we identify two potential sources of process change, one occurring within the organisational context and the other within the operational context. As such the focus in this chapter is on the co-evolution of operational (BPMN) and organisational (i*) models. Our intent is to provide a way of expressing changes, which arise in one model, effectively in the other model. We present constrained development methodologies capable of guiding an analyst when reflecting changes from an i* model to a BPMN model and vice-versa.
4.1 Introduction

Business process models play a key role in both organisational management [129] [58] and enterprise information systems development [39]. Many notations developed for the task of modeling business processes, have their own focus of application and appropriate audience [12] [77] [92] [76] [155]. High-level conceptual models provide an understanding of an organisation from an intentional and social perspective [154] for reasoning support during redesign [154]. In comparison, lower-level technical models are especially suited for applications in the description, execution and simulation of business processes [155].

Business process development should be based on principled high-level models of the enterprise and the business context. Commonly, processes are formulated in an ad-hoc fashion without reference to these high-level models. Some of the most prominent modeling notations enlisted are focused towards technically-oriented data, and process modeling notations such as ER, Data-Flow, Systems Flowcharting and UML and workflow modeling [38]. In this work, we offer constrained development methodologies to guide development of process models from higher-level conceptual models. This supports life-cycle management in the following sense: when changes occur to the high-level model, these can be reflected in the process model, and vice versa. In this chapter, following sections provide a background to business process modeling with an overview of our chosen notations and then illustrate concepts/methods provided in our methodologies (with examples).
4.2 Background

The notations used for modeling business processes have been categorized in many works, based on their conceptual features [12] [77] [92] [76] [155]. The common principle recognized in all analysis is that some notations are more suited toward specific audiences (i.e. with either technical/non-technical backgrounds) or applications (i.e. possibly for description, re-design or execution) throughout the business process lifecycle. Many notations focus on specific aspects, with limited relation/traceability to other important business process aspects. This has brought about the need for an enterprise view [92] to support the development and maintenance of rich models that provide an enhanced ability to conceptualize, communicate and understand business processes, and their context of operation.

In related work, some preliminary ideas in [30] have been proposed for developing a BPMN model given the existence, and agreement to, an $i^*$ model of the process. Six steps are provided for mapping between constructs, with no consideration for reflecting change and consistency made. Also, an approach for deriving a BPMN model from a business model is proposed in [4], achieved through the intermediate translation of the business model into an activity dependency model that can then be translated into a business process model. In this work, we provide a simpler approach aimed at reducing added complexity and/or misinterpretations during modeling. Furthermore, much work has been completed on supporting guided translation and coevolution of $i^*$ into various other behavioral modeling notations and languages [82] [35][83]. The primary aim in these approaches is to further develop detailed design artifacts that can lead onto implemented systems, or directly be used in the configuration of agent-based...
systems. However, our primary focus is on modeling lifecycle support during BPM projects whereby the concern is for the development and/or assessment of detailed business process designs. We take the following approach to lifecycle management: when changes to a business process model (i.e. BPMN [145]) occur, these changes must ensure some notion of consistency with a higher-level enterprise model, and vice versa. In this instance, an $i^*$ model [154].

### 4.2.1 Agent-Oriented Conceptual Modeling (AOCM) with $i^*$

$i^*$ supports modeling rich organisational contexts by offering high-level social/anthropomorphic abstractions (such as goals, tasks, soft goals and dependencies) as modeling constructs for reasoning support during business process redesign [154] [76]. Figure 4.1 represents a simple $i^*$ Meeting Scheduling model. The central concept in $i^*$ is that of intentional actor. These can be seen in the Meeting Scheduling model as nodes representing the intentional/social relationships between three (3) actors required to schedule a meeting: a *Meeting Initiator* (MI); *Meeting Scheduler* (MS); and, *Meeting Participant* (MP).
The \( i^* \) framework consists of two modeling components [154] Strategic Dependency (SD), and Strategic Rationale (SR) models. The SD model consists of a set of nodes and links. Each node represents an actor, and each link between the two actors indicates that one actor depends on the other for something in order that the former may attain some goal. The depending actor is known as \textit{dependee}, while the actor depended upon is known as the \textit{dependee}. Dependancies may involve \textit{goals} to be achieved (e.g. \textit{MeetingBeScheduled}), \textit{tasks} to be performed (e.g. \textit{EnterAvailDates}), \textit{resources} to be furnished (e.g. \textit{Agreement}), or \textit{softgoals} (optimization objectives or preferences) to be satisficed (e.g. \textit{MaximizeAttendance}).

The SR model further represents internal motivations and capabilities (i.e. processes
or routines) accessible to specific actors that provide illustration of how dependencies can be met. In $i^*$, a routine [154] specifies an intended course of action an actor may pursue given a set of alternatives. These elements and their relationships represent the strategic requirements of a business process when invoked in a specific context. For example, to ScheduleMeeting (illustrated in Figure 4.1 with its Scope) that includes three sub-tasks and six dependencies with two additional actors. Tasks in $i^*$ may be primitively workable whereby the actor responsible for the element believes that it can achieve its requirements at execution time i.e. it is sufficiently reduced during decomposition. In comparison to BPMN however, a primitively workable element may still be represented as a sub-process as the term does not imply a primitively executable action (i.e. application of analyst / designer discretion). Furthermore, for a routine to be workable, all involved actors must be committed to satisfying their dependencies [154].

The Tropos project [52] aims to provide methodological support for advancing the $i^*$ framework further towards architectural and detailed design where dynamic / behavioral aspects are of importance. Specifically, Formal Tropos (FT) see [50], is a part of the Tropos project that provides a specification language for modeling dynamic aspects of an $i^*$ model via formal annotation of Creation and Fulfillment conditions. These conditions are specified using first-order typed linear temporal logic and prescribe the constraints on an elements lifecycle. In this work, we take the same approach to annotation (with the use of fulfillment conditions annotated to $i^*$ models). In comparison, our work is illustrated via informal annotations.
4.2.2 Business Process Modeling with BPMN

The Business Process Modeling Notation (BPMN), developed by the Business Process Management Initiative (BPML.org) [145] is primarily a technically-oriented business process modeling notation that supports the assignment of activity execution control to entities within an organisation via swim-lanes. BPMN has the capability to map directly to executable process languages including XPDL [44] and BPEL [145] [112]. Furthermore, an analysis of BPMN [10] also stated its high maturity in representing concepts required for modeling business process, apart from some limitations in terms of representing state, and the possible ambiguity of the swim-lane concept.
4.2. Background

Figure 4.2: A BPMN ‘Patient Treatment Process Model

Figure 4.2 represents a simple BPMN Patient Treatment process. Processes are represented in BPMN using flow nodes: events (circles), activities (rounded boxes), and decisions (diamonds); connecting objects: control flow links (unbroken directed lines), and message flow links (broken directed lines); and swim-lanes: pools (high-level rectangular container), and lanes partitioning pools. These concepts are further discussed in [145].
4.3 Constrained Development Methodologies

We propose constrained development methodologies to guide the derivation or maintenance of one type of model given the availability of the other. The development is supported with the introduction of two concepts: **effect annotations** and **fulfillment conditions** (i.e. as in [50]).

An *effect* is broadly defined as the result (i.e. product or outcome) of an activity being executed by some cause or agent. An effect annotation is a specific statement relating to the outcome of an activity, associated to a state altering construct in a given model. Effects are annotated to atomic tasks/activities or subprocesses within an actors lane. The execution of a number of activities in succession results in a *cumulative* effect that includes the specific effects of each activity in the sequence. We also note the fact that certain effects can undo prior effects (i.e. in the case of compensatory activities). Effect annotations may possibly be formalized using the formal layers of some currently well-developed Goal-Oriented Requirements Engineering (GORE) methodologies [85] [50], however, we only state their applicability in this work, and aim towards possible integration in the future.

**Fulfillment conditions** are annotated to tasks and goals assigned to actors in an SR diagram, and dependencies (i.e. not including soft-goals as these are used during assessment of alternatives and describe non-functional properties to be addressed) in an $i^*$ model. A *fulfillment condition* [50] is a statement specifying the required conditions realized upon completion of a given task, goal or dependency. Fulfillment conditions recognize the required effects on a business process model. For example, a fulfillment...
condition for a task dependency to EnterADateRange, may be the DateRangeCommunicated effect (subsequently required by the task assigned to a dependee actor).

### 4.3.1 Annotation and Propagation

Tasks, goals and dependencies are annotated with fulfillment conditions in an $i^*$ model. Additionally, the tasks assigned to participants in a BPMN model are annotated with effects for assessment against fulfillment conditions.

Tasks associated to dependencies on the dependee side may require additional effects when related to a BPMN model. That is, the fulfillment conditions for a dependency may not be explicitly stated against the tasks. For example, the fulfillment condition for ProposedDateProvided (i.e. annotated to the ProposedDate resource dependency in Figure 4.1) will be propagated to the ObtainAvailDate task. This should occur during annotation, whenever a fulfillment condition is annotated to a resource, goal or task dependency.

Effect annotations in BPMN models are propagated via trajectories. A trajectory is a sequential execution of activities terminating at an end state that represents the operational goal of the process. Control flow links between events, activities, and gateways within a BPMN model indicate the flow of trajectories. Effects within a process are accumulated during forward traversal through a trajectory. This accumulation ensures that any compensatory activities, that may undo effects, are also taken into account during traversal.
## Annotating the Meeting Scheduling Model (Figures 4.1 and 4.3)

Table 4.1: Annotation of Fulfillment Conditions to Respective Tasks/Dependancies

<table>
<thead>
<tr>
<th>Task/Dependency (Figure 4.1)</th>
<th>Fulfillment Conditions</th>
<th>Task Annotation (Post Development Figure 4.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI: Scheduler Schedules Meeting</td>
<td>DateRangeEnteredIntoScheduler; DateRangeCommunicated-ToScheduler</td>
<td>1; 1;</td>
</tr>
<tr>
<td>MS: Schedule Meeting</td>
<td>AgreedDateKnownToInitiator</td>
<td>4</td>
</tr>
<tr>
<td>MS: Obtain Available Dates</td>
<td>ProposedDateProvided; AvailableDatesObtained; AvailableDatesStored; AvailableDatesValidated</td>
<td>2 (message); 2; 2</td>
</tr>
<tr>
<td>MS: Obtain Agreement</td>
<td>AgreementObtained; AgreementRecorded</td>
<td>4; 4</td>
</tr>
<tr>
<td>MS: Merge Available Dates</td>
<td>AvailableDatesMerged</td>
<td>3</td>
</tr>
<tr>
<td>P: AgreeToDate</td>
<td>DateAgreedTo; AgreementProvided</td>
<td>6; 6 (message)</td>
</tr>
<tr>
<td>P: Find Agreeable Date Using Scheduler</td>
<td>AvailableDatesEnteredIntoScheduler; AgreeableDateFoundUsingScheduler</td>
<td>5;6</td>
</tr>
<tr>
<td>MS-Dep to MI: Enter Date Range</td>
<td>DateRangeCommunicatedToScheduler</td>
<td>1</td>
</tr>
<tr>
<td>MI-Dep to MS: Meeting Be Scheduled</td>
<td>AgreedDateKnownToInitiator</td>
<td>4</td>
</tr>
<tr>
<td>MS-Dep to P: Enter Available Dates</td>
<td>AvailableDatesEnteredIntoScheduler</td>
<td>5</td>
</tr>
<tr>
<td>P-Dep to MS: Proposed Date</td>
<td>ProposedDateProvided</td>
<td>2</td>
</tr>
<tr>
<td>MS-Dep to P: Agreement</td>
<td>AgreementProvided</td>
<td>6 (message)</td>
</tr>
</tbody>
</table>
4.3.2 Scope Projection

In order to evaluate consistency between the two notations, we provide some rules for projecting the scope of the $i^*$ model. In the current case, $i^*$ models are likely to represent a broader scope in comparison to a specific BPMN model as they are applied to capture the greater organisational context. Scope projection is based on an identification of the business process (represented in BPMN) as a routine assigned to an actor in an $i^*$ model.

- Rule 1: The root node of the routine traceable to the process in consideration and all tasks in its first level of decomposition from are to be within scope.

- Rule 2: All dependencies that are associated to a task within the scope of the routine, where the actor in control of the routine (initiator) is the dependee are within the scope of the process; as well as the tasks assigned to dependee actors.

- Rule 3: All dependencies that are associated to a task within the scope of the routine, where the initiator is the dependee are within the scope of the process iff the task assigned to the dependee is part of some decomposition of a task in the scope of the process as per Rule 2; as well as the tasks assigned to the dependee actors.

4.3.3 Consistency Evaluation

We introduce consistency rules to provide a mechanism for ensuring consistency between $i^*$ and BPMN models (developed with consideration to [50]).

- Rule 1: Every actor in an $i^*$ model required as a participant in the routine (traceable to the business process) and any of their tasks must be represented in the BPMN model (and vice versa), assessed via application of scope projection rules.
4.3. Constrained Development Methodologies

- Rule 2: There must exist a trajectory in the process model, whereby the operational objective (as encoded in the accumulated fulfillment conditions of traceable tasks) of the routine is achieved, and the sequence of activities is consistent with the requirements specified in the routine as further outlined below:

- Rule 2.1: The accumulated effect of all tasks and goals traceable to the routine must achieve accumulated routine fulfillment conditions during forward traversal of at least one trajectory in the process model; AND,

- Rule 2.2: The fulfillment of a task on the depender side of a dependency must not be realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory.

4.3.4 Constrained Development of a Business Process Model given a High-Level Conceptual Model

These steps are based on the aforementioned consistency rules aimed towards providing analyst guidance during initial model development.

- Step 1: Identify internal and external actors in $i^*$ diagram.

- Step 2: Map elements to equivalent constructs within the BPMN model. See substeps below.

- Step 2.1: Map Participants. The greater organisation for which the $i^*$ model is represented is signified as a pool in BPMN. Any external participants are also represented as pools. Internal organisational actors are represented as lanes within the organisational pool.

- Step 2.2: Map Activities. Tasks within $i^*$ are represented as either sub-processes or atomic activities within BPMN assigned to actors within pools and lanes.
4.3. Constrained Development Methodologies

- Step 3: Sequence required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions. Tasks placed within each pool or lane are now sequenced to conform to routine requirements by taking Consistency Rule 2 (see: Section 3.3) into consideration. This requires that tasks be sequenced using control flow links in a manner that results in a trajectory satisfying fulfillment conditions on an $i^*$ model. Control flow links are used to indicate realization of dependencies between actors within the same organisation. In order to realize dependencies between organisational boundaries, a message flow link is used to represent the dependency going from the depender lane to the dependee lane. This may require single/multiple messages between tasks derived via analysis of fulfillment conditions.

- Step 4: Elaborate on sub-processes. The choice to introduce tasks or subprocesses into the BPMN diagram for specific tasks in the $i^*$ model is made in Step 2.2. The analyst can develop each sub-process guided by the list of required fulfillment conditions annotated to the $i^*$ task that the sub-process realizes.

Figure 4.3 illustrates the application of the constrained development methodology in the context of the Meeting Scheduling model represented in Figure 4.1, with annotations applied in Table 1. Much of the detail has been omitted for brevity. The following section describes a possible change requirement and its reflection within an $i^*$ model for further analysis.
4.3. Constrained Development Methodologies

Figure 4.3: BPMN Process Model derived using the Constrained Development Methodology

**Reflecting Changes in an i* Model to an associated BPMN Model**

The scope projection techniques are used to assess whether a change in an i* model will impact a BPMN model. These guidelines aim to support the reflection of change between i* and BPMN models for the specific instances of impacting change outlined below.

- **Step 1:** For each classification outlined below apply associated changes.

- **Addition of an actor.** If a new actor has been added to the i* model, a swimlane (i.e. for an internal actor) or pool (i.e. for an external actor) will need to be placed on the process model. Additionally, new dependencies must exist between the actor and existing actors (described below). These dependencies will be included for all new
actors where the dependency is related to the routine and actor is the dependee. However, where the actor is the depender they will only be included if linked to a task in an existing dependency graph (see Scope Projection rules).

- **Addition of a goal/task/resource dependency.** If a new dependency has been added to the \( i^* \) model, then this may require the addition of new activities/subprocesses and message flow links within the BPMN model (as described below).

- **Addition of a goal or task.** The addition of a goal or task will require the addition of a task within the BPMN model. The addition of these tasks must be scoped to their respective actors, and any dependencies must be realized via message-flow links where one of the actors is external to the organisation.

- **Step 2:** Re-apply consistency rules to both models to assess whether consistency has been maintained.

Consider the following example applied to the Meeting Scheduling example in Figure 4.1 (\( i^* \)) and Figure 4.3 (BPMN). A new requirement within in the form of a task dependency between the Meeting Initiator (i.e. the dependee) and the Meeting Scheduler (i.e. the depender) to ProvideParticipantPrioritization. Participant prioritization means that the Meeting Initiator must now prioritize the current list of participants in order for the Meeting Scheduler to MergeAvailableDates and FindAnAgreeableSlot effectively.

Given the application of our approach for guiding an analyst’s decision, it can be inferred that the effect for ParticipantPrioritizationProvided will propagate within the \( i^* \) model as a fulfillment condition on the SchedulerSchedulesMeetingTask. Furthermore, given Consistency Rule 3, requires that ParticipantPrioritizationProvided occurs
prior to the fulfillment of the *MergeAvailableDates* fulfillment conditions. This information can then be used to highlight the scope of change within the BPMN model to a point within a trajectory prior to the required effects of *MergeAvailableDates*, where an activity controlled by the initiator is able to realize the required effect.

### 4.4 Constrained Development of a High-level Conceptual Model given a Business Process Model

The following steps provide systematic guidance for developing an *i* model given an already existing process model. Figure 4.4, illustrates the constrained development of the *Patient Treatment* BPMN model in Figure 4.2.

- **Step 1:** Map elements to equivalent constructs within the *i* model.
  - **Step 1.1:** Map Participants. Both pools and lanes in a BPMN model represent actors in an *i* model. These can be directly translated into the model.
  - **Step 1.2:** Map Activities. Represent activities and sub-processes as primitively workable tasks assigned to actors in *i*.

- **Step 2.** Apply intentional reasoning.
  - **Step 2.1:** Query the Intention of Tasks. Intentional reasoning is applied to identify higher-level intentional elements and dependencies by querying the intention of tasks. This step aims to guide the further understanding and representation of an actors motivations.
  - **Step 2.2:** Query the Intention of Flow-Links. Analyze control and message flow between actor boundaries to identify goal, task and resource dependencies. These types
of links can be used as a primary heuristic for identifying possible dependencies between actors.

- **Step 3**: Identify soft-goal dependencies in the \( i^* \) model. The representation of soft-goals (including dependencies) are not in the scope of the BPMN notation.

### 4.4.1 Reflecting Changes in a BPMN Model to an associated \( i^* \) Model

- **Step 1**: For each classification of change, apply the following changes.

  - **Addition of a swimlane or pool**. If a swimlane or pool is added, then a new actor will be required within the \( i^* \) model. This will include the addition of new dependencies and tasks within the \( i^* \) model. A primary heuristic for identifying dependencies includes message flow links and control flow links between pools and lanes (message flow indicates a resource dependency for some information).

  - **Addition a task to an existing swimlane or pool**. If a new task is added to a swimlane or pool, this will require a task to be decomposed from the root node of the routine traceable to the current process.

- **Step 2**: Re-apply consistency rules assess whether maintenance.
Consider now a scenario where the business process model is modified to improve the performance of the \textit{IssuePrescription} task which has been identified to be a major operational bottleneck. The task is improved by including a task before hand which checks the patients previous medical history to identify previous prescriptions for the patient for similar illnesses (e.g. common flu). We name the task \textit{CheckPatientMedicalHistory}. Furthermore, the client is now encouraged to provide information on his medical background, which we represent as a task named \textit{ProvideMedicalHistoryInformation}. We now proceed to add an additional task within the bounds of the Doctor agent and an additional task within the bounds of the Patient agent.

As in the previous case we use intentional reasoning to identify that the added task, within the Doctor agent, contributes to the higher level task of \textit{TreatingPatients}. We
apply the same technique to justify the placement of the \textit{ProvideMedicalHistoryInformation} task as a decomposition task under the \textit{RequestMedicine} task.

The added message flow in the BPMN diagram is represented as a resource dependency between the Patient and the Doctor, where the Doctor requires the Patient to provide his previous medical history. We also introduce the soft-goal between the Patient and the Doctor, titled \textit{TimelyDrugPrescription}, indicating the fact that the Doctor will try to improve the time required to prescribe medication to the Patient.

4.5 Summary

In this work, we have illustrated an approach for supporting the lifecycle of business process models with the complementary use of $i^*$ - a well developed notation for modeling organisational contexts, and BPMN a newly developed notation for modeling business processes. The approach for reflecting changes in organisational context to changes in the design of business processes provides an effective mechanism for aligning business processes with organisational objectives. Similarly, operational improvements can be mapped back to organisational objectives to facilitate analysis and ensure no conflicts exist with existing objectives. Although these steps are preliminary we believe their systematic nature makes them available for automation in all phases, and are pursuing this task, through the development of a software tool, along with further refinement of the approach. In the next chapter we illustrate how business process and organisational models can be correlated to manage change.
Chapter 5

Correlating Business Process and Organisational Models to Manage Change

Business Process Management (BPM) provides the methods, tools and modeling notations to support a process-centric organisational view and management capability. As organisations grow in size and complexity, process improvement initiatives may involve change that has direct / significant impact across an organisation. Thus, we provide methods and extensions to existing process modeling notations to analyse change against high-level models of the organisation. Our approach permits improved analysis against higher-level organisational structures, motivations, inter-dependencies and capabilities that should be ideally considered as primary requirements during process design. Additionally, the organisational model becomes the ‘scaffolding’ with which to construct effective process architectures and management portfolios. This chapter discusses our approach in the context of two modeling notations - the i* framework as an organisational modeling notation, and the BPMN notation for business process modeling.
5.1 Introduction

A Business Process can be described as a set of dynamically coordinated activities, controlled by a number of socially dependant participants, aimed towards the achievement of a specific operational objective [58] [129]. Business Process Management is a re-emerging discipline, aimed towards supporting the effective and automated [129] management of business processes within an organisation via specialized tools and methods. Business Process Management promotes that a clear understanding through the explicit modeling of the processes underlying an organisation is required to support effective organisational management / improvement practices [57][59].

Business processes evolve throughout their ‘lifecycle of management’ [129], brought

![The Business Process Lifecycle (BPLC)](image)

Figure 5.1: The ‘Business Process Lifecycle (BPLC)’

on by change within their environment of operation. Figure 5.1 illustrates the major
phases of the business process lifecycle. The lifecycle can be partitioned into three primary perspectives - organisational, conceptual, and technical. In addition, four categories of feedback and improvement mechanisms are represented - adapt and control, re-deployment, redesign, and rediscover.

Business Process modeling (BPM) aims to facilitate effective process change via communication. As illustrated in Figure 5.1, the conceptual process perspective illustrates the lifecycle of a ‘business process model’. The model’s primary purpose is to capture, communicate, and maintain the current shared understanding of the process between all stakeholders to support associated phases of the entire business process management lifecycle.

In [59], three broad categories for business process change are defined - (1) business process [re]design, (2) business process improvement, and (3) business process automation. Both process redesign and improvement are a result of an analysis phase during BPM whereby process monitoring information is assessed against some performance criteria. This results in redesign (see Figure 5.1), required in order to better align business processes. In this case explicit evolutionary changes are made to the process design for all subsequent process instances to follow. Harmon [59] also makes the distinction that improvement is driven by the operational motivations/constraints of line management who oversee process execution and redesign is driven by changes to the higher-level strategic/organisational motivations of the enterprise (as illustrated in Figure 5.8). On the other hand, process automation takes existing process designs and aims to improve their effectiveness and efficiency though automation (i.e. via re-deployment - Figure 5.1) in machinery or software systems.
In comparison, Aalst [138] defines two categories of process change: *structural* (evolutionary); and, *individual* (ad-hoc). Structural changes apply to the design evolution of a business process, as a result of some improvement, redesign or re-engineering effort (i.e. as discussed above and in [59]). Ad-hoc change on the other hand, refers to run-time adaptation, required given some unforeseen or isolated circumstance[s] (e.g. ignoring an insurance check requirement for an emergency patient). Ad-hoc change may occur prior to (i.e. pre-planned pre-enactment), or post initiation of a business process instance (i.e. on-the-fly at run-time).

Our argument is that we need to base business process design on higher-level organizational motivations. Subsequently, the development of business process models should reference principled high-level contextual models of the enterprise that illustrate its motivations, resources, social context, and internal/external inter-dependencies. Any purposeful changes made to business process models must be reflected within the high-level model for analysis against the greater context of the enterprise. To support an analyst in achieving this task, we offer methods to assess change between organisational and business process models. These methods support change management in the following sense: when changes occur to the high-level model these can be reflected in related business process models for eventual deployment, and when changes are proposed within a business process model these may be reflected at a higher level to improve analysis and decision making. In particular, we employ the use of an agent/network-oriented organisational modeling notation - \(i^*\) [154], and a standardized, operational and executable process modeling notation - BPMN [145].

We discuss the chosen notations further in the sub-sections below. We then describe
our approach in detail, including our proposed process modeling extensions and analysis techniques. Finally, we illustrate our approach with examples, and conclude in the final sections.

5.1.1 Organisational Modeling (with $i^*$)

$i^*$ [154] is an organisational modeling framework that supports a representation of the social, intentional, and strategic aspects of organisational structures. Specifically, goal, soft-goal, task, and resource dependencies can be modelled to help in understanding important strategic relationships between actors in an organisational context. From this perspective, the actors motivations, level of commitment and vulnerability can be effectively portrayed to support enhanced analysis and redesign capabilities.

Figure 5.2 represents a simple $i^*$ Transport organisation model where (3) actors are represented in the context of ‘Package Routing’: a Sort Facility (SF); Bond Department (BD); and, Regulatory Agency (RA). In $i^*$ actors are represented as circular nodes with links that illustrate their dependencies with other actors.

$i^*$ provides two perspectives with which to view an organisation: a Strategic Dependency (SD) model providing a high-level view of actors and their dependencies; and, a Strategic Rationale (SR) model illustrating each actors underlying motivations and capabilities. The SR model facilitates an understanding of why an actor delegates, or is delegated, responsibilities in some organisational context.

When interpreting a dependency, the ‘D’ annotated to a link directs the dependency relationship from a depender (e.g. the ‘Regulatory Agency’) for a dependum (e.g. ‘Bonded[Packages]’) to a dependee (e.g. the ‘Bond Department’). Each dependency may require either: a goal to be achieved (e.g. ‘Bonded[Packages]’); a soft-goal to
be satisfied (e.g. ‘Timely Release[Packages]’); a task to be completed (e.g. ‘Provide[Packages]’); or, a resource to be provided (e.g. ‘Package Details’). An actors internal motivations and capabilities in an SR model, are represented as an AND/OR goal graph (as in Figure 5.2). Greater detail is available in an SR model concerning the source and destination task of dependencies between actors.

5.1.2 Business Process Modeling (with BPMN)

The Business Process modeling Notation (BPMN), developed by the Business Process Management Initiative (BPML.org) [145] is primarily a technically-oriented business
process modeling notation that supports the assignment of activity execution control to entities within an organisation via ‘swim-lanes’. BPMN has the capability to map directly to executable process languages including XPDL [44] and BPEL [145][112]. Furthermore, an analysis of BPMN [10] also stated its high maturity in representing concepts required for modeling business process, apart from some limitations in terms of representing state, and the possible ambiguity of the swim-lane concept.

Figure 5.3: A BPMN ‘Package Routing ’process model

Figure 5.3 represents a Package Routing process in BPMN. Processes are represented in BPMN using flow nodes: events (circles), activities (rounded boxes), and decisions (diamonds); connecting objects: control flow links (unbroken directed lines), and message flow links (broken directed lines); and swim-lanes: pools (high-level rectangular container), and lanes partitioning pools. These concepts are further discussed in [145].
5.2 Using Model Annotations To Assess Relationships between Business Process and Organisational Models

Activities and Sub-Processes (i.e. represented in BPMN as rounded boxes), and Tasks (i.e. represented in an i* model) signify a number of possible state transitions. The labelling of an activity (e.g. ‘Register New Customer’) generalizes one or more normal/abnormal outcomes (e.g. ‘A new customer registered’, or ‘An attempted registration has been refused’). As such, most process and organisational models do not adequately represent enough information for effective analysis. They are too ‘high-level’, and do not convey a usable understanding of achievable states. In order to improve the description and clarity in process models, we propose to augment state altering nodes (i.e. activities and sub-processes) with semantic effect annotations. This parsimonious extension to the BPMN notation permits modellers to annotate activities in a process model with richer specification of immediate effects.

5.2.1 Effect Annotation

An effect is the result (i.e. product or outcome) of an activity being executed by some cause or agent. Effects are commonly referred to as a ‘post-conditions’. An effect annotation relates a specific result or outcome to an activity on a business process model. It explicitly states a result of the activity in its domain of execution. Effect annotations are formed in the indicative mood, or as facts (e.g. ‘A courier has provided an unsigned contract to a partner organisation.’). A causal relationship exists between a process activity and an effect. An activity can cause many effects, and an effect can be caused by many activities. Effects can be viewed as both: normative - as they
state the required outcomes; and, *descriptive* - in that they describe the normal, and predicted, subset of all possible outcomes (i.e. actual outcomes may vary at run-time).

Effect annotations can be *formal* (for instance, in first order logic, possibly augmented with temporal operators), or *informal* (such as simple English). We recommend in both cases, that *informal* annotations of effect be applied as a first pass to ensure a rich expression, and for ease of communication. Many of the examples we use in this chapter rely on informal natural language effect annotations. Ideally, and for analysis purposes, it would be of benefit if the annotations were stated formally as this permit us to use automated reasoners, while informal annotations oblige analysts to check for consistency and completeness between effects (as discussed below). A middle-ground can be worked when effect annotations are formed using some predefined form that can be translated automatically into a formal representation. For example, via the use of Controlled Natural Languages (CNL) with grammar and vocabulary restrictions such as in [127], and [131].

When an analyst is annotating existing process models, the conditions labelling control-flows leaving a decision gateway may provide some understanding of the effect of a downstream activity. Effects may also refer to assumptions on how the immediate state of an observer (i.e. during process inter-operation across pools) may change as a result of some information / work item transfer. When implemented within a tool, effects may be viewed on a business process model graphically, or added to meta-information of activities or sub-processes.

An annotated BPMN model, for the purposes of this chapter, is one in which every task
(atomic, loop, compensatory or multi-instance) and every sub-process has been annotated with descriptions of its immediate effects. We will now describe a methodology for accumulating these effect annotations to obtain a cumulative effect annotation for a complete process. We will assume that informal annotations are available in describing this methodology.

5.2.2 Effect Accumulation

Effect annotations are statements concerning the immediate effect of a particular task. In order to identify the cumulative effect of a complete process, we combine the effect of tasks executed in a pair-wise manner. This provides the analyst with a cumulative effect as the accumulation is progressed through traversal of the activities in the process. This accumulation equates to stating that all (or some, as will be discussed below) of the prior effects ‘AND’ the immediate effects of the task to receive the cumulative effect, are true at the cumulative point in the process. That is, when given an ordered pair of tasks with effect annotations, the accumulation determines the cumulative effect after both tasks have been executed in contiguous sequence. Pair-wise effect accumulation only occurs across control-flow links between tasks within participant lanes.

Take Figure 5.4 as an example. Let a task $T_1$ be the preceding task in the sequence (i.e. ‘Scan Package’), and $T_2$ be the succeeding task (i.e. ‘Receive Package’). The cumulative effect of the process at $T_2$ results from combining its immediate effects with the cumulative effect of its preceding task.
5.2. Using Model Annotations To Assess Relationships between Business Process and Organisational Models

During process enactment, the effects of a task may override the effects of a preceding task. For example, say an effect in $T_1$ states ‘Some packages have not been screened’, and an effect in $T_2$ states ‘All packages have been screened’. In this case, the effect of $T_2$ will override the effect of $T_1$. The effects that require an override can be identified by searching for any contradictions in the effects to be accumulated in the prior task in the sequence, given the immediate effects of the current task. This will result in the inclusion of as many of the effects in the cumulative effect of the prior task $T_1$ that are not contradictory, when accumulated to the succeeding task $T_2$. The process continues without modification over splits. Joins require special consideration.

These result in alternate effect scenarios when XOR-joins or OR-joins have been used (as will be described below). In the following, we describe the methods to be followed in the case of 2-way joins only, for brevity. These methods generalize in a straightforward manner for $n$-way joins.

Figure 5.5 represents part of a process that includes an AND-join where $T_1 = \text{Deliver Package}$, $T_2 = \text{Accept Payment}$, and $T_3 = \text{Finalize Delivery}$. Firstly we accumulate the immediate effects of $T_3$ with the cumulative effects of both $T_1$ and $T_2$. The immediate effects of $T_3$ are combined with all alternate effect scenarios that have been
accumulated on either $T_1$ or $T_2$. This yields the cumulative effects $T_{1+3}$ and $T_{2+3}$. This accumulation includes the analysis of any contradictions as previously discussed, which takes any overrides into consideration. We then combine $T_{1+3}$ and $T_{2+3}$ to signify the cumulative effect at $T_3$. In this case, any effect scenarios accumulated on either $T_1$ or $T_2$ remain, with additional effects as per the immediate effects of $T_3$. Note that we do not consider the possibility of a pair of effect scenarios having any contradictions, since this would only happen in the case of intrinsically and obviously erroneously constructed process models.

Figure 5.5: Pair-wise accumulation to a task during an AND-join

In Figure 5.6, an XOR-join is represented, where either task $T_1 = ‘Scan Package’, or $T_2 = ‘Release Package’ have executed and task $T_3 = ‘Sort Package’ has completed during enactment. In this case, two effect scenarios are to be generated at $T_3$ with the cumulative effects of $T_1$ and $T_2$ respectively. Firstly we accumulate the immediate effects of $T_3$ with the cumulative effects of both $T_1$ and $T_2$. The immediate effects of $T_3$ are combined with all alternate effect scenarios that have been accumulated on either $T_1$ or $T_2$ to yield the cumulative effects $T_{1+3}$ and $T_{2+3}$. Any overrides are applied (as
5.2. Using Model Annotations To Assess Relationships between Business Process and Organisational Models

previously discussed). \(T_{1+3}\) and \(T_{2+3}\) then remain in the cumulative effect at \(T_3\) as alternate effect scenarios.

\[\text{EffectScenario}_{T_1} = \text{‘Some packages have been scanned at some sort facility.’} \ldots\]
\[\text{EffectScenario}_{T_2} = \text{‘Some packages are released from some bonding area.’} \ldots\]
\[\text{EffectScenario}_{T_3} = \text{‘All packages have been sorted to their destination routing facility’} \ldots\]
\[\text{CumulEffScn}_{T_1} = \text{‘Some packages have been scanned at some sort facility.’ AND ‘All packages have been sorted to their destination routing facility’} \ldots\]
\[\text{CumulEffScn}_{T_2} = \text{‘Some packages are released from some bonding area.’ AND ‘All packages have been sorted to their destination routing facility’} \ldots\]

Figure 5.6: Pair-wise accumulation to a task during an XOR-join

Finally, Figure 5.7 represents an OR-join, with tasks \(T_1 = \text{‘Consolidate Packages’},\)
\(T_2 = \text{‘Consolidate Documents’},\) and \(T_3 = \text{‘Consolidate Containers’}.\) In this case, either: \(T_1 \text{ ‘AND’ } T_2\) may have completed; or, \(T_1 \text{ ‘OR’ } T_2,\) prior to the completion of \(T_3.\) Therefore, we combine the methods for AND-joins, and XOR-joins to identify the cumulative effect at \(T_3.\)

Firstly we determine the result in the scenario where both \(T_1 \text{ ‘AND’ } T_2\) have completed (i.e. as in the AND-join previously discussed). We then determine the result whereby either \(T_1 \text{ ‘OR’ } T_2\) have completed (i.e. providing alternate effect scenarios). We then combine both results to provide the cumulative effect at \(T_3.\) That is, if there were only one effect scenario for both \(T_1\) and \(T_2,\) the result will be three cumulative effect scenarios at \(T_3.\)
5.2. Using Model Annotations To Assess Relationships between Business Process and Organisational Models

5.2.3 Fulfillment Conditions

A fulfillment condition [50] is a statement specifying the required conditions realized upon completion of a given task, goal or dependency in an organisational model (an i* model in this case). Fulfillment conditions recognize the required effects on a business process model. For example, a fulfillment condition for a task dependency to ‘BondA-Package’, may include an effect stating: ‘Some packages have been forwarded to some bond facility’. Fulfillment conditions annotated to dependencies will intuitively be required by the task the dependency is linked to on the dependee actor. This implies that the dependee task must include the capability to realize the fulfillment conditions of any of the dependencies it is required to fulfill.

Fulfillment conditions are annotated in the same manner as effects in business process models. In i*, fulfillment conditions are annotated to tasks and goals assigned to actors in an SR diagram and dependencies in an SD model. At this point in time, we do not
include soft-goals during annotation as they describe non-functional properties used during assessment of alternative structures.

5.2.4 Establishing Realization Relationships between Elements of both Organisation and Business Process Models

To effectively manage change, we need to deal with changes to both the processes themselves and also to the organisational context. In both cases, we need to evaluate the impact of these changes on the process model with reference to models of the organisational context. Impact can be determined via an assessment of realization between elements represented within organisational models and business processes.

We establish true realization by first establishing normative realization links between a BPMN model and an $i^*$ model. Such links relate activities or sub-processes, to tasks in an $i^*$ model. A normative realization link must be established by an analyst and suggests that the task in question must ideally be realized by the process it is linked to. We determine whether this normative statement actually describes reality over two steps (described below).

We establish weak realization of the link, which determines whether the effects of the task-process pair are contradictory. Contradictory effects preclude the possibility of the process realizing the task, while consistent effects and therefore weak realization leaves open such a possibility. Identification of weak realization is similar to the process discussed during effect accumulation, whereby any contradictory effects are identified. In this case however, any contradictions between fulfillment conditions and effect scenarios signify that the required fulfillment conditions defined in the organisational model cannot be realized during process execution. This is due to an inconsistency between
the effects of the process, and the fulfillment conditions. For example, a fulfillment condition requiring ‘All participants have agreed to the proposed date’ is inconsistent with an effect stating ‘Some participants have not agreed to the proposed date’. In the following, we will refer to object-level consistency between assertions of effects and fulfillment conditions. We will also refer to consistency between models and consistency labels on normative realization links. The context of use will clarify which specific notion of consistency is being used in each instance. When only informal annotations are available, consistency checking involves analysts evaluating natural language descriptions of effects and fulfillment conditions to determine if they are contradictory.

We then establish *true realization* to indicate that the process in question does indeed realize the task that it is linked to. True realization is established by identifying whether each process related fulfillment condition is *entailed* in each effect scenario in the cumulative effect of the process. This involves analysts evaluating whether the fulfillment conditions annotated to the task in the organisational model always hold when the effects are true. That is the fulfillment conditions *follow on from* the effects.

In this case, the analyst is to be provided with the cumulative effect scenarios resulting from effect accumulation of the process. The analyst then assesses each fulfillment conditions against each effect scenario in the cumulative effect of the process. True realization is established if the analyst can identify that the fulfillment conditions hold in each effect scenario.
5.2.5 Assessing Realization between Organisational and Business Process Models

We now describe a methodology for assessing the level of realization between an organisational model (as represented in $i^*$) and a business process model (as presented in BPMN).

The methodology is to be followed in determining realization between a BPMN model and an $i^*$ model. Note that we will label links as weakly/truly realized/unrealized – this is merely for convenience, bearing in mind that the corresponding labels actually describe consistency/inconsistency of the elements that are related via these links.

Note also that consistency and entailment checking between effects and fulfillment conditions can be automated with the use of formal reasoners when formal annotations are available (we require that all effect annotations and fulfillment conditions are specified in the same formal language).

Step 1: A set of normative realization links between the BPMN model and the $i^*$ model is established by an analyst. Tasks in an $i^*$ model are to be normatively linked to activities and sub-processes in a BPMN model. The internal structure of tasks in an $i^*$ model provides some guidance for establishing links. Lower level tasks should conceivably be represented as either sub-processes or activities in a BPMN model that is linked to some higher level task. This may not be the case however, where a sub-task in an $i^*$ model has been represented at some lower level of decomposed detail in a BPMN model.

Step 2: For each such link:

a. We first determine weak realization. A normative realization link is deemed to be
weakly realized if every effect scenario in the cumulative effect of the process is consistent with the fulfillment conditions of the corresponding task in the $i^*$ model. That is, the analyst has reviewed each effect scenario against the fulfillment conditions in the $i^*$ model, and can safely say that there are no contradictions between effects. The link is labelled \textit{unrealized} otherwise.

b. We next determine \textit{true realization}. A normative realization link is deemed to be truly realized if:

1. It is \textit{weakly realized}, and

2. Each fulfillment condition of the task in question is entailed (as discussed in the previous section) by each effect scenario, in the cumulative effect of the process in question.

3. Otherwise, the link is deemed to be \textit{unrealized}.

\textbf{Step 3:} Given a process model, an organisational model and a set of normative realization links relating the two:

a. The process model is said to be \textit{unrealized} (or inconsistent) with the organisational model if there exists \textit{at least one} normative realization link that is deemed to be unrealized.

b. The process model is said to be \textit{weakly realized} with the organisational model if all normative realization links are deemed to be weakly realized. Otherwise, the models are weakly unrealized.

c. The process model is said to be \textit{truly realized} with the organisational model if all normative realization links are deemed to be truly realized. Otherwise, the models are
truly unrealized.

Note that true realization implies weak realization, both for normative realization links and for pairs of models.

5.3 Managing Process Portfolios with Architectural Models of an Organisation

In previous sections we have shown how relationships can be established between business process and organisational models. We have also discussed the nature of change and identified that change may either occur at an organisational or business process level. We now discuss how the change can be effectively supported by the methodology we have established.

We propose the use of organisational models to provide the ‘scaffolding’ with which to organize business processes. This allows for improved traceability to the greater organisational context via explicit representation for issues of strategic importance such as participant motivation and inter-dependency. Thus the framework, illustrated in Figure 5.8, can be effectively used to manage the “entire set of processes” in a more holistic manner.

A Process Portfolio is “a coherent treatment of the entire set of processes, allowing them to be improved in total, rather than streamlining one and, consequently, unknowingly, suboptimizing others” [106]. It provides a holistic view of organisations processes, their relationships and salient properties. Some of the proposed contributions of process portfolio management include: an ability to “provide an initial structure in a
process-unaware organisation”; improved support for understanding “the most important set of its business processes” according to their qualitative aspects such as risk, criticality, impact and opportunity etc; and the utilization of “not only process model data, but also corresponding information about the actual process executions” during improvement efforts.

![Diagram](image)

Figure 5.8: Enterprise modeling Framework

The $i^*$ framework’s focus is on the strategic relationships between organisational actors and their underlying motivations and capabilities. As such, it provides an ideal high-level representation framework for initial process elicitation. Specifically, its sequence agnostic characteristics allow an analyst to focus on architectural requirements from an intentional actor perspective, and leave operational requirements such as coordination and communication for the later phases of detailed design. Coupled with our annotations and methodology, any processes constructed can then be verified against
organisational requirements. In this sense, once a valid architecture is agreed upon, process design and construction becomes a significantly easier task. In addition, any subsequent changes post-design can then be assessed against their impact at the organisational level, as will be illustrated (with examples) below.

5.3.1 Assessing the Impact of Process Change with Reference to an Organisational Model

The process in Figure 5.9 represents the ‘Bond Package’ sub-process in Figure 5.3, that has been changed as per “Change 1: Task/Flow Addition”. Previously true realization links were established between the ‘Bond Package’ sub-process and the tasks to ‘Receive[Package]’, ‘Bond[Package]’ and ‘Provide[Details]’ for the organisational model illustrated in Figure 5.2. Their fulfillment conditions are represented in Table 5.1.

<table>
<thead>
<tr>
<th>Task</th>
<th>fulfillment Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive[Package]</td>
<td>‘Some Packages have been received at the bond facility.’ AND ‘All received packages have been scanned.’</td>
</tr>
<tr>
<td>Bond[Package]</td>
<td>‘All received packages have been bonded at a bond facility.’</td>
</tr>
<tr>
<td>Provide[Details]</td>
<td>‘The details for all received packages have been provided to the regulatory authority.’ AND ‘All received packages have been screened’</td>
</tr>
</tbody>
</table>

Table 5.1: Associated Tasks and Effects for the Bond Package Process in Figure 5.9

We initially describe and assess the first change that resulted in the model in Figure 5.9, and then describe and assess another proposed change - “Change 2: Task Removal”.
5.3. Managing Process Portfolios with Architectural Models of an Organisation

Figure 5.9: Assessing the impact of process change in the Package Routing example

Change 1: In this case, the change involves the introduction of a new task ‘Return Package’ and exclusive decision gateway ‘Valid/Invalid’ (see Figure 5.9). The intent of this change was to reduce the processing time within the bond, by re-routing packages with invalid paperwork (i.e. unreadable, or for another destination / organisation) back to the sort facility that initially forwarded the package. The ‘Return Package’ task is also annotated with: ‘The details for some received packages have not been provided to the regulatory authority’ effect. The result of effect accumulation is two effect scenarios, with one containing the aforementioned effect. Upon analysis, the new effect scenario introduces a contradiction with the fulfillment conditions of the ‘Provide[Details]’ task, denying its realization. By analysing the organisational model (Figure 5.2), the fulfillment condition can be traced back to the ‘Package Details’ resource dependency the Regulatory Authority requires to ‘Handle[Package Clearance]’. In summary, this change has reduced the realization relationship between the models to unrealized, due to the contradiction.

Change 2: This change concerns the proposed removal of the ‘Scan Package’ task
in Figure 5.9. The task includes an effect stating: ‘All received packages have been scanned’, that realizes the a fulfillment condition annotated to the ‘Receive[Package]’ task in the organisational model. This change has the intent to reduce the package handling requirements of the Bond Department on the assumption that the Sort Facility will be scanning the packages prior their receipt. Upon accumulation the effect is not stated as being realized by the process, therefore the status of realization is reduced to weakly realized. That is, it is not explicitly stated whether the effect has or has not been realized.

Both the above changes illustrate simple scenarios of where the effect annotations and organisational model may be used to effectively manage process portfolios and change. In the first case, a change may have resulted in a process that is uncompliant with regulatory constraints. In the second case, a weaker reduction in the realization of process requirements was the case. In both cases, significant operational impact may still be foreseeable. However, the traceability available in the methodology and annotations provides a first step towards resolving actual / possible inconsistencies at design time, even in environments where processes are independently designed and managed.

5.4 Summary

In this chapter, we have provided a method to support effective process change against higher-level models of the organisational context. Process change occurring during design can be connected to elements on the organisational model to be realized. Once connections are established, the fulfillment conditions of related organisational elements
serve as requirements to be considered during process design. In addition, any changes at an organisational level can then be traced to elements at the process level for analysis of current capabilities, possibly triggering / focussing improvement efforts. Ideally, the methods and extensions are to support process design, and as such require integration with an associated tool. We are actively pursuing this task and hope to provide applied field results in the near future. In the next chapter we will apply our constrained development methodology to i* and UML Activity Diagram.
Chapter 6

Integration of Agent-Oriented Conceptual Models and UML Activity Diagrams Using Effect Annotations

Agent-oriented conceptual modeling notations such as i* represents an interesting approach for modeling early phase requirements which includes organisational contexts, stakeholder intentions and rationale. On the other hand, Unified Modeling Language (UML) is suitable for later phases of requirement capture which usually focus on completeness, consistency, and automated verification of functional requirements for the new system. In this chapter, we propose a methodology to facilitate and support the combined use of notation for modeling requirement engineering process in a synergistic fashion. For organisational modeling/early phase requirements capturing we use the i* modeling framework that describes the organisational relationships among various actors and their rationales. For late (functional) requirements specification, we rely on UML Activity Diagram.
6.1 Introduction

Understanding the organisational environment as well as the reasoning and rationale underlying requirements, design and process formulation decisions is crucial to model and build effective computing systems. Conceptual modeling notations employing knowledge representation techniques have been developed to support such an understanding [154]. A number of proposals have been made for combining \( i^* \) modeling with late phase requirements analysis and the downstream stages of the software lifecycle. The TROPOS project [19] uses the \( i^* \) notation to represent early and late phase requirements, architectures and detailed designs. However, the \( i^* \) notation itself is not expressive enough to represent late phase requirements, architectures and designs [81]. To address this problem, a custom designed formal language called FormalTropos [47] has been proposed. Proposals to integrate \( i^* \) with formal agent programming languages and formal methods have also been reported in the literature [81] [140] [144] [38]. This paper has similar objectives, but takes a somewhat different approach. We believe that the value of conceptual modeling in the \( i^* \) framework lies in its use as a notation complementary to existing specification languages, i.e., the expressive power of \( i^* \) complements that of existing notations. The use of \( i^* \) in this fashion requires that we define methodologies that support the mapping of \( i^* \) models with more traditional specifications. In the current instance, we examine how this might be done with Unified Modeling Language (UML) [13]. Our aim, then, is to support the modeling of organisational contexts, intentions and rationale in \( i^* \), while traditional specifications of functionality and design proceeds in the UML Activity Diagram. More generally,
this research suggests how diagrammatic notations for modeling early phase requirements, organisational contexts and rationale can be used in a complementary manner with more traditional specification notations that lead towards system modeling.

In this chapter, we propose some guidelines to facilitate and support the combined use of notations for modeling requirement engineering process in a synergistic fashion. For organisational modeling early phase requirements capturing we use the $i^*$ modeling framework that describes the organisational relationships among various actors and their rationales. For late (functional) requirements specification, we rely on a UML Activity Diagram. The heuristics described in this paper helps the system modeler to develop activity diagram based on $i^*$ models. We start the following sections with a discussion about $i^*$ modeling framework and UML Activity Diagrams with examples. We then illustrate the benefits of mapping $i^*$ models into Activity Diagram and present a methodology supporting the mapping of $i^*$ into UML Activity Diagram using effect annotations.

6.2 The $i^*$ Modeling Framework

6.2.1 Training System Strategic Dependency Model

The SD model provides an important level of abstraction for describing systems in relation to their environments, in terms of intentional relationships among them. This allows the modeler to understand and analyse new or existing organisational and system configurations even if the internal goals and beliefs of individual agents are not known.

An example concerning a computer based training system (CBT) for volunteers of
emergency services will be used to illustrate the Strategic Dependency (SD) Model notation (see figure 6.1 for the model). The modeling process begins with identifying the actors/agents involved with the CBT system and their mutual dependency relationships (using the taxonomy of dependency relationships described above). The TrainingCoordinator agent depends on Volunteer agents to achieve its TrainingAttended goal. The TrainingCoordinator has two goal dependencies on the TrainingSystem, TrainingScheduled and OnlineTrainingConducted (i.e., the TrainingCoordinator agent relies on the TrainingSystem agent to schedule training sessions and to conduct online training). The TrainingSystem has a dependency on the TrainingCoordinator to provide TrainingContent, modeled as a resource dependency. The TrainingSystem has a dependency on Volunteers to achieve its TrainingAttended goal. The TrainingSystem has a dependency on Volunteers to provide Confirmation of their attendance, modeled
as a resource dependency. Volunteers depend on the TrainingSystem to perform the ConductTraining task. Observe that we have chosen not to model this as a goal dependency since the TrainingSystem cannot autonomously decide how the corresponding goal might be achieved but must work with the depender in a tightly coupled fashion to perform the task. Volunteers have a further dependency on the TrainingSystem to TrainingScheduleReminder and TrainingInformation, modeled as resource dependencies. Volunteers have a preference for the TrainingSystem to satisfy the softgoal Training-ModulesEasyToUse. The notion of a softgoal derives from the Non-Functional Requirements (NFR) framework [25] [27] and is commonly used to represent optimisation objectives, preferences or specifications of desirable (but not necessarily essential) states of affairs.

### 6.2.2 Training System Strategic Rationale Model

Intentional elements (goals, tasks, resources, and softgoals) appear in the SR model not only as external dependencies, but also as internal elements linked by task decomposition and means-ends relationships. The SR model in figure 6.2 thus elaborates on the relationships between the TrainingCoordinator, TrainingSystem and Volunteer as represented in the SD model of figure 6.1.
For example, the TrainingCoordinator has an internal task to OrganiseTraining. This task can be performed by sub-tasks ScheduleTraining and GenerateTrainingContent (these are related to the parent task via task decomposition links). The task OrganiseTraining is related to the LowEffort, Quick softgoals via a task decomposition link. The intention is not to suggest that the softgoal plays the role of a sub-task but to relate the softgoal to the highest-level task for which the softgoal may be viewed as an optimization objective. The softgoal thus serves to constrain design decisions.
on how the task might be decomposed. In this instance, the contribution is positive, i.e., organizing the training material contributes (positively) to achieving the broader goal of making the TrainingMaterialEasyToUse.

The TrainingSystem agent is identified with a high level rationale OrganiseTraining, modeled as a task. The task is further decomposed into the sub-tasks ImpartTraining, ObtainConfirmation and MaintianSchedule. The Volunteer agent is responsible for the task AcquireTrainingSkills, which is decomposed to the sub-tasks ParticipateInTraining and ProvideConfirmation.

In \textit{i*}, a routine \cite{154} specifies an intended course of action an actor may pursue given a set of alternatives. These elements and their relationships represent the strategic requirements of a process when invoked in a specific context. For example, to OrganiseTraining (illustrated in figure 6.2) TrainingSystem includes four sub-tasks and ten dependencies with two additional actors. Tasks in \textit{i*} may be primitively workable whereby the actor responsible for the element believes that it can achieve its requirements at execution time - i.e. it is sufficiently reduced during decomposition. In comparison to UML Activity Diagram however, a primitively workable element may still be represented as an activity as the term does not imply a ‘primitively executable action’ (i.e. application of analyst / designer discretion). Furthermore, for a routine to be workable, all involved actors must be committed to satisfying their dependencies \cite{154}.
6.3  UML Activity Diagram

An activity diagram is an uncomplicated and perceptive illustration that depicts the actions, parallel activities and any possible alternative ways through the workflow. Activity diagrams defined in the Unified Modeling Language [137] are consequential from various methods to pictorially express sequence of activities or sub-activities and conditions taken within a process. Activity Diagrams explain the operational flow from an initiating point to the terminating point specifying many decision paths that exist in the development of proceedings contained in the activity. They are also used to explain states where parallel processing may occur in carrying out of some activities. Graphically an activity diagram is an anthology of vertices and arcs which generally contains activity states, action states, transitions and objects. Activity states are non-atomic as they can be interrupted and usually they may take some time to be accomplished. But action states are atomic, their work is non-interrupted. Action states can not be decomposed. Transitions depict the path initiated from one action or activity state and passed to next action or activity state as the action or activity of a state is completed. Transitions are represented as a simple directed line in the activity diagram [13].

6.4 Benefits of Mapping $i*$ Model into UML Activity Diagram

Constructing a system that adheres to organisational environment and meets end users need (such as determining the main goals of the intended system, relations and dependencies among stakeholders, alternatives in the early-stage requirements analysis etc.),
requires developing a clearly defined early stage functional requirements. The $i^*$ modeling framework which is a semi-formal notation built on agent-oriented conceptual modeling is well suited for this purpose.

We need to focus on the functional and non-functional requirements of the system as we continue the development process. In this phase we can adopt the UML activity diagram to discover and reason about the functional requirements of the system. An activity diagram is a dynamic illustration, which demonstrates the movement and the event of objects in the particular state. It clearly supports parallel activities and their synchronization. Activity diagrams are functional for analysing actions and the states of a Use Case, illustrating complex sequential algorithm and designing applications with parallel processes [13] [86] [96]. They represent the operational workflow of a system by capturing actions performed and provide a broad representation of the overall flow. Some benefits of integrating these two notations are given below.

- We feel that the usefulness and effectiveness of $i^*$ can be increased manifold by using it with UML activity diagram. Mapping rules provide a semantics to $i^*$ framework. Our view is that the $i^*$ modeling framework and UML activity diagram can function in a complementary and synergistic way.

- There is a need to map both SD and SR models into late phase requirements specification. Activity diagram can be used effectively to realize the actions and states in the late phase which cannot be represented in the $i^*$ diagram.

- For translating informal specifications provided in $i^*$ into Activity diagram, there is no need to add more details into the corresponding $i^*$ model. The mapping
from $i^*$ models into Activity diagram does not result in any information loss; this is ensured by consistency evaluation rules mentioned in this chapter.

- Using Activity diagram, we are in a position to express properties that are not restricted to the current state of the system, but also to its past and future history.

6.5 Methodology Supporting the Integration of $i^*$ and UML Activity Diagram

We provide some guidelines for the mapping of $i^*$ model into UML activity diagram. Mapping is done in two phases; phase-1 effect annotations, phase-2 mapping rules. These guidelines ensure the consistency of the generated activity diagram with the initial $i^*$ model.

Our proposed methodology uses the notion of cumulative effect annotation to determine whether the $i^*$ models and UML Activity Diagrams are consistent with each other. An effect is the result (outcome) of an activity being executed by some cause or agent. It indicates the achievement of a certain environmental state communicated through an event. In our work, every goal/task/resource dependency must have an effect annotation. A cause relationship exists between an activity and an effect. In other words, activity causes the effects to occur. An activity can cause many effects and an effect can be caused by a number of activities. For each selected dependency we have an object in the UML Activity Diagram with the same effect. This we consider as a weak notion of consistency. It clearly states the result of activity if the conceptual model were to be theoretically executed. We also annotate every task in the SR model
that is related to a dependency with a cumulative effect annotation. We then use the Activity Diagram and annotate actions with effects. Our approach ensures that a dependency is achieved through the cumulative effect of the actions on the UML Activity Diagram. This we refer to as strong consistency. Using this notion of cumulative effect annotations an analyst can ensure that a UML Activity Diagram is consistent with respect to the $i^*$ model under this regime.

6.5.1 Consistency Evaluation

We introduce consistency rules to provide a mechanism for ensuring consistency between $i^*$ model and UML Activity Diagram. The rules are developed with consideration to [50].

**Rule 1:** Every actor in an $i^*$ model required as a participant in the Activity Diagram must be represented in the model. Required participants are identified via the associated dependencies among the actors.

**Rule 2:** Every ‘primitively workable’ task decomposed (or required by decomposition where a dependency exists) from the chosen routine within the $i^*$ model, must be represented as an action or activity under the control of the appropriate actor in the process model.

**Rule 3:** There must exist a coordinated transition in the Activity Diagram, whereby the operational objective (as encoded in the fulfillment conditions or effect annotations) of the routine is achieved, and the sequence of activities is consistent with the requirements specified in the routine. There must exist a coordination of activities in the activity diagram that satisfy the requirements of the routine further outlined
6.5. Methodology Supporting the Integration of $i^*$ and UML Activity Diagram

below.

Rule 3.1: The fulfillment conditions of the operational goal at the root of the routine and all its sub-elements must be achieved through the accumulation of effects during forward traversal of the transition.

Rule 3.2: The fulfillment conditions of a task in the chosen routine must not be fulfilled prior to all tasks that decompose it, upon accumulation of effects during forward traversal of the transition.

Rule 3.3: The fulfillment of a task on the dependner side of a dependency must not be realized before the fulfillment of the dependency, upon the accumulation of effects during forward traversal of the transition.

6.5.2 Phase 1: Effect Annotations

The concept of effect annotation denotes the potential outcomes of activities and fulfilment conditions that are required to meet dependencies by achieving certain results. An effect generally defines that a result or consequence of an activity has generated because of its being accomplished by an agent or some previous phenomenon. As an example, effects can be annotated to activity/task nodes or even complete subprocesses in graphical notations. In $i^*$, we annotate effects to tasks assigned to actors which indicate the realization of a certain conditions aimed in the direction of (i.e. and perhaps required for) some higher order goal. The effect annotations is intended to provide a notation free methodology rather than limited to a specific notation. An effect annotation is a testimonial to the outcome of an activity related to a state that alters construction of a given model.
An effect annotation includes: a label that generalizes the effect (e.g. ‘CustomerDetailsStored’); a designation specifying whether the effect is a normal (i.e. desired) outcome for an activity (e.g. ‘RegistrationValidated’), or an abnormal (i.e. undesired) outcome for the activity that may require the application of some mitigation strategy; an optional informal definition describing the effect in relation to the result achieved in its environment (e.g. ‘The details relating to the current customer have been stored within the system.’); an optional formal definition may be used to define achieved states in a chosen formalism.

Fulfillment conditions are annotated to intentional actor elements and dependencies in an \(i^*\) model (i.e. not including soft-goals as these are used during assessment of alternatives and describe non-functional properties to be addressed). A fulfillment condition is a statement specifying the outcomes required to satisfy a given goal or dependency. Fulfillment conditions recognize the required effects on a business process model. For example, a fulfillment condition for a task dependency to ‘ConductTraining’, may be the ‘TrainingArranged’ effect (subsequently required by the task assigned to a dependee actor).

Intuitively, for a dependency to be fulfilled, explicit assignment of responsibility is made to a dependee actor who possesses an intentional element that can satisfy the dependency. Therefore, one guiding rule during the annotation of fulfillment conditions to an \(i^*\) model is that all fulfillment conditions annotated to a dependency must be annotated to the intentional element the dependency is linked to on the dependee. In this case we are only concerned with the fact that the dependee has the knowledge to achieve the dependency; not the ability (e.g. where another dependency may be
required with another actor). We have introduced two steps to derive the effect annotations from the CBT $i^*$ diagram. Step 1, Annotate the $i^*$ model with effects and then derive the annotations of dependencies with fulfilment conditions. Step 2, define fulfilment conditions to the tasks that Realizes/Requires the fulfilment conditions.

**Step 1:** Annotate model with effects and/or fulfilment condition

The tasks assigned to the actors in the CBT model are initially annotated with effects. Table 6.1 illustrates the annotation in a tabular form. Please note, TC stands for Training Coordinator, TS stands for Training System and Vol stands for Volunteer.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Task</th>
<th>Effect Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>Let Training System Schedule Training</td>
<td>Training System Schedule Training</td>
</tr>
<tr>
<td>TC</td>
<td>Generate Training Content</td>
<td>Training Material Generated</td>
</tr>
<tr>
<td>TC</td>
<td>Organize Training</td>
<td>Training Organized</td>
</tr>
<tr>
<td>TS</td>
<td>Obtain Confirmation</td>
<td>Confirmation obtained</td>
</tr>
<tr>
<td>TS</td>
<td>Create &amp; Forward User Access Info</td>
<td>User Name &amp; Password created</td>
</tr>
<tr>
<td>TS</td>
<td>Impart Training</td>
<td>Training Imparted</td>
</tr>
<tr>
<td>TS</td>
<td>Maintain Schedule</td>
<td>Training Schedule Maintained</td>
</tr>
<tr>
<td>TS</td>
<td>Arrange Training</td>
<td>Training Arranged</td>
</tr>
<tr>
<td>Vol</td>
<td>Provide Confirmation</td>
<td>Confirmation Provided</td>
</tr>
<tr>
<td>Vol</td>
<td>Participate in Training</td>
<td>Participated in Training</td>
</tr>
<tr>
<td>Vol</td>
<td>Acquire Training Skills</td>
<td>Training Skills Acquired</td>
</tr>
</tbody>
</table>

Table 6.1: Annotation of Tasks with Effects

The second segment of the model annotation involves annotating dependencies with fulfilment conditions that relate to required effects in the $i^*$ model. Table 6.2 depicts the dependency among the actors and their fulfilment condition to meet the dependencies in the Training System model.
6.5. Methodology Supporting the Integration of i* and UML Activity Diagram

Table 6.2: Annotation of Dependencies with Fulfilment Conditions

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Fulfilment Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Content</td>
<td>TC: Training Content Generated</td>
</tr>
<tr>
<td>Training Schedule</td>
<td>TS: Scheduled Training</td>
</tr>
<tr>
<td>Confirmation</td>
<td>Vol: Confirmation Provided</td>
</tr>
<tr>
<td>Username &amp; Password</td>
<td>TS: Username &amp; Password Created</td>
</tr>
<tr>
<td>Training Schedule Reminder</td>
<td>TS: Training Schedule Reminded</td>
</tr>
<tr>
<td>Online Training Conducted</td>
<td>TS: Training Conducted</td>
</tr>
<tr>
<td>Training Lesson</td>
<td>TS: Training Lesson provided</td>
</tr>
<tr>
<td>Training Information</td>
<td>TS: Training Information Provided</td>
</tr>
<tr>
<td>Conduct Training</td>
<td>TS: Training Arranged</td>
</tr>
<tr>
<td>Training Attended</td>
<td>Vol: Acquired Training Skills</td>
</tr>
</tbody>
</table>

**Step 2:** Propagate fulfilment conditions in i* models to task assigned to dependee and depender actors. The analysis of dependency proliferate effect annotations of dependencies into tasks that realise/require the fulfilment conditions. The task that realizes the dependency obtains the effect annotation as a required post and task requiring the dependency obtains the effect annotations as a required pre-condition. Table 6.3 illustrates the dependency with the fulfilment conditions and tasks that realize/require condition.
### Table 6.3: Tasks that Realizes/Requires the Fulfillment Conditions

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Fulfilment Conditions</th>
<th>Task - Realizes Fulfilment Condition</th>
<th>Task - Requires Fulfilment Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Content</td>
<td>TC: Training Content Generated</td>
<td>TC: Generate Training Content</td>
<td>TS: Impart Training</td>
</tr>
<tr>
<td>Training Scheduled</td>
<td>TS: Schedule Training</td>
<td>TS: Maintain Schedule</td>
<td>TC: Let Training System Schedule Training</td>
</tr>
<tr>
<td>Confirmation</td>
<td>Vol: Confirmation Provided</td>
<td>Vol: Provide Confirmation</td>
<td>TS: Obtain Confirmation</td>
</tr>
<tr>
<td>Training Schedule Reminder</td>
<td>TS: Training Schedule Reminded</td>
<td>TS: Maintain Schedule</td>
<td>Vol: Participate in Training</td>
</tr>
<tr>
<td>Online Training Conducted</td>
<td>TS: Training Conducted</td>
<td>TS: Impart Training</td>
<td>TC: Organize Training</td>
</tr>
<tr>
<td>Training Lesson</td>
<td>TS: Training Lesson provided</td>
<td>TS: Impart Training</td>
<td>Vol: Participate in Training</td>
</tr>
<tr>
<td>Training Information</td>
<td>TS: Training Information Provided</td>
<td>TS: Arrange Training</td>
<td>Vol: Participate in Training</td>
</tr>
<tr>
<td>Conduct Training</td>
<td>TS: Training Conducted</td>
<td>TS: Arrange Training</td>
<td>Vol: Participate in Training</td>
</tr>
</tbody>
</table>

Now we have the effect annotations for the intentional elements such as goals, resources and tasks. The dependency analysis will recognize the pre/post conditions of
6.5. Methodology Supporting the Integration of i* and UML Activity Diagram

the elements. Below is an illustration of the fulfilment condition propagation of the Training System Model in Table 6.4.

<table>
<thead>
<tr>
<th>Task</th>
<th>Effect Annotation</th>
<th>Required Pre</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC: Let Training System Schedule Training</td>
<td>Training System Schedule Training</td>
<td>TC: Organize Training Schedule Training</td>
</tr>
<tr>
<td>TC: Generate Training Content</td>
<td>Training Material Generated</td>
<td>TC: Conduct a computer based training</td>
</tr>
<tr>
<td>TC: Organize Training</td>
<td>Training Organized</td>
<td>TS: Impart Training</td>
</tr>
<tr>
<td>TS: Obtain Confirmation</td>
<td>Confirmation obtained</td>
<td>Vol: Provide Confirmation</td>
</tr>
<tr>
<td>TS: Create &amp; Forward User Access Info</td>
<td>User Name &amp; Password created</td>
<td>TS: Arrange Training</td>
</tr>
<tr>
<td>TS: Impart Training</td>
<td>Training Imparted</td>
<td>TC: Generate Training Content</td>
</tr>
<tr>
<td>TS: Maintain Schedule</td>
<td>Training Schedule Maintained</td>
<td>TC: Let Training System Schedule Training</td>
</tr>
<tr>
<td>TS: Arrange Training</td>
<td>Training Arranged</td>
<td>TC: Organise Training</td>
</tr>
<tr>
<td>Vol: Provide Confirmation</td>
<td>Confirmation Provided</td>
<td>None</td>
</tr>
<tr>
<td>Vol: Acquire Training Skills</td>
<td>Training Skills Acquired</td>
<td>Vol: Attend Training</td>
</tr>
<tr>
<td>Vol: Participate in Training</td>
<td>Participated in Training</td>
<td>TS: Create &amp; Forward User Access Info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TS: Training Lesson</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TS: Training Schedule Reminder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TS: Training Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TS: Conduct Training</td>
</tr>
</tbody>
</table>

Table 6.4: Propagation of Fulfillment Conditions to Respective Tasks
6.5.3 Phase-2: Mapping Rules

Rule-1: Discover the actors and represent them in activity diagram

We should go through the $i^*$ model to discover the actors. This step can be completed by looking at either SD model or SR models. Once the actors are found they will be placed as the names of the swimlanes of the activity diagram. We prefer using swimlanes pattern of the activity diagram as they are used to organize responsibilities for the actions. They can often correspond to organisational units in a business process model. Each swimlane represents a high level responsibility for part of the overall activity of an activity diagram. Every activity will belong to exactly one swimlane, but transitions may cross lanes.

For example, to discover the actors in the CBT system, we can look at the SD model in figure 6.1. From the SD model we get three actors, Training Coordinator, Training System and Volunteer. When we map the $i^*$ model into UML activity diagram, these actors are represented in the swimlanes to show the responsibilities of each actor (for each actions) associated with the overall system

Rule-2: Discover task/ actions

In this step we need to identify the tasks involved in the system. SR model of the $i^*$ diagram shows the internal tasks and their rationales. For each actor, the SR model will be analysed to discover the tasks. In our methodology identification of tasks/actions and their effects has been analysed in the effect annotation part. We will take the tasks from table 6.1 for the mapping and then categorize them according to the actors. We can discover the task of the CBT system by looking at its SR model. This model represents all internal tasks and their rationales. From table 6.1 we get the complete list
of tasks with their effect annotations. Tasks in $i^*$ model will be regarded as actions in activity diagram. Thus TrainingCoordinator has GenerateTrainingContent, LetTrainingSystemScheduleTraining and OrganiseTraining actions, TrainingSystem has ObtainConfirmation, Create&ForwardUserAccessInfo, MaintainSchedule, ImpartTraining and ArrangeTraining actions, and Volunteer has ProvideConfirmation, ParticipateInTraining and AcquireTrainingSkills actions.

**Rule-3: Identify the Initiating Actor**

Among the discovered actors we need to find the initiating actor. This actor will be responsible for the initial action in the activity diagram. The initiating actor can be identified through their ability to satisfy the pre-condition with an action that realizes the required effect. The initial actor will be represented in the first swimlane of the activity diagram.

There are three actors in the CBT system. To find the initiating actor, we need to analyse the actions, their effect annotations, required pre-condition and fulfilment conditions. By going through these we can conclude that TrainingCoordinator is the initiating actor which has the ability to satisfy the pre-condition of conducting a computer based training by triggering the action LetTrainingSystemScheduleTraining. TrainingCoordinator actor will be placed in the first swimlane of the activity diagram.

**Rule-4: Sequence actions by analysing pre/post conditions derived during annotation**

The tasks required for the fulfilment of the trigger condition for the course of action will be chosen initially and placed as action within the initiating actor’s swimlane in UML activity diagram. After the fulfilment of the pre-condition, the post-condition must be satisfied through the interaction of multiple actors, and the execution of their
assigned tasks. These tasks are mapped to activity diagram as actions and placed in the respective swimlanes that represents the controlling actors. The sequencing for actions is a guided task by identifying the required actions and dependencies in order to achieve the operational goal.

For the CBT system we will start from the initiating actor that initiates the first action. The initiating action is LetTrainingSystemScheduleTraining, so it is placed in the TrainingCoordinator’s swimlane. After fulfilment of the pre-condition of this action, the post-conditions will be satisfied through the execution of one or more actions with the interaction of other actors. Thus, we get GenerateTrainingContent and Maintain-Schedule actions and so on.

**Rule-5: Discover dependencies and represent them in activity diagram**

It is very straightforward to discover dependencies among actors from $i^*$ model. We can get the dependencies from SD or SR models. We will then represent goal, task and resource dependencies as objects in the activity diagram. The actions will specify which objects perform its operation and their states. The actions within a swimlane can be handled by the same objects or multiple objects.

Softgoal dependency in $i^*$ model is considered as a non-functional requirement of the system, which has a positive or negative contribution for achieving, accomplishing a goal, task, resource. For this reason, softgoal dependency will not represent an object. We have a total of eleven dependencies in CBT system including one softgoal dependency TrainingContentEasyToUse. All these dependencies except the goal dependency will be represented as objects. For example the resource dependency TrainingContent will be the object for GenerateTrainingContent and ObtainConfirmation. The state of
the object TrainingContent in this case will be [Generated].

Figure 6.3: UML Activity Diagram Derived Using the Methodology
Rule-6: Introduce required actions and object flow links between swimlanes

The final step includes introducing required actions and objects flow links between actions. The actions will be linked according to their sequence and then flow links will be represented among them which will include the objects and their states. In this step we need to consider the decision points of the activities if there is any. Decision points reflect the previous activity state. On each outgoing transition from decision points, we should cover all possibilities.

In this step we represent all the actors and their respective actions with actions and object flow links. The activity ProvideConfirmation in Volunteer swimlane renders a decision point. It has two guards, [provided] and [not provided], which directs the action links accordingly.
6.6 Reflecting Changes in an $i^*$ Model to an associated Activity Model

We provide some guidelines aiming to support the reflections of changes between $i^*$ and activity diagram for the specific instances of impacting changes outlined below:

**Step 1:** For each classification outlined below apply associated changes.

- **Addition of an Actor:** If a new actor is added to the $i^*$ model a swimlane need to be placed on the Activity Diagram. Additionally new dependencies must exist between the actors and existing actors. These dependencies will be including for all new actors where the dependency is related to the routine and actor is the dependee. However, where the actor is the dependee they will be only included if linked to a task in existing dependency graph.

- **Addition of a Goal/Task/Resource Dependency:** If a new dependency is added to the $i^*$ model, then this may require the addition of new actions/activities in the Activity Diagram.

- **Addition of a Goal/Task:** The addition of a goal or task may require the addition of an action or an activity within Activity Diagram. The addition of task in $i^*$ model must be scoped to their respective actors and any dependencies must be realized via transitions of action and activities.

**Step 2:** Re-apply consistency to both models to assess whether consistency has been maintained. For example in figure 6.2, a new requirement within the form of a task dependency between the TrainingSystem and the TrainingCoordinator, EnterDateRange
is added. This task dependency has been added to provide date range from the TrainingCoordinator to arrange the computer based training.

Given the application of our approach for guiding an analysts decision it can be inferred that the effect for EnterDateRange will propagate within the $i^*$ model as a fulfilment condition on the OrganiseTraining task. Following the proposed consistency evaluation and mapping rules analysts can make required changes to the Activity Diagram.

6.7 Summary

In this chapter we have applied our constrained development methodology to support the mapping of early phase requirement modeling notation $i^*$ into UML activity diagram. The methodology supports the mapping of these two otherwise disparate approaches in a synergistic fashion. We can now analyse the system’s behavior and explain the workflow from an initiating point to the terminating point which is otherwise not possible by only looking at the $i^*$ model and activity diagram separately. When proposing the mapping of two otherwise disparate approaches for requirements engineering, we need to maintain consistency between the two approaches. Effect annotations and mapping rules can be viewed as providing semantics to the $i^*$ diagrams while mapping into activity diagram of UML specifications, a language which already has one. We believe that these semantics are largely consistent with the somewhat implicit semantics for $i^*$. The proposed set of mapping rules constrains the modeler to map the elements of the $i^*$ model to appropriate activity diagram and ensures that the two models are consistent. In the next chapter we will illustrate our methodology on managing business process risk using organisational models.

Business processes represent the operational capabilities of an organisation. In order to ensure process continuity, the effective management of risk becomes an area of key concern. In this chapter we propose an approach for supporting risk identification with the use of higher-level organisational models. We provide some intuitive metrics for extracting measures of actor criticality, and vulnerability from organisational models. This helps direct risk management to areas of critical importance within organisation models. Additionally, the information can be used to assess alternative organisational structures in domains where risk mitigation is crucial. At the process level, these measures can be used to help direct improvements to the robustness and failsafe capabilities of critical or vulnerable processes. We believe our novel approach will provide added benefits when used with other approaches to risk management during business process management, that do not reference the greater organisational context during risk assessment.
7.1 Introduction

A Business Process can be described as a set of dynamically coordinated activities, controlled by a number of socially dependant participants, aimed towards the achievement of a specific operational objective [58] [129]. Business Process Management promotes that a clear understanding through the explicit modeling of the processes underlying an organisation is required to support effective organisational management / improvement practices [57]. An effective means to represent and manage operational risk is one of the most important capabilities within an enterprise. Some of the most prominent applications of risk management techniques include financial / operational management and modeling of organisations. Risk management techniques have also been extensively studied and applied within software process management, requirements engineering and project management disciplines [110] [130]. More recently, risk management has been applied to the business process management and modeling domain that as a whole, aims to bridge the gap between organisational and I.T. level conceptual / management concerns [102] [104] [105]. These approaches provide a more direct association between organisational risks at an activity level.

There are challenges associated to addressing risk at process level [102] [104]. We believe that by taking actor-level considerations such as vulnerability and criticality (at organisational level) as major considerations is important for process-level risk management. We provide an approach to support risk management by supporting the identification of risk factors (in terms of vulnerability and criticality) at organisational level prior to their propagation and reflection at a process level. We believe that such an approach will provide a higher-level scope for risk that may span numerous processes
within an organisation. Business process risk analysis should be based on higher-level organisational models. A high-level approach to iterative risk assessment should be integrated throughout the business process lifecycle. Therefore, risks may be identified and managed at an organisational level prior to their delegation to actual business processes. We provide an enhanced capability to relate risk at an organisational level by looking at the strategic relationships between functional units and process participants. We define risk at organisational model level on the basis of vulnerability and criticality. For organisational modeling we use the agent-oriented organisational modeling notation - $i^*$ [155] that describes the organisational relationships among various actors and their rationales. For business process model representation we use a standardized, operational and executable process modeling notation - BPMN [145].

The following sections starts with a brief background on Agent Oriented Conceptual Modeling and Business Process Modeling with BPMN. We then describe our approach to identify risk factors including our proposed measurement for vulnerability and criticality of actors at organisational level. Finally we illustrate the integration of risk factors in process models with examples and then some concluding remarks.

7.2 Agent Oriented Conceptual Modeling

As mentioned in the earlier part in the thesis, the agent metaphor is powerful in modeling organisational contexts. Agent-Oriented Conceptual modeling (AOCM) in notations such as the $i^*$ framework [154] (see: Figure 7.1) have gained considerable currency in the recent past. Such notations model rich organisational contexts and offer high-level social/anthropomorphic abstractions (such as goals, tasks, soft goals
and dependencies) as modeling constructs.
Figure 7.1: Strategic Dependency Model of an Emergency Service Provider
The central concept in $i^*$ is that of intentional actor. It can be seen in the Emergency Service Provider SD model (refer to 7.1) as nodes representing the intentional/social relationships between six (6) actors required to schedule a meeting: an Emergency Coordination Center Coordinator (ECCC); Field Control Center Coordinator (FCCC); Volunteer/Emergency Workers; Community; Weather Bureau and Call taking supervisor/system.

7.3 Business Process Modeling with BPMN

Business Process modeling Notation (BPMN) can be seen as primarily a technically-oriented notation that is augmented with an ability to assign activity execution control to entities (e.g. roles) within an organisation with ‘swim-lanes’. This effectively provides a view of the responsibilities and required communications between classes of process participants, but does not provide a view of other social and intentional characteristics including the goals of participants and their inter-dependencies.
7.3. Business Process Modeling with BPMN

Processes are represented in BPMN using flow nodes: events (circles), activities (rounded boxes), and decisions (diamonds); connecting objects: control flow links (unbroken directed lines), and message flow links (broken directed lines); and swim-lanes: pools (high-level rectangular container), and lanes partitioning pools. These concepts are further discussed within [145]. Figure 7.2 illustrates a BPMN model of the Emergency Service Provider. We derived this figure by applying our constrained development methodology mentioned in the previous chapters.
7.4 Identifying Risk within Organisational Models

Among the many informal to formal notations for modeling and analysing organisations, we have chosen the $i^*$ notation in describing our approach. $i^*$ has a rich ontological competence [45] that makes it appropriate for supporting both organisational analysis and design tasks (i.e. functional completeness). In the following we will describe our intuitive approach to analysis and design with regards to organisational risk. In order to achieve this task, we propose an analysis of strategic dependencies between actors in order to measure and identify each actor’s vulnerability and criticality. Once determined, the design task can be focused in toward areas of an organisational model that will require the most attention.

7.4.1 Vulnerability

The vulnerability of an actor is a significant factor for identifying and measuring risk. The $i^*$ model provides an intentional description of a process in terms of a network of dependency relationships among actors [155]. We believe because of its richer modeling concepts, the model provides a better basis for an analyst to explore the broader risk implications of alternative organisational structure. It can help analyze opportunities and vulnerabilities and recognize patterns of relationship, such as various mechanisms for mitigating vulnerability. A depender actor’s intention is to have the dependency goal achieved, task performed, or resource available. Failure to obtain the dependum can affect the process by making it more vulnerable and hence increasing the likelihood of risk occurrence. In our work we propose a way of measuring vulnerability of actors at organisational model. The analyst can then take necessary steps to mitigate these
vulnerabilities in process models. A stronger degree of vulnerability implies that a stronger initiative to mitigate vulnerability is necessary. An example of such initiative can be increased monitoring of dependee actor activities.

We propose a metric for actor vulnerability. This metric is defined below, and effectively divides the number of outgoing dependencies by the number of dependee actors. A depender actor with more outgoing dependencies implies a greater degree of vulnerability. We consider outgoing dependencies for vulnerability measurement as we believe that outgoing dependencies indicate delegation of tasks and activities. If the tasks are delegated to other actors the depender actor becomes vulnerable. As a result if the dependee actor fails to satisfy the dependency, the corresponding task/goal might not be satisfied (a considerable risk). The vulnerability of actors thus is related to the likelihood of a risk occurring. We believe if an actor is vulnerable, an increase in the overall likelihood of risk occurrence is apparent. Intuitively, if the likelihood increases risk will increase as well.

The formula we use to assess the vulnerability measurement (VM) of actors at organisational level is as follows:

\[
 VM_{\text{org}} = \frac{\text{No of Outgoing Dependencies}}{\text{No of Dependee Actors}}
\]

For example, for actor *EmergencyCoordinationCentreCoordinator*,

No of Outgoing dependencies = 12;

No of Dependee Actors = 4;

So, Vulnerability, \( VM_{\text{org}} = (12/4) = 3 \).
Table 7.1 illustrates vulnerability measurement of actors at organisational model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>No of Outgoing Dependencies</th>
<th>No of Dependee Actors</th>
<th>VM_{org}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Coordination Centre Coordinator</td>
<td>12</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Weather Bureau</td>
<td>0</td>
<td>0</td>
<td>Minimal Vulnerability</td>
</tr>
<tr>
<td>Call Taking Supervisor/ System</td>
<td>0</td>
<td>0</td>
<td>Minimal Vulnerability</td>
</tr>
<tr>
<td>Volunteer/Emergency Workers</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Flood Control Centre Coordinator</td>
<td>7</td>
<td>3</td>
<td>2.33</td>
</tr>
<tr>
<td>Community</td>
<td>8</td>
<td>3</td>
<td>2.66</td>
</tr>
</tbody>
</table>

Table 7.1: Vulnerability Measurement of Actors at Organisational Model

In a softgoal dependency, a dependee depends on the dependee to perform certain goals or task that would enhance the performance. The notion of a softgoal derives from the Non-Functional Requirements (NFR) framework [25] [27] and is commonly used to represent optimization objectives, preferences or specifications of desirable (but not necessarily essential) states of affairs. So, softgoals are non-functional requirements of the system, which have positive or negative contribution toward achieving a goal, task, or resource. While measuring the vulnerability of actors we do not include the softgoal dependencies. We believe these non-functional requirements of the system have minimal impact on risk either in the organisational level or on the process level. When we calculate the outgoing dependencies of actors we exclude the softgoal dependency. If any actor has no outgoing dependency with other actors, we consider that the actor has minimal vulnerability as we believe it can not affect the likelihood of occurrence in a greater extent. From Figure 7.1, we find that the actors WeatherBureau and
CallTakingSupervisor/System do not have any outgoing dependencies. It means they have not delegated their responsibilities or tasks to other actors. But, actor with no vulnerability does not necessarily mean that it is not critical enough to affect the consequences if it fails. In this case criticality of the actor is considered to measure the risk. Now we need to refine the vulnerability calculation by relating it at process level. The formula we use to calculate vulnerability measurement (VM) at process level is as follows:

\[ VM_{bp} = Organisational\ Level\ Vulnerability\ (VM_{org}) * Number\ of\ Incoming\ Flows\ (control\ flow\ and\ message\ flow)\]

For example, for actor EmergencyCoordinationCentreCoordinator, organisational Level Vulnerability,

\[ VM_{org} = 3;\]

No of Incoming Flows = 6;

So, Vulnerability at Process Level, \( VM_{bp} = 18.\)

Table 7.2 illustrates vulnerability measurement at process level.

<table>
<thead>
<tr>
<th>Actor</th>
<th>VM(_{org})</th>
<th>Incoming Flow</th>
<th>VM(_{bp})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Coordination Centre Coordinator</td>
<td>3</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>Weather Bureau</td>
<td>Minimal Vulnerability</td>
<td>1</td>
<td>Minimal Vulnerability</td>
</tr>
<tr>
<td>Call Taking Supervisor/System</td>
<td>Minimal Vulnerability</td>
<td>1</td>
<td>Minimal Vulnerability</td>
</tr>
<tr>
<td>Volunteer/Emergency Workers</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Flood Control Centre Coordinator</td>
<td>2.33</td>
<td>6</td>
<td>13.98</td>
</tr>
<tr>
<td>Community</td>
<td>2.66</td>
<td>2</td>
<td>5.32</td>
</tr>
</tbody>
</table>

Table 7.2: Vulnerability Measurement at Process Level
7.4.2 Criticality

Criticality is the consequence factor that is measured from the impact of an actor’s performance where the actor is assigned to satisfy responsibilities/incoming dependencies. The more critical an actor is, the more ability it carries to impact other actors and the organisational context. Incoming dependencies towards an actor are taken into consideration to measure the criticality of an actor. The incoming dependencies describe responsibilities are assigned to an actor from other actor. By receiving dependencies from other actor makes the dependency receiving actor crucial. If it fails to satisfy the incoming dependencies the depender actors are widely affected which possibly affect the context as a whole. In order to mitigate the risks associated with the system the criticality measurement of actors should be taken into consideration. Measuring critical factors of actors helps the analysts to analyze and construct alternative options to achieve the aim of the system. This will alleviate the risk management and increase the robustness of the system.

Criticality of actors at organisational Model is measured by multiplying number of incoming dependencies and number of depender actors. The formula we use to assess the criticality measurement (CM) of actors is as follows:

$$CM_{org} = \text{No of Incoming Dependencies} \times \text{No of Depender Actors}$$

For example, for actor Volunteer,

No of Incoming Dependencies = 10;

No of Depender Actors = 3;
So, Criticality at organisational Model, $CM_{org} = 10 \times 3 = 30$.

Table 7.3 illustrates criticality measurement of actors at organisational model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>No of Incoming Dependencies</th>
<th>No of Depender Actors</th>
<th>$CM_{org}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Coordination Centre Coordinator</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Call Taking Supervisor/System</td>
<td>9</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>Volunteer/Emergency Workers</td>
<td>10</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Flood Control Centre Coordinator</td>
<td>9</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>Weather Bureau</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Community</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7.3: Criticality Measurement of Actors at organisational Model

According to the result from Criticality Metrics, Volunteer actor is more critical than other actor in the model. Volunteer has ten incoming dependencies from three other actors and its existence is more crucial because if it fails to satisfy any of the incoming dependencies received from other three actors it will have greater impact on other actors and to system as a whole. We have not considered the softgoal dependencies while calculating the criticality of the actors for the same reasons of vulnerability measurement.

If an actor does not have any incoming dependencies from another actor of the model then it portrays that the actor has distributed his dependencies to other actor but no other actor has delegated any tasks, resources and goals into this actor. So the actor will have minimal impact on the consequences of the performance of other actors in the strategic context of the model. For this reason an actor with no incoming dependencies will be positioned with minimal criticality fact towards it but the vulnerability factor
of that actor will take it into the consideration of the risk measurement in the strategic framework. Now we need to refine the criticality calculation by relating it at process level.

The formula we use to calculate criticality measurement \((CM_{bp})\) at process level is as follows:

\[
CM_{bp} = Organisational Level Criticality (CM_{org}) \times No of Outgoing Flows
\]

For example, for actor Volunteer,

organisational Level Criticality = 30;

No of Outgoing Flows = 6;

So, Criticality, \(CM_{bp} = 30 \times 6 = 180\).

Table 7.4 illustrates criticality measurement at process level.

<table>
<thead>
<tr>
<th>Actor</th>
<th>(CM_{org})</th>
<th>No of Outgoing Flows</th>
<th>(CM_{bp})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Coordination Centre Coordinator</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>Call Taking Supervisor/System</td>
<td>27</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>Volunteer/Emergency Workers</td>
<td>30</td>
<td>6</td>
<td>180</td>
</tr>
<tr>
<td>Flood Control Centre Coordinator</td>
<td>27</td>
<td>6</td>
<td>162</td>
</tr>
<tr>
<td>Weather Bureau</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Community</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 7.4: Criticality Measurement at Process Level
7.5 Integrating Risks in Business Process Models

7.5.1 Treating Vulnerable Actors

We argue that every actor in the business processes should be given a relative level of effort to mitigate vulnerability via robustness and efficiency. We suggest for more vulnerable actors more monitoring of the tasks/ sub-tasks is necessary. Monitoring of the business process means tracking the individual tasks or subtasks in a process so that information on their state can be easily made visible. It is done to measure the satisfactory performance of a business process. Business process tasks of the vulnerable actors need more monitoring so that we can continually refine them based on feedback that comes directly from operational level.

![Business Process Model in BPMN](image)

Figure 7.3: Business Process Model in BPMN

The process model at Figure 7.3 has two actors WeatherBureau and ECCC with few tasks and subtask. The model also represents exception handling procedure for RecieveForecast task. From Table 7.1, we find that ECCC is the most vulnerable actor
which implies more monitoring of the tasks and subtask inside this process is required.

The process model in Figure 7.3 is improved in Figure 7.4 by using our notion of vulnerability. The exception for ReceiveForecast task is handled by QueryBureau sub-process. We extend this model by integrating ApplyRiskMitigation sub-process. This sub-process includes the risk mitigation procedures which takes place in case of the failure of QueryBureau sub-process.

The analyst should design the organisational model or process model carefully while delegating the dependencies from one actor to other actors. Actor with dependencies over only one actor is more vulnerable than actor with dependencies with multiple actors. For example in Figure 7.5, the vulnerability level of actor A1 is 4 and actor B1 is 1. Actor A1 has four dependencies over A2. If actor A2 fails then all the dependencies will remain unsatisfied. On the other hand actor B1 has delegated its dependencies over...
four actors. If any of the four dependee actor fails one dependency will remain unsatisfied, but the others might be satisfied. Thus actor A1 is more vulnerable than actor B1.

![Diagram of Delegation of Dependencies among Actors](image)

Figure 7.5: Delegation of Dependencies among Actors

### 7.5.2 Treating Critical Actors

Volunteer actor is the most critical actor according to the matrix. In this case the three actors ECCC, FieldControlCentreCoordinator (FCCC) and Community are dependant on Volunteer actor to accomplish their certain objectives. Failure to satisfy these objectives/incoming dependencies will have a big impact on the performance of the depender actors and to the system as a whole. To minimize the criticality levels of actors, the analyst needs to have pragmatic and profound process delegation strategy. The tasks and sub-processes of the most critical actors should be planned to make the whole process robust and fail safe. To make the process robust the analysts need to identify what is the overall objective of the process. This should describe problems to
be solved, issues to be addressed, key participants, whether all the tasks are well integrated within the process and how the processes add values and quality to the system.

The objective of the process in Figure 7.6 is to provide a Flood/Storm Safety advice to the Community. Volunteer provides the safety advice to the Community. For the well completion of the process Volunteer needs to have local information and rescue equipments which are done by FieldControlCenterCoordinator by accomplishing two tasks GatherLocalInformation and ArrangeRescue/EvacuationEquipment. Upon successful completion of the task ReceiveRescueEquipments the Community receives the message ProvideFlood/StormSafetyAdvice from Volunteer in the FollowEvacuationProcedures tasks which add values to the process of evacuation.

The process model at Figure 7.7 is extended from Figure 7.6 by introducing an exception handling technique in Volunteer’s ReceiveRescueEquipments task to manage its
satisfactory performance. If the Volunteer does not receive the rescue equipments from FCCC the process will throw an exception which sends query to FCCC. To handle the risks from negative response from the FCCC a MitigateRisk sub-process is introduced. Exception handling should be taken into thoughtful consideration by the analyst as exceptions may arise in any stages of the process.

![Extended Process Model Reflecting the Critical Actor](image)

Figure 7.7: Extended Process Model Reflecting the Critical Actor

The processes of the critical actors should have mutual consistency to reduce criticality and increases process performance. If a process is allocated to an actor, which the actor may not be capable of performing, it is likely to delay the process which could lead to a disaster. Clearly specified activities for the actors should be one of the most important priorities to the analyst. It makes easy to comprehend and allocate resourceful process design to ease the modification of processes.
7.6 Summary

In this chapter we have presented a discussion on how we can identify risk in terms of vulnerability and criticality in organisational models. We have also provided a way to integrate risks within the process model. We believe it helps the analyst to design organisational models, delegate dependencies among various actors, choose alternatives, decompose tasks, maintain consistency among organisational and process models, handle exceptions etc. Our proposal is based on a combined notation ($i^*$-BPMN) which might not be suitable for organisations using different notations. However, we wish to apply our methodology in other notations in the future. Some interesting research works were conducted in risk network propagation area to assess the vulnerability for all partners in a network [64] [20]. Authors in [64], believe examining and controlling risk propagation from the network and supply-chain perspectives has become vital to overall network security. The authors performed an analysis that illustrated a critical need for some type of coordination policy in information supply chains to monitor and direct information security activities among member firms. We believe network propagation (in organisational models) should be considered in risk identification and measurement of vulnerability and criticality. In the next chapter we extend our methodology by incorporating risk measure propagation through organisational network.
In this chapter we extend our risk measurement methodology detailed in the previous chapter by incorporating actors’ dependency relationships with each other across the whole organisational model. We argue that any actor associated with a vulnerable actor holds certain degree of vulnerability as a result of its association with the vulnerable actor. The degree of vulnerability becomes more if the actor is associated with more vulnerable actors. Similarly, we argue that any actor associated with a critical actor holds certain degree of criticality as a result of its association with the critical actor. The degree of criticality becomes more if the actor is associated with more critical actors. We believe our novel approach is capable of guiding the analyst to analyse vulnerability and criticality levels that exists among the actors’ interdependencies across the whole organisational model; hence enabling him/her to design or redesign risk aware business processes.
8.1 Background

We developed the $i^*$ organisational model in 8.1 in a project that expects to have a significant role in reducing the need for medical assistance in the elderly by enabling people to take a much more active role in the management of their health in their homes.

![Figure 8.1: $i^*$ Organisational Model (SD) of a Home Telecare System](image)

We get the BPMN model in 8.2 by applying our constrained development methodology mentioned in the previous chapters and in [51] [79] [2].
Figure 8.2: BPMN Model of Home Telecare System

8.2 Risk Measurements at Organisational and Process Models
8.2. Risk Measurements at Organisational and Process Models

8.2.1 Vulnerability

The formula we use to assess the vulnerability measurement \( VM_{org} \) of actors at organisational level is as follows:

\[
VM_{org} = \frac{\text{No of Outgoing Dependencies}}{\text{No of Dependee Actors}}
\]

For example, for actor Medicare,

No of Outgoing dependencies = 3 and No of Dependee Actors = 3;

So, Vulnerability at organisational Model, \( VM_{org} = \frac{3}{3} = 1 \).

Table 8.1 illustrates vulnerability measurement of actors at organisational model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>No of Outgoing Dependencies</th>
<th>No of Dependee Actors</th>
<th>( VM_{org} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>General Practitioner</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Receptionist</td>
<td>7</td>
<td>5</td>
<td>1.4</td>
</tr>
<tr>
<td>Patient</td>
<td>18</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Community Nurse</td>
<td>4</td>
<td>3</td>
<td>1.33</td>
</tr>
<tr>
<td>Home Telemed Device</td>
<td>5</td>
<td>3</td>
<td>1.66</td>
</tr>
<tr>
<td>Home Telecare EPRS</td>
<td>16</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>DSS</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Central Care Team</td>
<td>6</td>
<td>4</td>
<td>1.5</td>
</tr>
<tr>
<td>Ambulance</td>
<td>3</td>
<td>2</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 8.1: Vulnerability Measurement of Actors at Organisational Model

We then relate vulnerability measurement at process level. The formula we use to calculate vulnerability measurement \( VM_{init} \) at process level is as follows:
8.2. Risk Measurements at Organisational and Process Models

\[ VM_{init} = \text{Organisational Level Vulnerability (VM}_{org} \) * Number of Incoming Flows (control flow and message flow) \]

For example, for actor Medicare,
organisational Level Vulnerability, VM\(_{org} = 1\) and No of Incoming Flows = 2;
So, Vulnerability at Process Level, VM\(_{init} = 1*2 = 2.\)

Table 8.2 illustrates vulnerability measurement at process model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>VM(_{org})</th>
<th>Incoming Flow</th>
<th>VM(_{init})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>General Practitioner</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Receptionist</td>
<td>1.4</td>
<td>6</td>
<td>8.4</td>
</tr>
<tr>
<td>Patient</td>
<td>3</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Community Nurse</td>
<td>1.33</td>
<td>4</td>
<td>5.32</td>
</tr>
<tr>
<td>Home Telemed Device</td>
<td>1.66</td>
<td>3</td>
<td>4.98</td>
</tr>
<tr>
<td>Home Telecare EPRS</td>
<td>2</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>DSS</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Central Care Team</td>
<td>1.5</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>Ambulance</td>
<td>1.5</td>
<td>3</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 8.2: Vulnerability Measurement at Process Level

8.2.2 Criticality

Criticality of actors at organisational Model is measured by multiplying number of incoming dependencies and number of depender actors. The formula we use to assess the criticality measurement (CM\(_{org}\) at organisational level is as follows:

\[ CM_{org} = \text{No of Incoming Dependencies} * \text{No of Depender Actors} \]
For example, for actor Medicare,

No of Incoming Dependencies = 6 and No of Depender Actors = 5;

So, Criticality at organisational Model, $CM_{\text{org}} = 6\times 5 = 30$.

Table 8.3 illustrates criticality measurement of actors at organisational model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>No of Incoming Dependencies</th>
<th>No of Depender Actors</th>
<th>$CM_{\text{org}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare</td>
<td>6</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>General Practitioner</td>
<td>6</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>Receptionist</td>
<td>5</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Patient</td>
<td>12</td>
<td>6</td>
<td>72</td>
</tr>
<tr>
<td>Community Nurse</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Home Telemed Device</td>
<td>11</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>Home Telecare EPRS</td>
<td>14</td>
<td>6</td>
<td>84</td>
</tr>
<tr>
<td>DSS</td>
<td>6</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Central Care Team</td>
<td>7</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>Ambulance</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.3: Criticality Measurement of Actors at organisational Model

The formula we use to calculate criticality measurement ($CM_{\text{init}}$) at process level is as follows:

$$CM_{\text{init}} = \text{Organisational Level Criticality (}CM_{\text{org}}\text{)} \times \text{No of Outgoing Flows}$$

For example, for actor Medicare,

organisational Level Criticality = 30 and No of Outgoing Flows = 1;

So, Criticality, $CM_{\text{init}} = 30 \times 1 = 30$. 

Table 8.4 illustrates criticality measurement at process level.

<table>
<thead>
<tr>
<th>Actor</th>
<th>CM&lt;sub&gt;org&lt;/sub&gt;</th>
<th>No of Out-going Flows</th>
<th>CM&lt;sub&gt;init&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare</td>
<td>24</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>General Practitioner</td>
<td>24</td>
<td>3</td>
<td>72</td>
</tr>
<tr>
<td>Receptionist</td>
<td>15</td>
<td>6</td>
<td>90</td>
</tr>
<tr>
<td>Patient</td>
<td>72</td>
<td>6</td>
<td>432</td>
</tr>
<tr>
<td>Community Nurse</td>
<td>8</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>Home Telemed Device</td>
<td>22</td>
<td>5</td>
<td>110</td>
</tr>
<tr>
<td>Home Telecare EPRS</td>
<td>84</td>
<td>7</td>
<td>588</td>
</tr>
<tr>
<td>DSS</td>
<td>18</td>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>Central Care Team</td>
<td>35</td>
<td>7</td>
<td>245</td>
</tr>
<tr>
<td>Ambulance</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.4: Criticality Measurement at Process Level

8.3 Risk Measure Propagation Over Organisational Model

In this section we consider the actors representation and their dependencies within the organisation they exist or outside the organisation. We argue the followings:

- Any actor associated with a vulnerable actor holds certain degree of vulnerability as a result of its association with the vulnerable actor. The degree of vulnerability becomes more if the actor is associated with more vulnerable actors.

- Any actor associated with a critical actor holds certain degree of criticality as a result of its association with the critical actor. The degree of criticality becomes more if the actor is associated with more critical actors.
8.3.1 Vulnerability over Organisational Network

We get the initial vulnerability measurements ($VM_{init}$) from the calculation mentioned in the previous sections. However, to simplify the illustration of risk network propagation calculation, for the 8.3 we assume, actors A, B, C, D have a total vulnerability measure of 1 where each of A, B, C and D has vulnerability of 0.25. In 8.3 there are four actors A, B, C and D where A has outgoing dependency links to B and D.

![Figure 8.3: Vulnerability Scenario 1](image)

In this scenario, the formula we use to calculate vulnerability measurement ($VM_{net}$) of actor A at organisational network level is as follows:

$VM_{net}(A) = VM_{init}(B) + VM_{init}(D)$ which is 0.50.
Consider 8.4 where each of the actors has an initial vulnerability of 0.25 and where A has outgoing dependency to B, C and D; B has outgoing dependency to A as well as D; C has outgoing dependency link to A, B and D; D has outgoing link to B. The vulnerability value of the depender actor is divided among all the outbound dependencies to the dependee actors. Thus actor B gives a rating of 0.125 to actor A and a rating of 0.125 to actor D. Only one third of actor C’s vulnerability is counted for actors A’s criticality rating which is about 0.083. In other words, the vulnerability rating conferred by an outbound dependency is equal to the actor’s own vulnerability rating divided by the normalised number of outbound dependencies, d. Please note that dependencies from actors only count once per actor.

Figure 8.4: Vulnerability Scenario 2

Thus, the formula we use to calculate vulnerability measurement (VM_{net}) of actor
A at organisational network level is as follows:

\[ VM_{\text{net}}(A) = VM_{\text{init}}(B)/d(B) + VM_{\text{init}}(C)/d(C) + VM_{\text{init}}(D)/d(D) \]

which is 0.458.

By applying the formula above for actor Medicare in the Home Telecare organisational model we get:

\[ VM_{\text{net}}(Medicare) = VM_{\text{init}}(\text{Central Care Team})/d(\text{Central Care Team}) + VM_{\text{init}}(\text{Receptionist})/d(\text{Receptionist}) + VM_{\text{init}}(\text{Patient})/d(\text{Patient}) = 0.105 \text{ or 10.5 out of 100 (approx)} \]

Where,

\[ VM_{\text{init}}(\text{Central Care Team}) = 0.06; \]
\[ d(\text{Central Care Team}) = 4; \]
\[ VM_{\text{init}}(\text{Receptionist}) = 0.113; \]
\[ d(\text{Receptionist}) = 5; \]
\[ VM_{\text{init}}(\text{Patient}) = 0.403 \text{ and } \]
\[ d(\text{Patient}) = 6. \]

Table 8.5 illustrates the vulnerability measures as a result of applying the formula in all the actors of the Home Telecare organisational model.
8.3.2 Criticality over Organisational Network

We get the initial criticality measurements (CM_{bp}) from the calculation mentioned in the previous sections. However, to simplify the illustration of risk network propagation calculation, for the figure 8.5 we assume, actors A, B, C, D have a total criticality measure of 1 where each of A, B, C and D has criticality of 0.25. In figure 8.5 there are four actors A, B, C and D where A has incoming dependency links from B, C and D.
In this scenario, the formula we use to calculate criticality measurement ($CM_{\text{net}}$) of actor A at organisational network level is as follows:

$$CM_{\text{net}}(A) = CM_{\text{init}}(B) + CM_{\text{init}}(C) + CM_{\text{init}}(D)$$

which is 0.75.

Consider figure 8.6 where each of the actors has an initial criticality of 0.25 and where A has incoming dependency from B, C and D; B has incoming dependency only from A; C has incoming dependency from B as well as D; D has incoming dependency from A, B and C. The criticality value of the dependee actor is divided among all the inbound dependencies from depender actors. Thus actor B gives a rating of 0.25 to
actor A since it has only one incoming dependency link. Actor C gives a rating of 0.125 to actor A and a rating of 0.125 to actor D. Only one third of actor D’s criticality is counted for actors A’s criticality rating which is about 0.083. In other words, the criticality rating conferred by an inbound dependency is equal to the actor’s own criticality rating divided by the normalised number of inbound dependencies, d. Please note that dependencies to actors only count once per actor.

Figure 8.6: Criticality Scenario 2
The formula we use to calculate criticality measurement \((CM_{net})\) of actor A at organisational network level is as follows:

\[
CM_{net}(A) = CM_{init}(B)/d(B) + CM_{init}(C)/d(C) + CM_{init}(D)/d(D)
\]

which is 0.458.

By applying the formula above for actor Medicare in the Home Telecare organisational model we get:

\[
CM_{net}(Medicare) = CM_{init}(Receptionist)/d(Receptionist) + CM_{init}(Central Care Team)/d(Central Care Team) + CM_{init}(Home Telecare EPRS)/d(Home Telecare EPRS) + CM_{init}(Patient)/d(Patient)
\]

= 0.147 or 14.7 out of 100 (approx)

Where,

\[
CM_{init}(Receptionist) = 0.053;
d(Receptionist) = 3;
CM_{init}(Central Care Team) = 0.144;
d(Central Care Team) = 5;
CM_{init}(Home Telecare EPRS) = 0.346;
d(Home Telecare EPRS) = 6;
CM_{init}(Patient) = 0.254;
d(Patient) = 6.
Table 8.6 illustrates the criticality measure as a result of applying the formula in all the actors of the Home Telecare organisational model.

<table>
<thead>
<tr>
<th>Actor</th>
<th>$CM_{\text{init}}$</th>
<th>$CM_{\text{init}}%$</th>
<th>Initial Rating</th>
<th>$CM_{\text{net}}$</th>
<th>$CM_{\text{net}}%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare</td>
<td>24</td>
<td>1.411</td>
<td>0.014</td>
<td>0.147</td>
<td>14.7</td>
</tr>
<tr>
<td>General Practitioner</td>
<td>72</td>
<td>4.236</td>
<td>0.042</td>
<td>0.150</td>
<td>15</td>
</tr>
<tr>
<td>Receptionist</td>
<td>90</td>
<td>5.294</td>
<td>0.053</td>
<td>0.104</td>
<td>10.4</td>
</tr>
<tr>
<td>Patient</td>
<td>432</td>
<td>25.411</td>
<td>0.254</td>
<td>0.129</td>
<td>12.9</td>
</tr>
<tr>
<td>Community Nurse</td>
<td>24</td>
<td>1.411</td>
<td>0.014</td>
<td>0.100</td>
<td>10.00</td>
</tr>
<tr>
<td>Home Telemed Device</td>
<td>110</td>
<td>6.470</td>
<td>0.065</td>
<td>0.100</td>
<td>10.00</td>
</tr>
<tr>
<td>Home Telecare EPRS</td>
<td>588</td>
<td>34.589</td>
<td>0.346</td>
<td>0.135</td>
<td>13.5</td>
</tr>
<tr>
<td>DSS</td>
<td>36</td>
<td>2.118</td>
<td>0.021</td>
<td>0.088</td>
<td>8.8</td>
</tr>
<tr>
<td>Central Care Team</td>
<td>245</td>
<td>14.412</td>
<td>0.144</td>
<td>0.086</td>
<td>8.6</td>
</tr>
<tr>
<td>Ambulance</td>
<td>1</td>
<td>0.0588</td>
<td>0.006</td>
<td>0.029</td>
<td>2.9</td>
</tr>
</tbody>
</table>

**Table 8.6: Criticality Measurement at Organisational Network Level**

### 8.4 Addressing Risk

Actor’s vulnerability and criticality measurements found via risk measure propagation at organisational network level can be used to guide relevant professionals to assess organisational processes and capabilities. Capabilities of an organisation in similar context or industry have similarities. However the success of an organisation largely depends on how processes and technologies are applied to these capabilities. Business capabilities are a combination of business processes, people, technology solutions, and assets aligned by strategic performance objectives [31] [54] [75]. A capability is thus a combination of ‘what’ work is being performed in a certain area, e.g. “Process Invoice for Patients” - and ‘how’ it is done in terms of people, process, and Technology, are
implementations of that capability. So ‘what’ is done in an organisation in a given industry is similar for any organisation in that industry. But ‘how’ a work is done varies from organisation to organisation. As a result we can view the capabilities as a building block of an organisation. They have relationships to each other and to the environment, and analysts and architects need to pay attention to these interfaces, and to be clear on what responsibilities are being assigned to a capability. Together, people, process, technology and performance management yield a capability that has quality characteristics. These quality characteristics are important in driving the capability design process, just as in any other kind of architecture. The scope of this chapter includes process design which focuses on activities to produce outcomes. It can help analysts and architects to guide capability design as capability design includes process design, and adds technology to the consideration.

Since ‘how’ work is get done, and where the value is in the business, is unique in every business, we argue having a stable view of the processes and capabilities provides a lens on the business that allows analyst and architects to isolate where the value is and where the real performance drivers are. By having the vulnerability and criticality measure in an organisational network analysts and architects can easily get answers to the following questions:

- What are the high value processes that are/ have a high risk of under performing?

- What are the under performing processes that would require improvement on a priority basis?

- What exception handling activities are required to support processes?
How the delegation of responsibility among actors should be designed and maintained?

We argue analysts and architects can keep track of the business processes by implementing a scoring method based on the vulnerability and criticality measures. By analysing a large series of process models we implement the table scoring model in 8.7. Feedback from industry practitioners were also taken into account to have brief list of four types which essentially enables them to sort out the most obvious process types that require attention in a short period of time.

<table>
<thead>
<tr>
<th>Vulnerability or Criticality Level</th>
<th>Process Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between 15 and More</td>
<td>A</td>
<td>Highly likely in need of attention, a major redesign and improvement in process is required on an urgent basis to prevent risk</td>
</tr>
<tr>
<td>Between 10 and less than 15</td>
<td>B</td>
<td>Likely to need attention, process redesign and improvement will be required soon to prevent risk</td>
</tr>
<tr>
<td>Between 5 and less than 10</td>
<td>C</td>
<td>Medium attention</td>
</tr>
<tr>
<td>Less than 5</td>
<td>D</td>
<td>Unlikely to need attention</td>
</tr>
</tbody>
</table>

Table 8.7: Vulnerability Measurement of Actors at organisational Model

In terms of organisational capability process type A and B mean that the capability does not have a documented process and that the capability is implemented multiple times across the organisation in a non-standardized way. So in this case type A and B are indicators that more information is needed before change is planned. Process type C and D on the other hand mean that the capability does have a standard/ somehow standard process across the organisation. C and D indicate steady processes that are unlikely requiring changes.
8.5 Summary

In this chapter we illustrated our extended risk measurement methodology in organisational and operational setting by taking into account the organisational network and dependency among actors. We have also presented some suggestive approaches to address risk. In the next chapter we will present a case study on our constrained development methodology.
Chapter 9

Supporting Business Modeling using Multiple Notations: A Case Study

In this chapter, we present a case study that illustrates the combined use of i* and BPMN notations for performing business modeling in a synergistic fashion on a complex project for a large Government Department. We applied our constrained development methodology proposed in chapter 3 and 4 and in [79] [2] [51] to facilitate this modeling practice. The purpose of this case study is to further demonstrate the applicability of our proposed methodology in a real time, big scale industrial project.

9.1 Introduction

In this chapter we present a case study on a large scale project in a government body in Australia. This case study illustrates how the business modeling phase of the project was implemented with the support of multiple modeling notations. The following section starts with background information about the project. We then describe the business modeling strategy that was followed along with a brief discussion on the notations used. We then provide an illustration of the methodology, techniques and templates. Finally we present a discussion section and some concluding remarks.
9.2 Project Background

This case study is based on a large state government body in Australia. The organisation structure is a complex array of directorates and business units with varying needs. It required an enterprise software solution, which can accommodate its strict security requirements while supporting standardised and decentralised processes for time tracking, project management, resource management, financial management and reporting.

The department’s expectations from the single enterprise wide software solution were:

- Providing visibility of projects to the appropriate people
- Enhancing the availability and accuracy of project based information to enable more timely and effective decisions
- Helping map project performance
- Assisting in identifying risks and realising the benefits
- Providing a “single point of truth” through a single point of data entry, version control for reports and a record of decisions made in relation to all projects and
- Enhancing collaboration of project teams not only within their own projects but across other similar projects or areas

The department chose to configure the CA Clarity Project system [17] since its ability to provide the solutions to the organisation’s above-referenced requirements under the terms of strict tender. Since the department was very large in terms of the number of employees, types of services provided, complexity to manage day to day activities
9.3 Business Modeling Strategy

and service delivery; it started with piloting Clarity within one of its learning and business reform programs. The idea was, on the successful completion of the pilot project, the department will implement the software solutions across a wider collection of business units. The pilot program itself was a large scale program that could bring about a staged and coordinated replacement of the current personnel, finance and student administration systems in schools, technical institutions across department. CA Clarity Project and Portfolio Manager (PPM) was used to manage the suite of projects necessary to achieve this objective. The Clarity system also supported the department’s requirement to decentralize system administration, resource and project set up and maintenance, time tracking, project accounting, project cost allocations, posting and reporting processes.

The department’s highest priority at the time of the project was “client success”. This was to be accelerated by improving corporate capabilities to ensure the success of initiatives introduced for this purpose. The project was a result of the department’s need for a long-term project governance solution that will also be used to manage a variety of critical variables such as resource management, project portfolios, and demand for services in a number of major programs.

9.3 Business Modeling Strategy

The project management team decided to conduct detailed business modeling in order to configure the Clarity Systems based on requirements of the stakeholders. The project team put special emphasis to ensure the individual directorate and business units’ requirements were addressed. There were few challenges; the department was
very large with complex organisational structure making it harder to implement the software solutions by eliciting and analysing requirements from every directorates and business units. Also changes at the organisation level as well as the operational level were very common; so a methodology was needed that could track these changes both at organisational and operational level so that the changes to the software can be supported comfortably without losing consistency at these levels. On the other hand, CA Clarity itself is an extensive project and program management tool covering variety of organisational requirements with its own configuration complexity. The idea was to perform business modeling exercise using two different notations (i* organisational Modeling [154] and BPMN [145]) with the help of a constrained development methodology mentioned at [51] [2] [79] and previous chapters. A transition management plan was also produced suggesting the ways on how the department can make move towards the “to be world” from the “as is” world. However, as far as this chapter is concerned we will only concentrate our discussion on the business modeling side of the project.
The business modelling strategy examined the requirements for developing and maintaining one or more business models within the project, recommended the most appropriate approach and defined the techniques, standards, roles and responsibilities for developing and maintaining the required models during the course of the project. The business modeling strategy informed the Project Plan, the Stage Plans, the Project Quality Plan and required Business Models. As mentioned earlier, proposed products of business modeling were: High Level organisational Model (in i* organisational modeling notation), Operational Business Process Model (in BPMN).

Given the organisational size and complexity of the department it is quite normal to have a varied and large range of business requirements models. The large scope of the business units leads to greater complexity. It was decided that a combination of notations will be used in order to facilitate the maintenance of the models in lieu of changes.
in the context of their usage over the course of their lifecycle. Figure 9.1 illustrates the modeling strategy.

For initial requirements engineering exercise $i^*$ organisational modeling technique was used. These models represented the scope, organisational actors/roles and their dependencies and intentional rationale. We then mapped the $i^*$ organisational models into operational BPMN models and vice versa (when required) using our constrained development methodology. Table 9.1 illustrates a comparison among the two modeling notations used in the proposed approach.
### Table 9.1: Comparison among Different Modeling Notations

<table>
<thead>
<tr>
<th>Level</th>
<th>Benefits</th>
<th>Limitations</th>
</tr>
</thead>
</table>
| Organisational Model - $i^*$ | • Model complex organisational structure  
                           • Most important high level consideration  
                           • Represents actor dependencies in terms of goal, resource, task and soft-goal  
                           • Illustrates key social/strategic relationship for business process redesign | • Non operational  
                           • Graphical complexity for larger models |
| Process Model - BPMN   | • Conceptualise current and/ desired business processes  
                           • Make business process transparent; greatly improve visibility and efficiency  
                           • Flexible and expandable; more dynamic change control  
                           • Operational  
                           • Align very well with online documentation | • Low level of granularity  
                           • May become complex if not represented in hyperlinked documents |
Early-phase requirement engineering activities have traditionally been done informally [154], beginning with stakeholder interviews and discussions on the existing systems and rationales. Initial requirements are often ambiguous, incomplete, inconsistent, and usually expressed informally. We added some structure to this informal consultation process via the use of Requirements Capture Templates (RCTs).
### 9.4. Modeling Approach And Methodology Used

**Figure 9.2: Requirements Capture Template**

<table>
<thead>
<tr>
<th>Requirements Capture Template</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function Elaboration for the department</strong></td>
</tr>
<tr>
<td><strong>Department Name</strong></td>
</tr>
<tr>
<td><strong>Function Name</strong></td>
</tr>
<tr>
<td>(Use separate sheet for each function)</td>
</tr>
<tr>
<td><strong>Function Rationales</strong></td>
</tr>
<tr>
<td>(Use separate sheet for each function)</td>
</tr>
<tr>
<td><strong>Activity Details for the Function</strong></td>
</tr>
<tr>
<td><strong>Activity Name and Description</strong></td>
</tr>
<tr>
<td>(Use separate sheet for each activity under the function)</td>
</tr>
<tr>
<td><strong>Activity Rationales</strong></td>
</tr>
<tr>
<td><strong>Responsible Actor(s) involved in the activity</strong></td>
</tr>
<tr>
<td>(Unique list of Actor(s))</td>
</tr>
<tr>
<td><strong>Relationship / dependencies between responsible actor(s) to achieve / satisfy the above activity</strong></td>
</tr>
<tr>
<td>(Relationship is described as the dependency from source actor on to target actor, use separate row for each relationship and dependency)</td>
</tr>
<tr>
<td><strong>Source Actor</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Modeller Signature**

**Stakeholder Signature**
In effect, these were forms that the modeller seeks to fill out in the course of a stakeholder consultation session and that were eventually signed off by both the modeller and the stakeholder. The process of filling out these forms provided structure to stakeholder interview sessions. In addition, these forms were designed to seek information specific to the need of the underlying agent-oriented conceptual model \((i^*)\) that the modeller seeks to build. Stakeholders were thus able to provide focused input to the conceptual modeling task, while being shielded from the complexity of understanding and using the conceptual modeling language.

Once the templates were finalised and the \(i^*\) model was developed, we applied constrained development methodologies proposed at \([79]\) \([51]\) \([2]\) to guide the derivation or maintenance of one type of model given the availability of the other. Figure 9.3 illustrates a sample SR model that was developed for Demand Management (a function in CA Clarity system) modelling.
Figure 9.3: Sample SR Model of Demand Management

The development was supported with the introduction of two concepts: *fulfillment conditions* and *effect annotations* (i.e. as described in [50]); details about these concepts were covered in the previous chapters. The application of the methodology was divided into the following phases:

**Phase-1: Annotating the i* organisation Model** In this phase we annotated the i* model with fulfillment conditions. Annotation of fulfillment conditions was introduced gradually to an i* model by initially annotating dependencies on an SD model. An SR model was then developed/annotated with required effects given an understanding of required fulfillment conditions. An SR model was to be available for use in the methodology, as it provided a natural mapping to a business process model (i.e.
9.4. Modeling Approach And Methodology Used

with a representation of tasks). Additionally, the tasks assigned to participants in a BPMN model were annotated with effects for assessment against fulfillment conditions.

**Phase-2: Scope Projection and Consistency Evaluation**

In order to evaluate consistency between the two notations, we conformed to the rules for projecting the scope of the \( i^* \) model. Scope projection was based on an identification of the business process (represented in BPMN) as a routine assigned to an actor in an \( i^* \) model. Once the scope projection was completed we used the consistency rules provided in the methodology to ensure consistency between \( i^* \) and BPMN models. Details of scope projection and consistency evaluation rules were mentioned in chapter 3 and 4.

**Phase-3: Mapping Rules** In this step we did the actual development of a Business Process Model from the previously developed \( i^* \) models. The steps provided in the methodology guided the development of a business process model given an \( i^* \) model. The checklist at Table 9.2 was used to ensure the analyst address all the phases correctly while developing a BPMN model from an \( i^* \) model.

The methodology also provided systematic guidance for developing an \( i^* \) model given an already existing process model. We used the checklist at Table 9.3 to verify the analyst addressed all the phases correctly while developing an \( i^* \) model from a BPMN. Figure 9.4 represents a sample BPMN model derived from the Demand Management (CA Clarity System function) SR model using the constrained development methodology.
9.4. Modeling Approach And Methodology Used

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have you annotated all the tasks/dependencies of the $i^*$ model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you projected the scope of the $i^*$ diagram to the process in question?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you identified the internal and external actors in $i^*$ diagram?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you represented every required participant actor of $i^*$ model in BPMN model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you Mapped elements to equivalent constructs within the BPMN model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Are the accumulated effects of all tasks in BPMN model are consistent with the fulfillment conditions of the $i^*$ model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you sequenced the required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is the fulfillment of a task on the depender side of a dependency realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.2: Check List A

Figure 9.4: BPMN Model for Demand Management
9.4. Modeling Approach And Methodology Used

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have you annotated all the tasks of the BPMN model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you represented every required participant of BPMN model in $i^*$ model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you mapped activities of BPMN model into $i^*$ model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you applied intentional reasoning to query the intention of tasks and flow-links?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Are the accumulated effects of all tasks in BPMN model are consistent with the fulfillment conditions of the $i^*$ model?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you sequenced the required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is the fulfillment of a task on the depender side of a dependency realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.3: Check List B

All through the business modelings exercise, the following quality criteria were followed:

- Models are aligned with the Project Approach
- Modeling approach and technique meets the modeling requirements in the most efficient and cost-effective way
- Cost of tools and training provided are kept to a minimum
- Models are easy to maintain and lend themselves to an iterative approach
- Models require minimum specialist skills or training to be interpreted by the project team members
• Complete the checklist after developing the BPMN model and make any changes to the model you think are necessary.

9.5 Discussion and Lessons Learnt

Business modeling part of this project was very complex to manage. For a large organisation it is quite usual. Combined business modeling helped the project in the following ways:

• It helped the project to define its scope, identify associated roles, their dependencies, represent the processes embedded in the projects and clarify the developers to design the test cases and implement the configuration of the system.

• Combined models acted as a common language for communication for varied stakeholders’ goals, policy implications, and/or operational constraints by creating a contextual environment.

• It helped to increase the department’s organisational and operational management capability by representing “what business process exists”, and “what business process is required to exist”.

• The constrained development methodology used in this exercise helped the modellers in two ways. Firstly, it made the model transformation (i* to BPMN and also BPMN to i* when required) smooth and consistent. Secondly, in model management when a change was required, this methodology supported tracing and managing changes in organisational models and process models.
• The RCTs presented here eased the requirements elicitation process. However, these templates served other useful functions as well. They provided a structured repository and record of stakeholder interviews that was revisited when requirements were re-negotiated or revised (for instance, when changes were made to models, or when inconsistencies were detected). The detailed rationale recorded in these templates were also of good value in business process re-engineering. To anticipate and support future business process re-engineering efforts in the context of the department, we also captured alternative solution scenarios by completing additional RCTs that answer “how else” questions (while the primary RCTs represent the “as is” scenarios).

We do not claim this modeling effort was successfully completed without any problems. First and foremost we had to train the modellers the notations (i* and BPMN) and the methodology that were used in the process. We had to go through many scenarios to make them understand the concept of the methodology appreciate its usefulness. Secondly, we found there was no tool to support this methodology. A tool could have saved us a lot of time and could make the work more efficient. However, we did continue to get feedback from all parties involved on the use of this methodology and modeling exercise. Some of the concerns that rose from the analysts are:

• Model management is an important issue/challenge perceived by many academics, practitioner and vendors [65]. Managing business process models in an organisation requires significant resources and expertise. Hence, it is not surprising to see that some of the analysts believe that the development, implementation
and management of business process models using two different notations simultaneously might be a quite difficult asks for many organisations.

- Planning to integrate these methodologies would bring various management-related challenges such as change management and resource commitment. This initiative would require clear planning and goal setting which must be accepted by the executives of the organisation. Without this and the commitment to the methodology the initiative is unlikely to succeed. Organisations with little or no expertise in the process modeling area will likely to hire consultants/ modeling experts. While external consultants might bring expertise and specialist knowledge into the organisation, Return on Investment (ROI) need to be carefully examined.

- According to [65], business process model’s ease of use is another attribute that is highly regarded by many credential practitioners, vendors and academics. Some of our analysts believe that individuals without relevant knowledge and expertise in the BPM area might find this methodology quite challenging. This process model should be fully understood otherwise it could cause legitimate problems.

We believe the modeling implementation and management implementation needs to be sustained. The responsibility for this usually lies with modellers, quality group, auditors or even the senior project managers to ensure the methodology lives long past it implementers and original sponsors. We argue the implementation of this business modeling is a long term goal. Once the exercise is complete the aim is to keep them available and ensure the benefits are realised in their full potential.
We realise the potential fact that planning to integrate our methodology may bring various management-related challenges such as change management, resource commitment and knowledge management. We plan to develop a series of position papers to address business case, implementations, financial analysis issues etc to address such initiative with the executive leadership teams.

We plan to develop a tool to support our constrained development methodology. We believe the systematic nature of our methods make them available for automation in all phases, through the development of a software tool, along with further refinement of the approach. We also plan to address potential issue of model management in the future to find a viable automate solution preferably via tool support.

9.6 Conclusion

In this work we have presented an industrial case study that illustrated the business modeling phase of a project. We have illustrated the modeling strategy and modeling approach. We have also discussed how we used the constrained development methodology and the requirements capture templates. In our future work, we plan to elaborate more details on the management of the multiple models produced. In the next chapter we will present an experiment that we conducted on the constrained development methodology.
Chapter 10

Constrained Development Methodology: An Experiment

We proposed constrained development methodologies to guide the derivation and maintenance of one type of model given the availability of the other to effectively support the business process model lifecycle. In this chapter, we present an experiment that we conducted among participants from industry and academia on the Constrained Development Methodology. The purpose of this experiment is to further demonstrate that this methodology was tested and assessed by the experts in organisational and process modeling area. This experiment provided us an insight opportunity to see the actual perception and effectiveness of the constrained development methodology. The following section starts with a discussion on the experiment design followed by a section on experiment execution. We then illustrate the results and discussion in the following section with some concluding remarks.

10.1 Experiment Design

The evaluation approach taken in this experiment is twofold. One part of this study is based on a case study where the participants were asked to develop a business
process model given a high level conceptual model and vice versa. The case study is an empirical research method useful in realistic environments where strict control over variables is not possible. The other part of this study is based on questionnaires. The structure of this study preserves the opportunity for rigorous support of an argument in the absence of formal controls. We use the experiment methodology mentioned in [151].

10.1.1 Part One

Experiment on the Constrained Development of a Business Process Model given an $i^*$ Model

In this section the participants follow the constrained development methodology to develop a Business Process Model from a higher level conceptual $i^*$ model. The SR model in figure 10.1. is used for this purpose. In the $i^*$ framework, the SR model provides a more detailed level of modeling by looking “inside” actors to model internal intentional relationships. Intentional elements (goals, tasks, resources, and softgoals) appear in the SR model not only as external dependencies, but also as internal elements linked by task decomposition and means-ends relationships 10.1.
For example, the TrainingCoordinator has an internal task to OrganizeTraining. This task can be performed by sub-tasks ScheduleTraining and GenerateTrainingContent (these are related to the parent task via task decomposition links). The task OrganizeTraining is related to the LowEffort, Quick softgoals via a task decomposition link. The intention is not to suggest that the softgoal plays the role of a sub-task but to relate the softgoal to the highest level task for which the softgoal may be viewed as an optimization objective. The softgoal thus serves to constrain design decisions on how the task might be decomposed. In this instance, the contribution is positive, i.e., organizing the training material contributes (positively) to achieving the broader goal of making the TrainingMaterialEasyToUse.

The TrainingSystem agent is identified with a high level rationale OrganiseTraining,
modelled as a task. The task is further decomposed into the sub-tasks ImpartTraining, ObtainConfirmation and MaintainSchedule. The Volunteer agent is responsible for the task AcquireTrainingSkills, which is decomposed to the sub-tasks ParticipateInTraining and ProvideConfirmation.

Phase-1: Annotating the Computer Based Training System Model In this phase the participants annotate the $i^*$ model with fulfillment conditions and fill in table 10.1. Note Complete Process ‘CP’ signifies that fulfillment will occur when all other conditions are met.

<table>
<thead>
<tr>
<th>Task/Dependency</th>
<th>Fulfillment Conditions</th>
<th>Task (Post Development)</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10.1: Annotation of Fulfillment Conditions to Respective Tasks/Dependencies.

Phase-2:

Scope Projection In order to evaluate consistency between the two notations, we provided the participants with some rules for projecting the scope of the $i^*$ model. Scope projection is based on an identification of the business process (represented in BPMN) as a routine assigned to an actor in an $i^*$ model.

- Rule 1: The root node of the routine traceable to the process in consideration and all tasks in its first level of decomposition from are to be within scope.

- Rule 2: All dependencies that are associated to a task within the scope of the routine, where the actor in control of the routine (initiator) is the depender are within the scope.
of the process; as well as the tasks assigned to dependee actors.

- Rule 3: All dependencies that are associated to a task within the scope of the routine, where the initiator is the dependee are within the scope of the process iff the task assigned to the depender is part of some decomposition of a task in the scope of the process as per Rule 2; as well as the tasks assigned to the depender actors.

**Consistency Evaluation**

In this step the participants were asked look at the consistency rules provided in the methodology to ensure consistency between $i^*$ and BPMN models. Participants then used the consistency rules to check for violations. The rules are as follows:

- Rule 1: Every actor in an $i^*$ model required as a participant in the routine (traceable to the business process) and any of their tasks must be represented in the BPMN model (and vice versa), assessed via application of scope projection rules.

- Rule 2: There must exist a trajectory in the process model, whereby the operational objective (as encoded in the accumulated fulfillment conditions of traceable tasks) of the routine is achieved, and the sequence of activities is consistent with the requirements specified in the routine as further outlined below:

  - Rule 2.1: The accumulated effect of all tasks and goals traceable to the routine must achieve accumulated routine fulfillment conditions during forward traversal of at least one trajectory in the process model; AND,

  - Rule 2.2: The fulfillment of a task on the depender side of a dependency must not be realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory.

**Phase-3:**
In this step the participants developed a Business Process Model from the given \( i^* \) model. The steps provided in the methodology aim to guide the development of a business process model given an \( i^* \) model.

\textit{Step 1: Project the scope of the \( i^* \) diagram to the process in question.}

\textit{Step 2: Identify internal and external actors in \( i^* \) diagram.}

\textit{Step 3: Map elements to equivalent constructs within the BPMN model.}

\textit{Step 3.1: Map Participants.}

\textit{Step 3.2: Map Activities.}

\textit{Step 4: Sequence required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions.}

\textit{Step 5: Elaborate on sub-processes.}

To verify the syntactic and semantic mapping of BPMN models from \( i^* \) models we ask the participants to follow a checklist that contains the following questions:

A1Q1. Have you annotated all the tasks/dependencies of the \( i^* \) model?

A1Q2. Have you projected the scope of the \( i^* \) diagram to the process in question?

A1Q3. Have you identified the internal and external actors in \( i^* \) diagram?

A1Q4. Have you represented every required participant actor of \( i^* \) model in BPMN model?

A1Q5. Map elements to equivalent constructs within the BPMN model.

A1Q6. Are the accumulated effects of all tasks in BPMN model are consistent with the fulfillment conditions of the \( i^* \) model?

A1Q7. Have you sequenced the required tasks/sub-processes and introduce control and
sequence flow links by analyzing fulfillment conditions?

A1Q8. Is the fulfillment of a task on the depender side of a dependency realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory?

If answer to any of the questions above is ‘no’ except question A1Q8, then the participants were asked to review their model development

**Experiment on the Development of an i* Model given a Business Process Model**

In this section the participants follow the constrained development methodologies to develop an i* model from a Business Process Model in BPMN. The process model in figure 10.2 is used for this purpose. The objective of the process in figure 10.2 is to provide a Flood/Storm Safety advice to the Community. Volunteer actor provides the safety advice to the Community. For the well completion of the process Volunteer needs to have local information and rescue equipments which are done by FieldControlCenterCoordinator by accomplishing two tasks GatherLocalInformation and ArrangeRescue/EvacuationEquipment. Upon successful completion of the task ReceiveRescueEquipments the Community receives the message ProvideFlood/Storm-SafetyAdvice from Volunteer in the FollowEvacuationProcedures tasks which add values to the process of evacuation.
10.1. Experiment Design

Figure 10.2: Business Process Model in BPMN

Annotating the Rescue BPMN Model In this phase the participants annotate the BPMN model with effects and and fill in table 10.2. Note, effect annotations in this experiment are informal. Also note Complete Process ‘CP’ signifies that fulfillment will occur when all other conditions are met.
10.1. Experiment Design

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Effects</th>
<th>Fulfillment Conditions Annotation (Post Development)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10.2: Annotation of Effects to Respective Tasks/Dependencies.

The following steps provide systematic guidance to the participants for developing an \(i^*\) model given an already existing process model.

*Step 1: Map elements to equivalent constructs within the \(i^*\) model.*

*Step 1.1: Map Participants.*

*Step 1.2: Map Activities.*

*Step 2: Apply intentional reasoning.*

*Step 2.1: Query the Intention of Tasks.*

*Step 2.2: Query the Intention of Flow-Links.*

*Step 3: Identify soft-goal dependencies in the \(i^*\) model.*

To verify the syntactic and semantic mapping of \(i^*\) models from BPMN models we ask the participants to follow a checklist that contains the following questions:

*A2Q1. Have you annotated all the tasks of the BPMN model?*

*A2Q2. Have you represented every required participant of BPMN model in \(i^*\) model?*

*A2Q3. Have you mapped activities of BPMN model into \(i^*\) model?*

*A2Q4. Have you applied intentional reasoning to query the intention of tasks and*
flow-links?

A2Q5. Are the accumulated effects of all tasks in BPMN model are consistent with the fulfillment conditions of the $i^*$ model?

A2Q6. Have you sequenced the required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions?

A2Q7. Is the fulfillment of a task on the depender side of a dependency realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory?

If answer to any of the questions above is ‘no’ except question A2Q7, then the participants were asked to review their model development

10.1.2 Part Two

In this part we ask the participants to provide their feedback on the methodology. The question we ask them are given below:

BQ1. How do you rate the overall quality of the methodology?

1 = Failed to meet my expectations

2 = Neutral

3 = Met my expectation

4 = Strongly met my expectation

5 = Exceeded my expectation

BQ2. What are your most important ‘take away’ ideas from the methodology?

BQ3. Do you think this methodology is ready to use at organisational and process level?

BQ4. What are the most positive sides of this methodology?
BQ5. What are the most negative sides of this methodology?

BQ6. Do you think the $i*$ and BPMN models will be consistent with each other if you use this methodology? Please provide details.

BQ7. Do you have any other comments on the methodology?

10.2 Experiment Execution

This structured experiment was conducted with thirty four process modeling practitioners from five public sector organisations, four private IT consulting companies and four Australian universities. Eight participants from public sector organisations, eleven participants from private consulting companies and fifteen participants were from universities 10.3. Most of the participants were from ITC educational background. Some participants had more experience in modeling technical designs of IT system and some had more on functional business process designs.
10.2. Experiment Execution

Among these participants, sixteen had more than five years of industry experiences on process modeling, eleven participants had three to five years of industry experiences, six participants had less than three years of industry experience, and one participant had no industry experience on BPMN 10.4. However, the later participant had knowledge on BPMN and designed process models in research projects. He also had extensive experience on i* organisational modeling technique, UML activity diagram, class diagram and entity-relationship diagram. Participants in this survey were also from the different demographic background in terms of modeling business processes using BPMN. Three of the participants have worked in India in the past and have designed business processes for local financial and telecommunication sectors. The participants varied in levels of experience in both process and organisational modeling and usage of BPMN for modeling purposes.
In our research work we focus on how process design can be managed in dynamic organisational context. We argue using our constrained methodology business process model can be developed without any inconsistency given a high level organisational model and vice versa. We also argue that the constrained development methodology is capable of guiding an analyst when reflecting changes from an $i^*$ model to a BPMN model and vice-versa. To evaluate our belief we establish a sound methodological approach to empirically test our methodology. We conduct the experiment among thirty four process modeling experts practicing in Australia and overseas over a period of eight weeks. We then store the feedback from participants and carefully analyse them. Each of the participating practitioner were given a set of questions to verify the syntactic and semantic mapping of BPMN model from a given $i^*$ model and vice versa. This experiment provided us an insight opportunity to see the actual perception and effectiveness of the constrained development methodology in terms of managing process
design through business process model lifecycle. Extensive feedback from the participants allowed us to see further issues with the methodology.

**Experiment on the Constrained Development of a Business Process Model**

**given an i* Model** section of Part One questionnaire is designed to guide the development of a process model in BPMN given the availability of a high level organisational model (i.e. i* model). Participants were asked to define the scope of the i* diagram to the process in question to reduce the complexity of the model. It helps them to consider only the actors and dependencies that are related to the process in consideration. Identification of internal and external actors in an i* organisational model helps to identify the organisational boundary. This is required as BPMN separates internal organisational actors by representing them as lanes within pools whereas external actors are assigned their own pool. This step also provides an understanding of communication requirements. Communication between internal actors within pools is implied, whereas communication between internal and external participants is explicitly stated with message flow links in BPMN. Participants were then asked to represent every required participant actor of i* model in the BPMN model to maintain consistency between the models.

Mapping elements to equivalent constructs within the BPMN model is done in two steps. First step being mapping participants where external participants were represented as pools and internal organisational actors are represented as lanes within the organisational pool. There must be an initiating actor in the process model that is identified by the placement of a start event within the respective actor’s pool or lane. The initiator is identified as having responsibility over the execution of the routine
in the $i^*$ model. Second step being mapping activities where ‘primitively workable’
tasks within $i^*$ are represented as either sub-processes or automatic activities within
BPMN assigned to actors within pools and lanes. Participants were then asked to re-
view whether the accumulated effects of all tasks in BPMN model are in fact consistent
with the fulfillment conditions of the $i^*$ model for consistency checking.
Participants were also asked to sequence required tasks/sub-processes and introduce
control and sequence flow links by analysing fulfillment conditions. It ensures that
tasks are sequenced using control flow links in a manner that results in a trajectory
satisfying fulfillment conditions on an $i^*$ model. Sequence flow helps the users to realise
dependencies between actors within the same organisation. Participants can develop
each sub-process guided by the list of required fulfillment conditions annotated to the
$i^*$ task that the sub-process realises. In order to maintain consistency between the
two models the fulfillment of a task on the depender side of a dependency must not be
realised before the fulfillment of the dependency upon accumulation of effects during
forward traversal of the same trajectory.

**Experiment on the Development of an $i^*$ Model given a Business Process**

Model section of Part One questionnaire was designed to guide the development of
a high level conceptual model (i.e. $i^*$) given a business process model (i.e. BPMN).
Participants were asked to represent every required participant of BPMN model in $i^*$
model to ensure consistency.

Mapping elements from the BPMN model to equivalent constructs within the $i^*$ model
is done in two steps. First step being mapping participants where both pools and lanes
in a BPMN model represent actors in an $i^*$ model that can be directly translated into
the model. Second step being mapping activities where activities and sub-processes as ‘primitively workable’ tasks assigned to actors in $i^*$. Participants were then asked to apply intentional reasoning by asking ‘why’ questions to identify higher-level intentional elements and dependencies by querying the intention of tasks. This step aims to guide the further understanding and representation of an actor’s motivations. It also helps to identify task/goal dependencies on dependee actors outside the current organisation. These dependencies are to be reflected between actors in the $i^*$ model. Intentional reasoning could also be used to identify soft goal dependency.

Participants were also asked to review whether the accumulated effects of all tasks in BPMN model are in fact consistent with the fulfillment conditions of the $i^*$ model for consistency checking. In order to maintain consistency between the two models the fulfillment of a task on the depender side of a dependency must not be realised before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory. It is also important to sequence required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions.

Model management is an important issue/challenge perceived by many academics, practitioner and vendors [65]. Managing a single business process model in an organisation requires significant resources and expertise. It is interesting to see that about half of the participants with industry experience believe that the implementation and management of two business process models simultaneously might be a quite difficult ask for many organisations. However the participants realised the benefits of having
this methodology implemented in organisations. They argue although the initial development of these two modeling notations may require running a project by committing some resources to it, model management can be maintained through the use of appropriate tools.

Planning to integrate this methodology would bring various management-related challenges such as change management and resource commitment. This initiative would require clear planning and goal setting which must be accepted by the executives of the organisation. Without this and the commitment to the methodology the initiative is unlikely to succeed. Organisations with little or no expertise in the process modeling area will likely to hire consultants/modeling experts. While external consultants might bring expertise and specialist knowledge into the organisation, Return on Investment (ROI) for the organisation need to be carefully examined.

According to [65], ease of use is an attribute that is highly regarded by many credential practitioner, vendors and academics. While most of the participants find the methodology easy to apply, about 24% participants mentioned that individuals without relevant knowledge and expertise in the BPM area might find this methodology quite challenging. This process model should be fully understood otherwise it could cause legitimate problems.

10.3 Results and Discussion

Almost 78% participants agree constrained development methodology is quite suitable to manage process design in a dynamic organisational context. About 86% participants also find it simple and easy to derive a high level conceptual model (i.e. $i^*$) given
a business process model (i.e. BPMN) and vice versa. 64% participants believe that this methodology to be a very positive one with some refinements. However, about 9% participants argue since i* does not provide necessary information in “orders” which is required to develop BPMN, information might not be completely transformed between models for large systems. They argue a tool is required to support model transformation which we believe is also required and we are looking forward to develop a tool to support this in the future.

The figure at 10.5 illustrates the ratings that participants gave for the methodology. Majority of them (about 67%) mentioned that the constrained development methodology either exceeded their expectation or nearly exceeded their expectation. 21% participants rated 3 for this methodology with some suggested improvement. A small portion of the participants (12%) rated either 1 or 2. However, they provided suggestion for improvement.

![Ratings from Participants](image)

Figure 10.5: Participants Response on Methodology
We argue our constrained development methodology is a methodological approach for supporting the lifecycle of business process models with the complementary use of $i^*$ - a well developed notation for modeling organisational contexts, and BPMN - an industry accepted and widely used notation for modeling business processes. This approach for reflecting changes in organisational context to changes in the design of business processes provides an effective mechanism for aligning business processes with organisational objectives. Similarly, operational improvements can be mapped back to organisational objectives to facilitate analysis and ensure no conflicts exist with existing objectives. Even though these steps are preliminary we believe their systematic nature makes them available for automation in all phases.

About 76% participants agree that constrained development methodology is capable of guiding the development of process models from higher-level conceptual models and vice versa in a synergistic manner. By doing so this supports life-cycle management in the sense that when changes occur to the high-level model, they can be reflected in the process model, and when change occur in operational process model they can be reflected in high level organisational model.

We believe that in order to effectively conceptualize business process, development of business process models on principled high-level contextual models of the enterprise that illustrate its motivations, resources, and internal/external social/strategic inter-dependencies. Moreover, any purposeful changes made to business process models must be reflected within the high-level model for analysis against the greater context of the enterprise. About 78% of the participants believe that this methodology could provide support to analysts in achieving this task by guiding them during the design of process
models given the availability of $i^*$ models and vice-versa.

About 71% of the participants believe that unlike some preliminary approach [30] constrained development methodology provides methods to use both notations synergistically and in a “co-evolutionary” manner. Furthermore, another 78% believe that the approach provides limited elaboration on the sequencing of activities required in BPMN to fulfill the various dependencies expressed in the $i^*$ model.

Unlike the method proposed in [4], for deriving a BPMN model from a business model, 84% of the participants believe that the constrained development methodology is a simpler approach aimed at reducing added complexity and/or misinterpretations during modeling. About 63% of the participants believe that the introduction of consistency rule in the constrained development methodology also provide a mechanism for ensuring that the organisational and operational context is in fact consistent to each other.

10.4 Summary

The objective of this chapter was to evaluate whether using constrained development methodology a business process model can be developed without any inconsistency given the availability of a high level organisational model and vice versa. We conducted this experiment among some process modeling experts practicing in Australia and overseas. The experiment results illustrate that a significant number of participants found that constrained methodology can be used to systematically manage process modeling. Most of the participants also believe individuals with a little business
process modeling knowledge would find the methodology quite simple and easy to derive a high level conceptual model (i.e. $i^*$) given the availability of a business process model (i.e. BPMN) and vice versa. More than three quarter of the participants have indicated that constrained development methodology is capable of supporting business process life-cycle management in terms of effectively managing change. They believe our approach for reflecting changes in organisational context to changes in the design of business processes provides an effective mechanism for aligning business processes with organisational objectives. Similarly, operational improvements can be mapped back to organisational objectives to facilitate analysis and ensure no conflicts exist with existing objectives.
Chapter 11

Conclusion and Future Directions

In this chapter, a brief conclusion of our research is provided.

11.1 Conclusion

In this thesis we have focused on devising novel methods that have the capability to address some of the most challenging issues in managing business process design and modeling in dynamic organisational context. In order to do so, our chosen set of notations were $i^*$ modeling framework and Business Process Modeling Notation (BPMN). We have maintained our objectives to facilitate process design in a dynamic organisational context by bridging the gap between organisational relationship and operational processes by providing a process mapping and change management methodology. The objectives of this thesis have been achieved firstly by providing a detailed analysis of the concept that flexibility and combination of notations is required in order to facilitate the maintenance of the models, secondly by developing of methods to facilitate and support the change and/or combined use of notations ($i^*$ and BPMN) for modeling business process and thirdly by developing a methodology to integrate risks in process models through a set of intuitive metrics for extracting measures of actor criticality,
and vulnerability from organisational models.

This is accomplished through a detailed literature review, development of a new methodology that supports integrated use of \( i^* \) and BPMN models and testing the methodology in organisational context.

The outcomes of this research are as follows:

- Detailed analysis of the concept that flexibility and combination of notations is required in order to facilitate the maintenance of the models.

- Development of methods to facilitate and support the change and/or combined use of notations (\( i^* \) and BPMN) for modeling business process.

- A methodology to integrate risks in process models through a set of intuitive metrics for extracting measures of actor criticality, and vulnerability from organisational models.

The major contributions of this work are listed below:

- We propose a constrained development methodology that suggests a combined approach for supporting the business process model lifecycle. This provides the capabilities for an analyst to manage and reflect changes from an \( i^* \) model to a BPMN model and vice-versa. We argue that the management of change throughout the business process model lifecycle can be more effectively supported by combining notations. In particular, we identify two potential sources of process change, one occurring within the organisational context and the other within the operational context. As such the focus is on the co-evolution of operational
(BPMN) and organisational ($i^*$) models. Our intent is to provide a way of expressing changes, which arise in one model, effectively in the other model.

- We then illustrate in detail how business process and organisational models can be correlated to manage change. Taking account the fact that as organisations grow in size and complexity, process improvement initiatives may involve change that has major impact across an organisation. We provide methods and extensions to existing process modeling notations to analyse change against high-level models of the organisation. Our approach permits improved analysis against higher-level organisational structures, motivations, inter-dependencies and capabilities that should be ideally considered as primary requirements during process design.

- We then present the application of our constrained development methodology by using $i^*$ and UML Activity Diagram. This in general illustrates a level of flexibility when it comes of different notations used for similar purposes. Unified Modeling Language (UML) is suitable for later phases of requirement capture which usually focus on completeness, consistency, and automated verification of functional requirements for the new system. We propose methods to facilitate and support the combined use of notations ($i^*$ and UML Activity Diagram) for modeling requirement engineering process in a synergistic fashion.

- We argue there is a need for supporting risk identification with the use of higher-level organisational models and business process models. Business processes represent the operational capabilities of an organisation. In order to ensure process continuity, the effective management of risk becomes an area of key concern. We
propose an approach for supporting risk identification with the use of higher-
level organisational models. We provide some intuitive metrics for extracting
measures of actor criticality, and vulnerability from organisational models. This
helps direct risk management to areas of critical importance within organisation
models. Additionally, the information can be used to assess alternative organis-
isational structures in domains where risk mitigation is crucial. At the process
level, these measures can be used to help direct improvements to the robustness
and failsafe capabilities of critical or vulnerable processes.

- We then extend our risk measurement approach by incorporating actors’dependency
relationships with each other across the whole organisational model. We argue
that any actor associated with a vulnerable actor holds certain degree of vul-
erability as a result of its association with the vulnerable actor. The degree of
vulnerability becomes more if the actor is associated with more vulnerable actors.
Similarly, we argue that any actor associated with a critical actor holds certain
degree of criticality as a result of its association with the critical actor. The degree
of criticality becomes more if the actor is associated with more critical actors. We
believe our novel approach is capable of guiding the analyst to analyse vulnera-
bility and criticality levels that exists among the actors’interdependencies across
the whole organisational model; hence enabling him/her to design or redesign
risk aware business processes.

- Finally we present a detailed case study that illustrates the combined use of $i^*$
and BPMN notations for performing business modeling in a synergistic fashion
on a complex project for a large government department. We applied our con-
strained development methodology to facilitate this modeling practice. This case
study further demonstrate the applicability of our proposed methodology in a
real time, big scale industrial project. We also present the details of an exper-
iment conducted among participants from industry and academia with results
and findings.

11.2 Future Directions

Future work is required to fully realise the benefits of the methodologies illustrated in
the thesis. This involves work in the following areas:

- We realise the potential fact that planning to integrate our methodology may
bring various management-related challenges such as change management, re-
source commitment and knowledge management. This initiative would require
clear planning and goal setting which must be accepted and encouraged by the
executives of the organisation. Without this level of support the benefits of such
initiative is unlikely to be realised. Organisations with little or no expertise
in the process modeling area will likely to hire consultants/ modeling experts.
While external consultants might bring expertise and specialist knowledge into
the organisation, Return on Investment (ROI) need to be carefully examined. We
plan to develop a series of position papers to address business case, implement-
tations, financial analysis issues etc to address such initiative with the executive
leadership teams.

- We plan to develop a tool to support our constrained development methodology.
11.2. Future Directions

We believe the systematic nature of our methods make them available for automation in all phases, through the development of a software tool, along with further refinement of the approach.

- Model management is an important challenge perceived by many academics, practitioner and vendors. Managing, supporting and maintaining models in an organisation requires significant resources and expertise. It may be challenging for some organisations to develop, implement and manage models in two different notations simultaneously. We plan to address this potential issue in the future to find a viable solution.

The combined methodology illustrated throughout this thesis have high potential commercial merit. Given the completion of the future work mentioned above, organisations may realise the potential benefits of applying this methodology as a single packaged solution to managing process design in a dynamic organisational context.
Appendix A

Glossary

- **AOCM** - Agent Oriented Conceptual Modeling presents a conceptual model for developing software systems that are open, intelligent, and adaptive. It describes an approach for modeling complex systems that consist of people, devices, and software agents in a changing environment. The introduction of agent into requirement engineering supports requirements elicitation, exploration and analysis of software systems. Agent has characteristics such as intentionality, autonomy, and sociality. Therefore, agent orientation is currently seen as a software paradigm. Systems with characteristics such as autonomy, sociality, reactivity and proactivity, and communicative and cooperative abilities can provide greater functionality and higher quality when using agent orientation than other software paradigm, such as object orientation. Agent models and languages can represent the computational behaviors in an abstract manner so that they can be realised in software programs.

- **Business Process Management** - Business Process Management (BPM) is a re-emerging field of study, with determined focus from both industry and academia alike. BPM promotes that a clear understanding through the explicit
modeling of the processes underlying an organisation is required to support effective organisational management/improvement practices. The importance of process-orientation, as opposed to functional or departmental orientation is evident in the holistic view it provides of organisational operations, as described in the many antecedents to the field. That is, the process perspective supports improvement initiatives by enabling a holistic impact assessment in terms of operational changes, rather than the traditional approach of piecemeal improvement in a specific functional area to the detriment of other functional areas, or even to some extent, the overall operational effectiveness of the organisation. From the perspective on information systems developers, the semi-formal representation of organisational context that business process models provide allow for rapid/traceable development and deployment of systems that can effectively support the operations of an enterprise. Added benefits arise when the models can be translated to ‘process code that can be understood by ‘process-aware information systems, effectively accelerating the change process.

- **Business Model Process Management Lifecycle** - Business processes evolve throughout their lifecycle of management, brought on by the constant nature of change in today’s business environment. This evolution can be characterized by a number of distinct phases which are *Discovery, Design, Deployment, Enactment, Monitoring, Analysis* and *Disposal*.

- **Business Model Process Modeling** - Business process modeling aims to conceptualise current or desired business processes in a common (preferably graphical) language/notation that can be communicated to all involved participants.
These include business users (i.e. analysts, partners, managers and even first line workers etc.) and technical users (i.e. systems/network architects, programmers and support staff etc.). Business process models also form the basis of Business Process Management Systems (BPMS). Thus, it is of benefit if a business process model can be mapped to an executable process language (e.g. XPDL) that is understood, and can be executed by, a BPMS.

- **BPMN** - Business Process Model and Notation (BPMN) is a standard for business process modeling that provides a graphical notation for specifying business processes in a Business Process Diagram (BPD), based on a flowcharting technique very similar to activity diagrams from Unified Modeling Language (UML). Business Process Management Initiative (BPMI) developed BPMN, which has been maintained by the Object Management Group since the two organizations merged in 2005. As of March 2011, the current version of BPMN is 2.0. The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation and the underlying constructs of execution languages, particularly Business Process Execution Language (BPEL). The primary goal of BPMN is to provide a standard notation readily understandable by all business stakeholders. These include the business analysts who create and refine the processes, the technical developers responsible for implementing them, and the business managers who monitor and manage them. Consequently, BPMN serves as a common language, bridging the
communication gap that frequently occurs between business process design and implementation.

- **i* Modeling Notation** - *i* Modeling Notation models organisational context and offer high-level social/anthropomorphic abstractions (such as goals, tasks, softgoals and dependencies) as modeling constructs. It has been argued that such notations help answer questions such as what goals exist, how key actors depend on each other and what alternatives must be considered. The central concept in *i* is that of intentional actor. Intentional properties of an agent such as goals, beliefs, abilities and commitments are used in modeling organisations. The actor or agent construct is used to identify the intentional characteristics represented as dependencies involving goals to be achieved, tasks to be performed, resources to be furnished or softgoals (optimization objectives or preferences) to be satisficed. The *i* framework also supports the modeling of rationale by representing key internal intentional characteristics of actors/agents. The *i* framework consists of two modeling components: Strategic Dependency (SD) Models and Strategic Rationale (SR) Models.

- **NFR (Non Functional Requirement)** - Requirements consist of functional requirements and non-functional requirements. Functional requirements define the capabilities that the system should provide. Functional requirements concern the data input and output; they can be easily quantified and evaluated. Non-functional requirements (NFRs) are constraints on the functions offered by the system such as performance constraints, timing constraints, etc. Nonfunctional requirements do not describe any of the systems input, processing, or output.
**RE** - Requirement Engineering is one of the most important phases in software development lifecycle. Users typically know how to perform their day to day job operations. But they usually do not have an understanding of what systems should do or how would systems support their processes. Experienced consultants or analysts recognise ambiguous, incomplete or contradictory requirements. Requirement Engineering is the process of identifying the functions and real world goals of a software system.

**SOA** - A Service-Oriented Architecture (SOA) is an information technology approach or strategy in which applications make use of services available on the network such as the World Wide Web. A SOA is a form of distributed systems architecture, essentially a collection of services. Different from traditional application architecture, which links all the business logic together to compose a single application, the fundamental component of SOA is a service and services can be composed in specific ways to build applications. The foundation of SOA is made of basic services, their descriptions and basic operations, such as publication, discovery, selection and binding. SOA provides consistent interoperability and reuses existing services where possible. Implementing a SOA can involve developing applications that use services and making applications available as services.

**UML** - Unified Modeling Language is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created, by the Object Management Group. It was first added to the list of OMG adopted technologies in 1997, and has since become
the industry standard for modeling software-intensive systems. UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system’s architectural blueprins.

- **UML Activity Diagram** - Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types are rounded rectangles represent activities, diamonds represent decisions, bars represent the start (split) or end (join) of concurrent activities, a black circle represents the start (initial state) of the workflow, an encircled black circle represents the end (final state). Arrows run from the start towards the end and represent the order in which activities happen.


[9] Bartk, R., accepted as poster to JFPLC ’97, also available as Tech. Report No 97/1, Department of Theoretical Computer Science, Charles University, Prague, January 1997.


[34] Daniel A. Menasce, QoS Issues in Web Services, IEEE Internet Computing, and November. December 2002


on Agent-Oriented Software Development Methodologies International Journal of Multi-Agent and Grid Systems.


[115] Lin Padgham, Michael Winikoff, Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents.


velopment, 6.

[147] Molly Wilson, Alan Boring, Hierarchical Constraint Logical Programming, Jour-
nal of logic programming,1993.

[148] Molly Wilson, Hierarchical Constraint Logic Programming, Technical Report 93-


[152] Youngblood, M. D. Winning cultures for the new economy. Strategy and Leaders-
ership, 28(6), 4-9, 2000.


